Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

BÚSQUEDAS INTELIGENTES POR MEDIO DE ONTOLOGÍAS

Honard Manuel Bravo Bámaca
Asesorado por el Ing. Carlos Gustavo Alonzo

Guatemala, febrero de 2013

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

BÚSQUEDAS INTELIGENTES POR MEDIO DE ONTOLOGÍAS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

HONARD MANUEL BRAVO BÁMACA
ASESORADO POR EL ING. CARLOS GUSTAVO ALONZO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, FEBRERO DE 2013

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Walter Rafael Véliz Muñoz
VOCAL V	Br. Sergio Alejandro Donis Soto
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Luis Fernando Quiñónez López
EXAMINADOR	Ing. José Alfredo González Díaz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

BÚSQUEDAS INTELIGENTES POR MEDIO DE ONTOLOGÍAS

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha julio de 2012.

Honard Manuel Bravo Bámaca

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Guatemala, 18 de octubre de 2012

Ingeniero
Carlos Alfredo Azurdia Morales
Coordinador de Privados y Trabajo de Graduación

Respetable Ingeniero Azurdia:

Por este medio le informo como asesor del trabajo de graduación del estudiante universitario de la carrera de Ingeniería en Ciencias y Sistemas, HONARD MANUEL BRAVO BÁMACA, carné 200611509, que he revisado el trabajo de graduación titulado: "BÚSQUEDAS INTELIGENTES POR MEDIO DE ONTOLOGIAS", y a mi criterio el mismo está completo y cumple con los objetivos propuestos para su desarrollo según el protocolo.

Agradeciendo su atención a la presente,

Atentamente,

Ing. Carlos Gustavo Alonzo
Asesor de trabajo de graduación
Colegiado: 6358

Ing. Carlos Gustavo Alonzo
Ingeniero en Ciencias y Sistemas
Col. 6358

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 30 de Octubre de 2012

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **HONARD MANUEL BRAVO BÁMACA** carné **200611509,** titulado: **"BÚSQUEDAS INTELIGENTES POR MEDIO DE ONTOLOGÍAS",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA - INGENIERIA EN CIENCIAS Y SISTEMAS - COORDINACION

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación titulado ***"BÚSQUEDAS INTELIGENTES POR MEDIO DE ONTOLOGÍAS"****, realizado por el estudiante HONARD MANUEL BRAVO BÁMACA, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 04 de febrero 2013

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.046.2013

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **BÚSQUEDAS INTELIGENTES POR MEDIO DE ONTOLOGÍAS**, presentado por el estudiante universitario: **Honard Manuel Bravo Bámaca,** procede a la autorización para la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
Decano

USAC SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, febrero de 2013

/cc

ACTO QUE DEDICO A:

Dios	El gran ingeniero modelador y creador del universo y la vida, que me ha dado la fuerza, el valor, el coraje y la esperanza para seguir adelante
Mi padre	Ancelmo Manuel Bravo, por enseñarme el valor de la vida, de la familia, del trabajo duro y honesto, este triunfo es fruto de su gran esfuerzo.
Mi madre	Alicia Victorina Bámaca, que con su gran amor y apoyo incondicional, ha sido la inspiración para alcanzar mi meta.
Mis hermanos	Erick Ferdinando, Belyna Rosymery y Hanelore Adalí, gracias por ser parte de mi vida, gracias por los grandes momentos que hemos pasado juntos.
Doris Herrera	Mi amor, gracias por estar a mi lado y ser mi apoyo incondicional.
Mis amigos de la universidad	Por todos los momentos vividos en nuestra vida académica, si estoy aquí hoy es en gran medida a la calidad de amigos que he tenido.

AGRADECIMIENTOS A:

La Universidad de San Carlos de Guatemala	Por la posibilidad de desarrollar todos mis conocimientos y culminar mi carrera. Por el apoyo económico brindado permaneceré en eterna gratitud.
Familia Camel Bámaca	Por todo el apoyo brindado durante el tiempo de mis estudios, estaré siempre agradecido.
Ing. Carlos Gustavo Alonzo	Por apoyarme en la realización de este trabajo de graduación, motivándome a la realización de un buen trabajo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Algoritmo Secuencia finita de instrucciones, reglas o pasos que describen de forma precisa las operaciones de un ordenador debe realizar para llevar a cabo una tarea en un tiempo finito.

C Lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

C++ Lenguaje de programación diseñado a mediados de 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos.

CERN Centre Européen pour la Recherche Nucléaire, Centro Europeo para la Investigación Nuclear.

DARPA Agencia del Departamento de Defensa de los Estados Unidos responsable del desarrollo de nuevas tecnologías para uso militar.

DTD *Document Type Definition*, definición de tipo de documento, es una descripción de estructura y sintaxis de un documento XML o SGML.

E-Business	En español: negocio electrónico y se refiere al conjunto de actividades y prácticas de gestión empresariales resultantes de la incorporación a los negocios de las tecnologías de la información y la comunicación (TIC) generales y particularmente de internet.
FTP	*File Transfer Protocol*. Protocolo para transferencia de archivos, forma parte del conjunto TCP/IP. Este protocolo hace posible transferir archivos de una computadora a otra, a través de un servidor y cliente FTP.
HTML	*HyperText Markup Language* (*Lenguaje de Marcado de Hipertexto*), es el lenguaje de marcado predominante para la elaboración de páginas web.
Ingeniería de Software	Disciplina o área de la Ingeniería que ofrece métodos y técnicas para desarrollar y mantener software.
Linux	Hace referencia a la familia tipo Unix de sistemas operativos con el *kernel* de Linux.
Open Source	Traducido al español como código abierto, es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

PDF PDF (sigla del inglés *portable document format*, formato de documento portátil) es un formato de almacenamiento de documentos.

Solaris Es un sistema operativo de tipo Unix desarrollado desde 1992 inicialmente por Sun Microsystems y actualmente por Oracle Corporation como sucesor de SunOS. Es un sistema certificado oficialmente como versión de Unix. Funciona en arquitecturas SPARC y x86 para servidores y estaciones de trabajo.

SQL El lenguaje de consulta estructurado o SQL (por sus siglas en inglés *structured query language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas.

Spam Se llama spam, correo basura o mensaje basura a los mensajes no solicitados, no deseados o de remitente no conocido (correo anónimo), habitualmente de tipo publicitario, generalmente enviados en grandes cantidades (incluso masivas) que perjudican de alguna o varias maneras al receptor.

Stemming Es un método para reducir una palabra a su raíz o, en inglés, a un *stem* o tema. Por ejemplo una consulta sobre bibliotecas también encuentra documentos en los que solo aparezca bibliotecario porque el *stem* de las dos palabras es el mismo bibliotec.

URL	*Uniform Resource Locator,* es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, videos, presentaciones digitales, etc.
W3C	El World Wide Web Consortium, abreviadoW3C, es un consorcio internacional que produce recomendaciones para la World Wide Web.
Web	En informática, la World Wide Web, es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de internet.
Web invisible	Se conoce como Internet profunda o Internet invisible (en inglés: Deepnet, Invisible Web, *Dark* Web o *Hidden* Web) a todo el contenido de Internet que no forma parte de las páginas indexadas por las redes de los motores de búsqueda de Internet. Esto es debido a las limitaciones que tienen las redes para acceder a todas las webs por distintos motivos.
XML Schemas	Ampliación o mejora de los DTD, es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML.

RESUMEN

Después de la Segunda Guerra Mundial y con el surgimiento de gran cantidad de conocimiento nunca antes vista, surgieron voces alertando y proponiendo soluciones al problema de la sobrecarga de información. El punto de partida en el área de la recuperación de información sin duda ha sido el documento de Vannevar Bush "Como deberíamos pensar" y su planteamiento para solucionar la sobrecarga de información.

Después de ello y a partir de los 50's empezaron a surgir investigaciones académicas sobre métodos, modelos y metodologías que permitirían la representación, almacenamiento y búsqueda de información utilizando computadoras. En esta época surgieron los modelos de recuperación conocidos como: tradicionales.

A partir de los 90's, con la llegada de la World Wide Web, fue necesario cambiar el paradigma de los modelos tradicionales y trasladarlos al ámbito web: una arquitectura distinta, con relaciones entre documentos (hipervínculos) y circunstancias distintas. El gran logro de este segundo aire en la recuperación de información sin duda ha sido el algoritmo *PageRank* que ha llevado a sus creadores a tener una de las empresas de tecnología más grandes e importantes del mundo.

Desde la propuesta inicial de la web, Tim Berners Lee la propuso como un sistema dotado de significado, por ello, él y sus colaboradores en la W3C proponen la Web Semántica, la cual, no es una nueva web infraestructuralmente, sino, es una extensión de la actual, dotada de las

herramientas necesarias para proveersignificado no sólo a los humanos sino también y en especial a agentes inteligentes de software.

Por tal razón fue propuesta una pila de herramientas que soportarían la implementación de esta nueva web: Unicode, URI, XML, XMLS, RDF, RDFS, Ontologías, lógica, entre otras.

El elemento más importante parecen ser las ontologías, éstas nacieron en el área de la Inteligencia Artificial (en cuanto se refiera a herramientas de computación) y que se adaptó muy bien a las necesidades de la Web Semántica. Las ontologías están compuestas por clases, propiedades, individuos, axiomas y jerarquía, elementos esenciales que permiten formalizar áreas del conocimiento. Y cuando se habla de formalizar, se refiere a la capacidad de representarlo en un lenguaje claro y explícito que permite compartirlo, no sólo a personas sino también a agentes de software.

La implementación de búsquedas inteligentes por medio de ontologías permite recuperar no sólo resultados exactos, sino también resultados relacionados y recomendados a partir de las relaciones intrínsecas de la ontología. Éste conocimiento podría ser utilizado por los índices de los motores de búsqueda para contextualizar las búsquedas y los resultados. Sin embargo, esta tecnología está plenamente dirigida a la inferencia automática de conocimiento.

Se ha implementado un sistema que indexa la ontología en un índice de todas las propiedades de datos de la misma y al presentar los resultados de una búsqueda concreta, puede presentar las propiedades de datos y objetos relacionadas al resultado, ha esto se le ha llamado búsquedas inteligentes.

OBJETIVOS

General

Desarrollar un prototipo de sistema de gestión del conocimiento para un dominio del conocimiento específico e implementar búsquedas inteligentes.

Específicos

1. Realizar un análisis documental de la evolución de los sistemas de recuperación de información.

2. Describir la diferencia entre recuperación de información tradicional y recuperación de información en la web.

3. Elaborar un estado del arte de las ontologías.

4. Establecer la diferencia entre búsqueda de información y recuperación de información.

5. Desarrollar un prototipo de sistema de gestión del conocimiento con base en ontologías.

INTRODUCCIÓN

La propuesta de Tim Berners-Lee de dotar de significado a la web actual fue bautizada como Web Semántica, una web con las herramientas suficientes para poder inferir conocimiento automáticamente, para ello W3C, encabezada por Berners-Lee, estableció un *framework* de tecnologías, entre éstas las Ontologías.

Considero adecuado y necesario tener una vista general de la situación respecto al área de la recuperación de información y cómo ésta se puede ver favorecida con las tecnologías semánticas. Para ello se ha dividido este documento en cuatro capítulos que recorrerán, de forma a veces general y otra específica, la evolución de la recuperación de información.

El primer capítulo versa sobre los inicios de la recuperación de información, la necesidad original descrita por Vannevar Bush sobre el problema de la sobrecarga de información y las técnicas desarrolladas para la indexación automática y los modelos de recuperación de información, así como los primeros sistemas de recuperación implementados. Esto permitirá conocer los orígenes de esta área de las ciencias de la computación.

En el segundo capítulo se analiza las diferencias entre la recuperación de información tradicional y la recuperación de información en la web, la adaptación de la recuperación de información con las características propias de la web. El más importante avance en este período ha sido el desarrollo del algoritmo *PageRank,* el más representativo de los algoritmos de análisis de vínculos. Se muestran las características y forma de calcular el *PageRank* de

cada página. También se da un recorrido por los primero motores de búsqueda web y algunas características de su implementación, en especial de Google y su arquitectura.

En el tercer capítulo se presenta la propuesta de Web Semántica y la pila de tecnologías que la conforman y la importancia de las ontologías en esta nueva manera de representar los recursos web, por ello se describen las características, tipos, proceso de creación y herramientas que ayudan en la elaboración de una ontología.

En el capítulo final se presenta una conceptualización de búsquedas inteligentes definiendo los elementos que se consideran parte de éstas. Se muestra, también el proceso de creación de la ontología de dominio y la población con varios individuos. Finalmente se implementa un sistema de gestión del conocimiento con base en la ontología creada, se indexan las propiedades de datos de la ontología y se contextualizan los resultados por medio de las propiedades de objetos de la ontología, lo que da como resultado las búsquedas inteligentes.

1. EVOLUCIÓN DE LOS SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

1.1. Introducción

La humanidad ha elaborado una base sólida de conocimiento, la cual, hasta inicios del siglo XX, se almacenaba y consultaba únicamente en bibliotecas. Tener una copia de algún manuscrito significaba tener poder económico, político o social.

En el siglo III A.C la Antigua Biblioteca de Alejandría llegó a albergar 900 000 manuscritos. *Zenódoto* (considerado el primer bibliotecario de Alejandría) sería el diseñador del primer catálogo temático de la historia, llamado *Pinakes*, el catálogo estaba compuesto por varios rollos de papiro o pergamino. *Pinakes* estaba organizado por autores y temas, una entrada típica estaba compuesta por: nombre del autor, lugar de nacimiento, nombre del padre, nombre del maestro, formación académica, bibliografía del autor y lista de publicaciones. La entrada de un libro contenía el título de la obra, un resumen de su contenido, el nombre del autor e información de dónde provenía el libro.

Hoy día, se estima que la Biblioteca del Congreso de los Estados Unidos alberga 138 millones de documentos en 470 idiomas, la cual no solamente resguarda libros, sino también, audio y video.

Con la aparición de los sistemas de computación a mediados del siglo XX, se introdujo el término digital, lo cual provee la capacidad de manipular

información codificada en binario. A partir de este punto ha sido posible almacenar una gran cantidad de documentos, audio y video en formato binario y así almacenar muchos más documentos para luego ser consultados. Las bibliotecas están entre las primeras instituciones que implementaron sistemas de recuperación de información.

A mediados del siglo XX surgieron las primeras teorías acerca de cómo representar los elementos (frecuentemente documentos), la forma de almacenarlos, la manera de buscarlos y consultarlos, a esto, en ciencias de la computación, se le ha llamado recuperación de información *(RI).*

La importancia de forjar conocimiento a partir del conocimiento previamente generado lo describió Sir Isaac Newton con su célebre frase: “Si he visto más allá es porque me he parado en hombros de gigantes.”

En este capítulo se presentará el concepto de sobrecarga de información como la existencia de tal cantidad de información que vuelve difícil la toma de decisiones debido a la incapacidad de utilizar toda ésta. Se continuará con la descripción de los sistemas de recuperación de información y los modelos de recuperación de información.

1.2. Sobrecarga de información (*information overload*)

El lector podría preguntarse: ¿Habrá alguien capaz de leer los 138 millones de documentos de la Biblioteca del Congreso de los Estados Unidos?... probablemente no. Hay que reducir un poco el ámbito de la búsqueda. Qué tal si un investigador desea conocer todo sobre la Química, ¿Cuántos libros necesitará leer? Seguramente, miles de ellos. Lo cual continúa siendo poco probable que alguien sea capaz de tal hazaña.

Debido a ello, es que ha surgido la especialización, hay investigadores que se dedican a un área específica de conocimiento y todos los investigadores en su conjunto generan nuevas investigaciones a un ritmo más o menos exponencial (ver: Línea de Tiempo de la Información y los Sistemas de Recuperación). Es posible que existan miles de investigaciones sobre un tema, pero si un investigador sólo puede acceder a unas cuantas de ellas, es muy probable que su síntesis no esté a la altura de las exigencias, puede suceder también, que debido a la presencia de tanta información se dificulte de manera exagerada la realización de su síntesis.

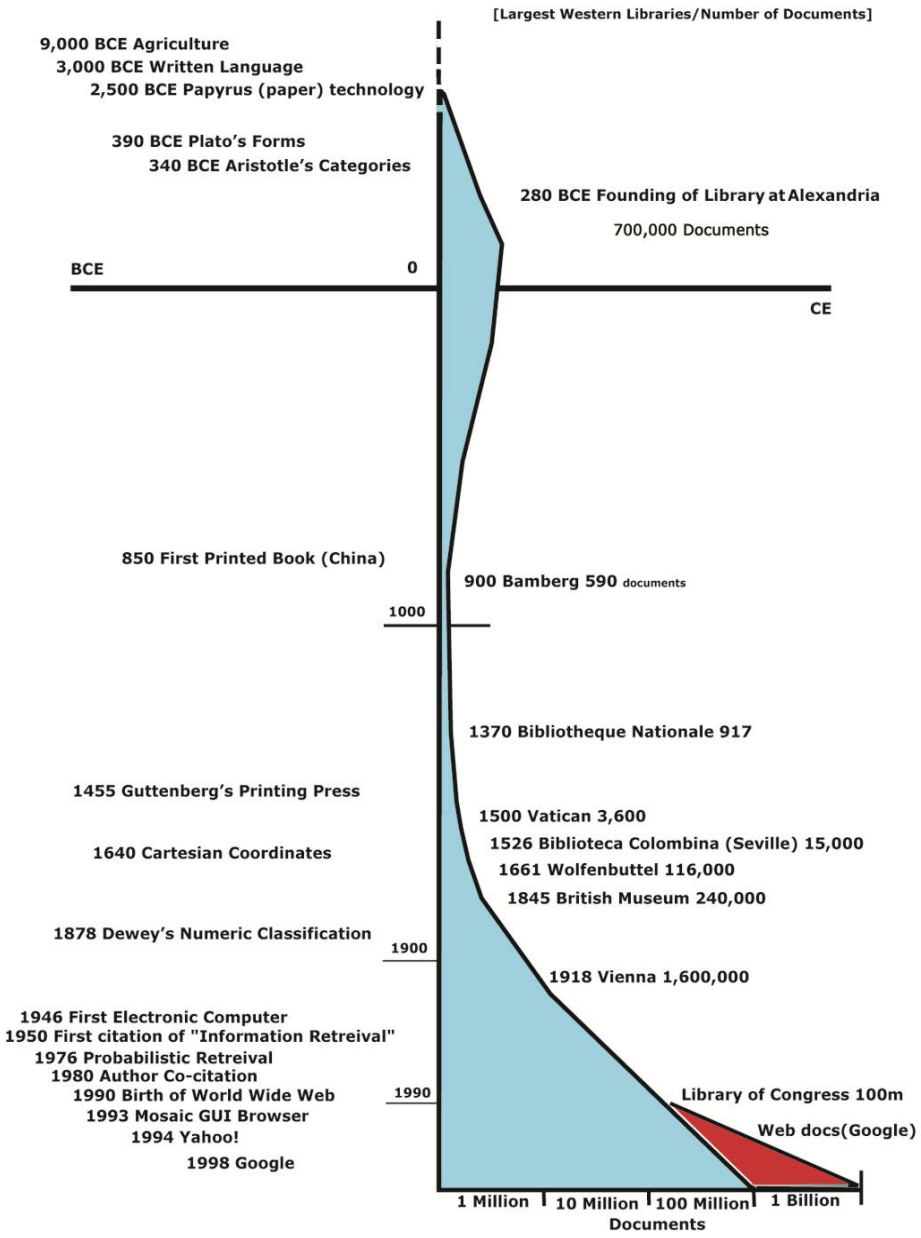
La figura 1 presenta la Línea de Tiempo de los Sistemas de Recuperación de Información y el Crecimiento de la Información. Esta línea del tiempo representa el crecimiento de la información (en documentos) desde el 9000 a.C., hasta 1998 de nuestra era, así como los hitos más importantes, para el autor, sobre la recuperación de información.

La sobrecarga de informativa (*information overload)* se refiere a la "dificultad que puede tener una persona (o sistema) en la toma de decisiones causada por la presencia de demasiada información"[1] y la incapacidad de utilizar (procesar) ésta.

Al tratar de lidiar con mucha información las decisiones se pueden retrasar o decidir equivocadamente, Alvin Toffler lo sabía y fue el primero en mencionarlo en su libro futurista *Future Shock* en donde menciona por primera vez el término *information overload.*

[1] GRAMAJO, Javier. *Knowledge bases system: Automatic contextualization using a domain Ontology*. 2010.

Figura 1. **Línea de Tiempo de la Información y los Sistemas de Recuperación**



Fuente: FIELDEN, Ned. History of Information Retrieval Systems & Increase of Information Over Time. http://userwww.sfsu.edu/~fielden/hist.htm. Consulta: 6 de septiembre de 2012.

La recuperación de información es un elemento que permite reducir la sobrecarga de información eligiendo solo aquella información que se considere relevante.

1.3. Sistemas de recuperación de información

La búsqueda de información no es un procedimiento trivial, éste es trascendental y fundamental para el progreso de la humanidad. La información es útil para generar conocimiento colectivo, sólo cuando ésta llega a alguien que es capaz de "comprenderla, ampliarla y difundirla"[2]. En este contexto, deben existir mecanismos para almacenar la información, pero más importante aún, mecanismos para consultar esa información.

La recuperación de la información (RI en adelante) es el área de las Ciencias de la Computación, cuya investigación formal inició en los 50's, ésta es la encargada de la "representación, almacenamiento, y la organización de la información, y su posterior acceso y recuperación que responda a las necesidades de un usuario"[3]. En consecuencia, a partir de una necesidad de información, seleccionar los elementos relevantes de una colección finita y que éstos sean capaces de satisfacer tal necesidad.

