UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Identificación de entidades relacionadas mediante extracción de recursos RDF.

TRABAJO DE FIN DE TITULACIÓN.

AUTOR: Malla Pacheco, Pablo Estuardo.

DIRECTOR: Chicaiza Espinosa, Janneth Alexandra, Ing.

LOJA – ECUADOR

2015

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

Ingeniera.

Janneth Alexandra Chicaiza Espinosa.

DIRECTORA DEL TRABAJO DE FIN DE TITULACIÓN

De mi consideración:

Que el presente trabajo, denominado: **"Identificación de entidades relacionadas mediante el análisis de grafos RDF"** realizado por el profesional en formación: **Pablo Estuardo Malla Pacheco**; cumple con los requisitos establecidos en las normas generales para la Graduación en la Universidad Técnica Particular de Loja, tanto en el aspecto de forma como de contenido, por lo cual me permito autorizar su presentación para los fines pertinentes.

Loja, Abril de 2015

f)…….....

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

“Yo Pablo Estuardo Malla Pacheco, declaro ser autor del presente trabajo de fin de Titulación: “Identificación de entidades relacionadas mediante el análisis de grafos RDF” de la Titulación de Ingeniero en Sistemas Informáticos y Computación, siendo la Ing. Janneth Alexandra Chicaiza Espinosa directora del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad”.

F.....

Autor: Pablo Estuardo Malla Pacheco.

Cédula: 1104619620

DEDICATORIA

A mis padres Oswaldo José y Rosita Pacheco y a mí hermano Oswaldo David, quienes me han ayudado y acompañado en todo momento y circunstancias con su comprensión y apoyo.

AGRADECIMIENTO

A la Universidad Técnica Particular de Loja donde me formé, en el campo intelectual, práctico, investigativo, valores éticos, morales, espiritual, aspectos trascendentales que marcaron el rumbo de mi existencia hacia una visión abierta a nuevas posibilidades, exigencias y a un eterno estudio.

A los docentes, catedráticos de la Escuela de Ciencias de la Computación que me impartieron los conocimientos en las aulas de estudio con su guía constante en toda mi formación.

A la Ing. Janneth Chicaiza Espinosa; Directora de la presente tesis, que bajo su dirección y que con conocimiento de causa me orientó en la investigación, ejecución, para el desarrollo del trabajo de tesis.

INDICE DE CONTENIDOS

RESUMEN

En la actualidad impera la necesidad de generar información concreta, y debido al crecimiento vertiginoso de la información en internet resulta cada vez más complicado seleccionar entre recursos útiles y recursos que están fuera de un contexto de búsqueda o presentados de manera ambigua, de aquí nace la importancia de utilizar procesos más eficaces de descubrimiento y selección de recursos, en base al conocimiento de su relación existente con otros elementos llegar a recuperar información relacionada.

En este trabajo se han creado servicios Web independientes que finalmente son integrados como parte de un flujo de trabajo secuencial, un servicio inicial de pre-procesamiento de texto, es decir, preparar los tags y keywords; un siguiente servicio se encarga de mapear el texto del usuario a la URI o (URI's) de uno o varios recursos semánticos, un tercer servicio, realiza el recorrido iterativo alrededor de estos nodos; un proceso de ranking se encarga de ordenar los resultados. Finalmente, un servicio de visualización presenta gráficamente la red de conceptos relacionados.

PALABRAS CLAVES: keywords, entidad, extracción, tag, desambiguación, relación semántica, DBPedia, grafo.

ABSTRACT

Currently prevails the need to generate specific information, and because of the rapid growth of information on the Internet is becoming increasingly difficult to choose between useful resources and resources outside of the context of seeking or presented ambiguously, born here importance of using more effective resource discovery and selection process, based on the knowledge of its relationship with other elements come to retrieve related information.

In this work we have created separate Web services that are eventually integrated as part of a sequential workflow, an initial pre-service processing, ie prepare tags and keywords; a subsequent service is responsible for mapping the user text or the URI (URI's) of one or more semantic resources, a third service, performs the iterative path around these nodes; ranking process is responsible for ordering the results. Finally, a service display graphically shows the network of related concepts.

KEYWORDS: keywords, entity extraction, tag, disambiguation, semantic relationship, DBPedia, graph.

INTRODUCCIÓN

Al utilizar un recurso podemos encontrar valiosos elementos asociados a él, las relaciones entre elementos pueden tener una naturaleza jerárquica, por ejemplo, el estudio e investigación de bioinformática, pueden encontrarse bajo el domino correspondiente a una especialidad informática, es decir, las relaciones concretas entre conjuntos de entidades (keywords o tags) permitirán encontrar un conjunto de elementos cercanos, gracias al universo de relaciones que rodea a una entidad.

La importancia de esta investigación a nivel de usuario es que al recorrer la estructura de cada entidad en base a su red de relaciones podemos utilizar las características de cada entidad y llegar a crear recomendaciones de elementos específicos; de esta manera, una persona podrá aprovechar de mejor manera un recurso al obtener un paquete de resultados específicos sobre un tema en concreto.

El propósito del proyecto es desarrollar componentes que permitan establecer un correcto flujo de trabajo para explorar estructuras semánticas (recursos), identificando las características de las relaciones entre los recursos y al descubrir los elementos específicos relacionados permitir presentar al usuario final un conjunto de resultados concretos.

Un proceso de sugerencia de tópicos, ha sido implementado mediante un conjunto de servicios Web, de tal manera que se pueda reutilizar cada componente en requerimientos futuros o en servicios externos.

El primer paso ha sido escribir el Estado del Arte que corresponde al capítulo I; básicamente se han estudiado las tecnologías actuales disponibles, conceptos generales de la estructura de los recursos y las características del uso de este tipo de elementos, así como el contexto de trabajo que encierra trabajar con entidades relacionadas mediante Linked Data y el impacto que tiene DBPedia como principal repositorio de trabajo.

El capítulo II corresponde el desarrollo del proceso de preparación de las entidades para su posterior aplicación en el proceso de recorrido iterativo de entidades; se indican los métodos que se han considerado para el procesado de texto; además, se explica cómo se ha diseñado el componente de pre-procesamiento de texto, encargado de procesar las palabras, corregir su estructura si es necesario mediante corrección ortográfica, técnicas de stemming y agrupamiento mediante análisis de similaridad.

El capítulo III describe el componente de análisis, proceso diseñado para determinar patrones de relación entre nodos; este componente recibe un conjunto de recursos pre-procesados e inicia el proceso de búsqueda de relaciones del conjunto de keywords recibidos para generar un paquete de URI's relacionadas en un primer nivel.

Cuando se extrae un primer nivel de relaciones, posteriormente se recorre y crea un nuevo árbol de relaciones más extenso, es donde interviene el componente de (recorrido), el cual extrae un nuevo conjunto de elementos relacionados más concreto en base a las URI's extraídas del componente anterior.

El capítulo IV se centra en la construcción de una aplicación Web, la cual integra todos los servicios creados y combina los resultados generados por cada servicio individual. Este mecanismo interactivo permite al usuario retroalimentar al sistema, de manera que se generen recomendaciones más precisas y útiles.

CAPITULO I
ESTADO DEL ARTE Y ESTUDIO DE TECNOLOGÍAS PARA ANÁLISIS DE ENTIDADES

1.1 Resumen.

El proceso de extracción de información se refiere a la actividad de encontrar e identificar automáticamente elementos que comparten similitudes o relaciones o que cumplen un patrón. Dos enfoques, pueden ser utilizados para cumplir este propósito: 1) aproximaciones basadas en aprendizaje automático según Pedraza, Rafael (2010), y 2) detección de elementos que cumplen una regla, un algoritmo o un patrón según Zubair, M. (2007).

En el artículo Lluis Codina, Mari (2009), nos explica que al considerar la implementación de la web semántica frente al estado actual de la web implica una nueva forma de trabajar y tratar los datos, debido a que los resultados que se esperan recuperar exigen un nivel de contexto exacto y único.

En este capítulo se describen los elementos que fundamentan el trabajo con entidades, su estructura y atributos, así como las ventajas de su uso. La Web Semántica y Linked Data, definen el marco de trabajo adoptado en este proyecto.

La idea detrás de Linked Data es poder mejorar la integración de datos y explotar las relaciones semánticas con diferentes fines.

Al extender información disponible en la Web se puede lograr que cada entidad o elemento publicado

El objetivo es extender la información disponible en Web, permitir que cada entidad o elemento publicado, tanto como conjunto de información o tripletas (RDF), tenga establecido vínculos hacia otros elementos, lo que fortalece la incidencia de datos en la Web.

Uno de los mayores retos de la Web Semántica es permitir que los contenidos (entidades y recursos), se encuentren dotados de semántica, y gracias a la vinculación relacional entre elementos, se puede lograr que los agentes o aplicativos que acceden a ellos sean capaces de deducir o de inferir conocimiento.

Cómo nos explica Berners-Lee (s, f), en la actualidad se puede considerar a la semántica como una de las características clave en la fase de reconocimiento de entidades o recursos, ya que mediante la semántica podemos saber en dónde reside el potencial y valor de un elemento, y mediante la manipulación de los elementos

representados semánticamente es posible establecer nuevos servicios que no existen en la actualidad o no están completamente explotados.

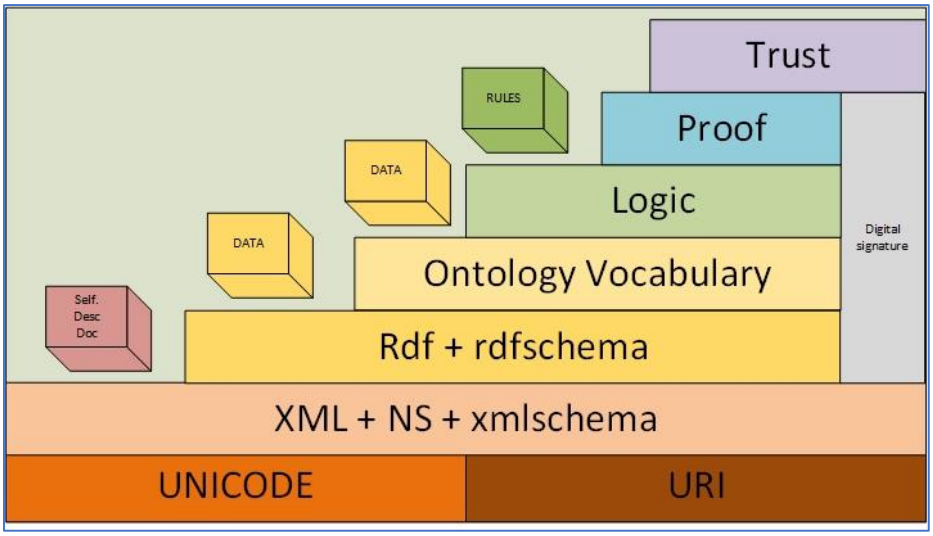
El estudio de herramientas como Gremlin[1] o Cypher[2] permitió conocer los fundamentos del uso de entidades bajo estructura de árbol de nodos, la forma de recorrer y manipular cada elemento descubierto.

El estudio de herramientas iniciales como Gremlin permitió descartar algunos elementos de recorrido de grafos pero se aprovechó los conceptos y los fundamentos de dichas herramientas para implementarlas en los servicios definidos en este proyecto.

Mediante el uso de URI's-HTTP es posible referenciar todo tipo de elemento como personas, agentes, recursos y facilitar la descripción de los elementos de un universo de información en particular.

1.2 Linked Data y su impacto en la generación de conocimiento.

Como el creador del concepto Tim Berners-Lee (2006), explica: "La web semántica no es una web separada sino una extensión de la actual, donde la información está dotada de un significado bien definido, los ordenadores están mejor capacitados y las personas trabajan en colaboración".



Arquitectura de la web semántica
Fuente: Tim Berners-Lee (2006).

[1] http://sql2gremlin.com/
[2] http://neo4j.com/developer/guide-sql-to-cypher/

El fundamento principal de la implementación de la web semántica es permitir evolucionar la web actual, de tal manera que los datos sean legibles para las máquinas, y sean interoperables entre distintos repositorios.

En la Figura 1, se presenta la arquitectura tecnológica que ofrece la Web Semántica. En los niveles iniciales constan las tecnologías base que permiten su desarrollo, tanto RDF, XML, URI's o Unicode nos permiten adaptar los recursos a una forma global que permite la compartición y reutilización de datos, documentos, entidades y tags.

En la tercera y cuarta capa se encuentran las tecnologías que permiten definir el modelo de datos, según el cual deben ser descritos los recursos.

Cuando nos enfocamos en el uso de una red de datos que se encuentran vinculadas de forma manual, nos encontramos con el proyecto Linked Data [3]que nos permite solventar el problema de relaciones dándonos la oportunidad de recorrer, manipular y visualizar con facilidad las ontologías o las estructuras de datos que se encuentran vinculados entre sí.

Mediante Linked Data podemos establecer una conexión a la información mediante los conjuntos de datos presentes en la Web que antes no se encontraban vinculados, o también permitir vincular dichos conjuntos de datos con los diversos métodos ya existentes como es la extracción de entidades, la búsqueda de relaciones, la visualización de la información.

Se ha considerado el uso de Linked Data y Web Semántica debido a que uno de los pilares fundamentales del trabajo con recursos son las entidades, según (Codina, 06), basándonos en entidades o recursos presentes en lenguajes como RDF Schema u OWL se puede lograr definir de manera formal conceptualizaciones que describan un dominio en específico, por ejemplo (recursos educativos, personas, lugares).

Al lograr obtener recursos relacionados en base a tags o keywords lograremos cumplir con una necesidad del siguiente componente principal que son las ontologías, al proporcionar elementos extraídos y previamente analizados podemos permitir que las ontologías representen y asocien dichos recursos en la Web.

Un concepto a considerar es que la Web de datos enlazados se puede convertir en un mecanismo para hacer que la información se encuentre públicamente disponible

[3] http://linkeddata.org/

mediante la utilización del protocolo HTTP, por tal motivo hemos considerado de igual manera aplicar DBPedia como fuente de recursos.

Las entidades relacionadas y su impacto en la Web Semántica han logrado establecerse como el camino a seguir para el tratamiento y diagnóstico global de la existencia de información en internet. Como explica O'Reilly (2005) en What Is Web 2.0: "los datos son el nuevo Intel inside', y en la publicación de Heath y Bizer (2011) hablan del 'data deluge' y nos explica que: "La World Wide Web ha revolucionado la forma en que conectamos y consumimos documentos, ahora está revolucionando la forma en que descubrimos, accedemos e integramos y usamos datos".

Linked Data tiene un impacto en el proceso de extracción de entidades relacionadas debido a que la fuente de este proceso son los metadatos (información/datos) que describen la estructura de los documentos o recursos a los que están asociados y de esta manera mostrar de una manera explícita el significado de las entidades.

Para la descripción de recursos se aplica el modelo RDF[4] (Resource Description Framework), basado en patrones de tripletas: sujeto-predicado-objeto. El sujeto se considera como todo aquello que se está describiendo, el predicado representa la propiedad o la relación que se establece en el recurso y finalmente el objeto establece el valor de la propiedad o un nuevo recurso con el que se establece una relación.

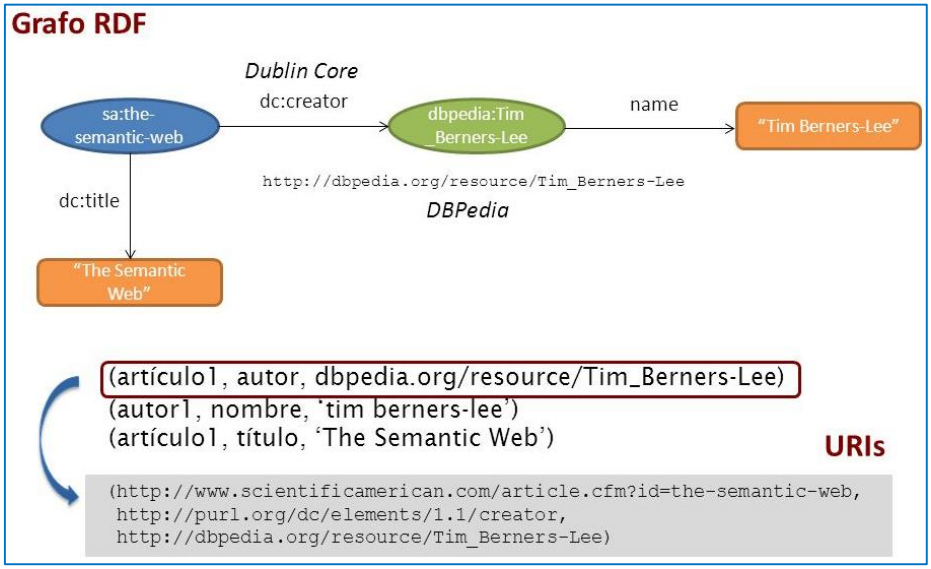


Figura 2. Grafo RDF – Enfoque de nodo
Fuente: Leyendo-y-escribiendo RDF y XML

[4] http://www.w3.org/RDF/

La figura 2, nos indica la estructura base que puede tener un recurso RDF cuando se lo considera como grafo, un nodo puede contener y representar su contenido como un rdf y formar parte de un árbol de grafos, contiene internamente la información en rdf y se beneficia externamente del igual manejo y las reglas que aplican el manipular y trabajar mediante grafos.

Marta Sabou, (2012) explica que Linked Data propone el seguimiento de algunos lineamientos de W3C[5] y entre los cuales define que lo principal es vincular datos distribuidos en la Web y de esta manera se está contribuyendo a la web semántica con el proceso de la publicación de los datos y la oportunidad de que se puedan vincular unos datos con otros para enriquecerse, así como vincularse con otros datos y permitir una mejor exploración para las personas o las máquinas mediante la aplicación de descripciones estándares en RDF.

Linked Data nos da directrices de trabajo para publicar datos enlazados en la Web, según Byrne, Hearth y Berners-Lee (2009) estas directrices son:

- Utilización de URI's para nombrar elementos
- Utilización de URI's des referenciales para que la gente pueda interpretar los nombres.
- Al buscar una URI, devolver información útil, respetando los estándares apropiados (RDF para describir recursos y SPARQL para acceder y consultar datos).
- Incluir y relacionar URI's de tal manera que se pueda recuperar información.

Cuando se analiza Linked Data llegamos a la conclusión de que el aporte a Big Data está determinado por un mayor nivel formal y al momento de estructurar los datos podemos mejorar la organización de todo el conocimiento modelado, esto se logra mediante técnicas de vinculación de datos y sus componentes.

Según explica (Christian Bizer, Freie Universität Berlin, 2009), en la actualidad explorar la Web Semántica se ha establecido en una necesidad cada vez mayor debido al gran crecimiento en el número de datos, la amplitud y calidad de información que puede ser recuperada a lo largo de las ontologías existentes, aunque aún dependen de la calidad de relaciones entre ellas.

[5] http://www.w3c.es/

1.3 DBPedia.

Se ha considerado DBPedia[6] como elemento central de nuestro trabajo debido a que permite el acceso a descripciones de todo tipo de recurso. Una característica fundamental de la selección de DBPedia es que permite el acceso a los datos creados colaborativamente y publicados en la Wikipedia; por tanto, se trata de un dataset de datos enlazados que se mantiene dinámicamente actualizado. Cada recurso en DBPedia está descrito al menos por una etiqueta, un resumen extenso o corto y un enlace directo a su página en Wikipedia.

DBpedia aprovecha la existencia de información ya estructurada en Wikipedia, como son infoboxes, categorías, imágenes, coordenadas geográficas, así mismo puede hacer uso de hipervínculos: tanto a otras webs como a traducciones del mismo contenido dentro de la propia Wikipedia., y toda esta información está almacenada en los DUMPS publicados por Wikipedia.

DBpedia aprovecha la existencia de información ya estructurada en Wikipedia: Infoboxes, categorías, imágenes, coordenadas geográficas, así mismo puede hacer uso de hipervínculos: tanto a otras webs como a traducciones del mismo contenido dentro de la propia Wikipedia., y toda esta información está almacenada en los DUMPS publicados por Wikipedia.

La naturaleza colaborativa de Wikipedia con DBPedia, facilita el acceso y selección de recursos de potencial valor, recursos que pueden ser aprovechados para su extracción y posterior manipulación en procesos como visualización de sus redes de relación o la recomendación de tópicos relacionados basándonos en los vínculos entre sus entidades.

En la actualidad la edición de las entradas en Wikipedia esa abierto a cualquier persona, y estas pueden ser modificados o editados sin algún tipo de restricción, sin embargo, existen moderadores que realizan la función de proteger algunas páginas de la modificación directa para evitar el “vandalismo de la información”, lenguaje o falsedades.

Para mantener la fiabilidad de la información presente en Wikipedia, existe una comunidad de editores voluntarios que se encargan de mantener la calidad y exactitud de las páginas a través del tiempo, y realizan esta actividad basados en el principio de:

[6] [6] http://wiki.dbpedia.org/OnlineAccess?v=17is

“sabiduría de las masas”, la cual se basa en que cada editor respalde los hechos que se redactan con referencias comprobables y fuentes autorizadas.

Un inconveniente que se puede presentar en la integridad de la información alojada en Wikipedia es la autopromoción que es entendida como un conflicto de intereses, de tal manera que al encontrarse elementos conocidos como Wiki spam estas entradas se eliminan rápidamente.

Una forma de detección de la manipulación malintencionada de la información en Wikipedia es el proceso de rastreo de una ubicación aproximada donde se presentan patrones sospechosos de conducta, y con cerca de 1300 administradores se puede llegar a encontrar una dirección IP de un infractor.

Debido a la estructura en la que se creó la Wikipedia, cada edición o incremento en la información se encuentra vagamente coordinada y su tamaño crece de manera simple, por tal motivo, la DBpedia ha generado un sin número de retos y requerimientos para solventar algunas de estas aplicaciones.

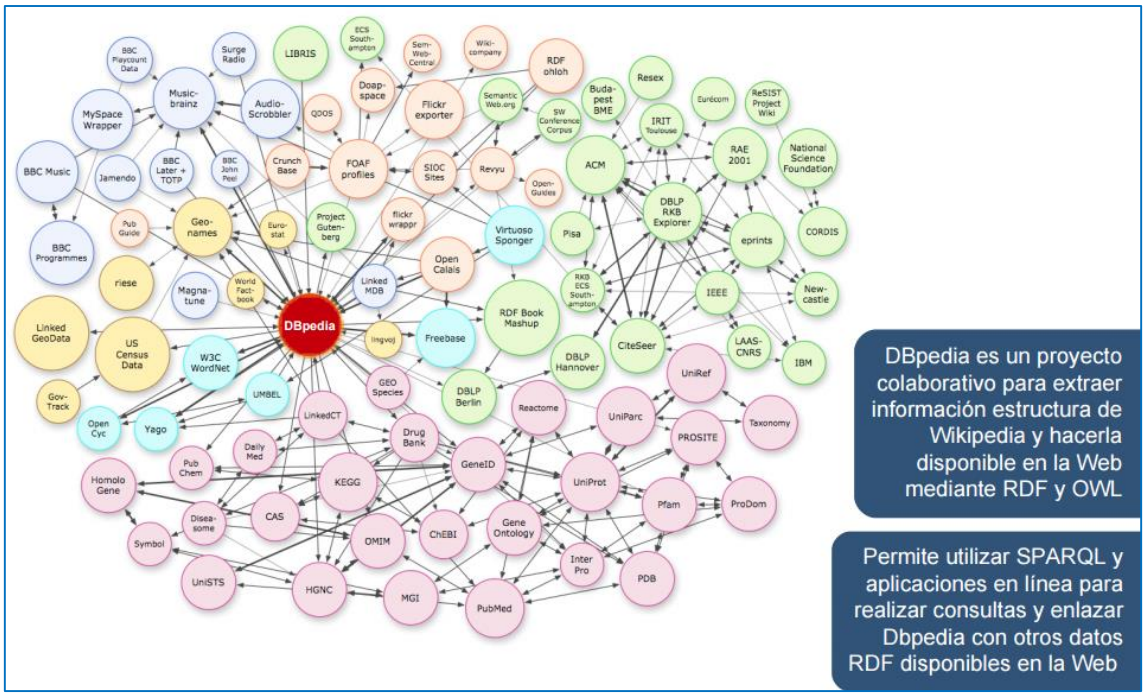


Figura 3. Proyecto DBPedia
Fuente: Juan Antonio Pastor Sánchez (pastor@um.es) - Semántica al alcance de todos

La estructura de la DBpedia se alimenta de la información estructurada de la Wikipedia, la cual es diversa y multidominio como está explicado en la figura 3, por lo tanto se requiere un vocabulario abierto y extensible que soporte múltiples propiedades y nuevos conceptos conforme estos sean detectados en la Wikipedia.

Según Miguel A. Martínez Prieto (2013), si comparamos las Taxonomías y otras jerarquías, DBPedia nos provee la gran ventaja de que cada elemento/recurso puede estar compuesto de una estructura descriptiva muy útil, incluyendo resúmenes, descripciones o vínculos a otras URI's.

Otra ventaja de la aplicación de DBPedia, comparada con jerárquicas estáticas, es que DBPedia va evolucionando conforme a los cambios de la Wikipedia, ésta evolución nos permite ir a la par con la generación del conocimiento, solventa de alguna forma el proceso manual de actualización, o la necesidad de implementar algún motor de crecimiento que exija mayores procesos de análisis de datos, antes incluso de ser extraídos para una recomendación, por lo que los problemas de extensiones de dominio o la comprensibilidad-máquina pueden ser abordados y superados con cierto grado de facilidad.

Para identificar y describir elementos o entidades del conjunto de datos de DBpedia es necesario hacerlo utilizando una URI como referencia, por ejemplo de la siguiente estructura: http: //dbpedia.org/resource/Name, en dónde nombre se extrae de la dirección URL del artículo de Wikipedia en dónde encontramos la forma: Wikipedia/Wiki/name.

Además, cada etiqueta, resumen o recurso en DBPedia puede estar almacenada en varios idiomas. Cada gráfica está conectada con otra variedad de conjunto de datos RDF en la nube de Linked Data.

Cada conjunto de datos de DBPedia está representado como una gran ontología multi-dominio y al utilizar RDF como un modelo de datos flexible hace que la representación de la información y el lenguaje de consulta SPARQL[7] sean más factible y fácil de implementar.

En Wikipedia se puede acceder a todo tipo de información en forma de texto libre, pero también se puede encontrar con información de manera estructurada como por ejemplo plantillas infobox, un conjunto de árboles y categorías, y enlaces a páginas web externas.

Como ya se ha explicado para poder identificar y describir las entidades podemos utilizar como referencia una URI.

[7] http://sparql.org/

Según (Pastor Sánchez, 2014), las clasificaciones en la DBpedia proporcionan varios esquemas de organización para cada entidad:

- Categorías de la Wikipedia las cuales están representados mediante el vocabulario SKOS[8].
- La clasificación: YAGO[9] se deriva de las categorías de Wikipedia usando WordNet.

Los enlaces de WordNet que se derivan de los infoboxes de Wikipedia y semánticamente son más precisos que los conjuntos de datos que derivan de las categorías de Wikipedia.

Según Javier D. Fernández, (2007), algunas características que pueden determinar que el uso de DBpedia presenta una dimensión factible para los procesos de investigación en la extracción de entidades son:

- El proyecto DBpedia utiliza el DBPedia Extraction Framework para obtener los datos estructurados desde la Wikipedia.
- La publicación de contenidos de Wikipedia como Big Data son multi-dominio y es abierto a su reutilización, además de ser una de las más extensas e importantes en la actualidad.
- Interrelación de DBpedia con otras colecciones facilita el uso de entidades que se desarrolla dentro del universo de Linked Open Data (LOD).
- Existen y se pueden generar módulos de acceso para SPARQL endpoints e interfaces para facilitar acceso a entidades o recursos.

Finalmente, DBPedia facilita la extracción de datos RDF de una manera automática. Cada información adicional que se encuentra inherente en el texto se puede extraer para mejorar aún más la descripción de un recurso que se requiera manipular.

Un requerimiento de la extracción de entidades relacionadas es conocer la mayor red de relaciones que se pueda generar y gracias a DBpedia es posible cubrir numerosos dominios y visualizar muchas instancias; además, los recursos descritos en DBPedia pueden llegar a conocer mejor una comunidad de entidades o área de conocimiento, es decir, los datos se pueden considerar como fuente real de un universo de conocimiento, y como explica (Marta Sabou, 2006), si utilizamos la DBpedia como la fuente de

[8] http://www.w3.org/2004/02/skos/
[9] http://en.wikipedia.org/wiki/YAGO_%28database%29

conocimientos principal podríamos proporcionar un tipo de mapeo intermediario para descubrir que relaciones se encuentran entre las clases de una ontología.

Otra forma de trabajar y explorar los datos en DBpedia para la extracción de entidades relacionadas es a través de enlaces externos ya que representan un gran ecosistema de información relacionada, entre algunas fuentes de información podemos encontrar:

- Páginas en HTML, enlaces a páginas web externas creadas como páginas de referencia, o como la página principal de una entidad, algunos ejemplos son:
    - 1 http://www.w3.org/TR/rdf-primer/
    - 2 http://www.w3.org/TR/rdf-sparql-query/
    - 3 http://www.w3.org/2004/02/skos/
- Enlaces RDF vinculados a datos externos, usando OWL y la propiedad sameAs; por ejemplo: los países pueden estar vinculados con las ontologías y autores Geonames1[10], Eurostat5[11] pueden estar vinculados a una ontología.

1.4 Incidencia de las relaciones semánticas en la manipulación de datos.

Con el proceso de interrelación de datos podemos llegar a conocer y descubrir nueva información y el conocimiento mediante la explotación de los datos enlazados, uno de los fines es por ejemplo la recomendación de entidades relacionadas.

En la actualidad ya existen guías e información que indiquen estándares y formas de diseñar y publicar datos enlazados en la Web, y es completamente necesario considerar a la web semántica para poder minimizar de alguna manera el impacto y dificultad al momento de integrar o de inter-operar información, datos o fuentes de extracción.

La existencia de estándares para su aplicación en vocabularios estructurados y controlados fomenta la correcta introducción de elementos a la web de datos gracias a que establecen la forma de modelación y presentación de datos.

Existen algunos formatos que van ligados a la publicación de nuevas normas sobre la elaboración modelos de datos como: BS 8723 XML Schema5 [12], que está vinculado a:

[10] http://www.geonames.org/ontology/documentation.html
[11] http://eurostat.linked-statistics.org/
[12] http://schemas.esd.org.uk/BS8723/

Structured vocabularies for information retrieval, o también, ISO 25964 XML Schema6 [13], que está vinculado a la ISO 25964: Thesauri and interoperability with other vocabularies, y la iniciativa Linked Data tiene bajo su respaldo la aplicación del estándar World Wide Web Consortium (W3C) como SKOS (Simple Knowledge Organization System).

Los vocabularios que controla Linked Data utiliza algunos estándares como el SKOS[14], según (Shiri, 2014), se explica que la aplicación de los principios Linked Open Data se introducen contenidos semánticos cuando se realiza un análisis de datos y cuando se los visualiza, de esta manera, bajo el respaldo de Linked Data, el modelo SKOS se está estableciendo como el modelo básico para representar sistemas de organización de conocimiento tales como las taxonomías, los tesauros, las listas de autoridad o las clasificaciones.

SKOS se presenta en forma de RDF, su función es proporcionar un modelo de representación de la estructura básica y también los contenidos de esquemas conceptuales tales como las taxonomías, los esquemas de clasificación, las listas de encabezamientos de materia, los tesauros y cualquier tipo de vocabulario controlado. En SKOS los conceptos son identificados mediante referencias a URI's.

Según (Manual de SKOS, 2009), se puede considerar un concepto como: "una noción o idea, una unidad de pensamiento", la cual puede llegar a ser representada unívocamente a través de una URI, y mediante este identificador se puede garantizar la identidad de dicho concepto en la Web semántica.

En el modelo SKOS se considera que los sujetos pueden tener tres clases: conceptos, esquemas de conceptos y colecciones, en donde los conceptos se consideran como elementos fundamentales de la estructura y estos conceptos tienen asignadas una etiqueta en uno o varios idiomas, por tal motivo, los conceptos pueden ser etiquetados en cadenas de texto y considerar uno o varios idiomas, documentar y ser estructurados a través de las relaciones semánticas de diferente tipología. (Web Semántica, diseño metodológico - SKOS-Core15).

[13] http://www.niso.org/schemas/iso25964/schema-intro/
[14] http://www.w3.org/2004/02/skos/
[15] http://www.w3.org/TR/swbp-skos-core-guide

Según explica (Miles y Bechhofer, 2009), existen 3 tipos de etiquetas consideradas en SKOS, los cuales son:

- Preferentes (skos: prefLabel): términos utilizados en la indexación. Su función es idéntica a la de los términos descriptores de los tesauros.
- Alternativas (skos: altLabel): utilizadas para representar términos sinónimos de los preferentes. Permiten enriquecer la diversidad léxica de un KOS y ofrecer múltiples puntos de acceso a un concepto que puede representarte con diferentes términos.
- Ocultas (skos: hiddenLabel): no visibles para los usuarios, pero si para las aplicaciones informáticas. Son útiles para el control de variantes terminológicas con errores ortográficos, diversas formas de acrónimos y abreviaturas, etc.

Existen propiedades que permiten vincular conceptos a través de la definición de relaciones semánticas, estas son:

- Jerárquica específica (skos: narrower): Esta relación que indica que existe un concepto cuyo significado es más específico que el concepto sobre el que se define la relación (por ejemplo: Botánica con Genética Vegetal).
- Jerárquica genérica (skos: broader): es la relación inversa de la anterior.
- Asociativa (skos: related): relación que indica que dos conceptos están relacionados semánticamente. (Por ejemplo: Botánica con Biología).

Los conceptos se asocian a un esquema de conceptos (skos: ConceptScheme) que generalmente identifica un único KOS. También es posible agrupar los conceptos en colecciones simples (skos: Collection) u ordenadas (skos: OrderedCollection).

SKOS también dispone de un conjunto de propiedades que permiten configurar redes entre diferentes KOS mediante relaciones de mapeado a establecer entre conceptos de diferentes esquemas:

- Equivalencia jerárquica específica (skos: narrowMatch): se utiliza cuando se desea indicar que un concepto de un esquema tiene un significado más específico que un concepto de otro esquema.
- Equivalencia jerárquica genérica (skos: broadMatch): es la relación inversa de la anterior.
- Equivalencia exacta (skos: exactMatch): cuando ambos conceptos tienen exactamente el mismo significado.

- Equivalencia asociativa (skos: relatedMatch): cuando dos conceptos de diferentes esquemas están relacionados semánticamente.
- Equivalencia cercana o próxima (skos: closeMatch): cuando ambos conceptos tienen un significado aproximado sin llegar a ser exacto.

Según explica (Miles y Bechhofer, 2009), SKOS define adicionalmente una serie de condiciones de consistencia y reglas - especialmente referidas a la transitividad de las relaciones semánticas, que delimitan el ámbito de aplicación de los vocabularios en procesos lógicos de inferencia.

El modelo que utiliza SKOS para los datos en realidad es una ontología que aplica OWL16, y al ser una derivación de este elemento puede ser usado conjuntamente con OWL para poder representar formalmente estructuras de conocimiento sobre un universo específico.

El proceso de establecer relaciones entre conceptos no se limita a un mismo vocabulario ya que también se pueden vincular diferentes estructuras de conocimiento en base a etiquetas, en donde cada elemento de datos presente en un documento o cualquier metadato con los que se describe una entidad puede ser enlazada a un valor de una estructura de conocimiento que esté bajo etiquetación con SKOS.

Un requerimiento indispensable en el manejo de entidades es que se pueda mapear conceptos en diferentes esquemas, de igual manera definir colecciones ordenadas o agrupar conceptos, con estas premisas al realizar un recorrido de una red de relaciones entre entidades y con la extracción de entidades poder establecer relaciones entre etiquetas asociadas a los conceptos.

Según (Hien T. Nguyen, 2010)el tema central del uso de una herramienta de extracción de entidades es el descubrimiento de mecánicas o funciones que nos permitan encontrar y descubrir las relaciones entre entidades o implementar un camino entre los nodos de un grafo semántico para poder suplir la necesidad de precisión en la información que se recupera cuando se realiza una búsqueda, aunque también existen otras aplicaciones a estas mecánicas como por ejemplo el enriquecimiento semántico de wordsets, o encontrar un camino para una inferencia lógica.

[16] http://en.wikipedia.org/wiki/Web_Ontology_Language

Existen algunas técnicas de mapeo para ontologías, el proceso de mapeo es muy necesario para posteriormente poder manipular y salvar la estructura semántica de los datos, ontologías o bases de datos que intervienen.

Una técnica eficaz para mapear ontologías es el realizar una comparación de sus topologías y reconocer que relaciones se encuentran duplicadas entre las clases.

Otro requerimiento en el proceso de extracción de entidades relacionadas es la de "limpieza" o preparación de los datos a utilizarse, aquí intervienen algunas mecánicas de procesado de datos, como por ejemplo la utilización de Stemming hacia los datos o entidades extraías para convertirlas en su forma normal o para poder categorizarlas y posteriormente mostrarlas como un recurso base del cual se derivan otros recursos similares.

Los datos que se encuentran se muestran como una extensa ontología de dominios múltiples y su base fundamental es el uso de Resource Description Framework (RDF) como un modelo de datos de representación flexible de información y de lenguaje de consulta SPARQL.

1.5 **Enfoque de grafos con RDF y búsqueda SQL.**

RDF demuestra que su formato puede ser usado para describir las relaciones que pueden existir entre entidades y el mundo, pero también puede ser aplicado en un nivel superior, por ejemplo para describir predicados RDF y clases o subtipos de recursos, ontologías, schemas y vocabularios.

Los estándares que han sido derivados de RDF como RDFS y OWL permiten que un formato como RDF agregue semántica para la elaboración de inferencias lógicas que se generen a partir de datos.

Cada modelo básico de RDF está compuesto de tres tipos de objetos:

- Recursos: Los componentes que pueden ser representados como una expresión RDF se puede considerar como recurso, y cada recurso puede ser una página web, puede ser una parte de aquella página web como un tag html o xml, o inclusive un id.
- Propiedades: Cada característica, atributo o relación que esté implícita en el proceso de descripción de un recurso. Las propiedades tienen una

denominación exclusiva que define los valores que puede tomar, el tipo de recurso que describen o su relación con otras propiedades.