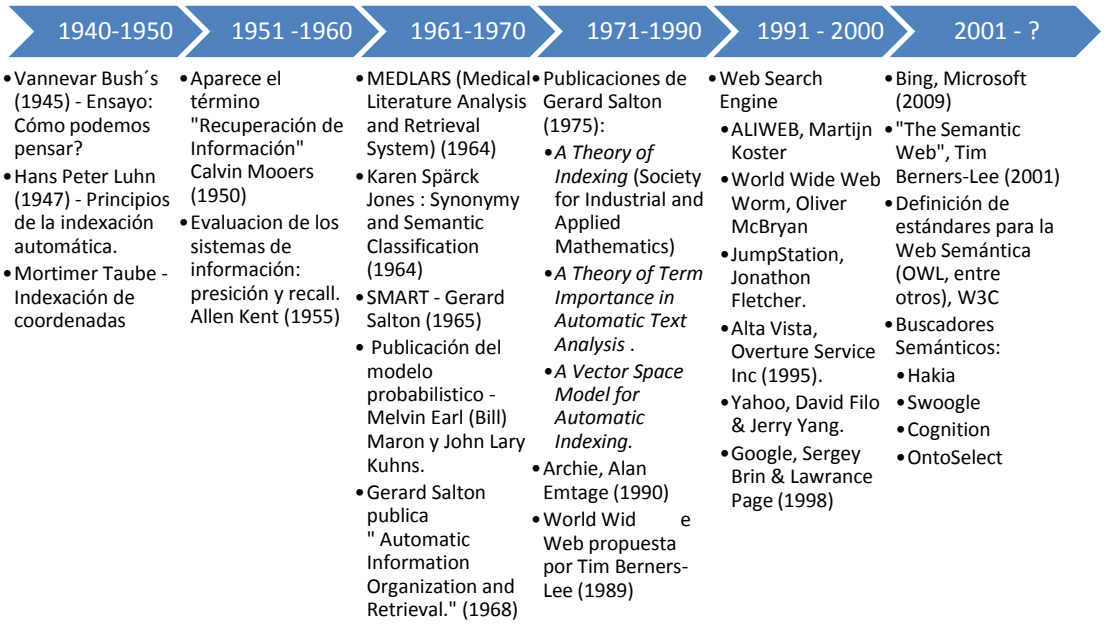
La necesidad de información generalmente la establecen los humanos en lenguaje natural, lo cual hace el concepto de relevancia algo muy ambigua y difícil de formalizar. Lo que es relevante para una persona no puede serlo para

[2] BUSH, Vannevar. *As We May Think*. Boston, Massachusetts: The Atlantic Monthly, 1945.

[3] BAEZA-YATES, Ricardo y RIBEIRO-NETO, Berthier. *Modern Information Retrieval*. s.l.: Addison-Wesley, 1999.

otra persona, es el elemento de relevancia lo que diferencia la recuperación de información a la recuperación de datos.

Figura 2. **Línea de Tiempo de la Recuperación de Información**



Fuente: elaboración propia.

La Línea de Tiempo de la Recuperación de Información muestra en términos generales los hitos importantes en RI y su evolución a través del tiempo. En 1945 Vannevar Bush escribió: "*As We May Think*" un artículo considerado el hito inicial que propicio el enfoque de esfuerzos en la investigación de RI, este ensayo surgió en el contexto de la Segunda Guerra Mundial y la necesidad de crear nuevos métodos para representar y buscar información.

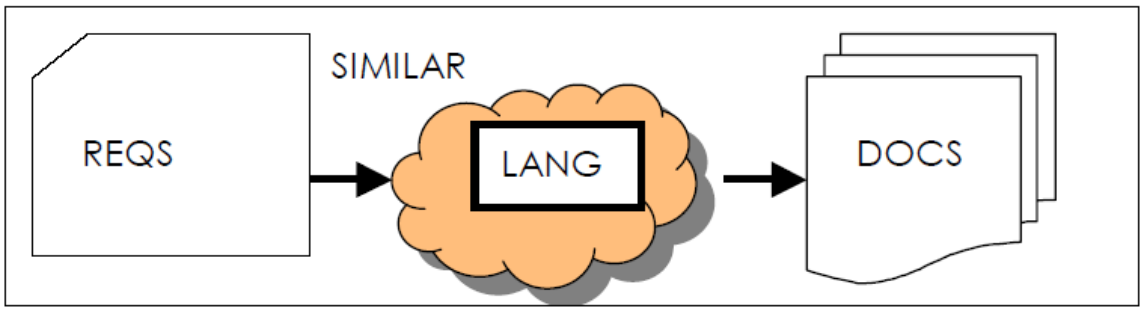
Para Heting Chu los pioneros en el campo de RI son: Mortimer Taube (1910 -1965), Hans Peter Luhn (1896 - 1964), Calvin N. Mooers y Gerard Salton (1927 – 1995). Mortimer Taube y dos de sus colegas propusieron e implementaron el método de indexación por coordenadas basado en *unitermy* lógica booleana, luego Hans Peter Luhn introdujo técnicas que permitieron la indexación automática de documentos la cual hasta entonces se hacía manual. A Calvin Mooers se le reconoce el acuñar el término *information retrieval* y la ley de Mooers: “Un sistema de recuperación de información tenderá a no ser usado cuanto más difícil y molesto sea para alguien tener información que no tenerla.”

Gerald Salton es considerado una prominente figura en el campo e RI gracias a sus investigaciones sobre los métodos de procesamiento automático y completo de textos implementados en SMART.

Con base en la definición de RI, es posible describir cualquier sistema de recuperación de información (SRI en adelante) en cuatro partes básicas: una colección de elementos (DOCS), un conjunto de petición (REQS), un mecanismo (SIMILAR) que determine qué elementos satisfacen las necesidades de información expresadas por el usuario de la petición y la representación de los documentos a un formato especial (LANG).

El proceso de búsqueda, es decir comparar la representación de REQS y LANG (representación de DOCS), se define en SIMILAR. Este algoritmo debe explora las similitudes entre la representación de la pregunta y la representación de los elementos de la colección, así decidir cuáles elementos cumplen la necesidad de información.

Figura 3. **Esquema básico de un SRI**



Fuente: SALTON, G.; Mc GILL, M.J. Introduction to Modern Information Retrieval, 1983.

Al inicio de la investigación sobre RI el ámbito más explorado fue precisamente LANG, la representación de los documentos en el SRI. La adecuada representación tendrá un impacto definitivo en la búsqueda de los documentos. A los modelos de representación se les conoce como: modelos de recuperación de información.

1.4. Modelos de recuperación de información

En el esquema básico de un SRI, LANG representa el modelo de RI. Un modelo de RI es una especificación sobre como representar documentos y consultas, y como comparar unos y otros.

También podría definirse como: cómo se obtienen las representaciones de los documentos y de la consulta, las estrategias para evaluar la relevancia de un documento respecto a una consulta, los métodos para establecer la importancia (orden) de los documentos de salida y los mecanismos que permiten una realimentación por parte del usuario para mejorar la consulta.

Básicamente un modelo es una representación abstracta de un proceso. Un modelo RI es una representación abstracta de RI, es decir, a partir de una necesidad de información el modelo intentará predecir si un documento cumple la necesidad o no y en qué grado.

Un modelo se define como:

- "La representación de los documentos en la colección.
- La representación de las necesidades de información (las consultas.)
- Un conjunto de funciones que permiten crear las representaciones de los documentos, las consultas y las relaciones entre estos.
- Funciones para determinar la relevancia de los documentos (*ranking).*"[4]

A continuación se muestra una clasificación de los modelos de recuperación de información:

[4]BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. *Modern Information Retrieval*. s.l.: Addison-Wesley, 1999. p. 23.

Figura 4. **Taxonomía de los modelos de recuperación de información**

Modelos de Recuperación de Información		
Modelos clásicos	Teoría de conjuntos	Booleano Clásico. Lógica Difusa. Booleano Extendido.
	Algebraicos	Vectorial Generalizado (indexación). Redes neuronales.
	Probabilístico	Inferencia de Red. Redes de creencias.
Modelos estructurados	Listas no sobrepuestas. Nodos próximos.	
Browsing	*Flat Browsing.* Estructura guida. *Hypertext browsing*.	

Fuente: DOMINICH, S. 2000.*A unified mathematical definition of classical information retrieval.* Journal of the American Society for Information Science, 2000. p. 624.

1.4.1. Modelos clásicos

Los modelos clásicos surgieron al inicio de la investigación formal sobre RI, estos modelos son: booleano, vectorial y probabilístico. Los modelos clásicos describen (representan) un documento como un conjunto de términos claves (*index terms)*. Un *index term* es una palabra clave que describe al documento.

1.4.1.1. Teoría de conjuntos (booleano)

Basado en la teoría de conjuntos y el álgebra booleana, por ello es sencillo de comprender e implementar en los primeros sistemas comerciales bibliográficos.

Define la relevancia de un ítem en términos binarios, es decir, un documento es relevante o no es relevante, sin ninguna escala intermedia. Cada ítem se representa en un conjunto de términos de tal forma que un término está presente o ausente del ítem. Las consultas se expresan mediante expresiones booleanas: AND, OR, NOT. El resultado de la consulta es un conjunto no ordenado de ítems, sin ningún grado de relevancia (equiparación exacta).

De este modelo surge el fichero invertido (ver anexo 1), el cual es usado en los buscadores *web* actuales.

1.4.1.2. Modelos algebraicos (vectorial)

El modelo vectorial, inicialmente formulado por Gerard Salton (1971), se basa en la teoría matemática vectorial. La principal diferencia con el modelo Booleano es valor dado a la relevancia, ya no se considera únicamente como relevante y no relevante, sino hay grados, esto es llamado grado de similitud entre los documentos almacenados.

1.4.1.3. Modelos probabilísticos

Formulado por Stephen Robertson y Sparck Jones, basados en la premisa que el proceso de RI es intrínsecamente impreciso. En principio para representar un documento se utilizan un conjunto de términos, para recuperar un ítem se considera su probabilidad, la probabilidad de que un ítem sea relevante a partir de una consulta, está probabilidad debe ser mayor que la probabilidad de no ser relevante.

La probabilidad se calcula a partir de cada uno de los términos ¿Qué probabilidad tiene una término de ser buen descriptor del ítem? (probabilidad de

que el término empleado en la consulta esté presente en un documento del conjunto de documentos relevantes en relación a la consulta). El resultado es una colección de ítems ordenados respecto a su relevancia, equiparación parcial, según su probabilidad.

1.5. Evaluación en recuperación de información

La evaluación de los sistemas de recuperación de información pueden hacerse desde distintos enfoques: evaluación funcional, evaluación de rendimiento y evaluación de rendimiento de recuperación. Las métricas de las dos primeras categorías son utilizadas para evaluar cualquier sistema de información, la tercera categoría contiene métricas específicas para los sistemas de recuperación de información.

Las métricas permiten analizar y comprender el sistema y así controlar y mejorar la calidad del mismo. Las métricas más usuales en la recuperación de información son la precisión y la exhaustividad, existen otras como pérdida, *fallout* y *richness*.

La tabla evaluación en RI describe el comportamiento en RI., A representa el número de documentos recuperados que son relevantes a la consulta, $A + C$ representa el número total de documentos recuperados relevantes y no relevantes. $A + B + C + D$ es el número total de ítems en la colección.

Tabla I. **Evaluación en RI**

	Recuperado	No Recuperado
Relevante	A	B
No Relevante	C	D

Fuente: elaboración propia.

La precisión se define como el cociente de la división del número de documentos relevantes recuperados entre el total de documentos recuperados.

Ecuación 1: precisión.

$$\boldsymbol{precisión} = \frac{\boldsymbol{A}}{\boldsymbol{A} + \boldsymbol{C}}$$

La exhaustividad es la proporción de documentos relevantes recuperados sobre el total de documentos relevantes.

Ecuación 2: exhaustividad (*recall*).

$$\boldsymbol{Exhaustividad} = \frac{\boldsymbol{A}}{\boldsymbol{A} + \boldsymbol{B}}$$

La pérdida es la proporción de documentos relevantes no recuperados y el total de documentos relevantes de la colección.

Ecuación 3: pérdida

$$\boldsymbol{Pérdida} = \frac{\boldsymbol{B}}{\boldsymbol{A} + \boldsymbol{B}}$$

La proporción documentos no relevantes no recuperados entre los documentos no relevantes se le conoce como *fallout*.

Ecuación 4: *fallout*

$$\boldsymbol{Fallout} = \frac{\boldsymbol{D}}{\boldsymbol{C} + \boldsymbol{D}}$$

La proporción de la colección que es relevante se le conoce como *richness*.

Ecuación 5: *richness*

$$\boldsymbol{Richness} = \frac{\boldsymbol{A} + \boldsymbol{B}}{\boldsymbol{A} + \boldsymbol{B} + \boldsymbol{C} + \boldsymbol{D}}$$

Con estas métricas (existen otras que no se muestran en éste documento) es posible comparar el resultado de los distintos modelos de RI. Así, por ejemplo, si un algoritmo *A* tiene mayor precisión y mayor exhaustividad que un algoritmo *B*, entonces el algoritmo A es mejor que *B*.

Los algoritmos que implementan un modelo RI pueden evaluarse desde distintos enfoques, como por ejemplo: los resultado (descrito arriba), el uso de los recursos requeridos (procesador, memoria), tiempo de respuesta, escalabilidad, etc. Lo que en ingeniería de software se llaman requerimientos no funcionales.

1.6. Teoría de la indexación automática (espacio vectorial)

Uno de los modelos más importantes en la historia de la RI es la teoría de la indexación (TIA en adelante), conocida también como vectorial. Las primeras

ideas sobre TIA fueron de Hans Peter Luhn. Luhn describió un método estadístico (frecuencia de aparición de palabras en el texto) para codificar y, posteriormente, recuperar información textual de forma totalmente automática y una técnica para obtener resúmenes automáticos.

Posteriormente, Gerard Salton define la TIA como: “La indexación consiste en asignar a los registros almacenados de atributos, elegidos para representar colectivamente la información contenida en los registros correspondientes.”[5]

Básicamente este método consiste en extraer todos los términos de la colección de documentos y definir un peso (ponderación) y así establecer la relevancia de cada término en el documento, para luego devolver aquellos en donde la relevancia sea mayor.

Algunas técnicas que forman parte de la TIA, y cuyo objetivo es ponderar los términos, las cuales fueron desarrolladas a mediados de los 60’s y aún tienen utilidad hoy en día, son:

- Frecuencia de los parámetros *(TF – term frecuency):* frecuencia de ocurrencia de un término *k* en un documento de la colección. Debe omitirse palabras comunes como “de”, “y”, “pero”; estas palabras formar para de la lista de palabras vacías -*stop words*-.Se supone que un término muy repetido será muy importante.

- Parámetros basados en valores de discriminación *(IDF – inverse documento frecuency-):*se descubrió también, que el término no sólo es

[5]SALTON, Gerard. *A Theory of Indexing*. Philadelphia: Society for Industrial and Applied Mathematics.: Cornell University, 1975.

relevante si aparece muchas veces (*tf*), sino que es más valiosa cuanto más raro es, es decir, cuanto menor es el número de veces que aparece en la colección.

La combinación de *a* y *b* define el método para ponderar términos llamado *tf-idf*.

Para indexar un registro es necesario representarlo en un vector índice -*index vector*-de *t* elementos, siendo *t* igual al número de términos indexables que existen en la colección, se representa esto como un conjunto de pares (a_{ij}, w_{ij}) que definen el término a_i y su respectivo peso w_i para un documento D_i, de la colección.

Ecuación 6: vector índice de un documento

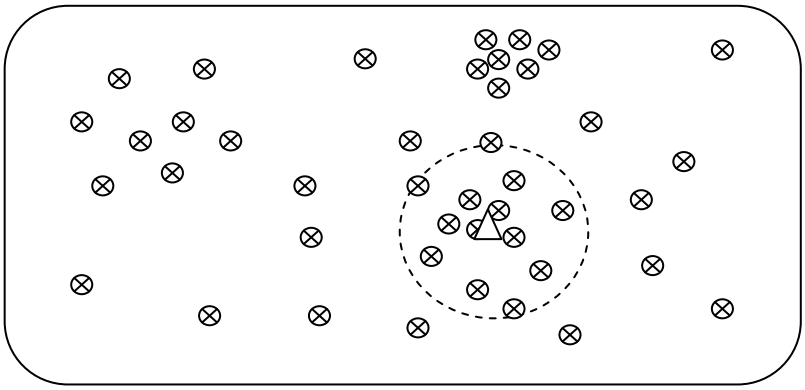
$$D_i = (a_{i1}, w_{i1};\ a_{i2}, w_{i2}; \dots, ; a_{it}, w_{it})$$

Es posible comparar la similitud *s* del vector índice de dos documentos D_i y D_j, a través de:

Ecuación 7: fórmula de similitud de dos vectores índices (lo que en geometría se conoce como la fórmula de coseno).

$$s(D_i, D_j) = \frac{\sum_{k=1}^{t} w_{ik} w_{jk}}{\sum_{K=1}^{t} {w_{ik}}^2 * \sum_{K=1}^{t} {w_{jk}}^2}$$

Figura 5. **Configuración espacio de indexación**



⊗ ítem; △ consulta; ◌ área de recuperación

Fuente: elaboración propia.

Los círculos con una equis (⊗) representa cada ítem (documento), la distancia entre dos ítems es inversamente proporcional a similitud *s* de ambos vector índice, es decir, mientras mayor sea la similitud menor será la distancia entre ambos ítems.

La recuperación de un ítem dará lugar a la recuperación de aquellos que están a su vecindad, lo que produce alta exhaustividad. Al mismo tiempo, aquellos ítems que se encuentran distantes son difíciles de recuperar, lo que asegura un alto grado de precisión.

Con el objetivo de mejorar los resultados, a través del tiempo se han desarrollado nuevas técnicas para la TIA, como: eliminar acentos, eliminar espacios, reducir las palabras a su origen gramatical (*stemming*); y así optimizar el índice.

En los sistemas de recuperación de información éste modelo es muy utilizado por su completa formalización y poca dificultad de implementación, además de ser escalable.

1.7. Diferencia entre búsqueda y recuperación de información

"La búsqueda es un efectivo método de recuperación cuando la consulta es específica y el usuario sabe explícitamente lo que busca… La búsqueda es una actividad estructurada, lo que da lugar a una baja probabilidad de *serendipia* – descubrir algo inesperado durante una búsqueda. –"[6]

Tipos de búsqueda:

- Cuando el usuario tiene un documento en mente, la búsqueda es llamada: punto conocido de búsqueda. Para la búsqueda generalmente se conoce un atributo del documento como el autor y título.

- Búsqueda negativa: cuando el usuario quiere asegurarse que un documento se encuentra en existencia.

- Difusión selectiva de información: cuando el usuario desea conocer los últimos documentos incorporados a la colección.

- Búsqueda por temas o tópicos: el usuario final tiene una necesidad de información y debería buscar por resultados relevantes.

[6] CHU, Heting. *Information Representation and Retrieval in the Digital Age*. s.l.: ASIST Monograph Series, 2003. p. 82.

Según Baeza-Yates y Ribeiro-Neto la recuperación de información se define como:

"La recuperación de la información es el área de las Ciencias de la Computación encargada de la representación, almacenamiento, organización y su posterior acceso y recuperación de información que responda a las necesidades de un usuario."[7]

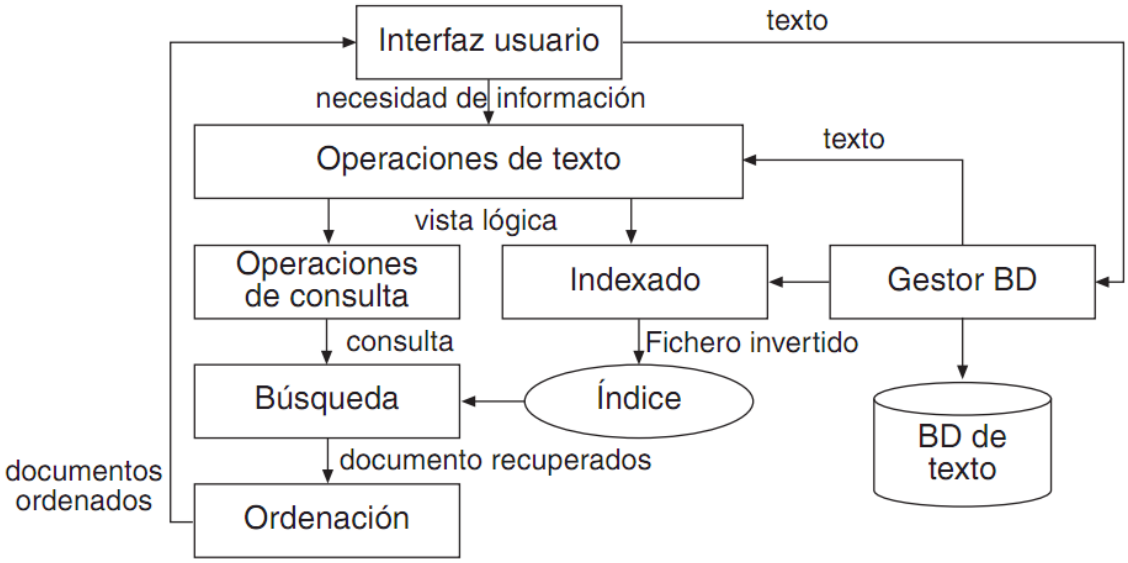
Esto implica un proceso de recuperación de información que en términos generales es indexar los documentos implementando un modelo de recuperación de información; capturar la necesidad de información del usuario y aplicarle operadores necesarios para convertirla en una consulta entendible por el sistema; comparar la consulta y el índice para recuperar los ítems que son relevantes para la consulta; sí es posible ordenar el resultado según criterios de mayor relevancia y mostrar el resultado. Este proceso se muestra en la figura 6: Proceso de recuperación de información.

La búsqueda es una sub tarea del proceso de RI en la cual se compara la consulta del usuario con la representación del ítem y se recuperan los ítems relevantes. Una búsqueda también se refiere a la navegación manual entre páginas (en la web), documentos o ítems para encontrar lo deseado.

La RI es un proceso mucho más amplio y que incluye a la búsqueda, la búsqueda es un subproceso que se encarga de comparar la necesidad de información (consulta) con la representación de los ítems recorriendo el índice de ítems.

[7] BAEZA-YATES, Ricardo y RIBEIRO-NETO, Berthier. *Modern Information Retrieval.* s.l.: Addison-Wesley, 1999. p. 1.

Figura 6. **Proceso de recuperación de información**



Fuente: BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. 1999. *Modern Information Retrieval.* Addison-Wesley, 1999. p. 10.

1.8. Primeros sistemas de recuperación de información

En el ámbito académico se desarrollaron un sinnúmero de SRI, que luego pasaron al ámbito comercial, uno de los primeros y más significativo es SMART (*System for the Mechanical Analysis and Retrieval of Text,* Sistema para el Análisis y Recuperación Automática de Texto), desarrollado por Gerard Saltón, éste es considerado el primer buscador digital, cuyo objetivo estaba enfocado en indexar trabajos científicos.

Estos sistemas, desarrollados hasta los 70's, se enfocaban en una colección de documentos de tamaña definido. Sin embargo, con la aparición de

Internet a principios de los 80’s y más aún, el desarrollo de la Web en 1989, estableció un nuevo paradigma en la recuperación de la información.

1.8.1. MEDLARS

Es el acrónimo inglés de *Medical Literature Analysis and Retrieval System*. Este sistema nace como un proyecto de la Biblioteca Nacional de Medicina de los Estados Unidos con la intención de automatizar la realización del índice de documentos médicos (*Index Medicus)* de todas las revistas y documentos médicos del mundo y que se publicaba mensualmente.

En 1960 se establecen los requerimientos del sistema. En 1961 los directivos solicitan a *General Electric Company* la implementación del mismo, esta dedica un equipo de seis ingenieros dirigidos por Richard F. Gerrard que comienzan a trabajar en el sistema. El sistema se desarrollo en tres fases. La entrega y puesta en producción del sistema fue en 1964.

Las características más destacadas del sistema fueron la flexibilidad, la alta velocidad de rendimiento y la capacidad de hacer frente a un gran número de citas bibliográficas.

El medio utilizado para el almacenamiento fueron cintas magnéticas, lo cual limitaba a que las búsquedas fueran secuenciales sobre todo el índice de bibliografías recuperando aquellas que cumpliesen con un patrón específico.

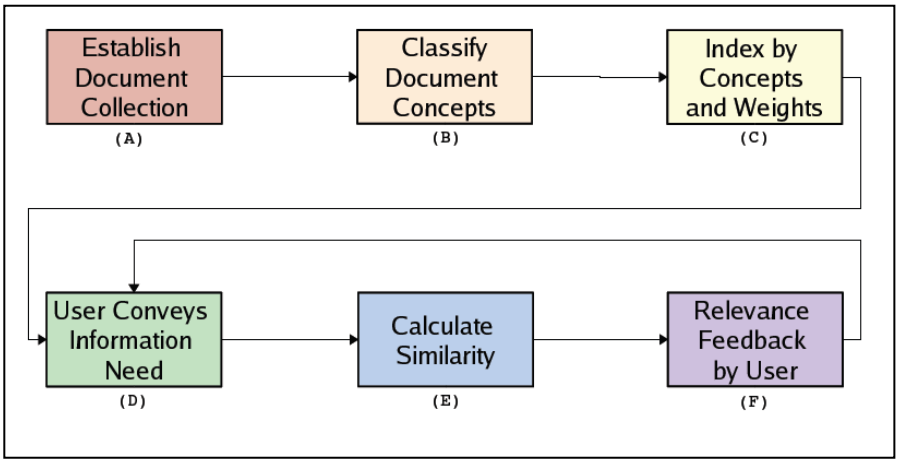
1.8.2. SMART

SMART (*System for the Mechanical Analysis and Retrieval of Text*) implementa un modelo de recuperación de información de espacio vectorial,

sobre una colección de documentos de tamaño definido y controlado. Fue desarrollado por Gerard Salton a mediados de la década de los 60's.

El sistema extrae los términos de los documentos y le establece una ponderación a cada uno (utiliza de métodos ponderación TF, IDF y TD-IDF), ignorando aquellos que son parte de la *stop-word*. La consulta se traduce a un vector (igual que los documentos) y éste se compara, por medio de una función de similitud, con todos los documentos y así definir si se recupera o no el documento (ver: Modelo General de SMART).

Figura 7. **Modelo General de SMART**



Fuente: THUL, Eric. *The SMART Retrieval System, Paper Presentation & Demonstration Session*. 2005. p. 13.

2. RECUPERACIÓN DE INFORMACIÓN EN LA WEB

2.1. Introducción

La idea original de un sistema de información accesible en todo momento y lugar fue ideada por Vannevar Bush en, Bush propone un aparato teórico llamado *memex*:

"Un memex es un aparato en el que una persona almacena todos sus libros, archivos y comunicaciones, y que está mecanizado de modo que puede consultarse con una gran velocidad y flexibilidad. En realidad, constituye un suplemento ampliado e íntimo de su memoria."[8]

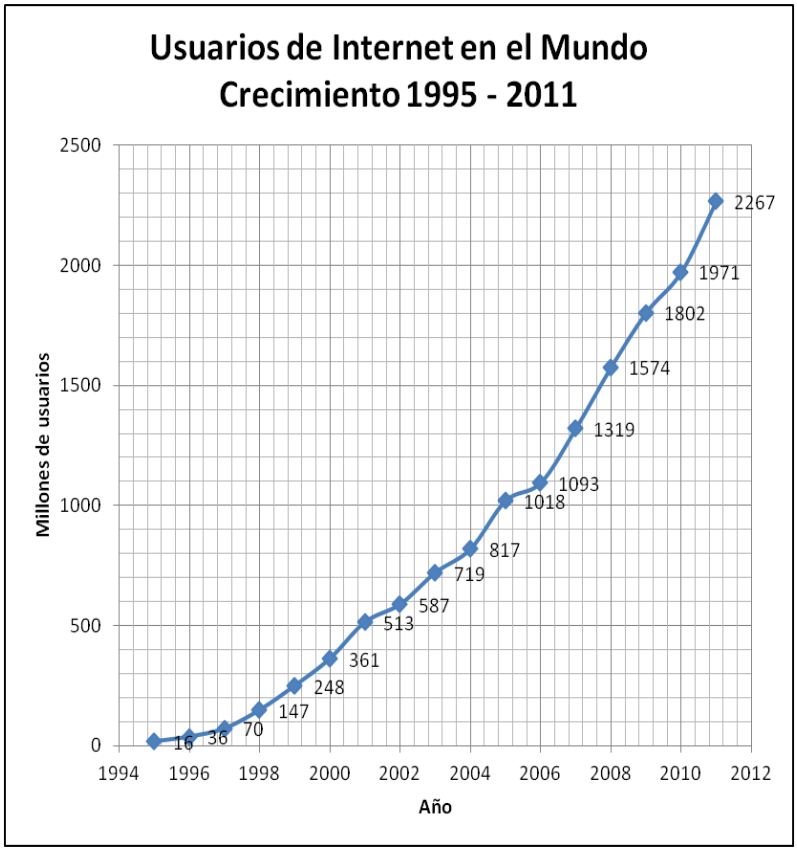
En 1989, Tim Berners-Lee propone al CERN el desarrollo de un sistema distribuido de información entrelazada accesible a través de Internet, llamado *World Wide Web* (o simplemente *Web*), de arquitectura distribuida que permitiese que los documentos (hipertextos) residan en cualquier computadora y sean capaces de establecer vínculos (enlaces) con otros documentos, además de permitir que sistemas heterogéneos puedan acceder a los documentos. Estas características han permitido un crecimiento exponencial de la *Web* durante sus aproximadamente veinte años de existencia.

[8] BUSH, Vannevar. *As we may think*. Boston, Massachusetts: The Atlantic Monthly, 1945.

“Pues bien, la World Wide Web ha realizado el sueño de Bush hasta un extremo que ni siquiera él se había atrevido a imaginar.”[9]

El crecimiento de la cantidad de usuarios ha venido desde 16 millones (1995) hasta 2 mil millones aproximadamente para 2010, como lo muestra la siguiente gráfica.

Figura 8. **Usuarios de internet en el mundo, crecimiento 1995-2011**



Fuente: Internet Growth Statistics.
http://www.internetworldstats.com/emarketing.htm. Consulta: 15 de septiembre de 2012.

[9]http://biblioweb.sindominio.net/pensamiento/vbush-es.html. Consulta: 20 de septiembre de 2012.

La importancia de la recuperación de información “ha tenido mucha más atención y mayor impacto”[10], además de mayores retos, a partir del crecimiento constante de las redes y especialmente de la Web en Internet, cuya cantidad de usuarios en diciembre de 2011 fue de 2 267 millones lo que representa el 32,7% de la población mundial y lo cual permite tener en perspectiva el impacto tan profundo que ha tenido y tendrá la Web. Este crecimiento ha impactado directamente en la cantidad de documentos en una infinidad de formatos presentes.

En 1999 el tamaño de la Web se estimó en 800 millones de páginas, al siguiente año se estimó en 2,1 billones de páginas. “Esto muestra claramente la necesidad de conocer cada vez más sistemas que permitan (o pretendan) hacer posible una rápida, efectiva y eficiente recuperación de información.”[11]

Hay dos herramientas para de buscar información en la web, por medio de motores de búsqueda o directorios. Ambas son herramientas de software, utilizadas por los usuarios para encontrar información deseada. El primero enfocado a una búsqueda general, en todo el espectro web. El segundo enfocado en clasificación de páginas de temas específicos. Aquel utiliza métodos automáticos para recolectar referencias, indexarlas y almacenarlas en su base de datos, el otro, en cambio, es controlada por humanos, quienes revisan y clasifican las direcciones de forma temática.

Los motores de búsqueda utilizan complejos algoritmos para devolver los resultados más adecuados, representando la consulta por medio de palabras

[10] CHU, Heting. *Information Representation and Retrieval in the Digital Age*. s.l.: ASIST Monograph Series, 2003. p. 4.

[11] SANTESTEBAN, Cristian Merlino. *Acceso y recuperación de información en la World Wide Web*. Buenos Aires, Argentina: Universidad Nacional de Mar de Plata, 2001.

clave; en cambio, para obtener resultados en un directorio es necesario navegar a través de los temas y subtemas, una representación implícita de la consulta.

La aparición de Internet y posteriormente de la *Web* impulso el desarrollo de nuevos modelos de recuperación de información basados en la navegación entre páginas. El primer modelo, estructura plana, es una simple lectura, sin establecer relaciones; el segundo, estructura guiada, incorpora la posibilidad de facilitar la exploración organizando los documentos en una estructura tipo directorio con jerarquía de clases y subclases; y el tercero, hipertexto, adquirir información de forma no secuencial sino por medio de nodos y enlaces (grafos).

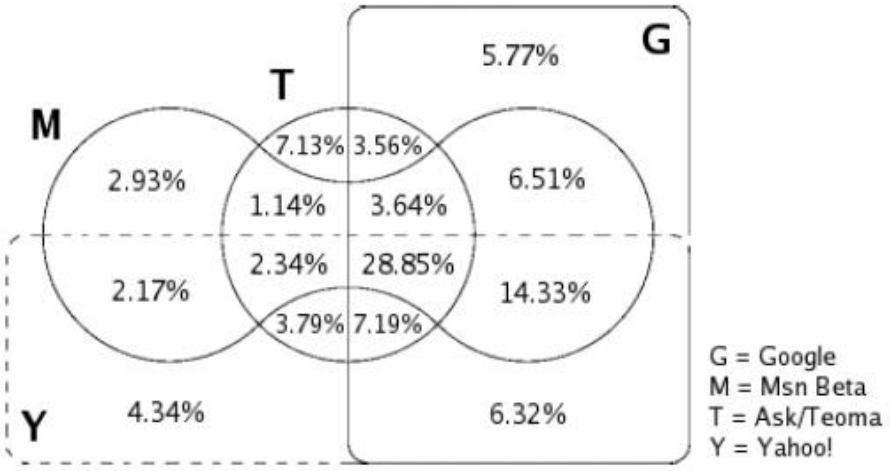
2.2. Recuperación de Información en la *Web*

Uno de los principales desafíos se encuentra en la indexación de la *Web*. Para esta tarea deben considerarse factores técnicos y económicos que impiden la total indexación de la Web.

El costo económico de rastrear e indexar toda la web es muy alto, es necesario costear la adquisición (o creación) de hardware y software para las bases de datos, los servidores para rastrear la web, entre otros.

Determinar el tamaño exacto de la Web es muy problemático debido a su naturaleza dinámica y además debe considerarse tanto la Web Invisible. Un estudio de 2005 calculó que el tamaño de la Web indexable es de 11,5 mil millones de páginas, los motores de búsqueda Google, Yahoo, Ask/Teoma y MSN se distribuyen ese total con algunos porcentajes compartidos, como lo muestra la siguiente figura.

Figura 9. **Distribución de páginas en los motores de búsqueda**



Fuente: GULLI, A.; SIGNORINI, A. *The Indexable Web is More than 11.5 billion pages*. Chiba, Japan: ACM, 2005. p. 2.

Otro factor técnico a considerar es el contenido, el contenido de la *Web* está escrito en distintos idiomas y en distintos formatos, no es únicamente HTML puede ser PDF, *Word*, etc., y cada formato distinto agrega dificultad en la indexación.

La siguiente tabla, compara cuatro elementos entre la recuperación tradicional y en la recuperación en la w*eb*: documentos, características de la *web*, comportamiento del usuario y los sistemas de recuperación de información.

Tabla II. **Diferencias entre recuperación de información tradicional y recuperación de información en la web**

Diferenciador	Web IR	Tradicional IR
Documentos		
Lenguaje	Documentos en muchos diferentes lenguajes. Usualmente los motores de búsqueda usan indexado de texto completo.	Base de datos usualmente cubre un solo lenguaje o indexa documentos en distintos lenguajes con un mismo vocabulario.
Tipo de archivo	Varios tipos de archivos, algunos difíciles de indexar por falta de información textual.	Usualmente indexa documentos de un mismo formato (eje: PDF) o sólo información bibliográfica es proveído.
Tamaño del documentos	Amplia gama de muy corto o muy largo. Documentos más largos suelen dividirse en partes.	La longitud del documento varía, pero no a un grado tan alto como con los documentos web. Cada índice de texto se representa con la unidad documental.
Estructura del documento.	Documentos HTML son semi-estructurados.	Documentos estructurados permiten búsquedas campo complejo.
Spam	Los motores de búsqueda deciden cuales documentos son aceptables para indexar.	Los tipos adecuados de documentos se definen en el proceso de diseño de la base de datos.
Hyperlinks	Documentos están conectados fuertemente. Estructura de hipervínculo se puede utilizar para determinar la calidad.	Los documentos no suelen estar conectados. A veces, los datos de citación se utilizan para determinar la calidad.
Características *Web*		
Cantidad de datos, tamaño de la base de datos.	Difícil de determinar. El actual tamaño de la *Web* se desconoce. Indexación completa de toda la *Web* es imposible.	La cantidad exacta de datos pude determinarse usando criterios formales.

Continuación de la tabla II.

Cobertura	Desconocido, sólo es posible estimarla.	Completa cobertura de acuerdo a las fuentes definidas.
Duplicidad	Muchos documentos existen en muchas copias o versiones.	Los duplicados son señalados. Por lo general hay una versión final para cada documento.
Comportamiento del usuario		
Intereses	Intereses muy heterogéneo.	Grupo de usuarios definidos claramente con conocida información sobre la conducta de búsqueda.
Tipos de consultas.	Usuarios tienen poco conocimiento de cómo buscar. Las consultas son cortas (2 – 3 palabras)	Usuarios conocen el lenguaje de recuperación; más largo, las consultas son exactas.
Sistema de Recuperación de Información		
Interface de usuario	Interface fácil de usar.	Normalmente interfaces complejas; práctica necesaria para realizar búsquedas.
Ranking	Debido a la gran cantidad de *hits* ranking por relevancia es la norma.	Ranking por relevancia a menudo no es necesario porque los usuarios conocen como restringir la cantidad de *hits*.
Funciones de búsqueda	Limitadas posibilidades.	Lenguaje de búsqueda complejos permiten dirigir las búsquedas.

Fuente: DIRK, Lewandoski. 2005. *Web searching, search engines and Information Retrieval*. Düsseldorf, Germany: Department of Information Science, Heinrich-Heine-University Düsseldorf, 2005. p. 12.

Los motores de búsqueda web deben salvar en mayor o menor efectividad todos esos aspectos. Sérgio Nunes considera que hay seis componentes principales, que ayudan a cubrir estos aspectos para los motores de búsqueda:

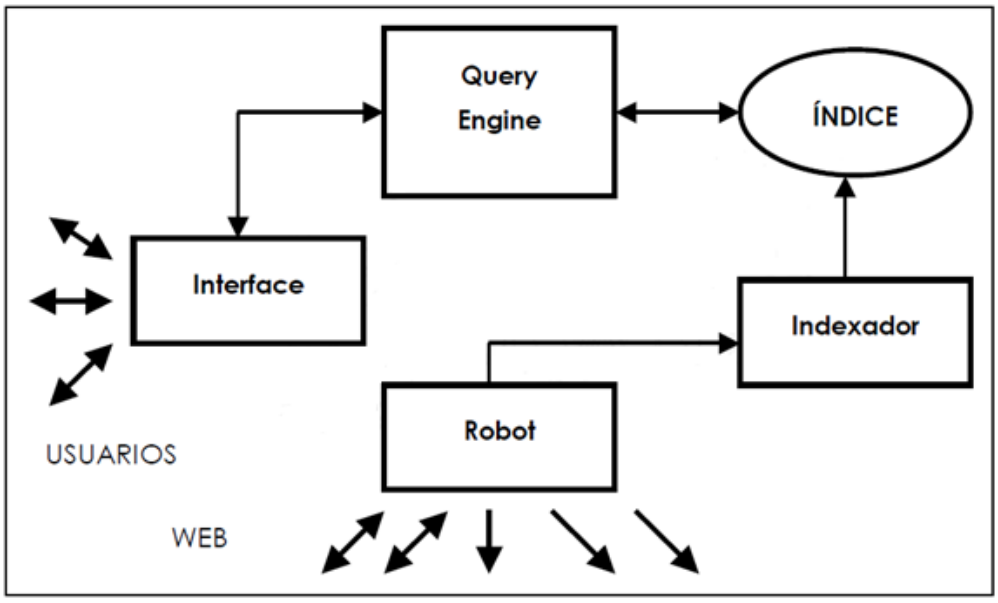
- "Crawler (*bot*-rastreador): incluye los rastreadores que recuperan las páginas web. Normalmente múltiples y distribuido sistema de rastreadores opera simultáneamente.
- Repository (depósito): gran cantidad de documentos web son almacenados en bases de datos especializadas, permitiendo acceso y escritura de alta concurrencia. Documentos HTML completos son almacenados aquí.
- Indexes (Índices): un motor de indexación construye varios índices optimizados para lecturas muy rápidas. Varios tipos de índices deben existir, incluyendo índices invertidos (Anexo 1: Archivo Invertido), lista de negras, léxicos. El contenido y los vínculos en los documentos son analizados.
- Ranking: para cada consulta, se califican los resultados combinando múltiples criterios. Un valor es asignado a cada documento.
- Presentación: ordena y presenta los documentos clasificados. Fragmentos breves del contenido de cada documento se seleccionan y se incluyen en esta etapa final."[12]

2.2.1. Arquitectura de los Sistemas de Recuperación de Información en la Web

Existe una arquitectura genérica, que la mayoría de motores de búsqueda emplea, sin embargo, cada motor en específico definirá su propia arquitectura. La arquitectura clásica está conformada por: motor de consultas (*Query Engine)*, robots, indexador e interface. Todos los componentes de la arquitectura se gestionan localmente.

[12] NUNES, Sérgio. *State of art in web information retrieval*. Portugal: s.n., 2006.

Figura 10. **Arquitectura simple de un motor de búsqueda web**



Fuente: BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval*. p. 513.

Aunque la arquitectura presenta varios problemas, ésta es la base para el desarrollo de motores de búsqueda. Con el transcurrir del tiempo y la necesidad, fueron diseñándose nuevas arquitecturas, la cuales trataban de solucionar los problemas de sus antecesoras.

Otra arquitectura, la arquitectura de *Harvest* es distribuida, constituida por dos elementos principales: el *recopilador* y los *brokers*. El primero es el encargado de recolecta y extraer de las páginas toda la información necesaria para crear los índices del motor de búsqueda, el segundo, el *bróker*, es el encargado de brindar el mecanismo de indexación de las páginas y proporcionar la interfaz de consulta.

La interfaz de consulta el usuario expresa sus necesidades de información y la interface de respuesta, que es la forma en que el sistema muestra los resultados.

El índice, es el componente fundamental de un motor de búsqueda, en el se emplean técnicas para almacenar referencia a las direcciones web, ésta labor es hecha por el indexador. Además aquí también se define el algoritmo de relevancia.

Como algoritmo de relevancia es posible utilizar el clásico esquema *tf-idf* añadiendo una extensión para el manejo de los enlaces. Sin embargo, existe un grupo de algoritmos que toman en cuenta el análisis de vínculos: *HITS* y *PageRank*.

2.2.2. *Crawler* (rastreador)

Robots informáticos, son simplemente programas que automatizan tareas repetitivas a velocidades imposibles de reproducir por los seres humanos. El término *bot,* en Internet, se utiliza generalmente para describir cualquier cosa que interactúa con el usuario o el que recoge los datos.

Los motores de búsqueda utilizan *crawlers* que buscan en la web para obtener información. Son programas de software que solicitan páginas, así como lo hacen los navegadores web. Leen el contenido de las páginas para su indexación, además, los enlaces son almacenados para descubrir nuevos documentos.

En el ámbito de la Web son conocidos como: *Web crawler* o simplemente *crawler.* Además, pueden ser de carácter general o específico.

2.2.3. Análisis del contenido (*content analysis)*

"Los primeros sistemas de búsqueda en la web se basaron en el análisis de contenido y se utilizaba una simple colección de palabras para clasificar documentos. Métodos de ponderación de términos como medidas TF.IDF fueron utilizados.

Los sistemas de segunda generación exploraron la estructura HTML de los documentos. Por ejemplo, un término incluido en un encabezado o título de las etiquetas correspondientes en HTML tiene un valor más alto que el mismo término en una etiqueta de párrafo."[13]

2.2.4. Análisis de vínculos (*link analysis)*

Es un área de la recuperación de información en la *Web* que se encarga de estudiar las relaciones generadas por los vínculos (*links)* en los hipertextos. Para Monika Henzinger "el análisis de las estructura de los hipervínculos de la web ha significado importantes mejoras en la recuperación de información en la web."[14]

El análisis de vínculos hace los siguientes supuestos:

- Un enlace de la página *A* hacia la página *B* es una recomendación de *B* por el autor de la página *A.*
- Si las páginas *A* y *B* están conectadas por un enlace la probabilidad de que sean del mismo tema es mayor a si no estuviesen conectadas.