- Sentencias: Cuando tenemos un recurso junto a una propiedad con un nombre más un valor de propiedad para e recurso denominamos a esa estructura como sentencia. Estas 3 partes serán denominadas como sujeto, predicado y objeto. Cada objeto representa el valor de una propiedad y puede ser otro recurso que puede llegar a estar especificado por una URI.

Si se compara XML con RDF nos damos cuenta que no está muy relacionado con el conocimiento debido a que los nodos de XML no necesitan estar vinculados con un concepto particular y el estándar de XML no permite discernir un concepto de un documento o universo de conocimiento, en cambio, si se trabaja mediante RDF, los nodos que no se entiendan de manera inicial pueden ser fácilmente procesados ya que RDF especifica algunos niveles de conocimiento.

Al utilizar los recursos URI's podemos encontrar y acceder a todo tipo de recurso debido a que una URI se maneja como un identificador de recurso más no solo un elemento que apunta a una dirección.

Cuando accedemos a un recurso mediante una URI podemos acceder a la información específica y útil que encierra ya que cada recurso está representado mediante descripciones y estándares basados en RDF y es requerido que un grupo de datos proporcione información relevante.

Otro aspecto clave del trabajo bajo el formato de RDF es que cuando se trabaja con información distribuida se puede facilitar su vinculación entre entidades, por ejemplo, se puede enlazar documentos o elementos que trabajen bajo vocabularios comunes o también permitiendo que un documento trabaje bajo su vocabulario específico, esto significa una gran flexibilidad de trabajo cuando se pretende manipular una amplia gama de cosas o de información que puede estar presente en varias fuentes.

En una tripleta, los nodos RDF, que son los sujetos y los objetos presentes en las sentencias, éstos se presentan bajo la etiqueta tipo: rdf: Description el cual encierra un atributo rdf: about que indica una URI del recurso al cual representa.

Cuando se pretende usar un recurso como una entidad o un objeto se puede aplicar dos formas:

1. Se puede utilizar el atributo rdf: resource, que es el que especifica la URI del objeto, y es posible describir propiedades que encierra el objeto en cualquier otro lugar.
2. Se puede utilizar la parte de rdf: Description dentro de un nodo de propiedad.

XML proporciona la base para la creación de otros lenguajes como RDF, RDFS e OWL. RDF es estándar para la publicación de datos en la Web, puede ser presentado bajo un modelo de datos basado en grafos, además RDF proporciona medios para describir recursos usando tripletas, proporciona medios para describir clases y sus relaciones.

RDF ha sido diseñado para representar conocimiento en un universo de datos distribuido, es decir, el diseño de RDF se basa en conocimiento y no para datos.

Los estándares que han sido construidos bajo el esquema de RDF incluyendo RDFS y OWL proveen a RDF de semántica que nos puede permitir llegar a realizar inferencias lógicas a partir de datos.

Al utilizar los recursos URI's podemos encontrar y acceder a todo tipo de recurso debido a que una URI se maneja como un identificador de recurso más no solo un elemento que apunta a una dirección.

Cuando accedemos a un recurso mediante una URI podemos acceder a la información específica y útil que encierra ya que cada recurso está representado mediante descripciones y estándares basados en RDF ya que es requerido que un grupo de datos proporcione información relevante.

El trabajo con URI's requiere un proceso iterativo, algunas de las herramientas de recorrido del árbol de nodos nos puede servir para revisar y manipular las conexiones, entre las principales se ha estudiado Gremlin[17] y Cypher[18] en el proceso de recorrido y manipulación de datos. [Ver - Anexo 3].

Al establecer que los recursos, tales como las entidades que se utilizarán para un proceso extractivo se manejarán bajo el formato de RDF, hay que establecer la forma en que estas entidades o elementos serán considerados y manipulados, y la manera más óptima de trabajar con entidades es manejándolas como un nodo.

[17] https://github.com/tinkerpop/gremlin/wiki
[18] http://neo4j.com/docs/stable/cypher-query-lang.html

La figura 4 indica un ejemplo de cómo podemos considerar un nodo y sus relaciones, puede ser como el usuario en el ejemplo de la red social, o la foto o un estado; las relaciones serían las acciones que pueden recaer sobre estos elementos como: “etiqueta” o “LikedStatus”, en el presente trabajo será considerado un nodo a una entidad o un recurso.

Las relaciones entre nodos que se van creando durante un proceso de recorrido nos permiten visualizar que conforme se va extendiendo un árbol de nodos, se descubre y se extiende más universos de conocimiento y de igual manera conocer el impacto que tiene un entidad sobre otras.

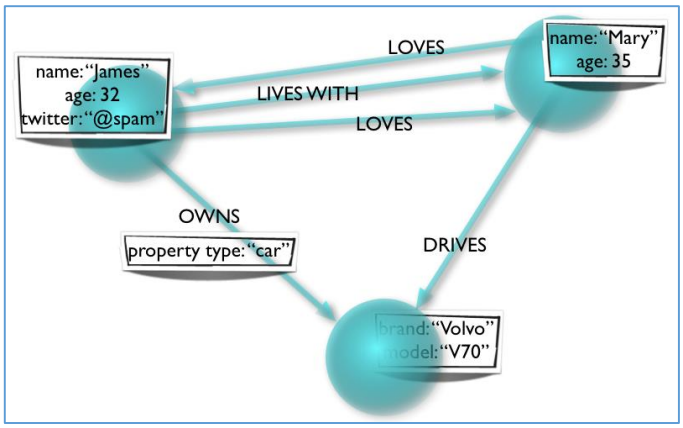


Figura 4: Relaciones entre nodos (2011) –
Fuente: [Top 5 Reasons to Get Your Graph On]

La Web Semántica brinda muchas ventajas en el momento de manipular entidades, recursos o elementos, y el trabajo bajo un enfoque de grafos para el trabajo con URI’s demuestra que se puede generar conocimiento de forma específica si sabemos explotar la estructura e información disponible en las entidades y si la organizamos y exploramos de una manera correcta y factible.

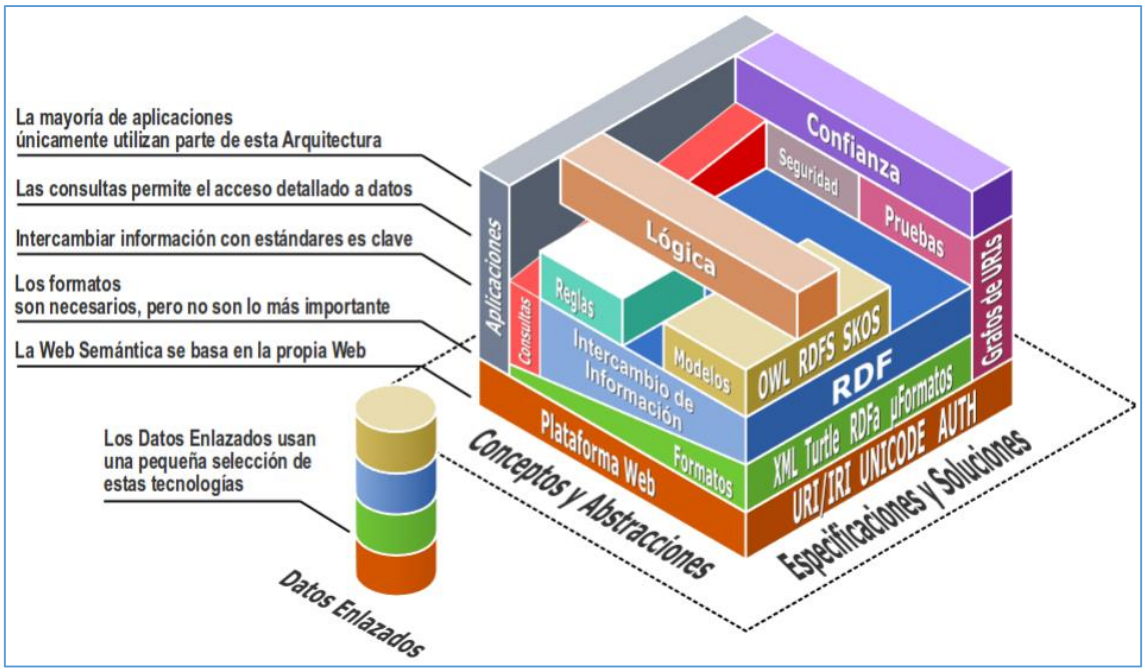


Figura 5. [Datos enlazados y sus posibles soluciones – Grafos de URI's]
Fuente: Arquitectura-tecnológica-de-la-web-semántica

Al considerar como nodos a las entidades que se van a aplicar en el descubrimiento de relaciones o vínculos entre recursos, se presenta algunas ventajas inherentes en el proceso extractivo al hacer esta consideración de trabajo.

En la figura 5 se puede visualizar los componentes globales que integran el análisis y uso de datos enlazados, los diferentes niveles representan la incidencia que tienen los datos sobre conceptos y abstracciones, y las especificaciones y soluciones nos indican los diferentes vocabularios, lenguajes y formatos que pueden ser aplicados a los datos, como la implementación de grafos o URI's para acceder o representar información.

El artículo: Emil Eifrem, CEO of Neo Technology (Top 5 Reasons to Get Your Graph On - 2012), considera algunas razones por las cuales la implementación del uso de recursos en forma de nodos o de árbol de relaciones, nos explica las ventajas de tratar a las entidades como una estructura de grafo.

- Rendimiento:

La aplicación de recursos basados en una estructura de nodos o árbol de relaciones permite optimizar el manejo de grandes volúmenes de datos, como por ejemplo, la manipulación de varios conjuntos de datos.

Las consultas pueden llegar a tardarse conforme los datos van aumentando de volumen o que involucren varias uniones lo que lo vuelve inviable para escenarios dónde se requiere una serie de consultas continuas, entornos OLTP (Transacciones en línea).

Si se trabaja con una estructura en forma de árbol de nodos no sufrimos de ese efecto de bajo rendimiento con grandes cantidades de datos como los que se presentan al trabajar con bases de datos relacionales.

En las bases de datos que trabajan bajo un modelo de grado, existen algunos casos dónde el uso de datos mejora el rendimiento en las peticiones debido a que la latencia es mejor comparado con el procesamiento por lotes tradicional.

La diferencia de rendimiento de una base de datos relacional puede llegar a presentar un deterioro en su rendimiento causado por consultas intensivas y un procesamiento de datos muy algo, sin embargo, una base de datos basado en grados ofrece un rendimiento que tiende a permanecer constante conjuntamente con el crecimiento de los datos, esto se debe a que las consultas se realizan de manera gráfica realizando un proceso de recorrido entre relaciones que posea la estructura de la base, de tal manera que el resultado en tiempo de ejecución de una consulta es proporcional al tamaño de la parte gráfica que tenga que recorrer para resolver una consulta, en lugar del recorrido total del tamaño del grafo.

La base de datos encuentra su punto de partida en el gráfico, y se acerca al resultado siguiendo relaciones a través de punteros.

Esto no sólo supone el obtener resultados con mayor rapidez sino también obtener características más específicas.

Ya no importa que cantidad de datos con los que se intente trabajar, como un grupo de tags o keywords, la misma consulta tendrá aproximadamente la misma cantidad de tiempo de respuesta, porque bajo este esquema el tiempo de consulta es proporcional al tamaño de aquella consulta más que el tamaño de los datos.

- Agilidad:

Una entidad trabajada en forma de nodo puede llegar a apoyar y participar en muchos sistemas y procesos de negocio durante su vida útil y por lo tanto, cada uno de estos datos tiene la tendencia a evolucionar. Un grafo permite tener una rica estructura de

datos sin limitaciones de esquema ya que éstos son extensibles y susceptibles a la evolución continua de datos.

Esta capacidad supera con creces al modelo relacional donde las migraciones de esquemas que son comunes afectan de manera significativa a las aplicaciones en las cuales estén involucradas.

Los procesos de negocio modelados mediante el uso de entidades en forma de nodos pueden ser obtenidos mediante re-ingeniería e incluso verificados formalmente con rapidez. Este beneficio de agilidad de negocio da la facultad a una organización de responder a un proceso de gestión de cambio de una manera estructurada, diligente y de manera rápida y repetidamente.

- Sencillez:

El modelar un dato en un nodo ya representa una simplicidad, y ésta característica viene del hecho de que los árboles de relaciones de datos pueden ser realizadas en una pizarra.

Una vez que desarrollamos nuestro modelo en una pizarra podemos estar seguros que ese mismo modelo va a persistir en un sistema, una base de datos o un proceso adaptativo, esto significa que la distancia semántica entre los expertos del dominio y los datos almacenados es muy corta.

Un modelo gráfico es accesible para usuarios regulares, si encontramos una falla tenemos la capacidad de despliegue y mediante una planificación podemos encontrar rutas que optimicen las redes de logística, podemos comprender y desarrollar un nuevo modelo, y podemos realizar mantenimiento y conservar la integridad estructurar de la gráfica y la infraestructura correspondiente.

- Fiabilidad:

Las estructuras modernas que trabajan entidades en forma de nodos ofrecen características fuertes para la construcción de sistemas confiables. Como ejemplo Neo4j es transaccional ACID y también soporta clustering de alta disponibilidad y rendimiento.

Estas características cierran la brecha que existe entre OLAP y OLTP ya que un dato generado o manipulado en forma de grafo es semánticamente rico y accesible no sólo

para sistemas en línea sino también para el funcionamiento de larga solución de trabajo de análisis, cuando conectamos simplemente esclavos OLAP a un clúster.

- Inclusión:

Si profundizamos en las razones de la implementación de entidades en estructura de nodo podemos darnos cuenta que podemos aprovechar la disciplina matemática muy rica en la teoría de grafos.

Un modelo gráfico está respaldado por siglos de trabajo en la teoría de grafos, matemáticas discretas y las disciplinas que se ocupan del análisis del comportamiento (sociología, psicología, antropología, etc.)

Esta estructura es semánticamente rica, los datos conectados pueden ser utilizados y ejecutados bajo algoritmos de grafos clásicos como: camino hallazgo, búsquedas por profundidad, búsquedas por amplitud.

Figura 6. – Relación y acceso existente entre Wikipedia y un recurso
Autor: Pablo Malla

También son susceptibles al análisis predictivo mediante técnicas de teoría de grafos, un ejemplo sería que dada una estructura organizacional mediante la teoría de grafos nos permite encontrar cuellos de botellas, estructuras inestables, y al encontrar estos problemas podemos aplicar algunos principios simples como: el cierre triádico, puentes locales, equilibrio, a nuestros datos.

La DBpedia debe su información de la Wikipedia, todo recurso que puede encontrarse en DBPedia deriva de esta fuente de información, como se muestra en la figura 6, todo el proceso de información se deriva de una fuente más extensa, completa y diversa.

La percepción predictiva bajo el uso de la teoría de grafos es particularmente sencilla de implementar una vez que los datos se almacenan y procesan.

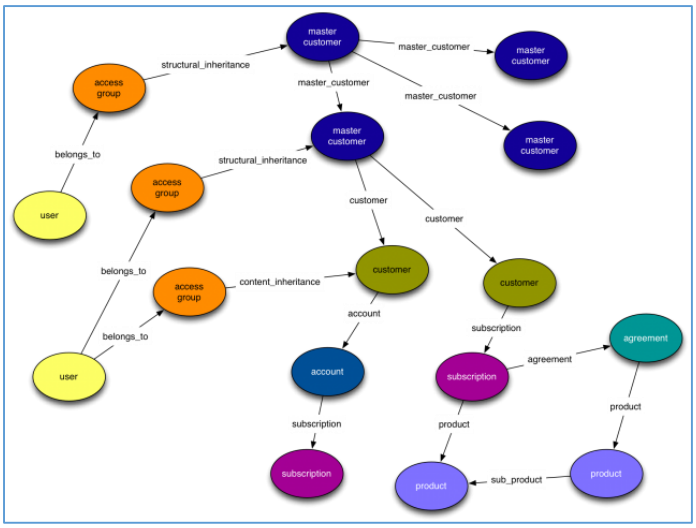


Figura 7: Interconexión de datos (Ejemplo: control de acceso productos y servicios) – Fuente: [Graphs as a New Way of Thinking]

El gráfico 7 representa lo que se explica en el artículo de Emil Eifrem, CEO, Neo Technology (Graphs as a New Way of Thinking JANUARY 9, 2013), “si queremos entender cómo los grafos actúan ante la complejidad de datos, es necesario entender la naturaleza de ésta complejidad, cada dato conforme va creciendo se vuelve cada vez más grande, más semi-estructurado y se conecta con más densidad.”

- Conectividad:

El valor resultante para el usuario final no es simplemente el resultado de aumento gradual de volumen y la variación de los datos, muchas de las veces se requiere hacer “preguntas” mucho más importantes y específicas que nos obliga a entender cómo nuestros datos están conectados entre sí y que resultado nos puede devolver.

Depende únicamente de nosotros comprender y estudiar las relaciones entre entidad y de igual manera conocer la calidad de éstas relaciones, y al considerar la maleabilidad de una estructura gráfica sabemos que podemos crear representaciones de un dominio semi-estructurado.

Una representación general de uso de entidades bajo los fundamentos de grafos se visualiza en la figura 8, en donde se puede ver que un árbol de relaciones íntegramente relacionado puede llegar a representar un dominio de conocimiento extenso, vinculando cada nodo y dando un sentido a la estructura completa obtenida al final.

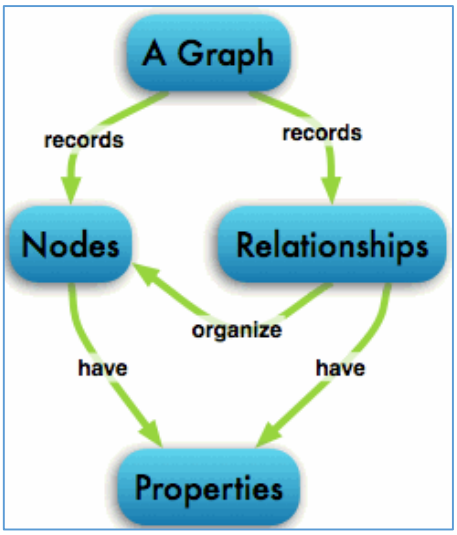


Figura 8. Estructura de nodos y sus relaciones
Fuente: What´s a Graph

La mayoría de las ontologías o bases de datos carecen de una extensión necesaria ya que se limitan a un dominio específico y son creadas por grupos pequeños que les cuesta su crecimiento o aporte frecuente de datos e información, por ello, la Wikipedia también se ha convertido en una de las fuentes más ricas y versátiles en cuanto a conocimiento disponible se refiere ya que es mantenida por miles de colaboradores. (Using Wikipedia entries as vocabulary for knowledge management. IEEE[19] Internet Computing, 11(5):54– 65,2007.)

La forma de leer un gráfico es el seguimiento de las flechas alrededor de un diagrama para formar oraciones, y si aplicamos esta característica a un conjunto de entidades que estén relacionadas podremos obtener información semántica o más precisa pues ya consideramos a un conjunto de nodos relacionados como una sola entidad.

El artículo - Query a Graph with a Traversal (What is a Graph Database? – 2012) nos describe el proceso en que las relaciones organizan las estructuras de los nodos y permite a cada gráfico convertirse en una lista, un árbol, un mapa o inclusive en una entidad compuesta, la cual puede posteriormente ser combinada y convertirse en estructuras mucho más complejas, y ricamente interconectadas.