[13] NUNES, Sérgio. *State of Art in Web Information Retrieval*. Portugal: s.n., 2006.

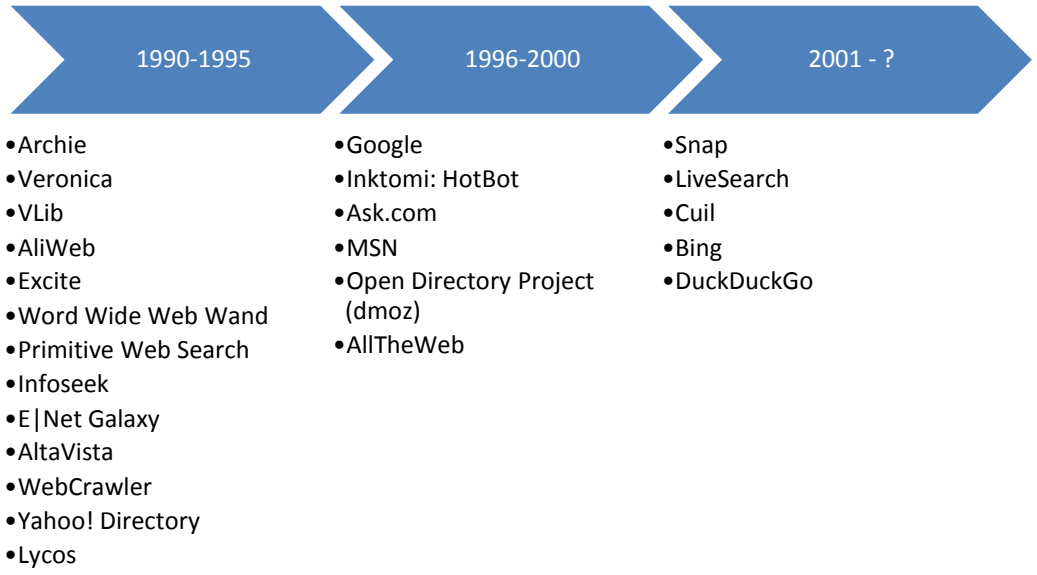
[14] HENZINGER, Monika. *Link Analysis in Web Information Retrieval*. Mountain View, California: s.n., 2000.

Para analizarse la web desde esta perspectiva es necesario ver a la web como un grafo dirigido. Los algoritmos más importantes son *HITS* y *PageRank.*

2.3. Motores de búsqueda

Se estima que cerca del 70% de los usuarios de la web usan un motor de búsqueda como punto de partida para encontrar lo que desean. El mejor motor de búsqueda recibe millones de solicitudes por día y muestra miles de millones de resultados en respuesta a las solicitudes.Los motores de búsqueda son los sistemas más utilizados en la búsqueda de información, porque, de alguna manera y tomando en cuenta las limitaciones, van creciendo al ritmo de crecimiento de la web.

Figura 11. **Cronología de los buscadores web**



Fuente: elaboración propia, con base en información de *WordStream.* http://www.wordstream.com/articles/internet-search-engines-history. Consulta: 10 septiembre de 2012.

A partir de 1990 se empezaron a construir motores de búsqueda, el primero de ellos *Archie,* en 1993 *Worl Wide Web Worm* implementó el primer *crawler* que indexaba los títulos y las URL. En 1998 se lanza *Google* y su algoritmo *PageRank*. Luego surgieron nuevos buscadores como *Bing* y *DuckDuckGo*.

2.3.1. Archie

Probablemente, el primer motor de búsqueda de internet fue *Archie*, así lo afirma Tim Macinta, quien tituló una sección del primer capítulo de su libro como: Inclínate y da gracias a Archie, lo cual denota la importancia de *Archie* como hito en las historia de los motores de búsqueda.

Archie fue desarrollado por Alan Emtage, Peter Deutsch y Bill Heelan cerca de 1990, entonces los tres estudiantes de la Universidad McGill de Montreal. El sistema estaba enfocado en indexar archivos que se encontraban en servidores FTP públicos de internet.

Lo que *Archie* hacía, era buscar periódicamente en bibliotecas de archivos, averiguar los archivos disponibles y agregar los nombres a una base de datos (*Archie* no indexaba el contenido de los documentos, sino, sólo el nombre del archivo), así, cualquiera podía buscaba el nombre de un archivo en *Archie* y este devolvía la ubicación en la red, en donde se encontraba disponible. A continuación, el usuario debía dirigirse a ese servidor (con un cliente FTP) solicitar acceso y transferir el archivo a la máquina local.

2.3.2. ALIWEB

ALIWEB es considerado el primer motor de búsqueda para la web (octubre de 1993), su creador es Martijn Koster, su diseño y funcionamiento se basa en Archie, pero a diferencia de éste, ALIWEB se enfoca en la Web.

La recopilación de datos se hacía a través de meta datos que los propios administradores de las páginas definían en el índice. Más adelante, se implementó *bots* para esta tarea.

2.3.3. World Wide Web Worm

En diciembre de 1993 surge éste motor de búsqueda, desarrollado por Oliver McBryan, el cual tuvo un índice de 110 000 páginas web y documentos web accesibles, ya para 1997 su índice, recopilado por medio de *web crowlers*, indexaba cerca de 2 a 100 millones de webs.

Al mismo tiempo también surgieron: *JumpStation*, el cual reunía información del título y cabecera de las páginas web y recuperaba estas utilizando una búsqueda lineal simple.

Para ambos, el posicionamiento era definido por el orden de indexación, lo cual generó un gran problema. Hasta aquí, ningún motor de búsqueda había hecho un análisis adecuado de los enlaces (a pesar que World Wide Web Worm si hacía uso del texto ancla). Si no se sabía el nombre exacto de lo buscado, era extremadamente difícil encontrarlo.

2.3.4. Yahoo!

Posee un portal de Internet, un directorio web y una serie de servicios, incluido el popular correo electrónico Yahoo!. Fue fundada en enero de 1994 por dos estudiantes de postgrado de la Universidad de Stanford, Jerry Yang y David Filo.

Yahoo! es el directorio más popular de internet, la manera de recolectar los documentos y categorizarlos es de manual, hay personas dedicadas a esa tarea. Después integró entre sus servicios un motor de búsqueda *Yahoo! search*, el cual busca entre los documentos del directorio.

2.3.5. Google

El más importante buscador en la web, al momento de escribir este documento es: *Google*. Éste fue desarrollado por Lawrence Page y Sergey Brin, ambos, en ese momento, estudiantes de la universidad de Stanford.

En septiembre de 1998 presenta Google, proyecto cuyo objetivo era construir un sistema que mejorara la calidad de búsqueda y escalabilidad (debido al crecimiento de la web), para ello proponen una arquitectura que contempla el crecimiento de la web, el cambió de la tecnología y eficientes estructuras de datos.

2.3.5.1. Algoritmo de recuperación (modelo RI)

El modelo de recuperación utilizado es el fichero invertido, esto es, escanear todos los documentos y páginas web, crear una lista de términos y por cada termino una lista de los documentos relacionados.

Por ejemplo: la palabra guerra, ocurre en los documento 3, 8, 22, 56, 68 y 92, mientras que la palabra civil, ocurre en los documentos 2, 8, 15, 22, 68 y 77.

Figura 12. **Ejemplo de fichero invertido**

Término	**Referencia a documentos**						
civil		3	8		22	56	68
guerra	2		8	15	22		68
ambas palabras			8		22		68

Fuente: elaboración propia.

Ahora, si se busca guerra civil, se hará una lista en de documentos en dónde estén presentes ambas palabras, el orden de esos documentos será definido por un algoritmo de posicionamiento.

2.3.5.2. Algoritmo de posicionamiento

Para el ranking de calidad de la web, Google hace uso de la estructura de enlaces, este ranking se llama *PageRank* y utiliza también los enlaces para mejorar el resultado de la búsqueda. Utiliza naturaleza de la Web (enlaces) para mejorar la calidad de las búsquedas.

El hecho de que el autor de una página web introduzca un enlace, representa, implícitamente, un respaldo para la página enlazada. La exploración de enlaces insertados en una página web y, la exploración de los enlaces que apuntan hacia esa misma página.

Google recupera documentos basado en palabras claves, posicionando los resultados por medio de *PageRank.*

2.3.5.3. Descripción de PageRank

Se considera que una página *A* es referenciada por las páginas *T1*…*Tn*, con un factor de amortiguamiento *d* que se establece entre 0 y 1. *C(A)* se define como el número de enlaces que salen de *A*.

Ecuación 8: *PageRank*

$$\mathrm{PR(A)} = (1-d) + d\left(\frac{PR(T1)}{C(T1)} + \ \ldots + \frac{PR(Tn)}{C(Tn)}\right)$$

El *PageRank* es una distribución de probabilidades, por lo cual, la suma de todos los *PageRank* es uno. Además utiliza el texto ancla como una descripción más precisa de los enlaces. También guarda una copia local de la información de todos los resultados obtenidos.

2.3.5.4. Arquitectura

El sistema operativo que soporta a *Google* es *Solaris* o *Linux* y en su mayor parte desarrollado en C y C++.

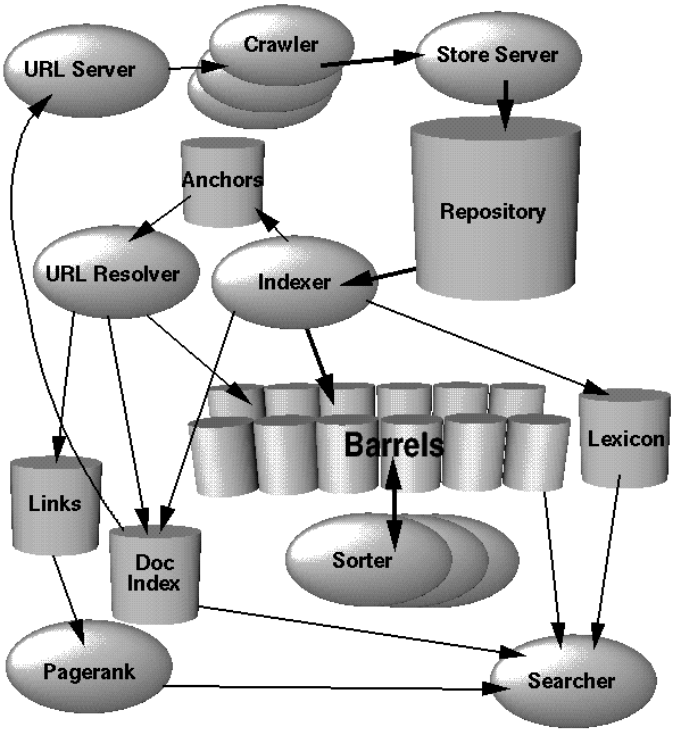
Un sistema distribuido (*Crowler)* rastrea las páginas web enviadas por el servidor de de URL, estas páginas son comprimidas y guardadas por el servidor de almacenamiento (*Store Server)* en el repositorio (*Repository).*

La función de indexación es llevada a cabo por el *indexer* y *sorter*. El indexador toma los archivos del repositorio, los descomprime y analiza, genera un conjunto de ocurrencia de palabras llamadas *hits*. El indexador distribuye los

hits en cubetas (*barrels)* y el clasificador *(sorter)* es el encargado de establecer un orden.

El indexador también realiza un análisis de los enlaces de todas las páginas y las almacena en un archivo ancla (*Anchors*). El *URL Resolver* lee los archivos ancla y asigna un *docID* a cada página, además almacena todos los enlaces en una base de datos (*links*) la cual será utilizada para calcular el *PageRank*.

Figura 13. **Arquitectura de alto nivel del motor de búsqueda *Google***



Fuente: BRIN, Sergey y PAGE, Lawrence*. "The Anatomy of a Large-Scale Hypertextual Web Search Engine"*. Computer Science Department, Stanford University, Stanford, 1997. p. 8.

3. ESTADO DEL ARTE DE LAS ONTOLOGÍAS

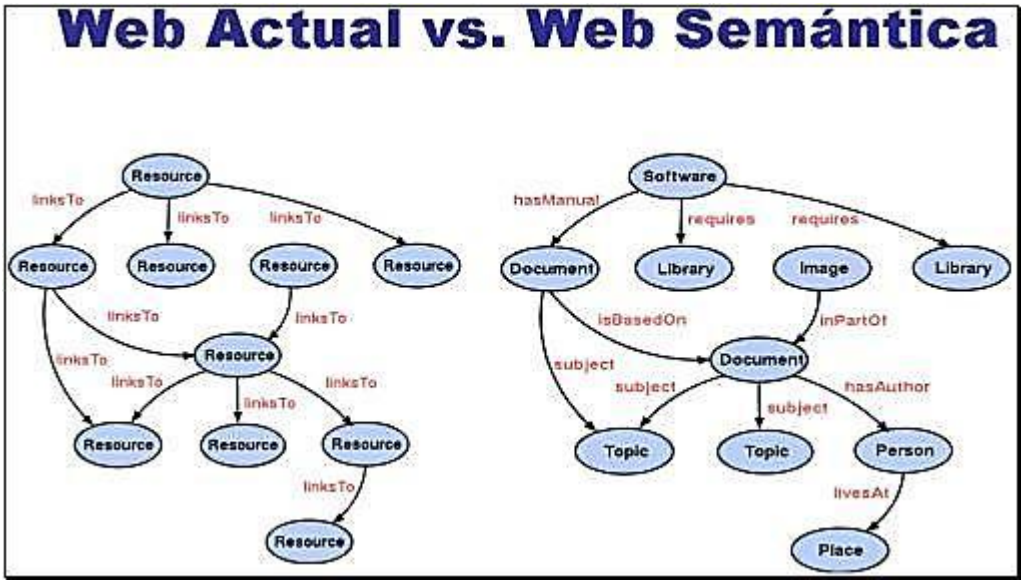
3.1. Introducción

Tim Berners-Lee, ahora director general de W3C, junto a James Hendler y Ora Lassila, han propuesto la *Web Semántica*, una web capaz de expresar significado no sólo a humanos, sino también, a programas de computadoras.

El problema con la indexación automática, descrita en el capítulo uno generalmente y específicamente para la web en el capítulo dos, es que los programas de computadoras son capaces de analizar el contenido sintácticamente pero no de analizarlo semánticamente. Esto es posible comprobarlo al buscar, por ejemplo, la palabra hipertexto en un buscador web, el buscador no distingue entre contextos y significados en los cuales aparece el término.

En la Web actual las relaciones entre documentos se dan por medio de enlaces (hipervínculos) sin ningún significado específico más allá de la relación de recomendación. En cambio, la Web Semántica no sólo está compuesta por documentos, sino por objetos, cada objeto representado de manera única, los objetos pueden ser: documentos, personas, conceptos, etc. Y las relaciones son de jerarquía, pertenencia, etc. La siguiente figura compara las relaciones de la Web actual con la Web Semántica.

Figura 14. **Web Actual vs. Web Semántica**



Fuente: La web semántica y la accesibilidad. http://autoevaluacion.unad.edu.co/redvida/unidad6/graf/webhoy-websem.jpg. Consulta: 25 de septiembre de 2012.

La Web Semántica no necesita de una nueva infraestructura dedicada, tal como lo describe Berners-Lee: “La Web Semántica no es una Web independiente o separada, sino una extensión de la actual, en la que la información tendrá un significado bien definido, de manera que pueda ser interpretada tanto por humanos como por programas de computadora.”[15].

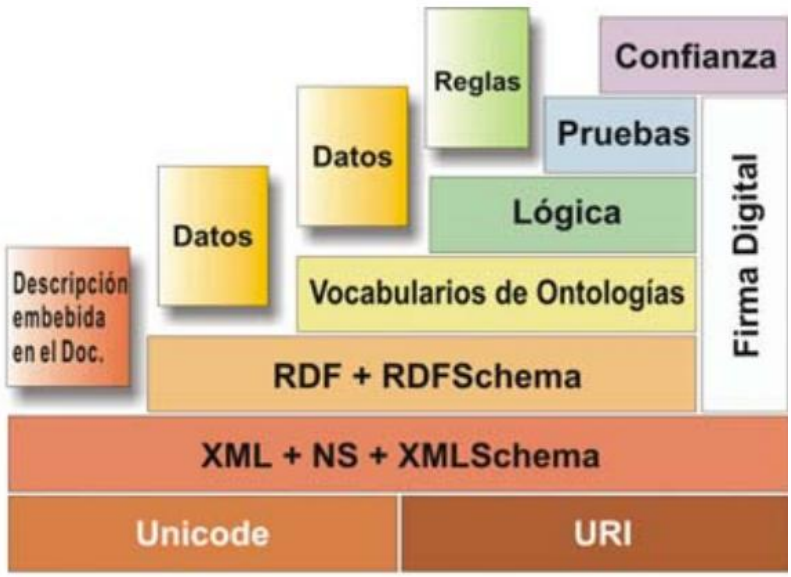
Eso significa que las características básicas de la Web serán aplicadas a la Web Semántica. La arquitectura, por ejemplo, debe permanecer

[15] BERNERS-LEE, Tim, HENDLER, James; LASSILA, Ora. *The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. s.l.: W3C, 2001.

descentralizada, tanto como sea posible, lo que permitirá el crecimiento que ha experimentado hasta ahora.

Los componentes básicos de la Web Semántica son la expresión de significado a través de una representación formal del conocimiento, facilitando relaciones semánticas entre conceptos por medio de ontologías, las cuales pueden ser consultadas por agentes (programas) que infieran razonamiento automáticamente a partir de reglas de inferencia.

Figura 15. **Arquitectura de la Web Semántica**

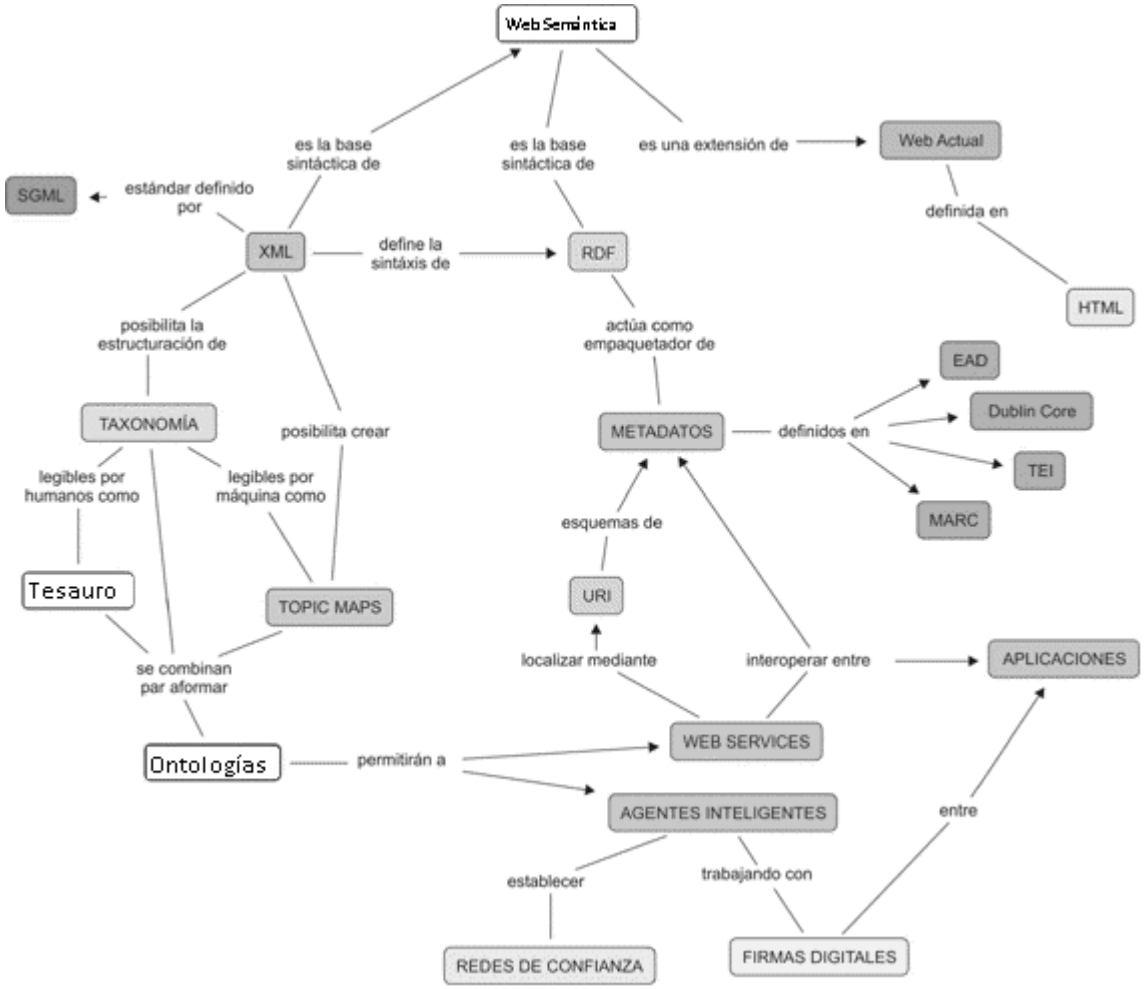


Fuente: BERNERS-LEE, Tim. Semantic Web-XML 2000 Architecture. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide11-0.html. Consulta: 10 de julio de 2012.

En conferencia del 2002 Berners-Lee presentó la arquitectura de la Web Semántica, tal cual se muestra en la figura anterior. Con ésta arquitectura se pretende estandarizar: el alfabeto (Unicode), las referencias (URI), el lenguaje

(XML + NS + XMLSchema), el formato (RDF + RDFSchema), las anotaciones sobre significado (lenguaje de ontologías), reglas y sistemas de deducción (Lógica).

Figura 16. **Mapa conceptual de la web semántica**



Fuente: RODRIGUEZ, Keilyn; LEÓN, Rodrigo. Web semántica: un nuevo enfoque para la organización y recuperación de información en el web.http://bvs.sld.cu/revistas/aci/vol13_6_05/aci030605.htmlConsulta: 27 de septiembre de 2012.

Un componente importante en la arquitectura son las ontologías, un concepto tomado de la filosofía, utilizado inicialmente en el ámbito de la Inteligencia Artificial. Hoy es muy utilizado en el desarrollo de la Web Semántica.