[19] https://www.ieee.org/index.html

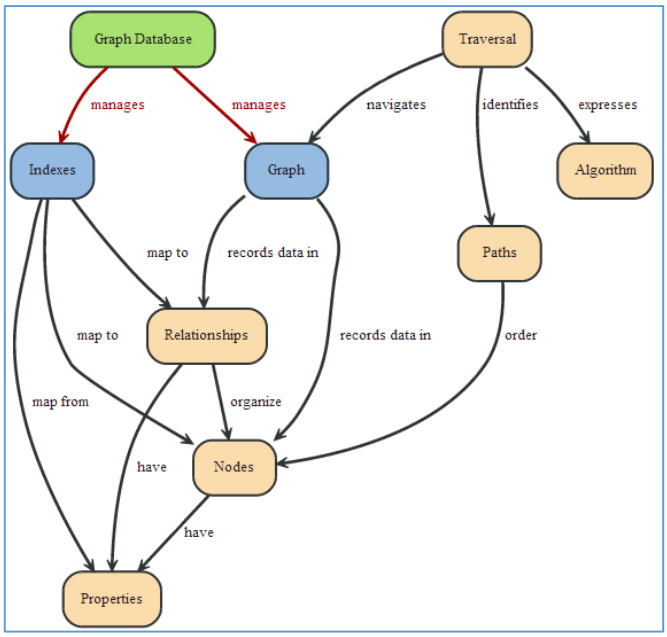


Figura 9: Estructura de una relación basada en grafos
Fuente: What´s a Graph

Al requerir que los datos con los que trabajos se encuentren interconectados convierte a la implementación del trabajo con entidades bajo un esquema de grafo como la mejor opción. Podemos reducir a nodos, propiedades y relaciones y asignarlas en una forma de pares clave/valor.

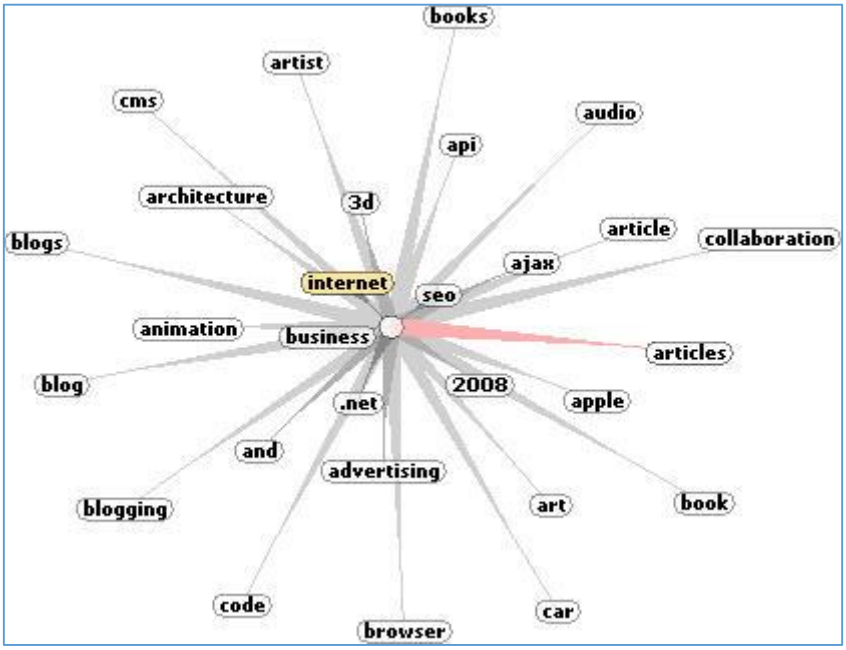


Figura 10: Estructura visual de un conjunto de entidades relacionadas en base a un tag
Fuente: Delicious tool tagas

Las relaciones semánticas entre conceptos y entidades son existentes tanto en documentos que se presentan de forma textual a través de palabras claves o llaves, frases, etiquetas que han sido generadas por un proceso de etiquetado, por lo tanto

podemos decir que el proceso de la extracción de una relación se refiere al poder de identificar y conocer estas relaciones presentes entre las entidades y los conceptos, y si logramos encontrar de forma automatizada estos vínculos podríamos añadir nuevas funcionalidades como la recomendación o la visualización de información de forma más factible.

El proceso de extracción de relaciones entre entidades que conforman un documento textual se orienta dentro de la extracción de relaciones tradicionales y estos recursos siempre están presentes en elementos que son estructurados como páginas web, documentos armados como una base de datos relacional.

Al centrarnos solo en la extracción de relaciones entre entidades que conforman un documento textual nos vemos inmiscuidos dentro de la extracción de relaciones tradicionales, y estos recursos siempre están presentes en elementos que son estructurados como páginas web,, documentos armados como una base de datos relacional.

El trabajo que aborda el presente tema de tesis es obtener conceptos o keywords relacionadas, se puede utilizar un conjunto de palabras para su pre procesado y en base a estas palabras obtener una red de recursos que se vinculan de alguna manera con cada uno de los elementos.

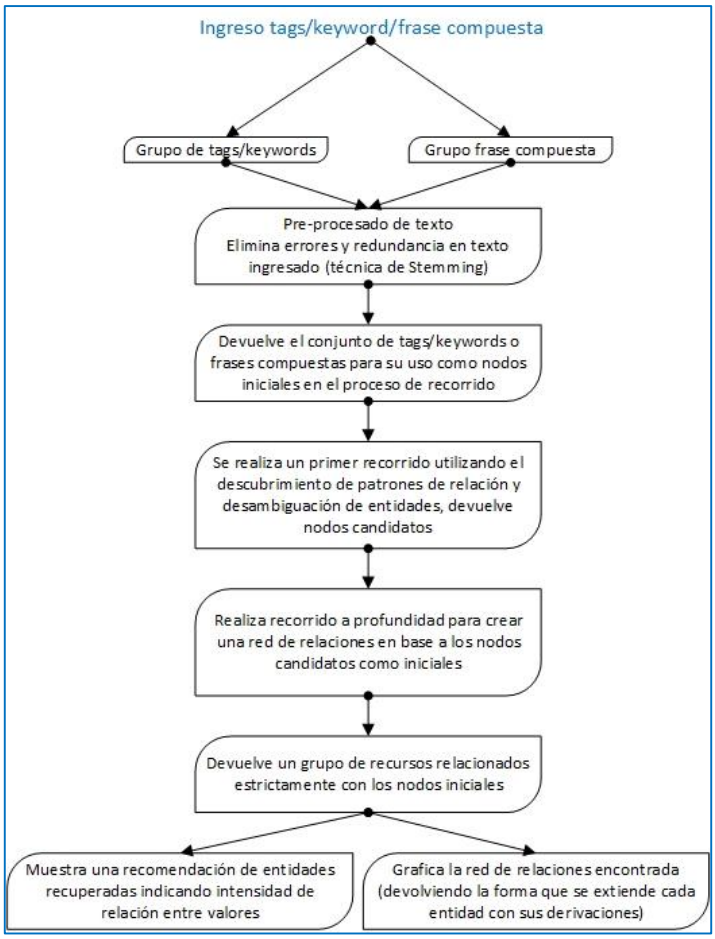


Figura 11 – Descripción de función de trabajo de cada componentes extractivos
Autor: Pablo Malla

El flujo general representado en la figura 11 muestra el proceso que debe seguir un proceso de recuperar entidades relacionadas, el primer paso es el ingreso de una lista de tags/keywords y frases compuestas, luego con el proceso de stemming se procesa el texto ingresado, se agrupa y categoriza los elementos para obtener paquetes para el proceso de recorrido y descubrimiento de entidades, con el uso de los paquetes generados se realiza un primer proceso de descubrimiento de entidades candidatas, las cuales deben cumplir con filtros de desambiguación, en base a las URI's candidatas descubiertas se realiza un recorrido a profundidad en donde se descubre una red de relaciones extensa, para finalmente poder visualizar la red en una gráfica y poder conocer la intensidad de relación que tienen cada entidad descubierta.

En base a nuestro estudio y experimentación de los diferentes procesos que muchas herramientas y técnicas de extracción aplicas llegamos a la conclusión que la combinación de procesos para encontrar recursos debe encerrar los procesos de:

- Procesado de estructura de dato
- Descubrimiento de datos

- Selección de patrones
- Algoritmos de recorrido
- Algoritmo de extracción y clasificación
- Presentación de resultados

En la actualidad contamos con un conjunto de herramientas que nos permiten implementar una base de datos basada en grafos, o manipular entidades que se manejen bajo ese esquema, uno de ellos es Neo4j, esta BDDG (Base de datos orientada a grafos), es un motor embebido en java completamente transaccional y que almacena datos estructurados en grafos en lugar de tablas. Ofrece un API para Ruby[20], Phyton[21] y Java[22]. (Ver Anexo 1)

1.6 Detección de entidades relacionadas.

Un proceso que aplica básicamente la detección de entidades relacionadas y que es bastante conocido es el campo de las relaciones en redes sociales. Un proyecto de redes sociales basa su trabajo en el uso de entidades como nodos basándose en algunos aspectos como que una red de nodos es libre escalas, ya que tiene sentido definir un valor promedio a una red ya que el número de posibles vecinos de uno nodo difiere en varios órdenes de magnitud, además una red de nodos sociales tiene un alto clusteing, es decir, la probabilidad de que 2 nodos vecinos estén conectados es muy alta. Una red social también muestra una de las características más exclusivas de una red de nodos y es que son redes “asorativas”, este término hace referencia a la capacidad de los nodos de agruparse con nodos parecidos, los nodos relacionados tienen a estar conectados entre ellos.

Existen algunos proyectos como Semantría[23] la cual trabaja con las categorías de las entidades que analiza, utilizando de igual manera la taxonomía de Wikipedia, otro proyecto AlchemyAPI[24] se realiza el análisis de datos con propiedades estructurales a gran escala, se verifica la conectividad de los nodos y se examina para ver qué tipo de conexiones existen, también se puede verificar el comportamiento del agrupamiento de los nodos de cada red.

[20] https://www.ruby-lang.org/es/
[21] https://www.python.org/
[22] https://www.java.com/es/
[23] https://semantria.com/
[24] http://www.alchemyapi.com/api/keyword-extraction

En el campo del análisis de las entidades multimedia también se ha encontrado aspectos de modelos de sitios de compartición tales como Delicious, Flickr, Instagram, en ellos se evidencia que se ofrece la posibilidad de compartir entidades o elementos en base a la colocación de etiquetas o comentarios en diferentes imágenes.

Un trabajo similar es Red Semántica Link[25], que es un modelo en el que los recursos se encuentran semánticamente vinculados y que derivan de sus conexiones nuevos modelos semánticos, su enfoque es generar conocimiento que puede encontrarse oculto y crear un nuevo enfoque de elementos significativamente vinculados entre sí mediante el concepto de grafos.

Una tecnología que tiene bastante impacto en los procesos de análisis semántico y de extracción de entidades es “Big Analytics”, que es aquella derivación de la ya clásica “Minería de Datos”, ésta tecnología se centra en la capacidad que podemos dar a las máquinas para analizar correlaciones, relaciones, los segmentos y los procesos estadísticos.

La tecnología de Big Analytics [26]se orienta a definir los proceso para descubrir patrones de información relevante y que puede ser potencialmente útil ya que el mayor porcentaje de información se encuentra en formato textual y es necesario un proceso de normalizado, pero para poder llegar a ese fin ha sido requerido primero trabajar con las entidades y los recursos que posteriormente al ser explotados pueden generar un contexto de trabajo único. (Hai Zhuge. Communities and emerging semantics in semantic link network: Discovery and learning. IEEE Trans. On Knowl. And Data Eng., 21(6):785– 799, 2009.)

Las técnicas que en la actualidad son utilizadas para detectar términos relacionados, se basan en el procesado estadístico involucrando de igual manera el procesado de lenguaje natural, también las técnicas de visualización final de resultados, aplicando un formato semántico estructurado acorde a la normativa de Linked Data.

1.7 Trabajos relacionados a la extracción de entidades relacionadas.

Existen algunas API’s de terceros que realizan un proceso similar a la extracción de entidades, estas herramientas aplican el análisis de contenido semántico y de la revisión de relaciones. Estas herramientas pretenden impulsar el proceso de extracción y

[25] http://www.lsi.us.es/~menchen/proyecto1.pdf
26 http://searchbusinessanalytics.techtarget.com/definition/big-data-analytics

descubrimiento de contenido con el fin de integrar los resultados que generen con la Web Semántica.

- AlchemyAPI[27]

Es una herramienta que proporciona una extracción de entidades y capacidades de desambiguación para el análisis de texto, HTML o también imágenes escaneadas. Entre sus características adicionales se incluye la extracción de la cita de donde se ha extraído y la identificación del idioma.

- Sitio Web: http://www.alchemyapi.com/api/entity/
- Condiciones de uso: http://www.alchemyapi.com/company/terms.html
- Precio y Uso Restricciones: gratuito hasta 30.000 peticiones / día

En el proceso que realiza en el proceso de extracción de entidades, con pocas excepciones, es desambiguado y proporciona enlaces a otras bases de datos estructurados como es FreeBase[28] y DBPedia.

- EVRI[29]

Es una herramienta que proporciona varias API's para el análisis de textos PNL, recomienda contenido y relaciones entre entidades semánticas enfocado al tema de popularidad de entidades.

- Sitio Web: http://www.evri.com/developer
- CrunchBase: http://www.crunchbase.com/company/evri
- Condiciones de uso: http://www.evri.com/developer/tos
- Precio y Uso Restricciones: Actualmente en versión preliminar, sin límites o de precio

En el proceso de extracción se centra en textos presentados, pero no aplica URIS de Linked Data hacia otras bases de datos. Aplica varias consultas en base a los criterios de valoración de EVRI, es una herramienta bastante restrictiva.

27 http://www.alchemyapi.com/api/entity/

28 https://www.freebase.com/

29 http://www.evri.com/developer

- Zemanta

Para su trabajo usa un texto transcrito, en base a ese texto extrae entidades relacionadas en una sola consulta.

Zemanta proporciona una API de contenido único, ofrece la extracción de contenido relacionado basado en bloques de texto presentado en lugar de palabras individuales o términos de búsqueda. En su proceso de extracción utiliza el texto para identificar temas relevantes al contexto, y de esa forma devuelven entradas de blog o artículos que pueden estar relacionados. La desventaja es que los metadatos o resultados son limitados en cada artículo.

- DBpedia Spotlight

Es un sistema para anotar automáticamente documentos de texto con una URI de DBPedia, además permite a los usuarios configurar las anotaciones a sus necesidades específicas a través de la ontología DBpedia siempre y cuando se consideren medidas de calidad, como la prominencia, pertinencia tópica, la ambigüedad contextual y la confianza de desambiguación.

DBpedia Spotlight se comparte como código abierto y desplegado como un servicio Web de libre acceso para uso público.

- OpenCalais

Es una herramienta que proporciona un motor de procesamiento de lenguaje natural, cuya función es extraer entidades semánticas de un texto. Esta API es de código abierto y ha sido utilizada en una gran variedad de plataformas, dando forma a una plataforma llamada OpenPublish en integración con Drupal[30] y WordPress[31]. Este aplicativo trabaja de gran manera con procesado de lenguaje natural (PNL), su trabajo es extraer metadatos semánticos.

- Sitio Web: http://www.opencalais.com/
- CrunchBase: http://www.crunchbase.com/company/opencalais
- Condiciones de uso: http://www.opencalais.com/APIkey
- Precio y Uso Restricciones: 50000 peticiones / día libre (hasta 4 / seg)

[30] https://www.drupal.org/
[31] https://es.wordpress.com/

- Requisitos de atribución: Visualización del logo y enlace a http://opencalais.com
- Actualmente proporciona desambiguación para entidades pertenecientes a empresas que manejan datos geográficos y empresas que manejan productos electrónicos.
- Soporta entrada de documentos de texto hasta 100K caracteres de longitud.
- Su implementación está orientada a negocios/empresas comerciales
- Proporciona recomendación de "Etiquetas Sociales" y "Hechos y Eventos" en sus resultados.

DBpedia Spotlight, OpenCalais, la API Zemanta, Extractiv y PoolParty Extractor analizan texto libre mediante el reconocimiento de entidades y luego elimina la ambigüedad de los candidatos a través de la resolución de nombres y enlaces de los organismos encontrados en el repositorio de conocimiento DBpedia.

Las entidades proporcionadas por el Servicio Web Calais son relevantes, y en general de buena calidad. Lo que falta, sin embargo, es desambiguación entidad y la especificación de conexiones a Linking Open Data (LOD). OpenCalais actualmente ofrece desambiguación a una entidad y a una URI de Linked Data por sólo un pequeño subconjunto de tipos de entidad.

La mayoría de estas aplicaciones pertenecen al proyecto Open Publish[32] siendo API's de libre acceso, en el análisis se llegó a la conclusión que una herramienta como Zemanta puede ser usado para cuando se requiere un análisis primario de PNL, es decir cuando se requiere trabajar con texto o documentos. OpenCalais tiene un gran impacto en la industria debido a sus métodos de desambiguación y su trabajo directo con LinkedData, sin embargo para poder explotar todas sus funciones es necesario pagar.

Todas estas herramientas trabajan usando un documento de texto, el cual se envía para su posterior análisis, estableciendo así un análisis relacional en base al contexto de un documento, más no un análisis por entidades únicas como TAGS o KEYWORDS o de URI's que representen un elemento.

Por tal motivo si se requiere buscar relaciones de entidades entre nodos es necesario implementar un aplicativo que permita utilizar palabras individuales, de igual manera hay que considerar que el acceso a repositorios de entidades como DBPedia son críticos al momento de recuperar vínculos entre entidades, y estas herramientas utilizan bases de

[32] http://www.phase2technology.com/blog/openpublish-2-0-and-beyond-a-labor-of-love/

datos que deben ser previamente construidas bajo sus propias reglas, lo que el acceso a entidades de conocimiento está limitada a la extensión y características de una base de datos interna y no global o abierta como DBPedia.

1.8 Propuesta general para el proceso de extracción de entidades relacionadas.

El marco de trabajo diseñado para el desarrollo del presente proyecto, se basa en un enfoque de Web semántica e incluye etapas de pre-procesado de texto, recorrido y extracción de datos enlazados, y presentación y recomendación de resultados. La integración de estas actividades en un flujo de trabajo permite encontrar entidades relacionadas aprovechando la gran cantidad de recursos y objetos Web descritos semánticamente.

- Pre-procesamiento estructura de recursos recuperados. (TAGS – KEYWORDS)

Se procesan el texto de entrada (palabras clave) mediante técnicas de stemming y el proceso de categorizar los elementos de acuerdo a su similaridad con el fin de especificar aún más la futura búsqueda de recursos relacionados. Este proceso también agrupa las frases compuestas en su estructura más similar para evitar la redundancia y duplicidad.

Se utiliza el ingreso una lista de palabras, siendo cada palabra considerada como una entidad, o también se puede utilizar una frase compuesta y ser considerada como un solo elemento.

- Recuperación de entidades relacionadas (extracción entidades candidatas relacionadas)

Se extraen entidades de repositorios enlazados que contengan información relacionada a la temática de interés en base a un proceso de selección de elementos candidatos desambiguados y luego elementos relacionados, para posteriormente encontrar relaciones que nos permitan generar una recomendación específica.

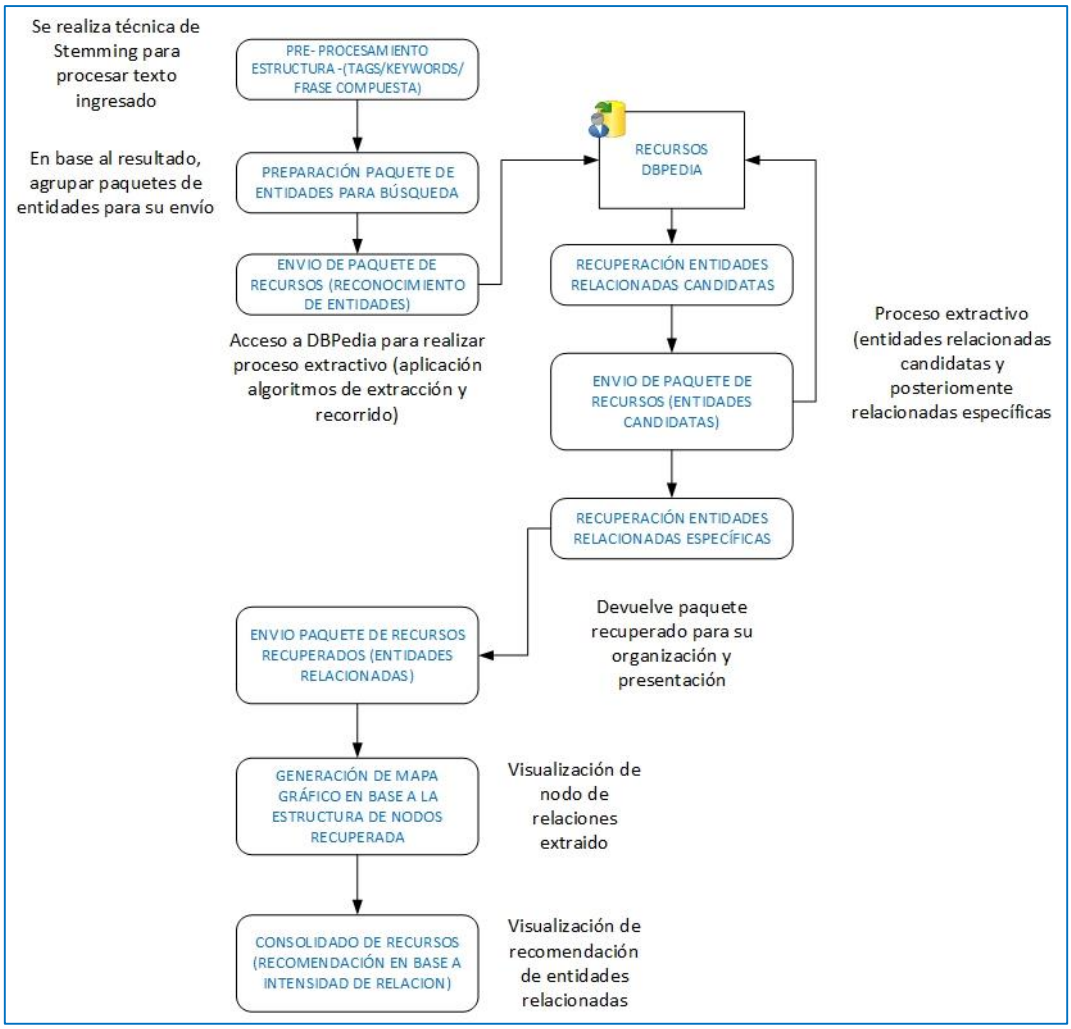


Figura 12: Estructura general de servicios extractivos integrados
Autor: Pablo Malla

- Mapeo de recursos. (extracción entidades relacionadas específicas en base a entidades candidatas desambiguadas)

El servicio que establezca la forma de mapear los resultados, esto en base a la estructura de los recursos candidatos recuperados, un reconocimiento de las entidades, su concepto y un valor que permita establecer desambiguación si existe similaridad con otros recursos, para posteriormente descubrir y recuperar entidades relacionadas específicas en base a los nodos candidatos desambiguados.

- Generación de gráfica de red de relaciones

Cuando se ha realizado el proceso de recorrido y extracción se procede a generar una gráfica la cual representa la red de entidades recuperadas, de tal manera se puede tener una idea completa de todos los elementos que conforman un grupo de recursos relacionados, desde sus nodos padres, y sus ramificaciones derivadas.

- Generación de recomendación de recursos relacionados.

Con todos los resultados obtenidos, la red de relaciones y las entidades relacionadas se genera una recomendación de uso de resultados, los cuales en base a la intensidad de relaciones entre nodos nos da una idea de que recursos están relacionados entre sí, que otros recursos comparten una dimensión de contexto y que entidades no contienen una red explotada.

1.9 Conclusiones y discusión del estado del arte.

El proceso de análisis y extracción de elementos dentro de la web de datos es un tema que se ha logrado expandir de manera exponencial y que comprende en la actualidad una gran cantidad en número de artículos y procesos de investigación, sin embargo la mayor problemática es lograr obtener un conglomerado de conceptos y conjuntos de entidades que se representen de manera explícita el universo de las entidades en las que se ha trabajado.

La existencia de varias aplicaciones de extracción de entidades y análisis actuales permite obtener entidades relacionadas de manera profunda, los resultados de sus procesos de trabajo se centran resultados de un universo específico y mantienen una línea de uso de datos único que al variar puede no devolver resultados esperados.

Este trabajo respecto a las tecnologías actuales propone el uso de entidades como TAGS y KEYWORDS lo que permite especificar qué elementos encontrar y que presencia tienen en un universo del conocimiento específico.

El hecho de conocer los nodos que derivan de un árbol de relación de un nodo principal (una entidad específica) podremos conocer que universo de conocimiento es extenso y cual falta de explotar, dándonos así la pauta para discernir caminos de investigación. Además la facilidad de considerar una frase compuesta como una entidad específica nos permite conocer aún más conocimiento o entidades específicas y no generalizadas. La deducción total es que en base a un elemento podemos generar un gran conjunto información, si administramos su estructura y sus relaciones con otros elementos podremos conocer cómo se vinculan entre elementos y como se extiende una red de conocimiento.

CAPITULO II
OBTENCIÓN Y PREPARACIÓN DE PROCESOS PARA EXTRACCIÓN DE RECURSOS RDF

2.1 Resumen.

Una gran parte del contenido de la Web está en formato textual, como los documentos académicos, científicos y el contenido de las redes sociales, si somos capaces de explotar esos conjuntos de datos codificados en cada estructura que las almacena, seremos capaces de recuperar recursos valiosos y preparar la información para su mejor utilización y descubrimiento.

En este capítulo se va a tratar los diferentes métodos de pre procesado textual que se puede aplicar a un conjunto de texto, en nuestro caso tags y keywords, esta metodología prepara la creación de nuestro primer servicio de pre-procesamiento de texto.

El procesado de texto permite especificar la correcta estructura que debe tener un conjunto de elementos para su posterior uso de búsqueda de entidades relacionadas, por tal motivo, éste capítulo indicará la solución que se ha implementado para organizar paquetes de entidades simples tales como una sola palabra o keyword así como entidades compuestas si es requerido utilizar una frase compuesta por varias palabras que requieran ser consideradas como entidad, éste capítulo indica la forma en que se estructura el primer servicio de procesado textual para la generación de paquetes de entidades que servirán como elementos iniciales al recorrido extractivo.

Al contar con una gran extensión de información interrelacionada, la aplicación de herramientas de extracción de conocimientos se ha vuelto imprescindible, pero con muchas veces los recursos disponibles en la web de datos se encuentran codificados en conjuntos de datos semánticos los cuales requieren ser explotados para poder ser interpretados y aprovechados por las personas que los solicitan.

2.1 Pre-procesamiento de tags y keywords.

La extracción automatizada de información de un texto y su transformación en una descripción formal es un objetivo importante en la investigación de la Web Semántica y la lingüística computacional. La información extraída puede ser utilizada para una variedad de tareas tales como la generación de una ontología, preguntas y respuestas y recuperación de información.

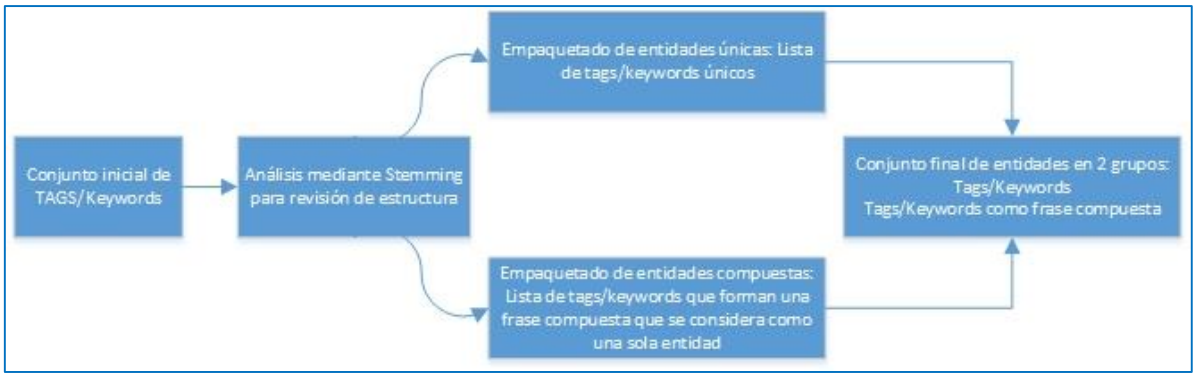


Figura 13: Flujo de entrada de Keywords y su organización
Autor: Pablo Malla

El flujo de entrada descrito en la figura 13, explica el proceso que debe cumplir una lista de tags/keywords o frases compuestas para ser considerada como un paquete de datos para su posterior uso en el descubrimiento de entidades relacionadas.

El conjunto inicial está compuesto por un listado, el cual pasa a ser analizado mediante stemming para la revisión de su estructura, mediante este análisis se logra encontrar la forma raíz de cada uno de los tags/keywords, así mismo en los elementos considerados como frases compuestas, se encuentra las frases que comparten una estructura similar y se las agrupa para evitar ambigüedad o repetición, creando así grupos de frases no repetidas o altamente similares, con esto se evita realizar el mismo análisis y descubrimiento de entidades a elementos ya validados. Al final lo que se obtiene con este flujo son 2 paquetes de elementos que son tags/keywords y frases compuestas revisados textualmente y agrupados.

Cuando se utiliza un conjunto extenso de elementos es requerido organizar su aplicación, de manera que cada elemento cumpla la función de una entidad específica, esto evitará problemas de ambiguación en la información, duplicidad en las búsquedas, errores en la extracción al obtener resultados inexistentes por considerar una estructura no válida.

Un grupo de keywords o tags únicos representan una entidad en sí, un ejemplo sería el uso de palabras como: Computer, Programing, Java, Informatic, además se considera un grupo de keywords o tags pueden crear una frase compuesta la cual debe ser considerada como una entidad única, como: “Java Programing”, “Computer Science”.

La base para la realización de pre-procesado de texto es consolidar el listado de tags/keywords y frases compuestas para su aplicación en el proceso de búsqueda de entidades relacionadas.

Las actividades que se realizan para pre-procesar el texto a utilizarse son: a) el validar que la escritura y estructura esté correcta, b) eliminar duplicidad de términos si en el listado ya consta un grupo de entidades que se definen como una entidad única, c) considerar también que una frase compuesta puede contener palabras similares pero indicando un contexto diferente.

La actividad principal a realizar en este apartado es conseguir un grupo de keywords o tags consolidado y correctamente procesado para que en base a este grupo de elementos iniciar el proceso de búsqueda y extracción direccionada al repositorio de elementos (DBPedia), considerando así la estructura de cada elemento y encontrando una entidad que se relacione de manera textual, posteriormente con los patrones de relación encontrar la relación semántica.

2.1.1 Proceso de Stemming en entidades relacionadas.

Se puede definir al proceso de stemming como el mecanismo que reduce un conjunto de palabras a su stem (raíz léxica común), y su finalidad es mejorar significativamente los resultados en los sistemas de recuperación de información (Manning, 2007).

En todo proceso de extracción de relaciones semánticas que se encuentran presentes en documentos en la actualidad no es una tarea o área de trabajo nueva, existe un gran conjunto de trabajos realizados sobre este tema, pero lo que si hay que considerar es que un proceso es considerado ya necesario y eficaz si queremos analizar algún elemento para encontrar sus elementos, y este proceso no es más que el procesamiento del lenguaje natural (NLP). (Prywes, Noah S, 1969).

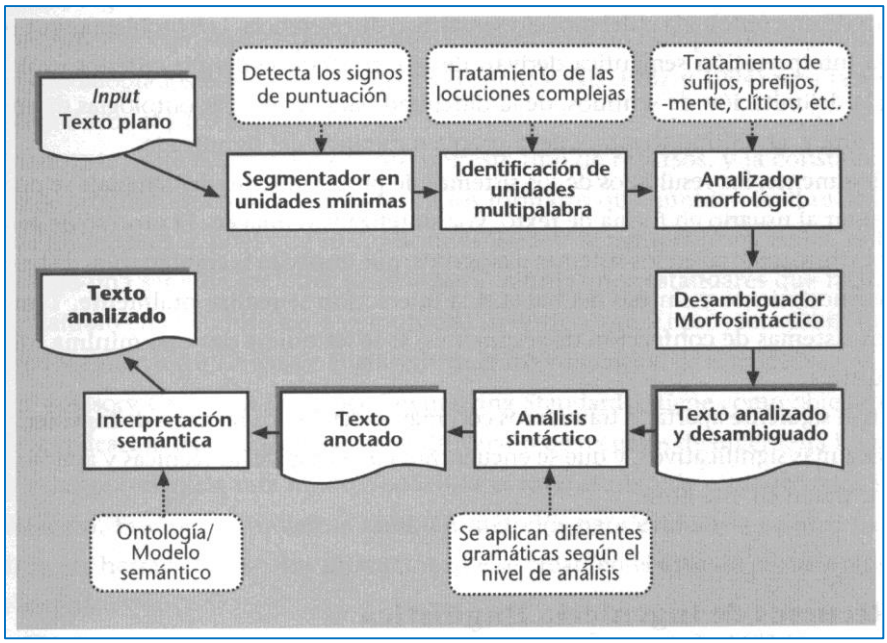


Figura 14: Flujo general de un proceso de procesamiento de texto (documentos, palabras claves, frases)
Fuente: Martí, M. A. (2003). Introducción. In M. A. Martí (Ed.), Tecnologías del lenguaje *(pp. 9-29).*

El fundamento de la recuperación de conocimiento en internet tiene su base en que cada entidad que se extrae debe representar objetivamente los rasgos del contexto con el que se trabaja, si obtenemos una estructura idónea del texto o elementos con los que trabajamos, podemos luego reconocer como se manifiestan los rangos semánticos en los documentos, y este proceso no debe ser considerado una tarea trivial. (Raghavan y Schütze, 2007)

El Stemming se refiere al proceso heurístico que nos permite eliminar los extremos de las palabras con la finalidad de encontrar la raíz de las palabras en la mayoría de las ocasiones. José Alberto Benítez Andrades, 2011)

La estructura o forma inicial (raíz de una palabra) se obtiene cuando se efectúa una reducción de las palabras a sus elementos mínimos con significado o las raíces de una palabra, de hecho "Stem" significa tallo ó raíz. De esta forma, los procesos de stemming, acotan las terminaciones de las palabras a su forma más genérica o común.

El Stemming es similar a la lematización ya que ambos nos ayudan a gestionar mejor el conocimiento recuperado de nuestra búsqueda de entidades. La mayoría de los motores de búsqueda aplican esta técnica para su trabajo, pues mediante el uso de recursos derivados o lemmatización nos puede ayudar a ampliarlas consultas de búsqueda y el agrupamiento de entidades relacionadas, lo que nos permitiría al final obtener una alta recuperación de entidades con gran precisión.

El considerar un proceso de stemming en la recuperación de la información ha planteado de diversas opciones de aplicación y comienza a estudiarse en los años 60, según (BaezaYates, 2006), con el fin de reducir los tamaños de los índices de texto (Manning, 2007) o también como una forma de normalizar los términos, por tal motivo, según (Rolleke, 2006,) el implementar un proceso de stemming inicia el proceso de estandarización y correcta representación de los elementos descriptores en un documento o de un contexto.

La extracción relación/texto también involucra la extracción de entidades para la identificación entidades o conceptos". (Brin, 1998; Iria y Ciravegna, 2005; Nguyen et al, 2007; Roth y Yih, 2002).

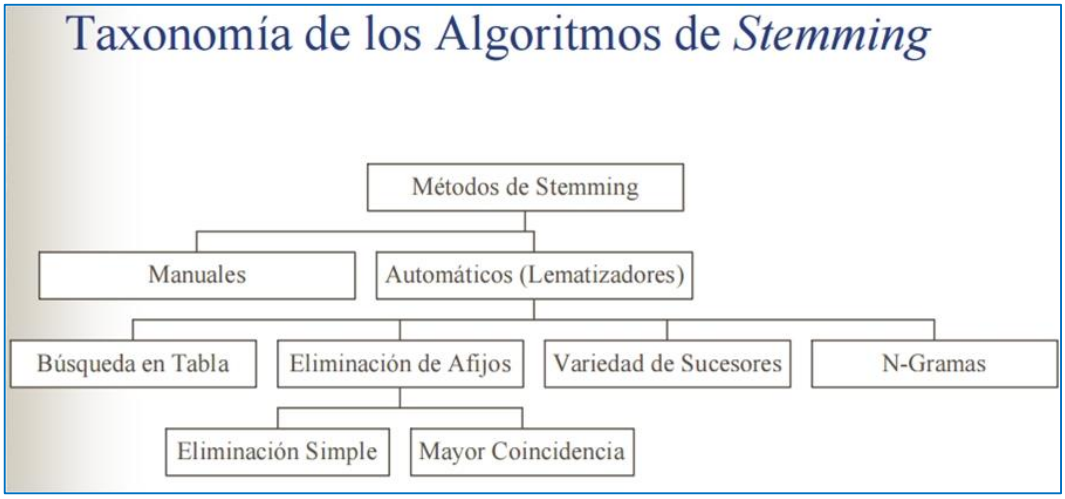


Figura 15: Taxonomía de algoritmos de Stemming
Fuente: Recuperación de información -

La extracción relación/texto se refiere al planteamiento de identificar aquellas estructuras léxicas verbales de carácter relacional que tiene como finalidad el de generar un corpus de estructuras que puedan asociar un rol o un concepto semántico. (Green R, 2010).

Las palabras están estructuradas en base a prefijos, sufijos y también una raíz, lo que se requiere es aplicar un método de procesamiento de esta estructura textual y el uso de una técnica de stemming nos ayudará a procesar las derivaciones semánticas que puedan llegar a generar confusiones durante el proceso de búsqueda y agrupamiento de la información.

Lovins [1968] define al algoritmo de división como: "un procedimiento que permite reducir todas las palabras con la misma raíz a su forma común, esto es realizado por lo general por extracción de cada palabra de sus derivativos y flexivos sufijos".