3.2. Ontologías

"Las ontologías facilitan el compartir y reusar conocimiento, estas se definen como: una representación formal y explicita de una conceptualización cognitiva de los componentes de conocimiento relevantes en un dominio y sus relaciones."[16]

Las ontologías se han extendido en su uso, más allá de la Web Semántica. Actualmente, son utilizadas en todos los ámbitos: ingeniería del conocimiento, procesamiento del lenguaje natural, administración del conocimiento, economía, telemedicina, medicina, ingeniería de software, etc. con el fin de crear un lenguaje común para investigadores que necesitan compartir información de un dominio.

Existen tres razones principales al por las cuales desarrollas una ontología:

La primera, es la capacidad de compartir conocimiento no sólo con otras personas, sino también con agentes de software, así, si sitios web comparten una ontología, esta puede ser analizada e interpretada, por ello, se podrá buscar e integrar datos con base en significado. Con ese propósito

[16] GRAMAJO, Javier. *State of the Art: Ontologies*. s.l.: Engineering Faculty, Computer Since and Systems Department, Universidad San Carlos de Guatemala, 2005. p. 2.

organizaciones como W3C (*World Wide Web Consortium)* y DARPA (*Defense Advanced Research Projects Agency)* han definido varios estándares.

La segunda es que a partir de la publicación de las ontologías (en internet comúnmente), se permite la reutilización del conocimiento de un dominio, es decir, si alguien desarrolla una ontología de algún dominio, ésta puede utilizarse en otros dominios.

Y por último, analizar el comportamiento de un dominio, es posible analizar el dominio una vez se conoce claramente su especificación.

Según W3C un dominio es el área específica de interés o un área de conocimiento. Así, por ejemplo, un área de interés podría ser: educación, producción de vinos; un área de conocimiento: física, medicina, economía, etc.

3.2.1. Componentes de la ontología

Una ontología se enfoca en dar una visión de un dominio en específico los conceptos que lo conforman y sus relaciones, por ejemplo, si la ontologías se trata de educación los conceptos podrían ser: recurso educativo, profesor, estudiante; y relaciones como: un profesor pública un recurso.

A los conceptos se les llama clases, las propiedades de cada concepto slots (rol o propiedades) y restricciones sobre los slots (facetas). Las clases describen conceptos de un dominio específico y puede contener subclases (concepto aún más específico).

Además, las relaciones son componentes imprescindibles en las ontologías, estas asociaciones entre conceptos permiten la comprensión su contexto, comportamiento y estado.

Por último, las instancias, éstas representan elementos individuales en las ontologías, es decir, objetos determinados de un concepto.

Según Gruber, las ontologías se componen de:

- Conceptos: son las ideas básicas que se intentan formalizar.
- Relaciones: representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.
- Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como: asignar-fecha, categorizar-clase, etc.
- Instancias: se utilizan para representar objetos determinados de un concepto.
- Reglas de restricción o axiomas: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: Si A y B son de la clase C, entonces A no es subclase de B; Para todo A que cumpla la condición B1, A es C, etc. Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

3.2.2. Representación formal de las ontologías

Como se ha dicho: las ontologías son una representación formal y explicita de conocimiento. La característica de formalidad se obtiene al definirla en un lenguaje comprensible para los usuarios de la ontología. Según el nivel de especificación, confiabilidad y restricciones del lenguaje, así se define el nivel de formalidad.

Los niveles de formalidad van desde el menor nivel, expresado con lenguaje natural, lo cual significa ambigüedad e incertidumbre al momento de interpretar el conocimiento, debido a la naturaleza del lenguaje; hasta lenguajes formales estructurados, definidos por medio de premisas y teoremas lógico matemáticos capaz de comprobarse.

Con el objetivo de clasificar las ontologías según su formalidad, se han definido cuatro niveles, los cuales se presentan a continuación:

- Altamente informales: escritas en lenguaje natural, lo cual está sujeto a imprecisiones y ambigüedades del lenguaje.

- Semi-informal: restringidas y estructuradas utilizando lenguaje natural.

- Semi-formal: Utilizando un lenguaje artificial definido formalmente.

- Rigurosamente formal: semi-formal, incluyendo además teoremas y pruebas, esto permite una definición de términos mediante lenguaje lógico-matemático sin ambigüedades.

El nivel de formalidad que se tratará en este documento es el *Semi-formal*, en el cual entra un estándar definido por W3C llamado: RDF Schema y posteriormente uno enfocado a la Web: OWL, los cuales se tratarán posteriormente.

3.2.3. Tipos de ontologías

A parte de la clasificación según el nivel de formalidad, las ontologías se pueden tipificar según el nivel de generalidad, es decir, el espacio de representación que describen.

Según Fensel, existen seis tipos de ontologías:

- Ontología de dominio: describen el vocabulario de un dominio de conocimiento concreto.
- Ontología de metadatos: son herramientas que representan el conocimiento de un área a través de metadatos, con ello contribuyen a facilitar las búsquedas al usuario final.
- Ontologías genéricas o de sentido común: captura conocimiento general del mundo como el tiempo, espacio, estado o evento.
- Ontologías representativas: su objetivo principal no es proveer de entidades representativas y no se refieren a ningún dominio en particular.
- Ontologías de métodos y tareas: ambas proveen términos específicos, procedimientos y funciones para, la primera, solución de problemas y, la segunda, para tareas particulares.
- Ontologías de aplicación: especialización de un dominio o tarea específica. Ontología para una aplicación concreta.

3.2.4. ¿Cómo desarrollar una ontología?

Es necesario mencionar que no existe una forma o metodología correcta para desarrollar ontologías, por lo cual, se presentan los puntos a tomar en consideración al desarrollar una ontología:

- Determinar el dominio y alcance de la ontología. Definir el dominio que cubrirá y en qué se usará, qué tipo de preguntas responderá y quién usará y mantendrá la ontología. El proceso de desarrollo de ontologías es iterativo e incremental, por lo cual, la definición del dominio y alcance se realizará, muy seguramente, más de una vez.

- Considerar la reutilización de ontologías existentes. Reutilizar ontologías, por ejemplo, si la aplicación debe interactuar con otras aplicaciones que ya han definido una ontología. Hay bibliotecas de ontologías en la Web.

- Enumerar términos importantes para la ontología. Escribir una lista de términos relevantes, sin preocuparse de las relaciones, cualquier propiedad que los conceptos puedan tener, o si los conceptos son clases o slots.

- Definir las clases y la jerarquía de clases. Para desarrollar una jerarquía de clases existen tres enfoques: *top-down*, *bottom-up*, y la combinación de ambos. A partir de la lista de términos, se deben seleccionar aquellos que realmente son importantes y se consideren como clases y sus subclases y descartar aquellos poco relevantes.

- Definición de propiedades de las clases (slots). En este paso, es necesario describir la estructura interna de los conceptos.

- Definirlas facetas de los slots. Los slots pueden tener diferentes facetas que describen el tipo de valor, valores admitidos, el número de valores (cardinalidad), y otras características de los valores que los slots pueden tomar.

    - Cardinalidad, ésta se refiera a cuantos valores un *slot* puede tener. La cardinalidad puede ser simple (admitiendo a lo sumo un valor) y cardinalidad múltiple (admitiendo cualquier cantidad de valores).

    - Tipos de valor de los *slots*. Describe el tipo de valor que debe llenar los *slots*: *String, Number, Booelan, Enumerated, Instance*. De todos estos los más importantes a mencionar son *Enumerated e Instance.* El primero especifica una lista de valores admitidos para el *slot*, el segundo, admite la definición de relaciones entre individuos. Los *slots* con tipo de valor *Instance* deben también definir una lisa de clases admitidas de las cuales las instancias pueden provenir.

    - Dominio y rango de un *slot*. Las clases admitidas para los slots de tipo *Instance* son a menudo llamadas rango. Las clases a las cuales un *slot* está adosado o las clases cuyas propiedades son descritas por un slot son llamadas dominio del *slot*.

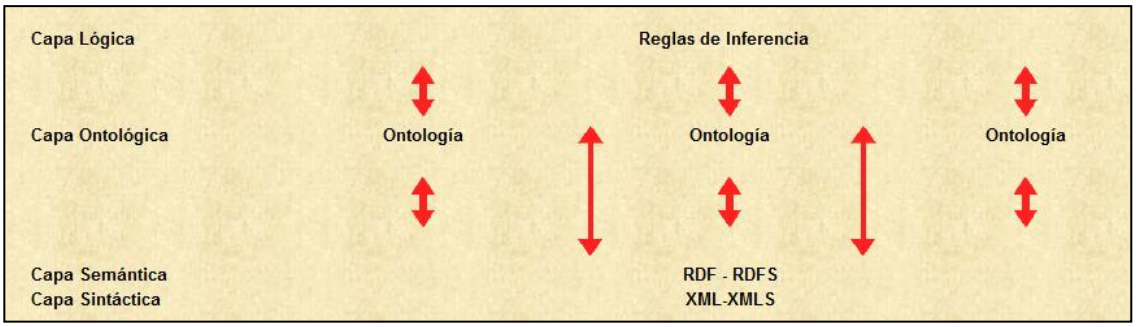
- *Crear instancias.* El último paso consiste en crear instancias individuales de clases en la jerarquía. La definición de una instancia individual de una clase requiere (1) elegir una clase, (2) crear una instancia individual de la clase y (3) rellenar los valores del *slot.*

3.2.5. Herramientas para estandarizar la representación del conocimiento

La arquitectura de Web Semántica propuesta por Tim Berners-Lee es una jerarquía de tecnologías necesarias que conforman un esquema de capas para la representación de conocimiento.

El objetivo de la Web Semántica es dotar de significado a la web, la propuesta más fuerte, actualmente, es por medio de las ontologías. Establecer relaciones semánticas entre conceptos y recursos y no, únicamente, páginas enlazadas con otras páginas. Así lo muestra la siguiente figura.

Figura 17. **Capas de la web semántica**



Fuente: LAMARCA, María. Hipertexto: El nuevo concepto de documento en la cultura de la imagen.http://www.hipertexto.info/documentos/ontologias.htm. Consulta: 28 septiembre de 2012.

3.2.5.1. URI, Unicode

Uniform Resource Identifier, Identificar uniforme de recursos, es el mecanismo para identificar de forma inequívoca cualquier recurso en la red. El subconjunto más conocido es URL – *Uniform Resource Locators*-. Mientras

Unicode es un estándar para el intercambio de símbolos, este estándar es capaz de representar la mayoría de símbolos de la mayoría de idiomas.

3.2.5.2. XML, XML Schema

XML es la sigla del inglés*e Xtensible Markup Language (*lenguaje de marcado ampliable o extensible) desarrollado por el W3C.

Es la base sintáctica sobre la cual se sustentan las siguientes capas del esquema de Web Semántica, permite definir otros lenguajes (metalenguaje) de etiquetado validándolo por medio de DTD o *XML Schemas*. Cualquier aplicación puede compartir documentos, transacciones y carga de trabajo a otras aplicaciones, esto permitirá a XML convertirse en el lenguaje común de *e-business* y gestión del conocimiento.

3.2.5.3. RDF y RDF Schema

El significado a todo lo establecido en un lenguaje XML es dado por *Resource Description Framework* (RDF) y RDF Schema. RDF es un lenguaje de etiquetado creado mediante sintaxis XML. Es considerado la primera capa que es parte real de la Web Semántica.

RDF define un modelo de datos para describir recursos (mediante URI) y enunciados en forma sujeto-predicado-objeto (tripletas), en donde el sujeto es el recurso que se está describiendo, el predicado es la propiedad o relación que se establece acerca del sujeto y, por último, el objeto es el valor de la propiedad o el otro recurso con el que se establece la relación.

También proporciona el mecanismo para describir clases, objetos y propiedades, relaciones entre clases, restricciones de dominio y rango sobre las propiedades.

RDF *Schema* es el estándar dedicado a la representación de ontologías usando declaraciones RDF, con el propósito de representar conocimiento para recursos web.

"Mientras que la relación entre XML y XML Schema es una relación de control sintáctico, la relación entre RDF y RDFS es de control semántico: el esquema XML asegura que los elementos son usados correctamente, mientras que un esquema RDF asegura que, por ejemplo, en un aserto donde el sujeto es "carné de conducir" y el predicado es "nombre", el objeto es el nombre de una persona y no el modelo de un coche."[17]

3.2.5.4. OIL

OIL significa *Ontology Inference Layer* (lenguaje de intercambio de ontologías). En el esquema de Web Semántica se encuentra en la capa de "Vocabularios de ontologías", el cual, algunos aseguran debería llamarse "Lenguajes de ontologías."[18]

OIL es el primer lenguaje de representación de ontologías basado en estándares W3C. La lógica descriptiva (declaración de axiomas y reglas) es utilizada para la representación del conocimiento. Tiene sintaxis XML y está

[17] PEIS REDONDO, Eduardo; HERRERA VIEDMA, Enrique. *Ontologías, metadatos y agentes: recuperación "semántica" de la información*. España: Universidad de Granada, 2003. p. 3.

[18] Ibid.

definido como una extensión de RDFS (*RDF Schema*). Esto permite la interoperabilidad de agentes que funcionan para RDFS también funcionar para OIL.

A continuación surgió DAML+OIL, que es la fusión del lenguaje DAML (*Darpa Agent Markup Language*) de DARPA, cuya finalidad era extender la expresividad de RDFS, y OIL. Por tanto, es un lenguaje que unifica dos lenguajes, con el objetivo de estandarizar un lenguaje para la definición de ontologías.

3.2.5.5. OWL

En 2002 el grupo de trabajo *Web Ontology* de W3C publica OWL (*Web Ontology Language*), derivado de DAML+OIL y cimentado en RDFS.

La propuesta de OWL incluye también *OWL Lite*, versión reducida de OWL, cuyo objetivo es ser adoptada rápida y ampliamente por los desarrolladores.

El objetivo de OWL es incentivar y promover la publicación de datos usando ontologías en la WWW.

3.2.6. Editor de ontologías

Desarrollar una ontología manualmente es una tarea que consume mucho tiempo, más aún, si la extensión la ontología es amplia. Hay herramientas de software que facilitan la tarea de definir una ontología y traducirla automáticamente a un lenguaje de ontologías.

Los editores de ontologías son una herramienta que ayuda a los ingenieros del conocimiento para construir ontologías. Un editor de ontologías debe soportar la creación de clases, herencia, slots, relaciones, y definición de axiomas. En el mercado de editores existen otros como: OILEd, OntoEdit, Protégé.

3.2.6.1. Protégé

Protégé es un editor de ontologías y aplicaciones basadas en el conocimiento (*knowledge-base)*. De código abierto (*open source)*, basado en Java y con la capacidad de ampliarse por medio de una arquitectura *plug*-in.

Las ontologías desarrolladas en él pueden ser exportadas a una variedad de formatos incluyendo RDFS, OWL y XML Schema. Soporta dos principales formas de modelado de ontologías: *Protégé-Frames*y *Protégé-OWL*.

- *Protégé-Frames editor*. Permite construir y poblar ontologías que se basan en *frames-based*, de acuerdo con el protocolo OKBC (*Open Knowledge Base Connectivity protocol)*. En este modelo, una ontología consta de un conjunto de clases organizadas en una jerarquía de subsunción para representar conceptos más destacados de un dominio, un conjunto de *slots* asociados a las clases para describir sus relaciones y propiedades, y un conjunto de instancias para las clases.

- *Protégé-OWL editor.* Permite construir ontologías enfocadas para la Web Semántica, en especial en el lenguaje OWL. Una ontología descrita con OWL incluye clases, propiedades (*slots)* e instancias. También es posible definir reglas semánticas: axiomas.

Este es un trabajo enfocado en desarrollar un motor de búsqueda inteligente de pequeña escala en una colección de datos bien controlada y definida.

3.2.7. Lenguajes de recuperación de ontologías

Un lenguaje de recuperación, es aquel utilizado para recuperar información de su almacén. Éste es un conjunto de ordenes¸ operadores y estructuras que siguiendo normas lógicas (sintaxis) que permiten la consulta de registros. El lenguaje de recuperación está directamente relacionado con el modelo de datos.

3.2.7.1. SPARQL

Similar al lenguaje SQL para modelos relacionales, SPARQL es un lenguaje de consulta para RDF, el cual permite realizar búsquedas complejas sobre el grafo RDF. Para RDF existe una variedad de lenguajes de consulta: RDQL (por Hewlett Packard), RQL (por ICS-FORTH), SeRQL (por la empresa holandesa *Aidministrator*) y SPARQL (por W3C).

SPARQL es acrónimo recursivo de *SPARQL Protocol and RDF Query Language,* propuesto y normalizado por el grupo de trabajo *RDF Data Access Working Group* (DAWG) de W3C. Consisten en tres especificaciones:

- *SPARQL Query Language:* es el núcleo, en ésta especificación se explica la sintaxis para la composición de sentencias y su concordancia.
- *SPARQL Protocol*: formato para la devolución de los resultados de las búsquedas, a partir de un esquema de XML.
- *SPARQL Query XML Results Format*: describe el acceso remoto de datos y la transmisión de consultas de los clientes a los procesadores.

Utiliza WSDL para definir protocolos remotos para la consulta de bases de datos basadas en RDF.

Al igual que la tripleta RDF, SPARQL también define las sentencias en tripletas (sujeto-predicado-objeto) y terminadas en punto. Las tripletas representan patrones mediante los cuales se buscarán tripletas coincidentes en los datos.

En un *query* SPARQL, tanto el sujeto, predicado y objeto pueden substituirse por una variable:

Ecuación 9: tripleta SPARQL

```
?element table:name ?name
```

Así, la consulta anterior devolverá todo recurso RDF con una propiedad *name.* En SPARQL todas las posibles asociaciones se consideran, con lo que si un recurso tiene varias instancias de una propiedad, varios resultados serán devueltos para ese recurso.

Ecuación 10: Consulta SPARQL

```
PREFIX
 table:<http://www.daml.org/2003/01/periodictable/PeriodicTable#>
SELECT ?name
FROM<http://www.daml.org/2003/01/periodictable/PeriodicTable.owl>
WHERE{ ?element table:name ?name. }
```

La palabra PREFIX es utilizada para asociar un URI a una etiqueta utilizada en la consulta, pueden incluirse varios prefijos en la misma consulta. Al igual que SQL, para definir los campos que la consulta retornar, los campos se especifican con la palabra SELECT.

La palabra FROM identifica los datos sobre los cuales se ejecutará la consulta, regularmente es una ontología (aunque pueden ser varias). La palabra WHERE indica el patrón sobre el que se filtrarán los tripletes del RDF.

3.3. Sistemas de gestión del conocimiento

Hay varias definiciones sobre el conocimiento:

“Hechos, o datos de información adquiridos por una persona a través de la experiencia o la educación, la comprensión teórica o práctica de un tema u objeto de la realidad.”

Lo que se adquiere como información relativa a un campo determinado o a la totalidad del universo.

Conciencia o familiaridad adquirida por la experiencia de un hecho o situación.

Incluye el "saber qué" (*know that*), el "saber cómo" (*know how)* y el "saber dónde" (*know where*).”[19]

Existen dos tipos fundamentales de conocimiento: tácito y explícito.

[19]http://es.wikipedia.org/wiki/Conocimiento. Consulta: agosto de 2012.

El conocimiento tácito, es el tipo de conocimiento inconsciente, del cual podemos hacer uso de una forma mecánica sin darnos cuenta de su contenido, y que Michael Polanyi describe con un aforismo: Conocemos más de lo que podemos decir.

El conocimiento explicito, a diferencia del conocimiento tácito, de este sabemos que lo tenemos y para ejecutarlo somos conscientes de ello. Por esto es más fácil de transmitir o representarlo en un lenguaje.

El conocimiento se aprende y se crea, pero, también se olvida, por esa razón el almacenar, organizar y recuperar el conocimiento, constituye un aspecto importante en la gestión del conocimiento.

Los sistemas de gestión del conocimiento (SGC en adelante) o sistemas de organización del conocimiento (conocidos como SKO - *Systems of Knoledge Organization*) son herramientas y mecanismos para llevar a cabo la gestión del conocimiento (GC en adelante), el objetivo es apoyar los procesos de creación, almacenamiento, manipulación, transferencia, recuperación y aplicación del conocimiento. La creación de conocimiento implica desarrollar un nuevo contenido o reemplazar el contenido existente dentro del conocimiento tácito y explícito de una organización.

La gestión del conocimiento puede implementarse en variedad de aplicaciones como: repositorios de documentos, bases de datos de experiencias, listas de discusión, en general en aplicaciones cuyo objetivo sea transferir el conocimiento tácito y explicito entre los usuarios que la utilizan, de modo que sea accesible a quién quiera, en cualquier momento.

4. BÚSQUEDAS INTELIGENTES EN SISTEMAS DE GESTIÓN DEL CONOCIMIENTO

4.1. Introducción

Hasta aquí, se ha hablado de la teoría que rige las búsquedas en los sistemas de recuperación de información. Este trabajo pretende dar una aproximación a los conceptos relacionados con las búsquedas inteligentes.

Los conceptos relacionas con búsquedas inteligentes son los agentes inteligentes, ontologías, procesamiento del lenguaje natural y motores de búsqueda inteligentes. La combinación de estos conceptos y su implementación permite una búsqueda inteligente.

4.2. Búsquedas inteligentes

El constante incremento de la Web exhibe lo limitado de los actuales procesos de búsqueda, primordialmente por la falta de acceso a la semántica de los documentos y la limitación de los usuarios para expresar sus necesidades, los motores de búsqueda actual, como se ha visto, no implementan búsquedas de semántica estricta, si es que implementen algo de semántica.

La contextualización, interpretación e inferencia de la semántica hace de una persona inteligente.

Los modelos de búsqueda clásico y web no promueven la búsqueda semántica, sino sintáctica; la similitud de palabras claves. En la Web los resultados de búsquedas se basan en las páginas más visitadas, las páginas de mayor antigüedad, peso de los hiperenlaces (*pageRank)*, etc. Este comportamiento se debe a las tecnologías actuales de representación, basadas en HTML y un sinfín de formatos, enfocadas en la presentación de la información y no en el contenido.

De esa cuenta, Tim Berners- Lee impulsa la idea de una Web Semántica, una Web, como una extensión de la Web actual, que almacene y recupere la información con base en su sentido semántico, así, no sólo los humanos sean capaces de comprender el contenido, sino también agentes de software.