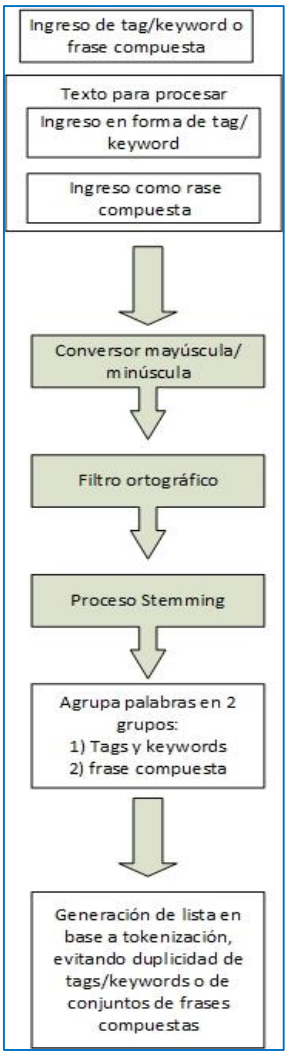


Figura16. Estructura de funciones de procesado de Tags/Keywords
Autor: Pablo Malla

Existen algunos algoritmos utilizados para desechar prefijos y sufijos:

- Paice/Husk[33]
- S-stemmer / n-gramas[34]
- Técnicas lingüísticas

Un afijo es una secuencia lingüísticas que se anteponen (prefijos), se posponen (sufijos) o insertan (infijos) en una palabra o lexema para modificar su significado, bien gramaticalmente (afijos flexivos), bien semánticamente (afijos derivativos).

Con el proceso de stemming lo que se pretende es eliminar los afijos derivativos a las entidades extraídas para posteriormente poder categorizar los elementos en un grupo

[33] http://www.scientificpsychic.com/paice/paice.html
[34] http://eprints.rclis.org/13961/1/figuerola2004recuperacion.pdf

de URI's específicas que luego nos permitirán realizar el proceso de clusterización de recursos devueltos.

Cuando recuperamos una entidad o un conjunto de elementos relacionados, se requiere organizar los resultados en base a su "común" estructura, aunque hay que considerar que la descripción de una palabra, un tag o un resultado devuelto no precisamente se refiere a un conjunto específico de resultados, es decir, no hay garantía de que un recurso represente un conjunto real de recursos similares, sin embargo, por lo general es suficiente manipular un recurso relacionado para encontrar los mapas raíces de muchos más recursos, siendo un mapa raíz todos aquellos nodos padres de una entidad de los cuales derivan nuevos elementos que al final determinan la red de relaciones de un contexto.

El proceso de lematización es quitar los prefijos y sufijos en un texto, una consulta o una palabra, y con este fin se puede agrupar las palabras que tienen un significado conceptual como por ejemplo, caminar, caminata o caminando en un único conjunto.

Estos son algunos algoritmos que se han desarrollado para el inglés, la mayoría de ellos implantan procedimiento secuénciales y de gramática definida y cerrada.

- Lematizador S-Stemmer. Un método simple para palabras en inglés es el lematizador S-Stemmer (HARMAN, 1991); en el cual solo unas pocas terminaciones comunes son 24 removidas. Este es conservativo y raramente reconoce en donde realizar el corte para la lematización.
- Lematizador de Lovins. El lematizador de Lovins (LOVINS, J, 1968) desarrollado por Jule Beth Lovins del MIT en 1968, este usa un completo algoritmo y listas de excepciones para remover más de 260 diferentes terminaciones. Específicamente, este algoritmo secuencial simple remueve solo un sufijo, contiene 260 sufijos, 29 casos para remoción de sufijos y 34 reglas para decodificar terminaciones.
- Lematizador de Porter. Fue desarrollado por Martin Porter en la universidad de Cambridge en el año 1980, se basa en la idea que los sufijos en el lenguaje inglés (aproximadamente 1200) están compuestos sobre todo de una combinación de sufijos más pequeños y más simples.

El lematizador de Porter es un extensamente usado y disponible, y se utiliza en muchas aplicaciones. Las implementaciones de este lematizador están disponibles en un Website establecido por el mismo Porter, con implementaciones en Java, C y el Perl.

Otras implementaciones de este algoritmo están disponibles en la Web. El lematizador de Porter es probablemente el más ampliamente usado en la investigación de la IR (PORTER, Martin, 1988)

Si se requiere una precisión lingüística aún más detallada se ha encontrado que la lematización nos podría servir.

2.1.2 Trabajos propuestos para utilización de stemming de entidades.

Una de las labores más importantes en el campo de la recuperación de información contextual, es la de aumentar la relevancia a los documentos recuperados (Jansen, 2006) la ejecución.

Para aumentar la relevancia de un documento recuperado no basta con la aplicación de herramientas disponibles en la internet como por ejemplo las que provee Lucene[35], por tal motivo para cumplir con el objetivo de buscar documentos relevantes, se pueden aplicar algunos filtros más estrictos en el proceso como la implementación de stopwords, las cuales son una lista de palabras de uso frecuente que, tanto en la indexación como en la búsqueda, no se tienen en consideración y por lo tanto se pueden omitir, también se puede optar la aplicación de un modelo de espacio vectorial: que es un modelo algebraico utilizado para filtrar, indexar, recuperar y calcular la relevancia de la información, o representa los documentos con un lenguaje natural mediante el uso de vectores en un espacio lineal multidimensional.

Existen muchas técnicas de reducción del espacio lineal multidimensional, aunque la técnica de DF (basada en la frecuencia de aparición en documentos) reúne sencillez y eficacia para determinar la relevancia de un documento.

La técnica DF se basa en tratar de obtener un coeficiente que indica el número de documentos en los que un determinado término de la colección aparece al menos una vez. Dicho coeficiente es lo que se denomina Frecuencia de Aparición en Documentos o simplemente DF, siglas por las que se conoce esta técnica. Yang, Yiming, y Jan O. Pedersen (2007)

[35] https://thekandyancode.wordpress.com/2013/02/04/tokenizing-stopping-and-stemming-using-apache-lucene/

Aunque éstas tienen un largo camino por recorrer, ya existen algunos diccionarios electrónicos, herramientas de lematización y/o stemming, como las que se menciona a continuación:

- Carmona (1998) propone, una Morfología Compu- Stemming para documentos recuperados de la web denominada MACO, como el lematizador de referencia, aclarando que es posible realizar el stemming mediante un algoritmo que use reglas gramaticales de derivación morfológica para un idioma en cuestión, o usando un diccionario informatizado que asocie a cada forma su lema (palabra) representante.
- Pombo (2009) plantea una estrategia de normalización de términos basada en n-gramas (donde formalmente, una n-grama es una sub-secuencia de longitud n de una secuencia dada. Es una metodología simple que pueda ser utilizada independientemente de la base de datos documental considerada y, de los recursos lingüísticos disponibles, incorporando un corrector ortográfico contextual que funciona como un etiquetador sintáctico, basado en una extensión dinámica del algoritmo de Viterbi sobre Modelos Ocultos de Markov de segundo orden.
- Velásquez y Sidorov (2005) proponen el nombre de Agme, un modelo que consiste en un conjunto de reglas para obtener todas las raíces de una forma de palabra para cada lexema, su almacenamiento en el diccionario, la producción de todas las hipótesis posibles durante el análisis y, su comprobación a través de la generación morfológica. Se usa un diccionario de cuarenta mil lemas a través del cual se puede analizar más de dos millones y medio de formas gramáticas posibles. Para descubrir estas formas incorpora procesos flexivos que ocurren principalmente en los nombres (sustantivos y adjetivos) y verbos. Las demás categorías gramaticales (adverbios, signos de puntuación, conjunciones, preposiciones, etc.), presentan poca o nula alteración flexiva; el tratamiento de estas últimas se realiza mediante la consulta directa al diccionario.

Además de estos modelos de Stemming, existen otras herramientas que pueden ayudar a realizar el proceso de pre-procesado textual, SENSEBOT[36] o Cloudlet[37] son dos plug-ins que buscan orientar a las personas a realizar sus búsquedas con términos que

[36] http://www.sensebot.net/
[37] http://www.akamai.com/html/technology/forward-rewrite-cloudlet.html

puedan estar relacionados con una consulta original. (Ryen W. White and Resa A. Roth. Exploratory Search: Beyond the Query-Response Paradigm).

Una gran extensión de la información contenida en documentos presentes en Web está codificada en gran media en lenguaje natural y por lo tanto no se encuentra estructurada, por esta razón no es factible acceder al conocimiento que se genera en base a esta estructura simple, por eso, la adquisición masiva de conocimiento en base a procesos de extracción debe compensar el aumento de la dificultad y la disminución de la calidad de la extracción.

Cuando se requiere trabajar mediante un motor de búsqueda se debe considerar una metodología que se aplica en los sistemas de recuperación de la información la cual nos indica que la recuperación de información usa el principio de frecuencia de palabras presentes en un documento, además aplica el peso único de una variable y la proximidad cercana a la petición del usuario.

2.1.3 Técnicas distancia y similaridad.

Un modelo de similaridad nos permitirá encontrar patrones y su función primordial es el encontrar un nivel de error permitido comparando las entidades y estableciendo si es aceptable o no la similaridad entre una entidad y otra.

La función principal de una metodología de análisis de distancia es establecer un proceso y un nivel de error, en el cual se realiza una medición de hileras y determina si es aceptable o no. La noción de distancia determina la comparación de hilos y su diferencia entre ellas.

Se ha analizado distintas formas de medir la distancia y la similaridad que pueda existir en los conjuntos de datos recuperados, entre ellas se ha estudiado la técnica conocida como “Distancia de Hamming”, explicado en: (A. Bogomolny (Feb/2006) - Distance between strings), pero esta técnica fue descartada debido al gran inconveniente de que para poder realizar su proceso de análisis de similaridad todos los recursos recuperados deben poseer el mismo tamaño por lo que no es factible su aplicación en nuestro servicio de pre-procesamiento de texto.

Un modelo de similaridad es una representación matemática que permite resolver el cálculo de la similaridad de un Keyword para posteriormente ser procesado y utilizado en el proceso de extracción de entidades. Este modelo nos da un soporte para las

búsquedas que se basan en ciertos patrones y que pueden presentar errores de digitación, deletreo, escaneo, etc.

La frecuencia de una variable no especifica exactamente la proximidad a datos o información cuando se analiza un recurso, por tal motivo también se ha estudiado el método de similitud del coeficiente de Jaccard para tratar de implementar su concepto y comparar la proximidad de datos enviados (keywords) y los datos devueltos en el proceso de extracción.

Mediante la aplicación de la medición de similitud de Jaccard se logró establecer que la similitud en la sintaxis de texto devuelve un resultado correcto cuando se compara con cada letra de la palabra, pero cada letra puede cambiar su posición y se puede llegar a contar como similar a pesar de ello por lo que no sería capaz de detectar los keywords de tipo exacto sobre el conjunto de datos recibido.

El coeficiente de Jaccard es adecuado para ser utilizado en la medición de similaridad de palabras pero exige un proceso de análisis de ortografía del cual carece, otra de sus debilidades es la medición de similitud con ciertas palabras.

La “Distancia Levenshtein” deriva su nombre debido al científico ruso Vladimir Levenshtein, quien ideó el algoritmo en 1965, y es otra técnica que nos permite realizar el análisis de similaridad en los recursos recuperados. (M. Gilleland (Feb/2006) - Levenshtein Distance, in Three Flavors)

Cuando nos enfocamos en la distancia y la diferencia que puede existir entre entidades se puede definir qué relaciones existen entre ellas o más específicamente en qué se diferencian unas con otras.

2.1.4 Proceso de Stemming para preparar paquete de tags/keywords.

Entre las formas principales que nos permiten realizar o verificar la diferenciación entre entidades pueden establecerse mediante las longitudes de las entidades comparadas y en el cambio de las letras unas por otras.

Para el trabajo con entidades TAGS/KEYWORDS como base principal, es necesario realizar un trabajo de análisis de los elementos que utilizaremos, procesar su estructura, eliminar ambigüedades y pre procesar el texto para inicializar un proceso extractivo en base a las palabras correctas, la estructura idónea que oriente el proceso de manera precisa.

El proceso de análisis que realiza Levenshtein aplica un proceso de análisis en base al número de eliminaciones, inserciones o sustituciones que son requeridas para lograr que una cadena o entidad considerada fuente “x” se convierta en una entidad objetivo “y”.

Un ejemplo que nos permitirá entender este proceso sería, Si x = "casa" y, y = "casa", entonces DL (x, y) = 0 ya que no es requerido realizar ninguna transformación debido a que son cadenas idénticas, pero, si nos encontramos con x = enojo y, y = enajo, DL (x, y) = 1 ya que requiere una transformación, si x = enojo y, y= enojar DL (x, y) = 2. Por lo tanto, entre mayor sea el valor de DL, existirá una mayor diferencia entre las cadenas analizadas.

La figura 17 muestra un ejemplo de agrupación de una lista previamente ingresada la cual está compuesta por frases compuestas que deben ser consideradas como entidades únicas.

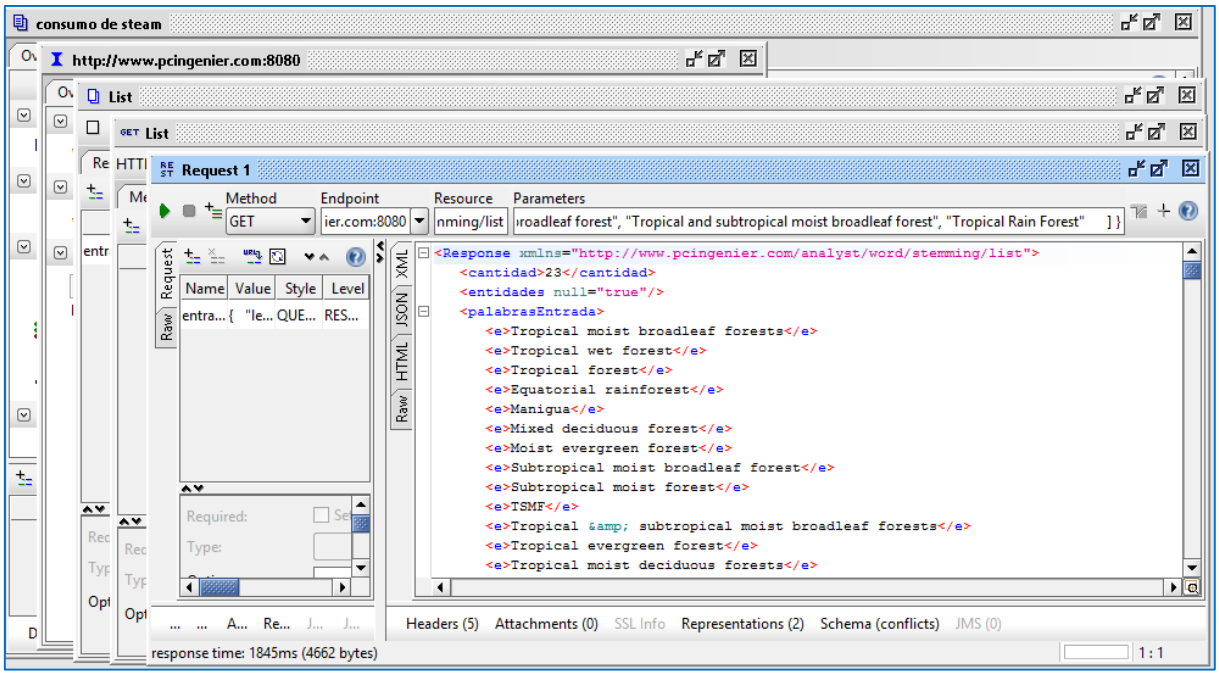


Figura 17: Ejecución de un proceso de Stemming en el servicio de Pre-procesamiento
Autor: Pablo Malla

Un paquete de pruebas de un conjunto de entidades relacionadas debe ser procesada antes de poder ser utilizadas como los elementos principales en el inicio de un proceso extractivo.

Un ejemplo de pre procesamiento de texto, es empaquetar los elementos que se envían, revisando primeramente la estructura de las palabras con los proceso previamente

descritos, luego organizar las frases para evitar ambigüedades. El fin principal es obtener un paquete compacto.

- Paquete de entidades:

Tropical moist broadleaf forests
Tropical wet forest
Tropical forest
Equatorial rainforest
Manigua
Mixed deciduous forest
Moist evergreen forest
Subtropical moist broadleaf forest
Subtropical moist forest
Tropical and subtropical moist broadleaf forests
Tropical evergreen forest
Tropical moist deciduous forests
Tropical moist evergreen forest
Tropical moist forest
Tropical moist forests
Tropical or subtropical moist broadleaf forest
Indonesian rainforest
Indonesian rainforests
Tropical forests
Tropical moist broadleaf forest
Tropical and subtropical moist broadleaf forest
Tropical Rain Forest

En este grupo de elementos podemos evidenciar que consideramos 22 entidades como “frases compuestas” y 1 elemento único. Al recibir este grupo de texto, el primer punto es procesar cada palabra que compone para evitar estructuras incorrectas, las inconsistencias en palabras que conforman a las entidades y la ambigüedad o duplicidad de elementos en entidades. [Ver anexo 7]

La ventaja del agrupamiento de los recursos procesados pueden ser vitales para su posterior uso o manipulación, entre algunos beneficios tenemos:

- Porque muchas expresiones son variaciones entre sí
- Porque -muy probablemente- se trabaja con palabras o conjunto de entidades que son posiciones similares para esas mismas variaciones sin necesidad de trabajarlas todas.
- Porque –sobre todo si manejamos grandes cantidades de expresiones- agruparlas servirá para clarificar dicho listado y hacerlo más manejable
- Porque esas variaciones son ideales para utilizarlas al hacer linkbuilding

Al proceso se le ha implementado un nuevo nivel de abstracción en base a tokens, que nos permitirán agrupar o clusterizar los resultados en conjuntos únicos evitando en lo posible la duplicidad de entidades entre grupos.

La lógica consiste en asignar un token a cada palabra, posteriormente realizar una nueva comparación exclusiva para este proceso mediante la técnica de Levenshtein en el cual comparamos la palabra, el token asignado y su presencia en el conjunto de resultados, si hay similaridad entre el grupo de palabras aumenta el tamaño de dicho grupo, pero si en algún momento se llega a evidenciar que difiere una palabra o unas palabras del conjunto de manera que la mayoría de sus entidades no coinciden, se vuelve a crear un grupo con las palabras nuevas o diferentes para comenzar nuevamente el proceso de tokenizado, comparación y asignación de entidades a sus respectivos grupos.

En la figura 18 se visualiza las agrupaciones que se han logrado generar en base al análisis realizado al grupo de elementos que se ha ingresado. Se puede ver que se han agrupado estructuras similares para las entidades compuestas que tienen como base “subtrop”, y se trata de optimizar los paquetes para evitar redundancia.

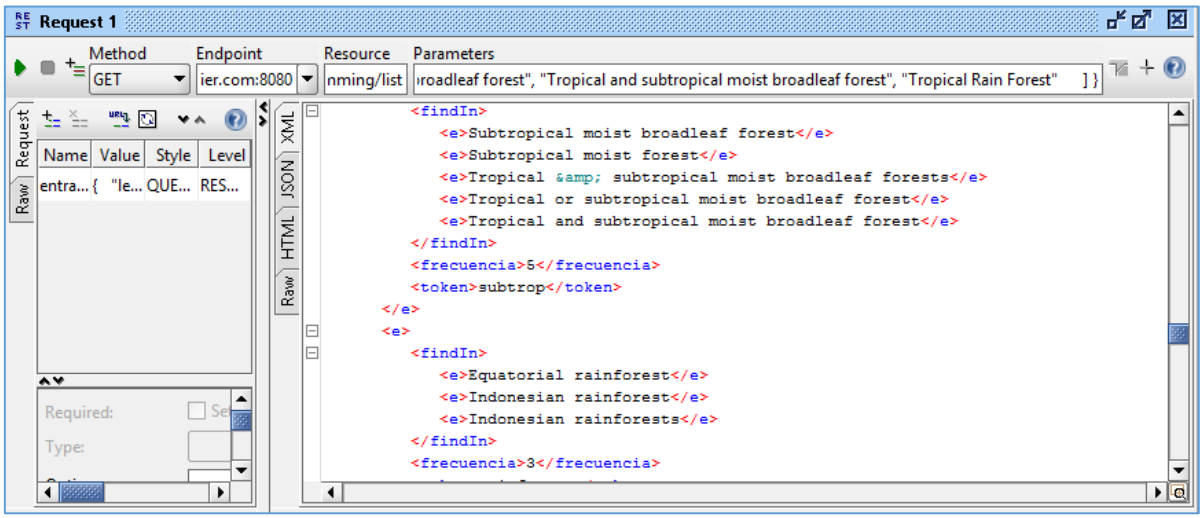


Figura 18: Ejemplo de tokenización (frecuencia) y clusterización de keywords procesados – Agrupamiento en base a análisis de estructura textual de entidades
Autor: Pablo Malla

Cuando hemos logrado procesar y clusterizar el conjunto de entidades, keywords o recursos que utilizaremos para la posterior extracción de tópicos relacionados, se procede al envío hacia el nuevo servicio de análisis de recursos relacionados. Cabe recalcar que el procesamiento puede ser de miles de entidades, recursos, texto, por lo que el soporte del servicio es robusto y trabaja de manera independiente.

Cuando se aplica este método podemos analizar la proximidad entre 2 conjuntos de datos o paquete de keywords, evitando la existencia de redundancia en los datos aplicados, y al obtener los resultados se encontró precisión en la obtención del análisis de resultados pero en la aplicación de una base de datos pequeña.

2.2 Conclusiones y discusión del capítulo II.

El elemento principal para la correcta extracción de entidades en cualquier proceso extractivo son sus entidades iniciales, si el grupo de elementos no está correctamente estructurado es posible que se obtengan elementos con otros significados, con errores o simplemente que el proceso extractivo no se realice de la manera correcta, por tal motivo, es requerido revisar en lo posible las palabras o los elementos que se consideran como punto de partida para cualquier proceso de recorrido y extracción de información.

La aplicación de un pre procesado de texto nos permite conocer exactamente qué elementos estamos considerando, y para el proceso actual del presente trabajo ha sido necesario utilizar el proceso de stemming para solventar el inconveniente de duplicidad de entidades entre palabras clave y frases compuestas, de esta manera estamos evitando sobrecargar de datos duplicados o ya considerados, agrupar entidades en paquetes y considerar a las búsquedas compuestas como una sola entidad, evitando el disgregamiento y la confusión al momento de iniciar un recorrido extractivo.

Un proceso que se puede considerar para trabajos futuros puede ser extraer un conjunto de texto, y establecer que tags, palabras, o frases queremos considerar para encontrar elementos relacionados a ellos y con el servicio de pre-procesado de texto podremos agruparlos, evitando la duplicidad y convirtiendo a cualquier tipo de información en una entidad o permitiendo el uso de tags o keywords sin duplicidad.

CAPITULO III

ANÁLISIS DE PROCESO PARA DETERMINACIÓN DE PATRONES DE RELACIÓN

3.1 Resumen.

Al utilizar tags/keywords o frases compuestas como los elementos que necesitan ser recorridos para conocer el universo de conocimiento que encierran, es necesario considerar la forma ideal de atravesar su red de relaciones, es decir, conocer qué otros nodos orbitan a su alrededor y tienen una relación común.

Al establecer un recorrido en el cual se atraviese una extensión determinada, y en su proceso se determine que patrones de relación vinculan fuertemente una entidad con otra, se los extraiga y se los agrupe de manera que al final de un proceso de recorrido se conozca un conjunto de relaciones con una entidad inicial, así tendremos la capacidad de expandirnos hacia otras dimensiones de conocimiento.

Este capítulo describe los tres procesos de extracción de datos que nos permiten obtener conjuntos de entidades relacionadas entre sí, y en base a esos paquetes de conocimiento poder recomendar, conocer la intensidad de relaciones, que entidades comparten un universo de trabajo con otra, que entidad carece de más nodos relacionales y necesita ser expandida con nuevos elementos, y que entidad se extiende hacia otros campos que pueden guiarnos hacia otros contextos.

3.2 Generalidades de estructura de entidades para aplicación en un proceso de descubrimiento de recursos.

En la actualidad la cantidad de información disponible en la web ha crecido de una manera exponencial, y es una característica conjunta a esta realidad es la incidencia que tienen los usuarios al acceder de manera masiva, cada uno con diferente necesidad entre un contexto y otro, por tal motivo se han establecido dos requisitos fundamentales que son: La organización de la información (Information Organization) y la organización del conocimiento (Knowledge Organization).

Para solventar esta necesidad creada por el requerimiento de acceder a información de manera contextualizada existen los Tesauros y las Ontologías, las cuales mediante el uso de contextualizaciones ingresadas de manera manual han logrado crear grupos de información para entidades basadas en un contexto único que permitiría a un usuario diferir entre que dimensión de trabajo quiere buscar o simplemente omitir cierta información que no le interesa a pesar de estar asignada a una entidad crítica.

Según (Novak JD, 1984) lo que hoy se conoce como mapas del conocimiento probablemente pudieran tener sus bases teóricas en las redes semánticas; el problema

con las redes semánticas es que ellas eran estructuras heterogéneas, cuya representación con sus dependencias semánticas, era sólo realizada por los investigadores y los sistemas que las utilizaban sin extensión a otros entornos o aplicaciones computacionales. Los métodos de razonamiento de las redes semánticas se basaron en estrategias particulares de implementación, no en un lenguaje formal.

Para clasificar relaciones semánticas se debe considerar primer el contexto en el que se va a trabajar, el artículo: Fillmore, C. J. “Types of lexical information”, nos explica que las relaciones semánticas dependen de aquellas características léxico-semánticas y las sintáctico-semánticas que están establecidos entre los conceptos o la dimensión de estudio que se necesite respecto a un dominio de conocimiento específico.

El trabajo con SKOS – CORE se fundamenta en el uso del esquema de RDF, ya que de su capacidad para definir conceptos, esquemas de estos elementos también se puede derivar el usar dichos elementos para encontrar nuevas entidades que en base a ellos puedan determinar una relación. Un concepto está definido como una unidad de pensamiento y puede ser descrito o establecido, y a su vez una colección de estos elementos genera un esquema.

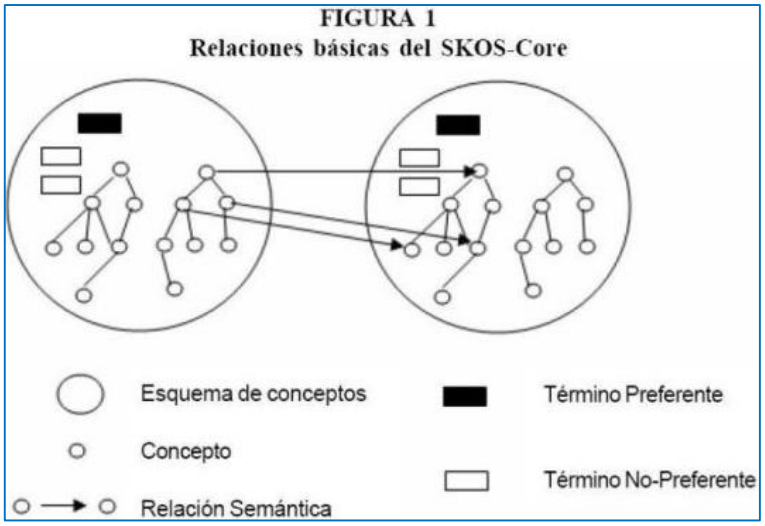


Figura 19: - Estructura de Relaciones SKOS
Fuente: Skos European Report

Como se visualiza en la figura 19, un concepto está vinculado únicamente a una etiqueta preferente, lo que es definido como: Descriptor o Término Preferente y también se considera un número de etiquetas alternativas, que son definidas como: Términos no descriptores o Términos no preferentes.

Las relaciones que se encuentran entre los descriptores tradicionales pueden ser descritas entre los conceptos de un esquema de conceptos, los cuales pueden estar

definidos en SKOS-CORE con la característica que estas relaciones se trabajan bajo un carácter semántico.

Vocabulario SKOS

a) Conceptos y esquemas de conceptos

- skos:Concept
- skos:ConceptScheme
- skos:inScheme
- skos:hasTopConcept

b) Etiquetas léxicas

- skos:prefLabel
- skos:altLabel
- skos:hiddenLabel

c) Relaciones semánticas

- skos:semanticRelation
- skos:broaderTransitive
- skos:narrowerTransitive
- skos:broader
- skos:narrower
- skos:related

d) Documentación

- skos:note
- skos:scopeNote
- skos:historyNote
- skos:changeNote
- skos:definition
- skos:editorialNote
- skos:example

e) Colecciones de conceptos

- skos:Collection

- skos:OrderedCollection
- skos:member
- skos:memberList

f) Propiedades de mapeado

- skos:mappingRelation
- skos:exactMatch
- skos:broadMatch
- skos:narrowerMatch
- skos:relatedMatch

g) f) Notaciones

- skos:notation

La codificación de las etiquetas correspondientes a los Términos Preferentes y No Preferentes pertenecientes a un concepto, skos: Concept, se realiza mediante las propiedades skos: preflabel, para los términos preferentes y skos: altLabel, para los no preferentes o descriptores.

Esta segunda etiqueta representa la relación de sinonimia o las equivalencias, debido que la utilización de notas aclaratorias es otra necesidad que se representa con el skos: ScopeNote y las definiciones de conceptos se representan mediante skos: definition.

Adicionalmente, el SKOS-CORE permite representar las relaciones básicas entre los conceptos mediante la familia skos: semanticRelation, bajo la cual se encuentran las relaciones jerárquicas definidas por las propiedades skos: narrower y skos: broader, para términos específicos y generales respectivamente y skos: related para términos relacionados.

El SKOS-CORE también posee la forma de especificar otros tipos de propiedades que representan una forma de ampliar las relaciones semánticas entre conceptos, es así como aparecen:

a) skos: broaderGeneric y skos: narrowerGeneric, para definir únicamente relaciones de subfunción entre dos conceptos, donde un concepto es subclase de otro.

b) skos:broaderInstative y skos:narrowerInstative, donde Instative expresa que un concepto es una instancia de otro, y
c) skos: broaderPartitive y skos: narrowerPartitive, donde Partitive indica que un concepto es parte de otro (es una forma de especificar que un concepto es parte de otro).

Por último, este también permite expresar refinadamente la relación de asociación con las propiedades skos: relatedHasPart y skos: relatedPartOf, las cuales expresan que un concepto tiene parte de otro concepto o que un concepto está relacionado con parte de otro concepto respectivamente.

Descripción de las propiedades y relaciones definidas en la ontología SKOS-Core.

Propiedad SKOS-Core	Tipo / Actua	Definicion	Relación
Skos:prefLabel	Terminos	Preferente o Descriptor	
Skos:altLabel	Terminos	No-preferente o No-descriptor	Equivalencia
Skos:externalID	Conceptos	Identificadores para los conceptos	
Skos:copeNote	Conceptos	Definicion de concepto, notas aclaratorias	
Skos:definition	Conceptos	Definicion de concepto	
Relaciones Semanticas	**Tipo / Actua**	**Definicion**	**Relación**
Skos:narrower	Terminos	Terminos especificos	
Skos:broader	Terminos	Terminos generales	
Skos:related	Terminos	Terminos relacionados	
Generic	**Tipo / Actua**	**Hereda de subclassof**	**Subsuncion**
Skos:broaderGenereic	Terminos	Herencia de terminos especificos	
Skos:narrowerGeneric	Terminos	Herencia de terminos generales	
Instative	**Tipo / Actua**	**Hereda de type**	**Un concepto es instancia de otro**
Skos:broaderInstative	Conceptos	Un concepto es instancia de un concepto especifico	Skos:relatedHasPart
Skos:narrowerInstative	Conceptos	Un concepto es instancia de un concepto general	Skos:relatedOf

Figura 20: Propiedades y relaciones definidas por SKOS Core – Fuente: Skos Core Guide

3.3 Ejemplo general de utilización de la estructura de un recurso para recuperar entidades relacionadas.

El proceso general de extracción se basa de manera general en un esquema de conceptos. Inicialmente se necesita obtener las categorías principales que cubre una temática general de dominio, en donde cada una represente un concepto. Se necesita recorrer todos los elementos del dominio, en el caso del presente trabajo sería la estructura inicial de un recurso y se extrae el concepto que puede estar referenciado como concepto subordinado, así logramos categorizar un concepto inicial para un recurso, para realizar este proceso se usan las propiedades:

- skos:broader y

- skos:narrower

Un segundo proceso consiste en adicionar relaciones de asociación entre conceptos, esto significa que se debe representar como se relaciona cada concepto con los demás bajo un tipo de relación de asociación, la cual debe representar una similitud temática entre conceptos, para este proceso se aplican la propiedad principal de un proceso de extracción que es:

- skos:related

En este proceso lo que se obtiene son conceptos, los cuales conforman un árbol jerárquico, en la mayoría de los casos de un tercer nivel de profundidad, aquí hay que considerar que un recursos puede tener ramificaciones a más niveles de profundidad, pero solo se han considerado hasta 3 niveles.

El primer nivel contiene los conceptos principales bajo un término, que es el TAG/KEYWORD/Frase compuesta inicial, cuando se selecciona estos conceptos hay que considerar la posibilidad de descubrir otros descriptores

A cada recurso inicial se le debe asociar la propiedad skos: subjet que está definida en SKOS:Core, con esta configuración se pretende relacionar cada recurso con los conceptos de la base de contexto que los describe semánticamente, así logramos vincular la base de contexto, con la base de recursos, permitiendo que en la aplicación de un módulo de extracción podamos recuperar elementos vinculados semánticamente, en donde se puede recuperar, un título, una descripción, un idioma, o cualquier parte de la estructura de un recurso.

Un agente de búsqueda, puede ser un sistema ya creado como SABIOS[38], puede ser un servicio como AlchemyAPI API o ZEMANTRA o un conjunto de algoritmos implementados, pueden formular una consulta en forma de términos semánticos (puede ser un concepto), y comenzar un proceso de recorrido y extracción, y en base a un contexto que se determine, si una entidad coincide con un concepto existente en una base de contextos (DBPedia u otro repositorio de contextos o recursos) entonces se puede encontrar o considerar ese elemento, para hacer ese discernimiento se puede utilizar las etiquetas:

- preferentes(skos.prefLabel) como

[38] SABIOS – SEMANTIC http://www.redalyc.org/articulo.oa?id=179014344008

- las etiquetas alternas (skos:altLabel) e incluso
- las etiquetas de los errores (skos:Hidden) de cada concepto
- (skos:concept)

Si al final de ese proceso de consulta, el agente que hayamos aplicado encuentra elementos que tienen uno o más conceptos, se procede a buscar cada uno de esos elementos vinculados semánticamente que tengan relación con otros conceptos, estas relaciones se determinan mediante:

- relaciones de herencia - skos:broader, y
- relaciones de generalización - skos:narrower y
- relaciones de asociación - skos:related)

El fin del uso de estos elementos es encontrar otros conceptos relevantes de manera semántica con la búsqueda inicial.

Cuando se ha logrado encontrar nuevos conceptos, nuestro agente aplicado debe retornar un recurso, puede ser un RDF que contiene una lista de tripletas RDF asociados con la información de cada uno de los conceptos originales y los elementos asociados a ellos.

El identificador único del concepto, la propiedad que identifica el tipo de relación semántica que establece con otros (skos: broader ó skos: narrower ó skos: related) y el identificador único del concepto asociado.

Finalmente, hay que indicar que para realizar este proceso se puede aplicar los mecanismos de consulta SPARQL, y también herramientas como JENA para su integración, con estas premisas podríamos lograr interpretar las tareas de razonamiento de los agentes que hayamos seleccionado para acceder al conocimiento.

3.4 Generalidades de algoritmos para implementación para un proceso extractivo.

Para un proceso extractivo se debe aplicar herramientas que permitan conocer de manera exacta la relación entre recursos. Existen varias técnicas para el descubrimiento de relaciones, no solo de entidades sino también de estructuras más complejas como Tesauros, Ontologías, etc. (A.Budanitsky, 2006)

La mayoría de éstas técnicas ya existentes para el reconocimiento automático de relaciones están orientadas hacia la utilización de criterios sintácticos que inician en la comparación de palabras o conceptos que describan conceptos clave. Éste proceso de descubrimiento de información debe tener.

Los enfoques actuales hacen referencia a mecanismos de emparejamiento semántico que nos permite descubrir entidades o elementos vinculados, se emplea inferencia lógica con lo cual se identifica relaciones de equivalencia, hay que considerar tres enfoques en este proceso de reconocimiento.

- Enfoque de emparejamiento basado en relaciones terminológicas: Su fin es determinar descripciones de los elementos, se puede aplicar sistemas como Wordnet o Thesaurus, esto nos debe permitir contrastar conceptos e identificar relaciones terminológicas como la sinonimia. D. Bianchini, V. De Antonellis, B. Pernici, P. Plebani. Ontology-based Methodology for e-Service discovery. Information Systems, ISSN 0306–4379, 31(4), 361–380 (2006). Referenciado en 69, 71
- Enfoque de Emparejamiento basado en Categorías: Su finalidad es evalúa la relación existente entre los parámetros definidos entre los descriptores del servicio publicado y solicitado, tales como los conceptos que enriquecen sus entradas y salidas, con el fin de determinar la correspondencia entre ambos y clasificarla dentro de cinco categorías denominadas grados o niveles de correspondencia (Exacto, Contenido, Contenedor, Intersección y Nulo) (Paolucci, 2002).
- • Enfoque de Emparejamiento basado en la Clasificación (Ranking) de las entidades: este enfoque sugiere que la clasificación de la correspondencia basada en categorías no es suficiente para determinar el grado de relación entre los servicios comparados.

  Propone entonces cuantificar esta relación a partir del cálculo de la similitud semántica de los recursos aplicados uncialmente: entradas y las salidas de los servicios publicados, de manera que pueda realizarse una clasificación o ranking de elementos de acuerdo con dicho valor de similitud.

  Existen muchas formas de definir matemáticamente esta relación, una de las más utilizadas consiste en medir la longitud del trayecto entre los dos conceptos comparados, asignando pesos a las aristas del camino que los separa, los cuales dependen del número de nodos secundarios presentes en la jerarquía de un

conjunto de recursos o de una ontología de dominio a la cual pertenecen (profundidad). (H Al-Mubaid, 2006)

Al momento de establecer los enfoques que se pueden considerar para la extracción de entidades relacionadas o de elementos que se encuentran alrededor de un nodo o elemento inicial, se debe aplicar un grupo de algoritmos o metodologías que nos permitan seleccionar o clasificar los nodos idóneos para armar una red de relaciones y conocer el contexto que se expande en base a ellos.