Para que ese objetivo sea posible es necesario proveer de tecnologías y mecanismos con las cuales definir la semántica de los documentos, y así ser usada por buscadores inteligentes con el fin de ofrecer resultados precisos y contextualizados.

La dificultad que representa ese proceso descrito por Berners-Lee se debe a que cada *Web Master* o administrador de una página o documentos debe elaborar su propia ontología que describa el dominio. Sin embargo, existen propuestas, al menos de semi-automatizar este proceso.

Los elementos que se han identificado y pueden ser parte una búsqueda inteligente son: agentes inteligentes, procesamiento del lenguaje natural, ontologías, motores de búsqueda inteligentes.

4.2.1. Agentes inteligentes

Los agentes inteligentes deben ser los encargados de recabar la información de comportamiento, necesidades y gustos de los usuarios, esto permite a los agentes realizar búsquedas en la web de información específica. Debe tener la capacidad de filtrar y realizar inferencias en representación del usuario u otro agente.

En un motor de búsqueda inteligente, los agentes inteligentes juegan un papel importante, debido a que éstos son los encargados de realizar las búsquedas en la web (con base en lo aprendido del usuario: necesidades, gustos, configuración del usuario, comportamiento; lo cual se llama perfil) y obtener los resultados.

En la Web Semántica, los agentes inteligentes, deben interactuar con las ontologías que describen los dominios y así, por ejemplo, decidir que zapatos el usuario desea comprar, que noticias quiere leer, que programas quiere ver, que investigaciones le interesan, etc.

4.2.2. Procesamiento de lenguaje natural

Tanto los agentes inteligentes y los motores de búsqueda inteligentes deben proveer del procesamiento de lenguaje natural. El procesamiento del lenguaje natural se encarga en la capacidad de comunicación de personas y máquinas. En este caso, la capacidad de interpretar las necesidades de información de una persona y convertirlas a un lenguaje computacional entendible.

Con un módulo de procesamiento de lenguaje el usuario puede plantear una necesidad de información, tal cual el piense, sin necesidad de buscar conceptos que él cree que podría servir para encontrar la información deseada.

4.2.3. Ontologías

Las ontologías son el elemento que contextualiza y permite inferencia conocimiento, esto por medio de los lenguajes ontológicos. Además posee el visto bueno y la promoción de toda la comunidad W3C, en especial de su promotor Berners-Lee.

4.2.4. Motor de búsqueda inteligente

Un buscador inteligente debe indexar los recursos, disponibles para ello, utilizando la riqueza semántica de esos mismos recursos, con el objetivo de mejorar la precisión en las búsquedas.

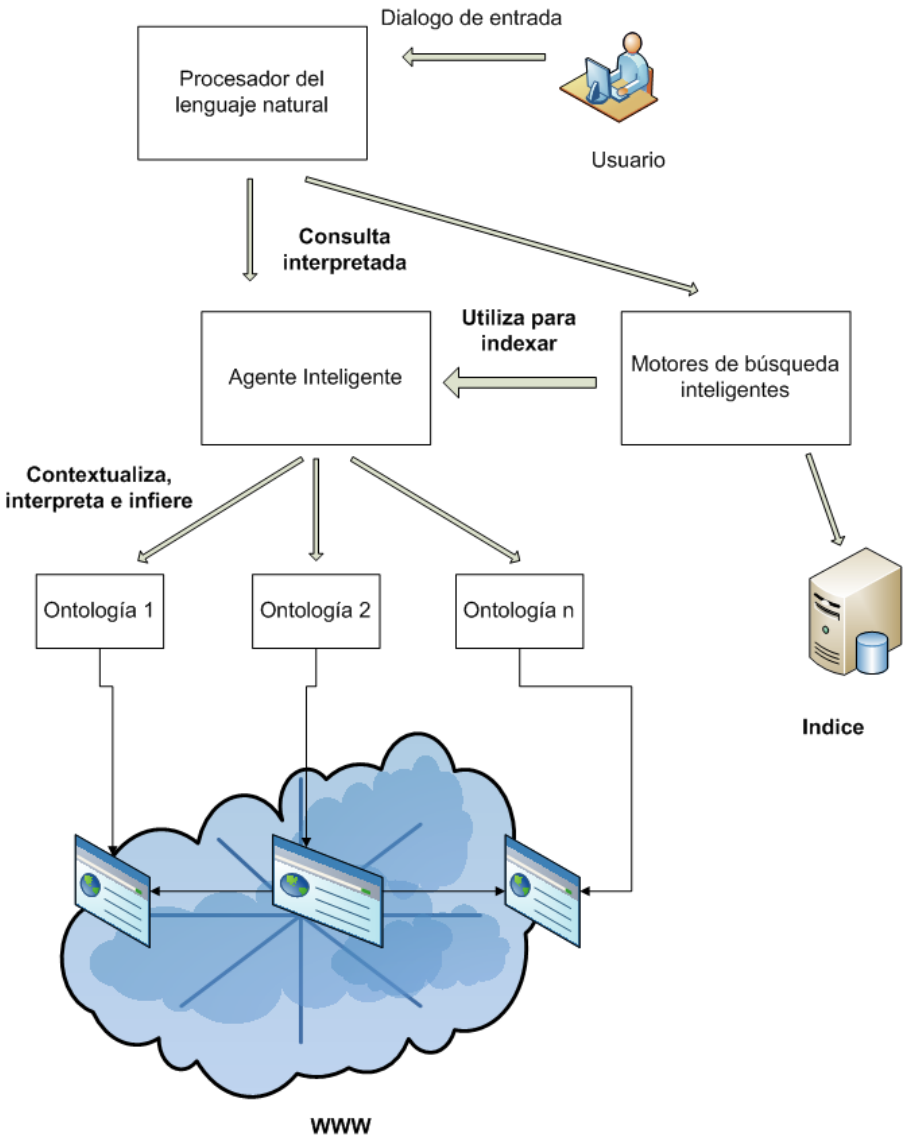
Un motor de búsqueda inteligente necesita hacer uso de los agentes inteligentes, los cuales serán los encargados de “indexar” la información con base en las ontologías que definen tal recurso. Además, un procesador de lenguaje natural, para comprender y traducir la necesidad del usuario.

4.2.5. Definición de búsquedas inteligentes

Una búsqueda inteligente es aquella en la cual el software es capaz de interpretar correctamente lo que el usuario desea, utilizando tecnologías y mecanismos para contextualizar la información y en los resultados obtenidos sean exactos, relacionados y recomendados basados en las estructura semántica de las ontologías.

A continuación se presenta un esquema de componentes para realizar una búsqueda inteligente.

Figura 18. **Esquema de búsqueda inteligente**



Fuente: elaboración propia.

El esquema representa las relaciones entre los cuatro componentes que de una búsqueda inteligente.

4.3. Prototipo sistema de gestión del conocimiento

El sistema de gestión de conocimiento se ha implementado basado en una ontología de un dominio específico, ésta trata de describir las relaciones y clases existentes en los trabajos de graduación en la Universidad de San Carlos de Guatemala. Los trabajos de graduación son documentos de investigación o prácticas que los estudiantes de la universidad elaboran con el objetivo de obtener un título de grado. Elementos importantes en este dominio son la universidad, asesor, estudiante, los temas de los cuales trata el documento, si es tesis o EPS, entre otros.

4.3.1. Implementación de la ontología en Protégé

La ontología describe a personas (estudiantes y asesores) que participan en la elaboración de trabajos de graduación en la Universidad de San Carlos de Guatemala.

Clases:

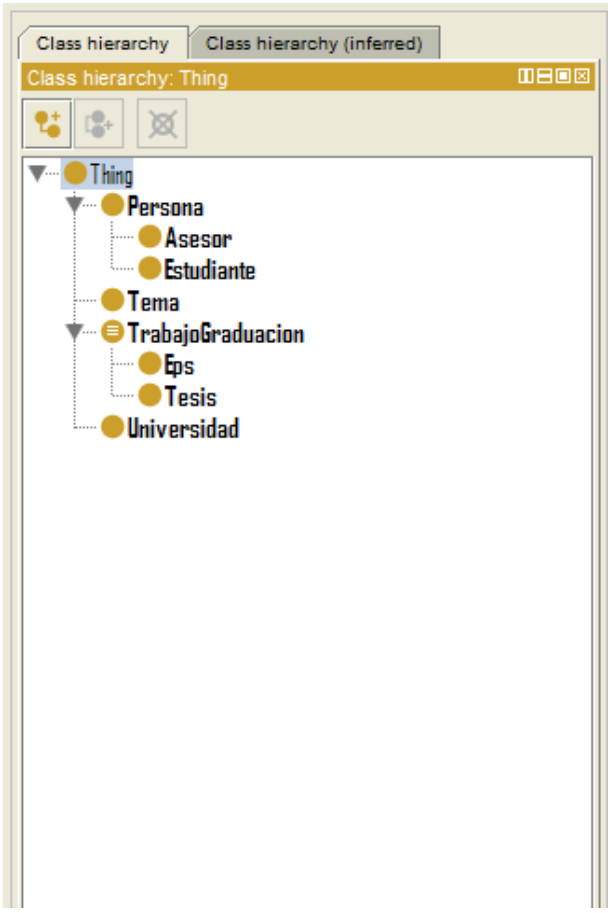
Persona
Estudiante
Asesor
Tema
Trabajo de Graduación
Eps
Tesis

Universidad

Propiedades:

hasAsesorA
hasAutorA
hasEstudiante
hasReferenciaA
hasSubtemaA
hasTema
isAsesorDe
isAutorDe
isEstudiante
isReferenciaDe
isSubtemaDe
isTema
correoElectronico
edad
fechaPublicacion
homePage
keywords
nick
nombre
resumen
sinónimo
teléfono
titulo
url

Figura 19. **Representación de la ontología en el navegador de Protégé**



Fuente: navegador de clases de Protégé.

A continuación se describen las clases y subclases presentes:

Tabla III. **Clases de la ontología**

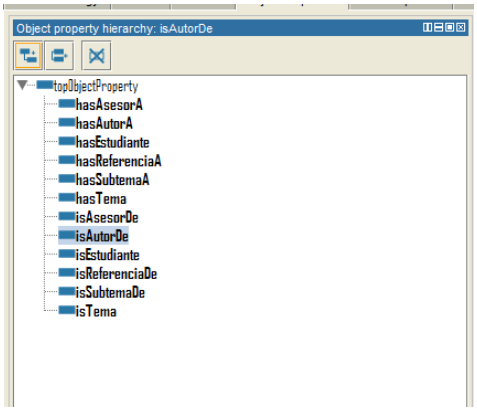
Clase	Descripción
Thing	En *OWL* siempre se define una superclase llama *Thing* (Cosa) que será la clase padre de cualquier otra clase. Todos los individuos creados pertenecerán a esta clase por inducción.
Persona	Clase que define los atributos de una persona, en ésta categoría se clasifican a las clases Asesor y Estudiante.
Asesor	Es la persona que guía al estudiante en la elaboración de un trabajo de graduación. El asesor puede asesorar a más de un estudiante.
Estudiante	Es la persona que elabora el trabajo de graduación, el estudiante estudia en una universidad y tiene un asesor.
Tema	El tema sobre el que versa el trabajo de graduación.
Trabajo de Graduación	Clase que define los atributos en común de los trabajos de graduación ya sea EPS o Tesis.
Eps	Trabajo de graduación que se basa en una práctica supervisada.
Tesis	Trabajo de graduación de investigación.
Universidad	Centro educativo superior que promueve la creación de trabajos de graduación.

Fuente: elaboración propia.

Nota: en el nombramiento de las clases no se utiliza tildes ya que la aplicación no lo permite.

Cada clase posee relaciones y propiedades, las relaciones indican con cuales otra clase tiene algún vínculo y permiten la navegación a través de esas relaciones. Las propiedades, son características que describen a cada uno los individuos que son parte de la clase. Las propiedades y relaciones se van heredando de clase a subclase y las subclases pueden definir propiedades específicas.

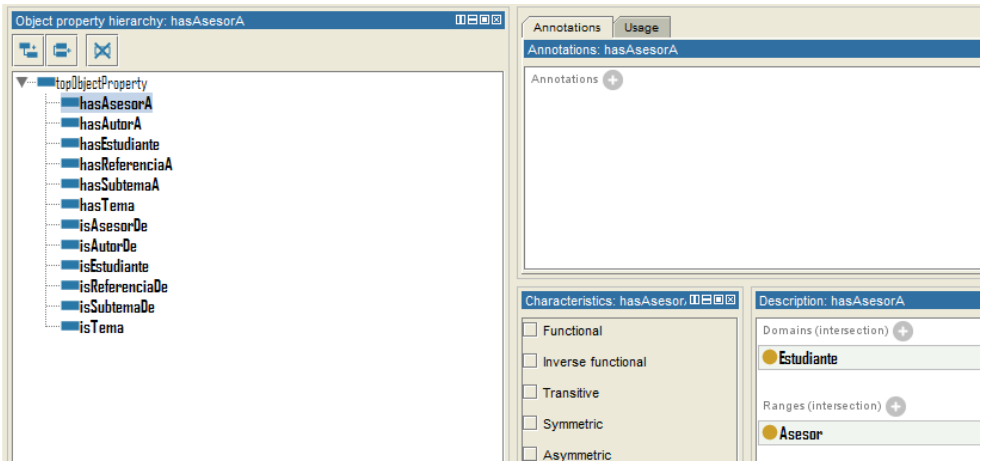
Figura 20. **Propiedades de objetos (relaciones)**



Fuente: navegador de propiedades de Protégé.

Las relaciones son conocidas como propiedades de objetos, una propiedad debe definir un dominio y un rango. Para las propiedades de objetos, tanto el dominio como el rango deben ser clases.

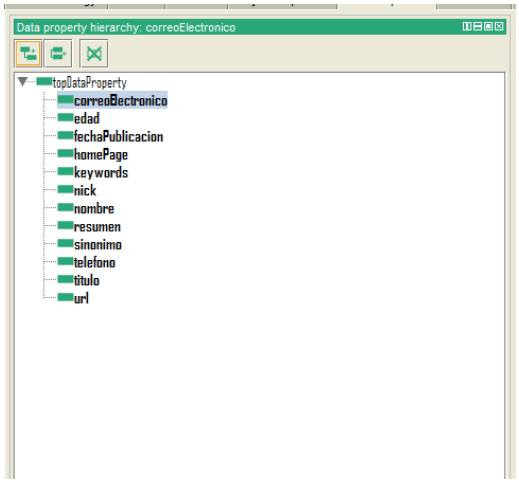
Figura 21. **Dominio y rango de la propiedad *hasAsesorA***



Fuente: navegador de propiedades de Protégé.

A continuación se listan las propiedades de datos:

Figura 22. **Propiedades de datos**



Fuente: navegador de propiedades de Protégé.

Una descripción más detallada del dominio y rango de cada propiedad se aprecia en la siguiente tabla:

Tabla IV. **Propiedad, dominio, rango y descripción**

Propiedad	Descripción
hasAsesorA	Dominio: Estudiante Rango: Asesor Un estudiante tiene un asesor que lo guía en la elaboración del trabajo de graduación.
hasAutorA	Dominio: TrabajoGraduacion Rango: Estudiante El trabajo de graduación tiene de autor a uno o más estudiantes.

Continuación de la tabla IV.

hasEstudiante	Dominio: Universidad Rango: Estudiante Un individuo de la clase Estudiante estudia en una universidad.
hasReferenciaA	Dominio: TrabajoGraduacion Rango: TrabajoGraduacion Puede que un trabajo de graduación cite a otro trabajo de graduación.
hasSubtemaA	Dominio: Tema Rango: Tema Taxonomía de los temas de los cuales puede versar una tesis, por ejemplo: Química tiene de subtema a Química Análitica.
hasTema	Dominio: TrabajoGraduacion Rango: Tema Trabajo de Graduación versa sobre uno o más temás.
isAsesorDe	La propiedad inversa de *hasAsesorA.*
isAutorDe	La propiedad inversa de *hasAutorA.*
isEstudiante	La propiedad inversa de *hasEstudiante.*
isReferenciaDe	La propiedad inversa de *hasReferenciaA*.
isSubtemaDe	La propiedad inversa de *hasSubtemaA.*
isTema	La propiedad inversa de *hasTema*.
correoElectronico	Dominio: Persona Rango: tipo de dato cadena Correo electrónico de la persona
edad	Dominio: Persona Rango: tipo de dato entero Edad de la persona
fechaPublicacion	Dominio: TrabajoGraduacion Rango: fecha Fecha de publicación del trabajo de graduación
homePage	Dominio: Persona Rango: tipo de dato cadena Página Web de una persona
keywords	Dominio: TrabajoGraduacion Rango: tipo de dato cadena Palabras claves de un trabajo de graduación
nick	Dominio: Persona Rango: tipo de dato cadena Nick de una persona. Ejemplo: Nombre: Juan Manuel Pérez Rodríguez; Nick: Juan Pérez

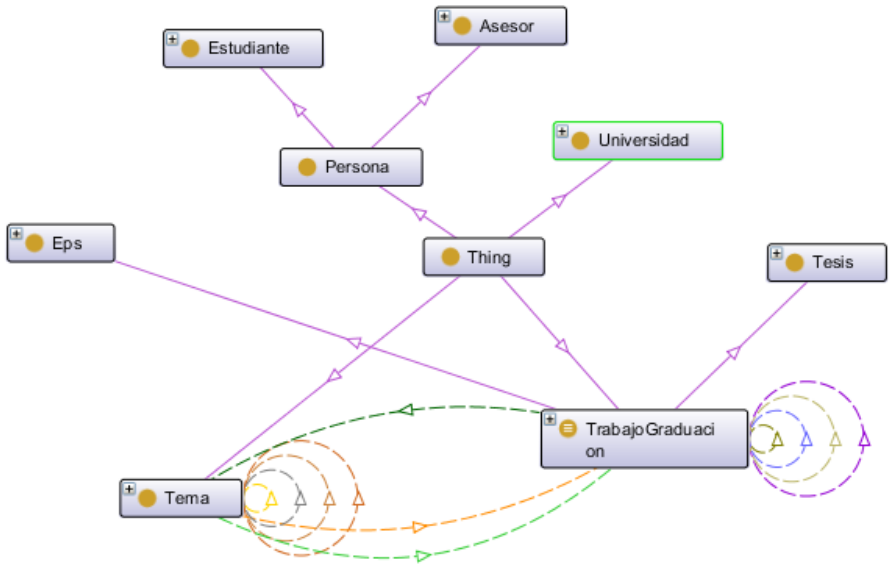
Continuación de la tabla IV.

nombre	Dominio: Persona o Tema o Universidad Rango: tipo de dato cadena Nombre de la persona o tema o universidad
resumen	Dominio : TrabajoGraduacion Rango: tipo de dato cadena Resumen del trabajo de graduación
sinónimo	Dominio: Persona o Tema o TrabajoGraduacion o Universidad Rango: tipo de dato cadena
teléfono	Dominio: Persona Rango: tipo de dato cadena Teléfono de la persona
titulo	Dominio: TrabajoGraduacion Rango: tipo de dato cadena Titulo del trabajo de graduación
url	Dominio: TrabajoGraduacion o Universidad Rango: tipo de dato cadena Dirección de la ubicación del trabajo de graduación

Fuente: elaboración propia.

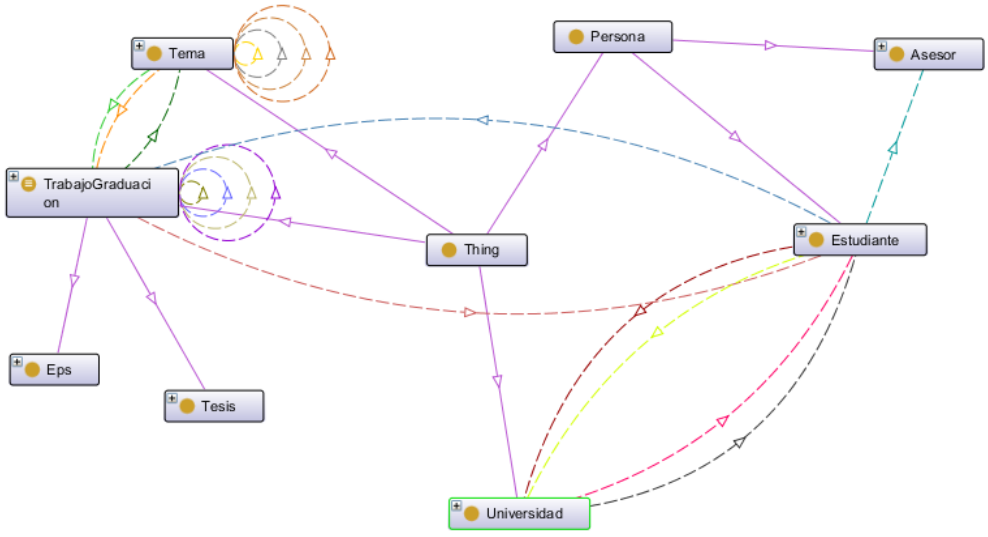
La siguiente imagen muestra la taxonomía del dominio, la clase *Thing* es la superclase de todas las clases, la flecha apunta hacia la subclase.

Figura 23. **Taxonomía de la ontología**



Fuente: *plugin* OntoGraf de Protégé.

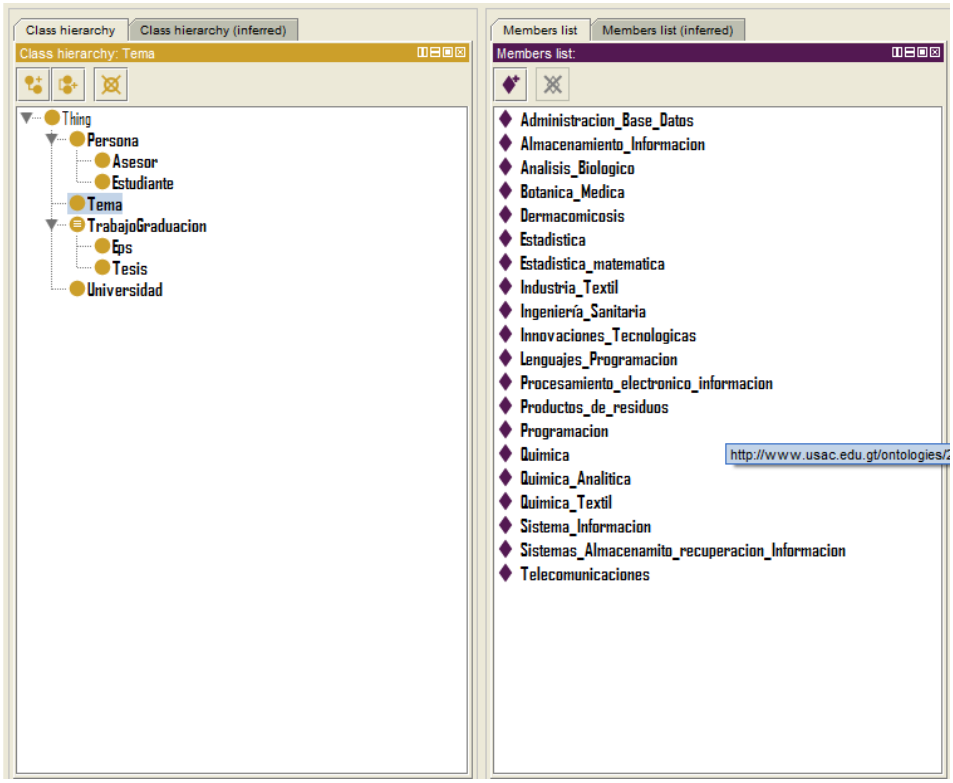
Figura 24. **Relaciones de las clases en la ontología**



Fuente: *plugin* OntoGraf de Protégé.

Luego de definir e implementar la ontología, el siguiente paso es poblarla, es decir, ingresar los individuos que forman parte de la misma. El proceso de poblar la ontología puede hacerse de forma manual o automática. La información utilizada para poblar la ontología se ha obtenido de una base de datos publica: http://bibios.usac.edu.gt, la biblioteca de la Universidad de San Carlos de Guatemala.

Figura 25. **Instancias para la ontología**

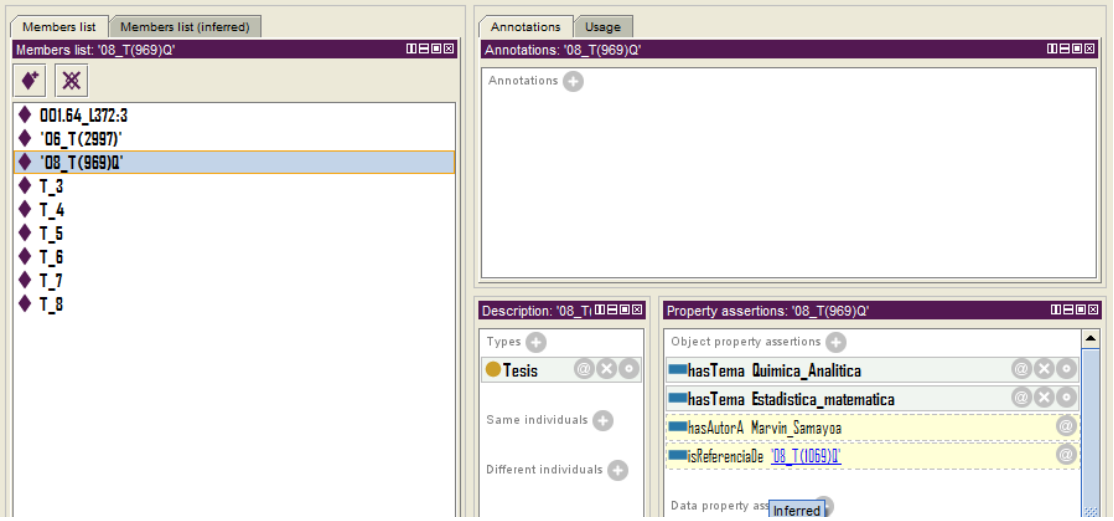


Fuente: navegador de individuos de Protégé.

Al momento de crear cada individuo deben de establecerse también sus propiedades. Protegé permite inferir algunas propiedades a partir de las definiciones establecidas al momento de la creación de la ontología, para ello

provee un razonador automático. La imagen muestra relaciones que el razonador infirió automáticamente.

Figura 26. **Propiedades inferidas automáticamente**



Fuente: navegador de individuos de Protégé.

El almacenamiento de la ontología puede ser en archivos XML o en base de datos, la mejor opción para mantener la interoperabilidad y distribución de la web son los archivos XML. Estos archivos pueden estar definidos en RDF o un lenguaje más enriquecido como OWL.

Figura 27. **Fragmento de la ontología en almacenado en formato OWL en un archivo XML**

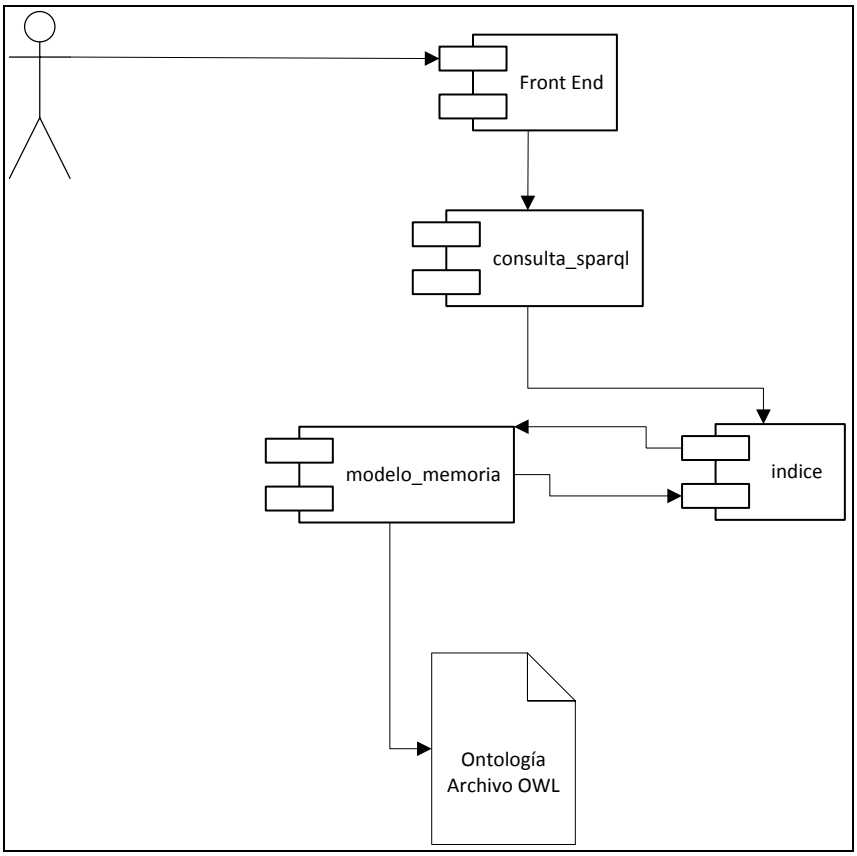
```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Laudon "http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#Laudon," >
    <!ENTITY _T2 "http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#08_T(969)" >
    <!ENTITY _T "http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#08_T(1069)" >
]>


<rdf:RDF xmlns="http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#"
     xml:base="http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:Laudon="http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#Laudon,"
     xmlns:_T2="http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#08_T(969)"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:_T="http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#08_T(1069)">
    <owl:Ontology rdf:about="http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl">
        <rdfs:label rdf:datatype="&rdfs;Literal">1.0</rdfs:label>
        <rdfs:comment rdf:datatype="&rdfs;Literal">Desarrollada para la tesis de grado &quot;Búsquedas Inteligentes por medio de Ontologías
        &quot;</rdfs:comment>
        <rdfs:isDefinedBy rdf:datatype="&rdfs;Literal">Honard Bravo</rdfs:isDefinedBy>
        <rdfs:comment rdf:datatype="&rdfs;Literal">Ontología del dominio, construida con el objetivo de describir las relaciones en la
        construcción de trabajos de graduación en la Universidad de San Carlos de Guatemala.</rdfs:comment>
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Annotation properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->

```

Fuente: elaboración propia.

4.3.2. Arquitectura

Para la construcción del sistema se ha utilizado el *framework* para *java:* Jena. Este *framework* permite construir la ontología a partir de un archivo OWL, la ontología en memoria dinámica se le llama modelo. El modelo es utilizado para indexar todas las propiedades literales, esto permite una búsqueda de palabras en el modelo y así recuperar las propiedades relacionadas. El componente SPARQL permite realizar consultas utilizando ese lenguaje de consulta. La figura a continuación muestra los componentes del sistema.

Figura 28. **Componentes del sistema**



Fuente: elaboración propia.

El siguiente fragmento de código permite cargar la ontología en memoria dinámica. El modelo utiliza el lenguaje OWL Lite y también utiliza un *reasoner*. Ambos elementos vienen incluidos en el *framework Jena*.

Figura 29. **Construcción del modelo con *Jena***

```
      public Model getModel(){
      Model model = ModelFactory.createOntologyModel(
OntModelSpec.OWL_LITE_MEM_RULES_INF );
               InputStream in = FileManager.get().open(owl_file_path);
      if (in == null){
                   System.out.println("Error al intentar cargar la ontología.");
      }
               model.read(in, null);
               return model;
      }
```

Fuente: elaboración propia.

El índice se construye a partir del modelo. El índice se compone de todas las literales existentes en la ontología, así al buscar una palabra, el índice es capaz de relacionar la palabra con la propiedad a la cual pertenece:

Figura 30. **Construcción del índice a partir del modelo con *Jena***

```
      public IndexLARQ createIndexLiterals(Model m){
       // Constructor del indice
               IndexBuilderString larqBuilder = new IndexBuilderString();
       // El indice se contruye a partir de todas las sentencias en formato RDF
      larqBuilder.indexStatements(m.listStatements());
               larqBuilder.closeWriter();
      return larqBuilder.getIndex();
          }
```

Fuente: elaboración propia.

4.3.3. Búsquedas inteligentes

Los agentes inteligentes son los encargados de inferir conocimiento automáticamente utilizando las ontologías como base de conocimiento. En una búsqueda inteligente se muestran los resultados exactos de la búsqueda y también los resultados relacionados.

Los resultados relacionados de obtienen a partir de las propiedades de relación con otros objetos, por ejemplo: el individuo T_3 que de tipo *Tesis* tiene de autor a (*hasAutorA*) el individuo Vivian_Toledo que es de tipo *Estudiante*. Para cualquier humano esa relación es natural, pero para un agente inteligente sólo a través de una definición clara y formal se puede inferir ese conocimiento. El agente es capaz de comprender esa estructura lógica y jerárquica.

La recuperación de información es posible gracias a la estructura RDF subyacente en OWL (según la arquitectura de Web Semántica presentada en el tercer capítulo) utilizando SPARQL como lenguaje ontológico de recuperación.

La siguiente consulta muestra todos los trabajos de graduación presentes en la ontología y el autor de los mismos.

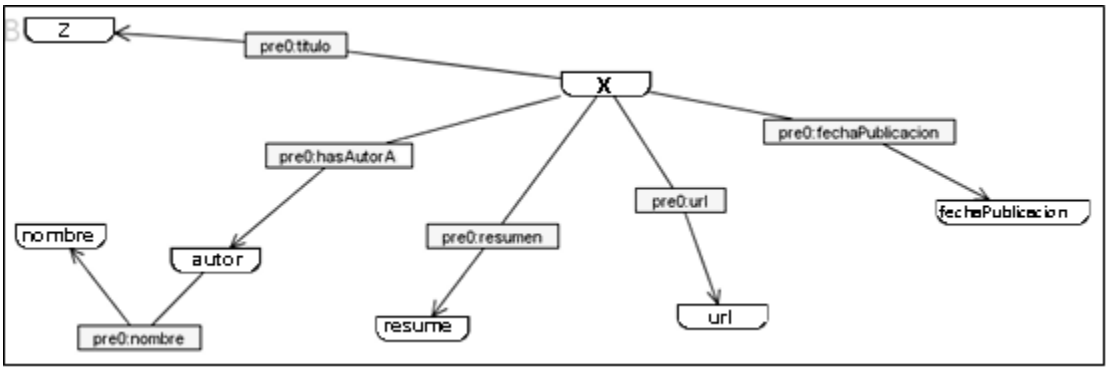
Figura 31. **Consulta en SPARQL**

```
        SELECT  ?fechaPublicacion ?titulo ?resumen ?url ?nombre
        WHERE
          {   ?x                      <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#fechaPublicacion>
?fechaPublicacion ;
        <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#hasAutorA>  ?oautor ;
        <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#hasAutorA>  ?a ;
        <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#titulo>  ?titulo ;
        <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#resumen>  ?resumen ;
          <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#url>  ?url .
        ?oautor  <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#nombre>  ?nombre .
        }
```

Fuente: elaboración propia.

La siguiente figura es más explícita mostrando las relaciones existentes entre la referencia de cada individuo y las propiedades de éste. En nuestro caso *x* representa a los individuos de tipo *TrabajoGraduación* y cada línea lo conecta con las propiedades indicadas en la consulta, *autor* representa al individuo de tipo *Estudiante* que es autor del trabajo de graduación en cuestión, al individuo *autor* se le ha solicitado la propiedad del nombre.

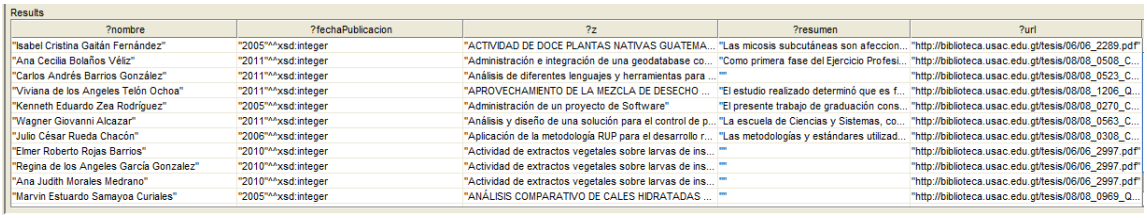
Figura 32. **Gráfica de la consulta en SPARQL**



Fuente: grafo de consulta SPARQL de Protégé.

Los resultados obtenidos al ejecutar la consulta anterior son los siguientes:

Figura 33. **Resultado de consulta en SPARQL**



Results

?nombre	?fechaPublicacion	?z	?resumen	?url
"Isabel Cristina Gaitán Fernández"	"2005"^^xsd:integer	"ACTIVIDAD DE DOCE PLANTAS NATIVAS GUATEMA...	"Las micosis subcutáneas son afeccion...	"http://biblioteca.usac.edu.gt/tesis/06/06_2289.pdf"
"Ana Cecilia Bolaños Véliz"	"2011"^^xsd:integer	"Administración e integración de una geodatabase co...	"Como primera fase del Ejercicio Profesi...	"http://biblioteca.usac.edu.gt/tesis/08/08_0508_C...
"Carlos Andrés Barrios González"	"2011"^^xsd:integer	"Análisis de diferentes lenguajes y herramientas para ...	""	"http://biblioteca.usac.edu.gt/tesis/08/08_0523_C...
"Viviana de los Angeles Telón Ochoa"	"2011"^^xsd:integer	"APROVECHAMIENTO DE LA MEZCLA DE DESECHO ...	"El estudio realizado determinó que es f...	"http://biblioteca.usac.edu.gt/tesis/08/08_1206_Q...
"Kenneth Eduardo Zea Rodríguez"	"2005"^^xsd:integer	"Administración de un proyecto de Software"	"El presente trabajo de graduación cons...	"http://biblioteca.usac.edu.gt/tesis/08/08_0270_C...
"Wagner Giovanni Alcazar"	"2011"^^xsd:integer	"Análisis y diseño de una solución para el control de p...	"La escuela de Ciencias y Sistemas, co...	"http://biblioteca.usac.edu.gt/tesis/08/08_0563_C...
"Julio César Rueda Chacón"	"2006"^^xsd:integer	"Aplicación de la metodología RUP para el desarrollo r...	"Las metodologías y estándares utilizad...	"http://biblioteca.usac.edu.gt/tesis/08/08_0308_C...
"Elmer Roberto Rojas Barrios"	"2010"^^xsd:integer	"Actividad de extractos vegetales sobre larvas de ins...	""	"http://biblioteca.usac.edu.gt/tesis/06/06_2997.pdf"
"Regina de los Angeles García Gonzalez"	"2010"^^xsd:integer	"Actividad de extractos vegetales sobre larvas de ins...	""	"http://biblioteca.usac.edu.gt/tesis/06/06_2997.pdf"
"Ana Judith Morales Medrano"	"2010"^^xsd:integer	"Actividad de extractos vegetales sobre larvas de ins...	""	"http://biblioteca.usac.edu.gt/tesis/06/06_2997.pdf"
"Marvin Estuardo Samayoa Curiales"	"2005"^^xsd:integer	"ANÁLISIS COMPARATIVO DE CALES HIDRATADAS ...	""	"http://biblioteca.usac.edu.gt/tesis/08/08_0969_Q...

Fuente: elaboración propia.

Un agente inteligente debe ser capaz de comprender que *Ana Cecilia Bolaños Véliz*es es una Estudiante y que ha escrito una tesis cuyo nombre es “Administración e integración…. ” A partir de ello si quisiera obtener aquellas tesis que comparten los mismos temas ó aquellas tesis que han tenido al mismo asesor que la estudiante.

La búsqueda de información por medio de buscadores web utilizando palabras claves no será reemplazada por las tecnologías semánticas, estas tecnologías enriquecen y contextualizan la información ya que los *web crawlers* podrán obtener más información contextual a través de la(s) ontología(s) de cada sitio. Existen esfuerzos para estandarizar ontologías que todo mundo debería usar, por ejemplo, la ontología FOAF (*Friend of a Friend)*, es una ontología que define a las personas y su relación con otras personas.

4.3.4. Resultados

Para las búsquedas, a partir de una palabra o varias, se utiliza el modelo y el índice construidos anteriormente, se debe construir la consulta en SPARQL.

Figura 34. **Consulta en SPARQL utilizando un índice**

```
      String queryString = StrUtils.strjoin("\n",
                      "PREFIX pf:     <http://jena.hpl.hp.com/ARQ/property#>",
                      "SELECT ?individuo ?propiedad ?match {",
                      "   ?match pf:textMatch '"+searchString+"'.",
      "   ?individuo ?propiedad ?match.",
                      "}");
```

Fuente: elaboración propia.

El *namespace* `<http://jena.hpl.hp.com/ARQ/property#>` permite la búsqueda en la ontología utilizando el índice construido con anterioridad. La propiedad *textMatch,* que es una propiedad especial utilizada por Jena, indica que se deben recuperar aquellos individuos (`?individuo`) en el cual alguna de sus propiedades contenga la cadena buscada (`searchString`).

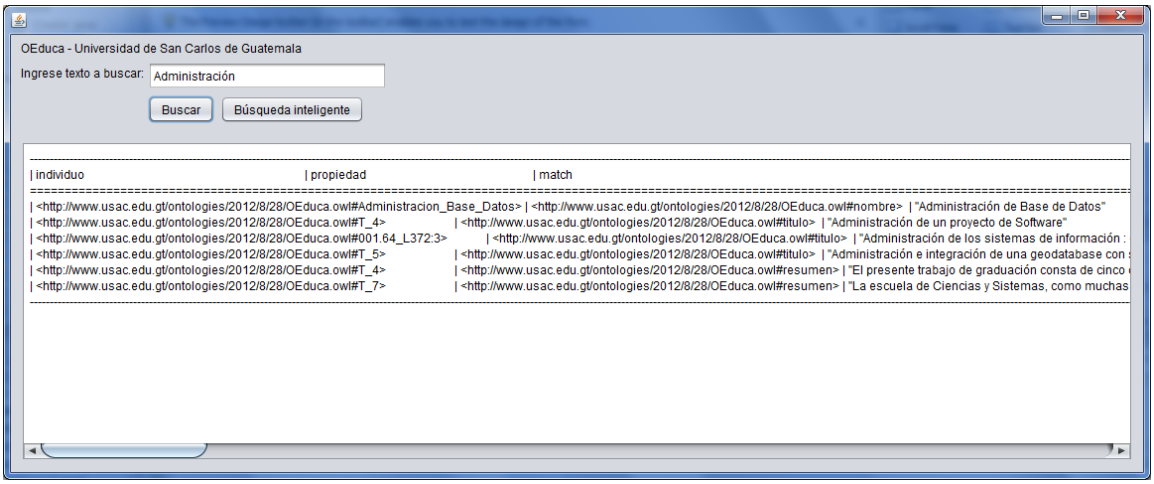
Hasta este punto es un método de recuperación de información clásico. La respuesta de la anterior consulta, cuando se busca el término "*Administración",* se ha dividido en tres partes para apreciarla mejor, la de los individuos devueltos, le siguen las propiedades y por último el valor del texto que concordó con la cadena de búsqueda. Las líneas numeradas se corresponden en un trío RDF (sujeto-propiedad-objeto).

```
-------------------------------------------------------------------------------
| individuo                                                                   |
===============================================================================
1|<http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#Administracion_Base_Datos> |
2|<http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_4>                 |
3|<http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#001.64_L372:3>       |
4|<http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_5>                 |
5|<http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_4>                 |
6|<http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_7>                 |
```

```
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
| propiedad                                                                    |
================================================================================
1 | <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#nombre>             |
2 | <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#titulo>             |
3 | <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#titulo>             |
4 | <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#titulo>             |
5 | <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#resumen>            |
6 | <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#resumen>            |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
| match
================================================================================
1 | "Administración de Base de Datos"
2 | "Administración de un proyecto de Software"
3 | "Administración de los sistemas de información: organización y tecnología"
4 | "Administración e integración de una geodatabase con sistemas de información
geográfica"
5 | "El presente trabajo de graduación consta de cinco capítulos, los cuales hacen
énfasis en … administración … "
6 | "La escuela de Ciencias y Sistemas, como muchas escuelas o instituciones dentro de
la Universidad … administración … "
--------------------------------------------------------------------------------
```

La aplicación ha encontrado seis individuos en los cuales el valor de alguna de sus propiedades contiene la cadena buscada. Puede observarse también que el individuo T_4 aparece dos veces y esto es porque tanto la propiedad #titulo y #resumen contienen a la cadena buscada.

Figura 35. **Resultado de la búsqueda**



Fuente: elaboración propia, con base en la aplicación OEduca.

Figura 36. **Consulta en SPARQL para búsqueda inteligente**

```
String queryString = StrUtils.strjoin("\n",
"PREFIX pf:     <http://jena.hpl.hp.com/ARQ/property#>",
                "SELECT ?individuo ?propiedad ?match
?todas_propiedades ?valor {",
"   ?match pf:textMatch '"+searchString+"'.",
"   ?individuo ?propiedad ?match.",
                "   ?individuo ?todas_propiedades ?valor.",
                "}");
```

Fuente: elaboración propia.

Esta consulta es similar a la anterior con algunos agregados. Se ha agregado una nueva condición: Devolver TODAS las propiedades y su valor de cada individuo que retorne la consulta (`?todas_propiedades ?valor`). Con ésta clausula la consulta devolverá no sólo los resultados exactos sino también

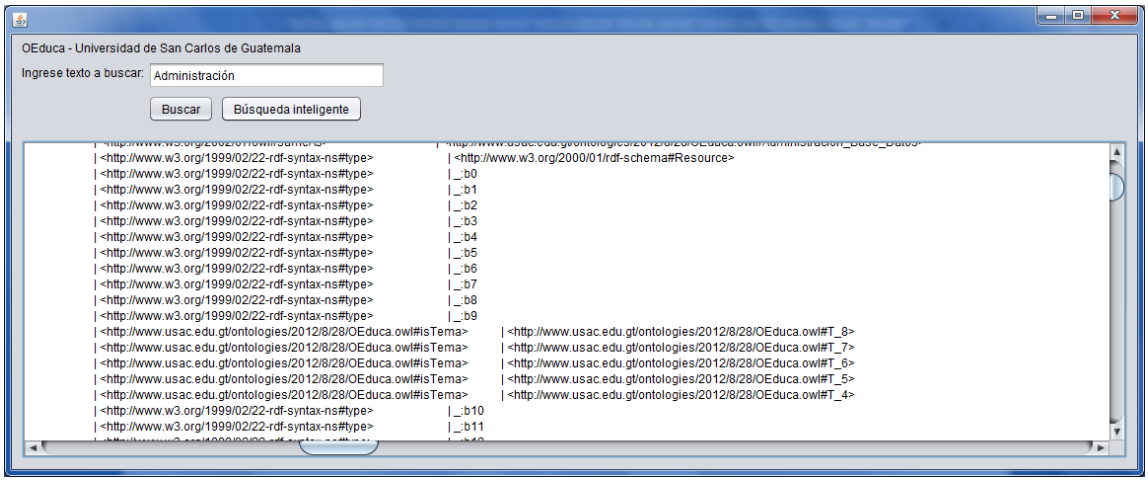
aquellos relacionados. Por ejemplo para el individuo <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#Administracion_Base_Datos> el resultado es el siguiente:

```
------------------------------------------------------------------------
 | todas_propiedades
========================================================================
1| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#sinonimo>
2| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#nombre>
3| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
4| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
5| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
6| <http://www.w3.org/2002/07/owl#sameAs>
7| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
8| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
9| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
10| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
11| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
12| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
13| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
14| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
15| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
16| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
17| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
18| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#isTema>
19| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#isTema>
20| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#isTema>
21| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#isTema>
22| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#isTema>


------------------------------------------------------------------------------
| valor
==============================================================================
1| "administracion de base de datos"
2| "Administración de Base de Datos"
3| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#Tema>
4| <http://www.w3.org/2002/07/owl#NamedIndividual>
```

```
5| <http://www.w3.org/2002/07/owl#Thing>
6| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#Administracion_Base_Datos>
7| <http://www.w3.org/2000/01/rdf-schema#Resource>
8| _:b0
9| _:b1
10| _:b2
11| _:b3
12| _:b4
13| _:b5
14| _:b6
15| _:b7
16| _:b8
17| _:b9
18| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_8>
19| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_7>
20| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_6>
21| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_5>
22| <http://www.usac.edu.gt/ontologies/2012/8/28/OEduca.owl#T_4>
```

La línea 3 indica, por ejemplo, que el individuo en cuestión es de tipo *Tema*. Además la línea 5 hace uso un axioma (recordemos que *Tema* es una subclase de *Thing)* por ello también el individuo en cuestión es de tipo *Thing*. Las líneas de la 18 a la 22 muestran otros resultados relacionados ya que indican que este individuo (de tipo *Tema*) es un tema (*isTema*) de los individuos *T_8, T_7, T_6, T_5* y *T_4* (de tipo *Tesis)* se ve que ese razonamiento se ha hecho automáticamente.

Figura 37. **Resultado de búsqueda inteligente**



Fuente: elaboración propia, con base en la aplicación OEduca.

CONCLUSIONES

1. La sobrecarga de información es un gran problema para la humanidad, no por la producción de conocimiento, sino por la incapacidad de localizar la información adecuada en los mementos adecuados. Los sistemas de recuperación de información juegan un papel fundamental para enfrentar el problema al tratar de automatizar la recuperación de información.

2. Hay varios elementos a considerar entre la recuperación de información tradicional y la recuperación en la web. Las principales son el método de clasificación (*ranking)* y el manejo de la distribución de la información. El algoritmo más popular es *PageRank* de Google que toma en consideración la conexión entre los documentos.

3. Las ontologías son la principal herramienta de la Web Semántica para dotarla de significado, la representación del contexto y sus relaciones a través de la formalidad permite que no sólo las personas, sino también agentes inteligentes, sean capaces de analizar e inferir conocimiento. En las búsquedas inteligentes juegan un papel fundamental ya que son el elemento que dota de significado y contexto a las búsquedas.

4. Desde el punto de vista técnico la búsqueda es un subproceso de la recuperación de información, en donde se compara la representación del documento con la representación de la consulta, según sea el modelo de recuperación. Sin embargo, la búsqueda también puede verse desde el

punto de vista del usuario, como una tarea sistemática para encontrar información específica.

5. Las búsquedas inteligentes proveen la capacidad de obtener información de las relaciones existentes entre los individuos de las ontologías, esas relaciones permiten contextualizar y devolver no sólo resultados exactos sino también relacionados, para ello el lenguaje ontológico es fundamental.

6. Los motores de búsqueda web basados en indexación y *pageRank* no desaparecerán, sino que, se beneficiaran contextualizando los índices. El sistema construido se basó en la creación de un índice el cual es el punto de entrada de la búsqueda de donde se desprendían los resultados relacionados en las propiedades de la ontología.

RECOMENDACIONES

1. El crecimiento de la web es mucho más extensa y más dinámica que una base de conocimiento basada en ontologías, es por ello que es necesario buscar los mecanismos para automatizar éste proceso, de tal cuenta, la aplicación actual de las búsquedas inteligentes se realizan sobre sistemas de gestión de conocimiento basados en ontologías de un conjunto de instancias controladas.

2. Cuando se nombran las propiedades de la ontología es recomendable utilizar los verbos en idioma inglés:*has* e *is* (tiene y es). Esto permitirá a los agentes inteligentes comprender mejor las propiedades.

3. La construcción de sistemas de gestión de conocimiento basado en ontologías se ha facilitado gracias a *frameworks* que proveen de funciones específicas para ello. Sin embargo, el desarrollador debe poner especial énfasis en la construcción de la ontología ya que es el núcleo de sistema y las búsquedas inteligentes que se pretenden implementar.

BIBLIOGRAFÍA

1. ANDERSON, Simon P.; DE PALMA, André. *A theory of information overload*. Virginia, USA: University of Virginia, Department of Economics, 2005. 26 p.

2. BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. *Modern Information Retrieval*. Massachusetts: Addison-Wesley, 1999. 430 p.

3. _____.*Modern Information Retrieval: the concepts and technology behind Search*. 2a. ed. Massachusetts: Addison-Wesley, 2011. 944 p.

4. BERNERS-LEE, Tim. *Information management: a proposal*. [en línea] http://www.w3.org/History/1989/proposal.html. [Consulta: agosto de 2012].

5. _____; HENDLER, James; LASSILA, Ora. *The Semantic Web a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. [en línea] http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html. [Consulta: septiembre de 2012].

6. BRIN, Sergey; PAGE, Lawrence. *The anatomy of a large-scale hypertextual Web search engine*. [en línea] http://infolab.stanford.edu/~backrub/google.html. [Consulta: septiembre de 2012].

7. BUSH, Vannevar. *As we may think*. [en línea] http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/. [Consulta: agosto de 2012].

8. CHU, Heting. *Information representation and retrieval in the digital age*. New Jersey: ASIST Monograph Series, 2003. 25 p.

9. CRIADO FERNÁNDEZ, Luis. *Procedimiento semi-automático para transformar las Web en Web Semántica*. Madrid: UNED, 2009. 258 p.

10. DIRK, Lewandoski*. Web searching, search engines and Information Retrieval*. Düsseldorf, Germany: Department of Information Science, Heinrich-Heine-University Düsseldorf, 2005. 15 p.

11. DOMINICH, S. *A unified mathematical definition of classical information retrieval*. USA: Journal of the American Society for Information Science, 2000. 624 p.

12. EPIFANIO TULA, Luis; MEDEOT, Matías Daniel. *Sistema de Recuperación de Información - Motor de búsqueda: Innuendo*. Argentina: Universidad Tecnológica Nacional, Facultad Regional Córdoba, 2007. 5 p.

13. FENSEL, D. *Ontologies: silver bullet for knowledge management and electronic*. Berlin: Springer-Verlag, 2004. 99 p.

14. GALO, Daniel. *BlindLight: Una nueva técnica para procesamiento de texto no estructurado*. Oviedo, España: Universidad de Oviedo, 2005. 213 p.

15. GRAMAJO, Javier. *Knowledge bases system: automatic contextualization using a domain ontology*. [en línea] http://www.academia.edu/313708/Automatic_contextualization_using_a_domain_Ontology. [Consulta: julio de 2012].

16. _____. *State of the art: ontologies*. Guatemala: Engineering Faculty, Computer Since and Systems Department, Universidad de San Carlos de Guatemala, 2005. 18 p.

17. GULLI, A.; SIGNORINI, A. *The indexable Web is more than 11.5 billion pages*. Chiba, Japan: ACM, 2005. 2 p.

18. HENZINGER, Monika. *Link analysis in Web information retrieval*. Mountain View, California: Google Inc., 2000. 6 p.

19. KLEINBERG, Jon M. K. *Authoritative sources in a hyperlinked environment*. USA: Journal of the ACM, 1999. 632 p.

20. LAMARCA Lapuente, María Jesús. *Hipertexto: el nuevo concepto de documento en la cultura de la imagen*. [en línea] http://www.hipertexto.info/.[Consulta: 21 de 09 de 2012].

21. MACINTA, Tim. *Web Developer.com guide to search engines*. USA: Wiley, 1998. 200 p.

22. MAES, Patti. *Agents that reduce work and information*. USA: Magazine Communications of the ACM, 1994. 40 p.

23. NOY, Natalia F.; McGUINNESS, Deborah L. *Desarrollo de ontologías-101: guía para crear tu primera ontología*. Standford: Standford University, 2005. 29 p.

24. NUNES, Sérgio. *State of art in Web Information Retrieval*. Portugal: University of Porto, 2006. 29 p.

25. PEIS REDONDO, Eduardo; HERRERAVIEDMA, Enrique. *Ontologías, metadatos y agentes: recuperación "semántica" de la información*. España: Universidad de Granada, 2003. 10 p.

26. PLAZA, Valdés. *¿Crecimiento de las oportunidades o nuevas desigualdades? El verdadero "dilema digital", capitalismo y democracia: disyuntivas y dilemas*. Ciudad de México: UNAM, 2008. 10 p.

27. SALTON, Gerard. *A theory of indexing*. Philadelphia: Cornell University, 1975. 55 p.

28. _____; LESK, M. E. *The SMART Automatic Document Retrieval System – an illustration*. USA: Magazine Communications of the ACM, 1965. 400 p.

29. SANTESTEBAN, Cristian. *Acceso y recuperación de información en la World Wide Web*. Buenos Aires, Argentina: Universidad Nacional de Mar de Plata, 2001. 105 p.

30. SELBERG, E. *Towards comprehensive Web search*. Washington: University of Washington, 1999. 53 p.

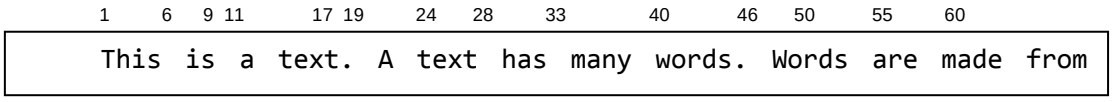
31. TAUBE, Mortimer; GULL, C. D.; WACHTEL, Irma S. *Unit terms in coordinate indexing*. USA: Journal of the American Society for Information Science and Technology, 1951. 218 p.

ANEXOS

Anexo 1: Archivo Invertido

Tomado de: BAEZA-YATES, Ricardo y RIBEIRO-NETO, Berthier. Modern Information Retrieval: The Concepts and Technology behind Search. s.l.: Addison-Wesley, 2011

Un archivo invertido (o índice invertido) es un mecanismo orientado a indizar un documento con el objetivo de mejorar la velocidad en la tarea de búsqueda. La estructura de un archivo invertido está compuesto por dos elementos: el *vocabulario* y las *ocurrencias*. El vocabulario es el conjunto de todas las distintas palabras en el texto. Para cada palabra una lista de todas las posiciones donde la palabra aparece es almacenada. El conjunto de todas estas listas es llamado las “ocurrencias” (Figura 8.1 muestra un ejemplo).



Vocabulario	Ocurrencias
letters	60…
made	50…
many	28…
text	11, 19
words	33, 40

Un texto de ejemplo y un archivo invertido construido en base a éste. Las palabras son convertidas a minúsculas y algunas palabras no son indizadas. Las ocurrencias apuntan a la posición de los caracteres en el texto.

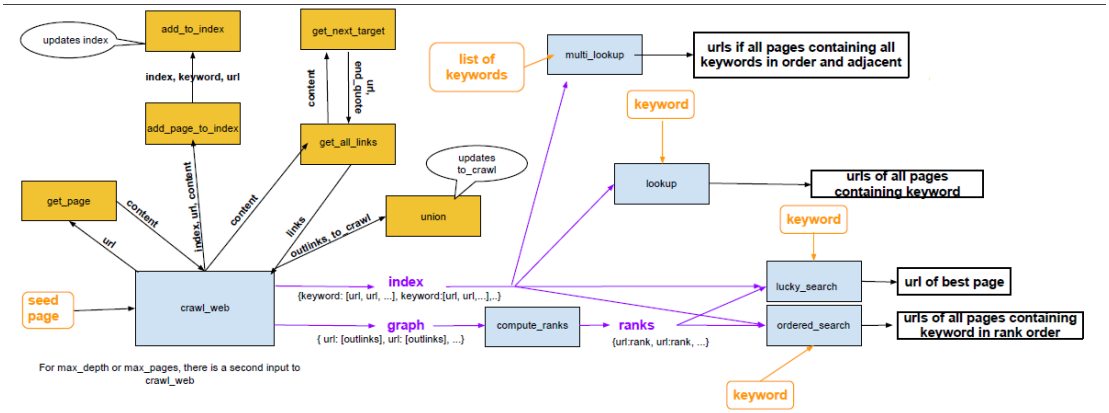
Algunos autores hacen la distinción entre archivos invertidos y listas invertidas. En un archivo invertido, cada elemento de la lista apunta a un documento o nombre de archivo, mientras una lista invertida coincide con nuestra definición. Preferimos no hacer tal distinción porque, como veremos después, este es una materia de *addressing granularity*.

Anexo 2: Implementación de un motor de búsqueda simple

Con información de: www.udacity.com, *Building a Search Engine*, Abril 2012

En abril de 2012 el proyecto *Udacity* liderada por Sebastian Thrun, impartió en línea un curso llamado “Construyendo un motor de búsqueda” del cual se desprende el código que precede. El lenguaje de programación utilizado es *Python 2.6*. La implementación es simple pero provee de los elementos y conceptos necesarios para comprender los motores de búsqueda, en especial, *Google*, ya que *Sebastian Thrun* trabaja para este motor de búsqueda. Es más, el método de ranking es *PageRank* El código es totalmente funcional.

Figura 1 - Vista general de la implementación del motor de búsqueda.



Fuente: udacity.com

Implementación

```
1.  # Code from this file is taken from Udacity's CS101 course. (www.udacity.com)
2.  # Udacity provided this code under a CC BY-NC-SA 3.0 license
3.  # see http://creativecommons.org/licenses/by-nc-sa/3.0/ for more details
4.  # April 8th, 2012
5.
6.  # Number of links to be followed from the seed page
7.
8.  crawl_depth = 1
```

```
9.  split_list = " ,!-<>"
10.
11. def compute_ranks(graph): # implementación de PageRank
12.     d = 0.8 # damping factor
13.     numloops = 10
14.
15.     ranks = {}
16.     npages = len(graph)
17.     for page in graph:
18.         ranks[page] = 1.0 / npages
19.
20.     for i in range(0, numloops):
21.         newranks = {}
22.         for page in graph:
23.             newrank = (1 - d) / npages
24.             for node in graph:
25.                 if page in graph[node]:
26.                     newrank += d * (ranks[node] / len(graph[node]))
27.             newranks[page] = newrank
28.         ranks = newranks
29.     return ranks
30.
31.
32.
33. cache = {
34.     'http://udacity.com/cs101x/urank/index.html': """<html>
35. <body>
36. <h1>Dave's Cooking Algorithms</h1>
37. <p>
38. Here are my favorite recipies:
39. <ul>
40. <li><a href="http://udacity.com/cs101x/urank/hummus.html">Hummus Recipe</a>
41. <li><a href="http://udacity.com/cs101x/urank/arsenic.html">World's Best Hummus</a>
42. <li><a href="http://udacity.com/cs101x/urank/kathleen.html">Kathleen's Hummus Recipe</a>
43. </ul>
44.
45. For more expert opinions, check out the
46. <a href="http://udacity.com/cs101x/urank/nickel.html">Nickel Chef</a>
47. and <a href="http://udacity.com/cs101x/urank/zinc.html">Zinc Chef</a>.
48. </body>
49. </html>
50.
51.
52.
53.
54.
55.
56. """,
57.     'http://udacity.com/cs101x/urank/zinc.html': """<html>
58. <body>
```

```
<h1>The Zinc Chef</h1>
<p>
I learned everything I know from
<a href="http://udacity.com/cs101x/urank/nickel.html">the Nickel Chef</a>.
</p>
<p>
For great hummus, try
<a href="http://udacity.com/cs101x/urank/arsenic.html">this recipe</a>.

</body>
</html>




""",
    'http://udacity.com/cs101x/urank/nickel.html': """<html>
<body>
<h1>The Nickel Chef</h1>
<p>
This is the
<a href="http://udacity.com/cs101x/urank/kathleen.html">
best Hummus recipe!
</a>

</body>
</html>




""",
    'http://udacity.com/cs101x/urank/kathleen.html': """<html>
<body>
<h1>
Kathleen's Hummus Recipe
</h1>
<p>

<ol>
<li> Open a can of garbonzo beans.
<li> Crush them in a blender.
<li> Add 3 tablesppons of tahini sauce.
<li> Squeeze in one lemon.
<li> Add salt, pepper, and buttercream frosting to taste.
</ol>
```

```
109.
110. </body>
111. </html>
112.
113. """,
114.    'http://udacity.com/cs101x/urank/arsenic.html': """<html>
115. <body>
116. <h1>
117. The Arsenic Chef's World Famous Hummus Recipe
118. </h1>
119. <p>
120.
121. <ol>
122. <li> Kidnap the <a href="http://udacity.com/cs101x/urank/nickel.html">Nickel Chef</a>.
123. <li> Force her to make hummus for you.
124. </ol>
125.
126. </body>
127. </html>
128.
129. """,
130.    'http://udacity.com/cs101x/urank/hummus.html': """<html>
131. <body>
132. <h1>
133. Hummus Recipe
134. </h1>
135. <p>
136.
137. <ol>
138. <li> Go to the store and buy a container of hummus.
139. <li> Open it.
140. </ol>
141.
142. </body>
143. </html>
144.
145.
146.
147.
148. """,
149. }
150.
151. def crawl_web(seed, max_depth = crawl_depth): # returns index, graph of inlinks
152.     tocrawl = [[seed, 0]]
153.     crawled = []
154.     graph = {}  # <url>, [list of pages it links to]
155.     index = {}
156.     while tocrawl:
157.         page, depth = tocrawl.pop()
158.         if page not in crawled and depth <= max_depth:
```

```
            content = get_page(page)
            add_page_to_index(index, page, content)
            outlinks = get_all_links(content)
            graph[page] = outlinks
            for link in outlinks:
                tocrawl.append([link, depth + 1])
            crawled.append(page)
    return index, graph


def get_page(url):
    if url in cache:
        return cache[url]
    try:
        import urllib
        return urllib.urlopen(url).read()
    except:
        return ""


def get_next_target(page):
    start_link = page.find('<a href=')
    if start_link == -1:
        return None, 0
    start_quote = page.find('"', start_link)
    end_quote = page.find('"', start_quote + 1)
    url = page[start_quote + 1:end_quote]
    return url, end_quote

def get_all_links(page):
    links = []
    while True:
        url, endpos = get_next_target(page)
        if url:
            links.append(url)
            page = page[endpos:]
        else:
            break
    return links


def union(a, b):
    for e in b:
        if e not in a:
            a.append(e)

def split_string(source, splitlist = split_list):
    output = []
    atsplit = True
    for char in source:
```

```
        if char in splitlist:
            atsplit = True
        elif atsplit:
            output.append(char)
            atsplit = False
        else:
            output[-1] += char
    return output

def add_page_to_index(index, url, content):
    words = split_string(content)
    for word in words:
        add_to_index(index, word, url)

def add_to_index(index, keyword, url):
    if keyword in index:
        index[keyword].append(url)
    else:
        index[keyword] = [url]

def lookup(index, keyword):
    if keyword in index:
        return index[keyword]
    else:
        return None

def lucky_search(index, ranks, keyword):
    pages = lookup(index, keyword)
    if not pages:
        return None
    best_page = pages[0]
    for candidate in pages:
        if ranks[candidate] > ranks[best_page]:
            best_page = candidate
    return best_page

index, graph = crawl_web('http://udacity.com/cs101x/urank/index.html')
ranks = compute_ranks(graph)
```