- Naive Bayes[39]: se considera como parte de los clasificadores probabilísticos, los cuales se basan en la suposición que las cantidades de interés se rigen por distribuciones de probabilidad, y que la decisión óptima puede tomarse por medio de razonar acerca de esas probabilidades junto con los datos utilizados.
- El esquema C4.5[40]: fue diseñado como una extensión del algoritmo ID3[41], este último forma parte de los clasificadores conocidos como árboles de decisión, los cuales son árboles donde sus nodos internos son etiquetados como atributos, las ramas salientes de cada nodo representan pruebas para los valores del atributo, y las hojas del árbol identifican a las categorías. Estos algoritmos proporcionan un método práctico para aproximar conceptos y funciones con valores discretos, e.g. en clasificación de elementos.
- k-Vecinos Más Cercanos:[42] k–Vecinos más cercanos (k-NN, por sus siglas en inglés) es uno de los métodos de aprendizaje basados en instancias más básicas, pero con resultados aceptables en tareas que involucran el análisis de entidades. En resumen, este algoritmo no tiene una fase de entrenamiento fuera de línea, por lo tanto, el principal cálculo se da en línea cuando se localizan los vecinos más cercanos. La idea en el algoritmo es almacenar el conjunto de entrenamiento, de modo tal que para clasificar una nueva instancia, se busca en los ejemplos almacenados casos similares y se asigna la clase más probable en éstos.
- DBPedia Ranker [43]: Es un proceso que explora la estructura gráfica de un conjunto de elementos de DBpedia que le son proporcionados, consulta de manera externa fuentes de información con el fin de calcular un valor de similitud para cada entidad que se encuentre durante la exploración. El detalle de la

[39] http://naivebayes.blogspot.com/
[40] http://es.wikipedia.org/wiki/C4.5
[41] http://es.wikipedia.org/wiki/Algoritmo_ID3
[42] http://www.scielo.org.co/pdf/iei/v28n3/v28n3a11
[43] *Not Only Tags: http://www-ictserv.poliba.it/publications/2010/MRDD10b/icwe2010.pdf*

implementación de DBPedia Ranker será expuesto en el siguiente apartado del capítulo, debido a que es la metodología que se ha implementado en el desarrollo del proyecto.

3.5 Identificación de patrones de relación en DBPedia.

El proceso de ranking que se realiza en DBPedia se basa en el análisis y la explotación del texto de los enlaces accedidos, específicamente en los resúmenes, los párrafos primeros de cada artículo y los wikilinks; es decir, los vínculos existentes entre cada página de Wikipedia. En el primer proceso se utiliza específicamente los dos primeros nodos: URI1 y URI2 y se realiza la siguiente comparación:

> Si la etiqueta de la URI1 del primer nodo está contenido en el resumen de la URI2 y viceversa. Si ha sido encontrado un vínculo se establece que existe una relación primaria entre las 2 entidades, sino, se prosigue con otro elemento, aunque no se descarga definitivamente éste análisis, y la razón se debe a que, puede darse el caso que un nodo no tenga un nuevo nodo hoja, por lo tanto se puede considerar al nodo vecino como un elemento cercado que, aunque no represente una relación primaria, puede ser utilizado como un nodo cercano.

El supuesto de este proceso comparativo es que si el nombre de un recurso a consultarse se encuentra en el resumen del otro recurso de DBPedia es razonable creer que los elementos están relacionados unos con otros.

Un proceso comparativo aplicable también es el comprobar que si la página de la Wikipedia de un recurso (uri1) tiene como salida un enlace a la página de Wikipedia de la (uri2).

Figura 21: Flujo de proceso de Rankeo para determinar nodos de relación- Fuente: DBPedia Ranker – Not Only Tags

En la figura 21 se puede el flujo del proceso para encontrar entidades, la aplicación de los 3 algoritmos des descubrimiento devolverán al final una red de relaciones.

Cuando se inicia la exploración de un grafo en la DBPedia se debe comenzar desde un grupo de nodos iniciales.

La exploración del grafo DBpedia comienza a partir de un conjunto de nodos semilla. Tenemos una variable R que se inicializa con los recursos en el proceso inicial de búsqueda, luego con los pasos iterativos se va llenando con cada nodo alcanzado durante ese proceso de exploración.

Se necesita establecer cuál es la estructura básica que tiene un elemento o un nodo que nosotros utilizaremos, esta estructura será:

$$r = \begin{Bmatrix} \text{URI} = \text{Una uri de DBPedia} \\ \text{encontrado} = \text{Las veces que la URI ha sido encontrada} \\ \text{rankeo} = \text{Un valor boleano que representa si la Uri ya ha sido rankeada} \\ \text{en. contexto} = \text{Un valor boleado que indica si una uri debe estar o no dentro de un contexto} \end{Bmatrix}$$

Estructura de un nodo para recorrer
Autor: Pablo Malla

Se puede aplicar dos algoritmos de exploración en la DBPedia, los cuales permiten acceder, discernir y descubrir nodos relacionados para su posterior uso en un proceso de extracción más específico que nos permite luego recomendar resultados.

Estos algoritmos de exploración y rankeo de entidades utilizan RST (Rhetorical Structure Theory) o extracción hasta MAXIMA PROFUNDIDAD.

La configuración inicial ideal para P (propiedades a considerar de cada entidad) utiliza dcterms: sujeto y skos ya que no representan en sí a un contexto especial y están inherentes en casi la totalidad del conjunto de datos de la DBPedia. En el caso específico se aplica skos: broader.

3.5.1 Primer algoritmo de exploración.

El primer proceso de exploración requiere recorrer los nodos que orbitan alrededor de una entidad y atraparlos para su posterior análisis, se consideraría como un proceso de reconocimiento de extensión de una hipotética red de relaciones existente.

Entrada:

- n: Un nodo a ser explorado
- P: Las propiedades que consideramos de la entidad
- MAX_DEPTH: el número máximo de saltos que se realizarán a partir de n

(Se utiliza las propiedades de P para realizar una consulta a DBPedia para cada uno de los elementos vecinos de n dentro del rango establecido con MAX_DEPTH, de esta manera se obtiene un conjunto Q de nodos candidatos que están relacionados directa o indirectamente con n.)

```
1     Q = ObtenerNodosRelacionados (n, P, MAX_DEPTH);
2     Por cada q_i ∈ Q recorrer
          Si q_i no se encuentra en R entonces se lo agrega a la variable, o sino,
          aumentamos el valor de "encontrado" para que represente el número
          que q_i ha sido recuperado en nuestra búsqueda.

3          Si q_i ∉ R entonces
4                q_i.URI = getURI(q_i)
5                q_i.encontrado = 1
6                q_i.rankeado = falso
7                q_i.en_contexto = NULL;
8                R = R ∪ {q_i};
9                guardar {q_i};
10         sino
11               q_i=encontrado = q_i=encontrado + 1;
12               actualizar q_i;
13         fin si
14         sim = similaridad (n.URI,  q_i.URI);
15         guardar (n, q_i,sim);
16    fin
17    n.rankeo = verdadero:
18    actualizar n;
```

Primer Algoritmo de Exploración aplicable a DBPedia (Exploración de profundidad en base a contexto encontrado en cada recurso) –Tomado: DBPedia Ranker – Not Only Tags: http://www-ictserv.poliba.it/publications/2010/MRDD10b/icwe2010.pdf

Al momento de explorar un grafo de varios niveles se llegó a la conclusión que al determinar una distancia de MAXIMA PROFUNDIDAD de tamaño 2 teníamos un buen punto de partida, porque los recursos que se encontraban dentro de 2 saltos en la exploración estaban altamente correlacionados con el nodo raíz o la URI.

Cuando se llegaba a un tercer salto cada correlación iba perdiendo fuerza y disminuyéndose con rapidez, inclusive si se realizaba un análisis a un nivel 1 en la mayoría de los casos considerábamos solo los nodos que se encuentran directamente vinculados o relacionados.

Entre más profundidad de exploración se realiza existe el riesgo de descubrir demasiados elementos ya no relacionados.

Si se aplica la exploración de 100 nodos o semillas y se establece que la máxima profundidad de exploración sea = 4 se ha podido evidenciar que en el top 10 de los primeros resultados, estos se encuentran bastantemente relacionados o son similares

con un promedio de similitud del 85% donde cada recurso entre si es similar entre 1 o 2 saltos.

3.5.2 Segundo algoritmo: ranking de entidades.

Cuando se ha explorado y conocido la red de nodos a partir de un nodo inicial, se comienza con el proceso de reconocimiento de nodos relacionados, aunque a primera instancia se puede considerar que los nodos cercanos u obtenidos en el reconocimiento de la red, aún no se puede afirmar que tengan una relación cercana o contextual, para ello analizamos cada uno de los elementos.

Entrada:

- Un elemento denominado S = $\{uri_i\}$ de un conjunto de URIs;
- Un elemento P = $\{p_j\}$ propiedad a considerar de cada entidad;
- La determinación del proceso de Máxima profundidad que definirá el número de saltos de exploración en cada nodo;
- Una variable global R la cual contendrá todos los recursos encontrados durante el proceso de exploración.

1 R = 0;

(Por cada nodo encontrado en base a la URI inicial (primer descubrimiento de un tag) se crea un nuevo nodo derivado. Este nodo también representa un contexto específico.)

2 Por cada descubrimiento $uri_i \in$ S recorrer

3 r_i. URI = uri_i;
4 r_i. encontrado =1;
5 r_i. rankeo =falso;
6 r_i. en.contexto =verdadero;
7 R = R ∪ $\{r_i\}$;
8 guardar r_i;

(En este paso se inicializa la exploración y el Rankeo de los nodos encontrados como semillas para permitir encontrar un nuevo elemento dentro del mismo contexto.)

9 explorar (r_i, P, MAX_PROFUNDIDAD);
10 FIN

(Una vez que hemos logrado establecer los nodos con un contexto se exploran todos los recursos rankeados restantes.)

11 Por cada $r_i \in$ R como (r_i.rankeo == falso) recorrer;

(Aquí se explora y rankea solo los nodos de DBPedia que se encuentren dentro del contexto de la búsqueda inicializada.)

12 Si esta_en_contexto (r_i) entonces

(r_i Será explorado y rankeado con todos sus nodos vecinos y los nodos descubiertos serán adheridos a R)
13 explorar (r_i, P, MAX_PROFUNDIDAD);
14 Fin SI
15 Fin

Segundo algoritmo de Rankeo aplicable a DBPedia (Exploración de profundidad en base a una consulta bajo SPARQL estableciendo datos en campo de propiedades) –
Fuente: DBPedia Ranker – Not Only Tags

Por ejemplo en la [línea 12 del Algoritmo 2] podemos revisar que se realiza una evaluación a un nodo estableciendo si aquel nodo alcanzado durante la exploración pertenece al contexto original que ha sido "enviado" y "descrito" en los nodos semillas, si se logra encontrar que parte del contexto de ese nodo se encuentre presente entonces realizamos un nuevo proceso más de exploración.

La figura 22 muestra los dos componentes que se relacionan, luego del proceso del pre procesamiento de tags/keywords y frases compuestas, la lista generada sirve como punto de partida para la identificación de entidades relacionadas en base al análisis de los algoritmos de exploración, análisis y rankeo.

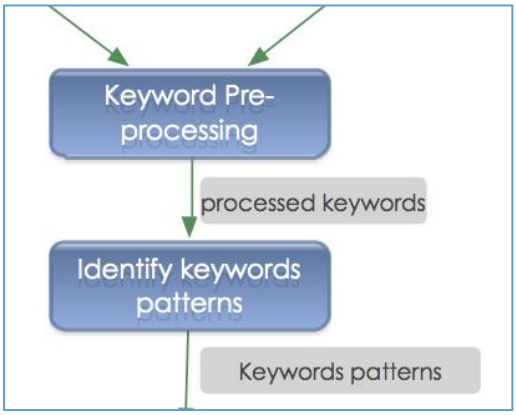


Figura 22: Primera identificación de entidades en base a Keywords procesados
Autor: Pablo Malla

Otra forma de realizar una consulta y extracción de URI's relacionadas es en base a la exploración de un grafo de nodos en DBPedia mediante una consulta SPARQL en un endpoint.

Si consideramos factible realizar esta búsqueda directa podemos aplicar el segundo algoritmo para el Rankeo en DBPedia.

Cada propiedad en DBpedia que requiera ser explorada puede ser establecida en el sistema previo al proceso de exploración, por ejemplo utilizando las propiedades dcterms: sujeto y skos: amplitud, y un dato importante es considerar que estos 2

términos no están establecidos para un contexto en específico, por lo tanto pueden ser muy útiles como un punto de inicio de exploración.

En algunos casos cuando se realiza una exploración aplicando este algoritmo puede encontrar otros tipos de propiedades también comparten una similitud con los nodos alcanzados bajo las propiedades, estos elementos no se deben descargar debido a que pueden ser usado como un conjunto de datos no presentes al contexto, pero relevantes al momento de categorizar grupos, por lo tanto la conclusión sería que de que una elección de dcterms: sujeto y skos no se discrimina radicalmente al único dominio que establecemos lo que lo convierte en un buen proceso de exploración.

3.5.3 Proceso de contextualización de resultados.

Para poder determinar el valor de similitud entre 2 recursos se debe considerar el peso de las aristas entre ellos. Si tenemos una uri1 y una uri2 se debe comparar la cantidad en que se encuentran relacionadas cada uno de sus elementos tanto en un nivel de análisis de la información textual contenida dentro de ellos y también la explotación de los vínculos de relación externos que salen de sus estructuras.

En este proceso es necesario evaluar qué tan precisa es la correspondencia semántica entre dos recursos de la DBPedia.

En el caso que trabajamos con dos recursos uri1 y uri2, es necesario verificar el número de entidades o páginas web que contiene o que han sido etiquetadas con el valor de su rdfs: label asociada a cada uri (uri1 y uri2).

El proceso de análisis de contexto se refiere al proceso de identificar ese subconjunto de nodos que están representados en la DBPedia bajo nuestro contexto de interés.

Por ejemplo si se establece un contexto de dominio específico como: programación o informática nos deberíamos centrar solo en la clasificación de cada subgrafo en la DBPedia cuyas relaciones deben estar de algún modo relacionadas con el contexto inicial establecido.

Al lograr establecer un contexto, se debe comprobar que la uri pertenece a ese dominio clasificando los nodos representativos bajo ese contexto, este proceso de categorización se puede realizar aplicando un algoritmo de análisis de contextos factible para la DBPedia.

Durante la exploración bajo las premisas de los algoritmos de Exploración y de Rankeo de entidades (1 o 2 dependiendo de la forma de inicialización de la exploración) cada nuevo descubrimiento debe ser evaluado para validar que se encuentre dentro del contexto, esto se puede lograr comparando las categorías y si la puntuación de cada umbral es superior al valor establecido podemos decir que aquel elemento se aleja del contexto indicado y podríamos proceder a descartarlo o agruparlo en una nueva categoría.

Entrada:
n: Un nodo r
Salida:
Verdadero si r está considerado parte de un contexto, falso al ser contrario
(Todo nodo está establecido como: en_contexto de manera inicial.

```
1   Si r.en_contexto  == verdadero
2       retorna verdadero
3   Fin
4   s = 0:
    (Aquí se consideran las entidades, categorías o estructuras de la DBPedia
    más populares o relacionadas en el proceso de exploración)
5   C = obtieneContexto();
6   Por cada nodo n ∈ C hacer
7           s = s + similaridad (r.URI, n.URI)
    (Si el valor de similaridad existente entre r y el valor representativo del
    contexto una URI es mayor en proporción a un umbral establecido se lo
    considera a r como parte del contexto)
8           Si s es mayor o igual a Umbral entonces
9               r.en_contexto = verdadero;
10            actualizar r;
11            devuelve verdadero;
12          Fin
13  Fin
14  r.en_contexto = falso;
15  r.rankeo = verdadero;
16  actualizar r;
17  devuelve falso;
```

Tercer Algoritmo: Determinación de contexto - Tomado: DBPedia Ranker – Not Only Tags: http://www-ictserv.poliba.it/publications/2010/MRDD10b/icwe2010.pdf

El umbral se considera como la parte más importante en la deducción y rankeo de una entidad como relacionada con otro elemento, debido a que este elemento determina el valor de similitud entre los recursos, en otras palabras su peso asociado a la arista entre dos entidades.

Para poder desarrollar este proceso de análisis de similitud entre entidades se puede aplicar algunas fuentes para extracción, unos pueden ser los motores de búsqueda como lo es Google, también podemos utilizar las nubes de etiquetas como fuente

como los presentes en Delicious o elementos de información de las nubes presentes en páginas web, pero para el presente trabajo se ha considerado la información contenida en DBPedia. En el desarrollo del presente trabajo se ha implementado bajo DBPedia SpotLight.[44]

Para poder establecer la selección es necesario que evaluemos la similaridad de relaciones entre las dos URI's con respecto a la fuente de extracción externa (DBPedia), aquí hay que considerar que los valores que representen a cada entidad comparada en número debe estar fuertemente asociada o contener información similar, de esta manera los recursos uri1 y la uri2 serán la fuente de comparación y relación1 y relación2 representan la similaridad en enlaces hipertextuales mapeados en DBpedia considerando la propiedad wikiPageWikiLink.

Si logramos establecer que un recurso1 tiene un enlace a otro recurso2 sabremos entonces que uri1 y ur2 contienen algún tipo de relación.

La ecuación quedaría de la siguiente manera:

$$\text{similaridad}\ (\text{uri1}, \text{uri2}, \text{fuente de información}\ (\text{DBpedia}) = \frac{\text{relación1}, \text{uri2}}{\text{relación1}} + \frac{\text{relación1}, \text{uri2}}{\text{relación2}}$$

Ecuación para determinar relación de similaridad
Autor: Pablo Malla

Sabemos que relación1 y relación2 representa las veces que una información (recurso de DBpedia) está contenida por el label del recurso entre uri1 y uri2, de esta manera si nuestro resultado es simétrico el valor devuelto estará comprendido entre 0 y 2 lo que nos facilita establecer que entre las 2 URI's hay una relación entre recursos compartidos.

Existe una condición de extracción en la cual un recurso que es seleccionado como nodo único no posee una relación directa con un nodo u otro conjunto de nodos posteriores, por lo que el recorrido se cortaría de manera esporádica, sin realizarse un recorrido extractivo completo.

Para solventar el inconveniente de recorrido corto en una red de entidades debido a su carencia de selección de nodo padre se ha creado un elemento que nos permita encontrar entidades en base a la relación de categorías encontradas de la entidad, de esta manera al expandir a un nivel más profundo la estructura de la entidad y

[44] https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki

seleccionar del “subjet” su propiedad de categoría podremos permitir el recorrido de una red de relaciones adjunta a las categorías que representan la entidad.

La función de consulta sería la siguiente:

```
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT  ?recurso (str(?definicion ) as ?abstract)(str(?label) as ?titulo)
WHERE
{   ?recurso dcterms:subject <http://dbpedia.org/resource/Category:Melanthiaceae>.
   ?recurso dbpedia-owl:abstract ?definicion.
  ?recurso rdfs:label ?label.
   FILTER (langMatches(lang(?label), "en")   && langMatches(lang(?definicion), "en"))
 } LIMIT 20
```

Figura 23. Consulta auxiliar de recorrido al carecer de un recurso contextual inicial –
Autor: Pablo Malla

Este elemento de consulta mostrado en la figura nos permite ingresar en el flujo de extracción de entidades (recorrido, determinación de contexto y extracción de entidades) a las entidades que no encuentren un nodo padre inicial al ser un recurso corto, y al expandirse a los recursos de sus categorías y seleccionando candidatos con el proceso de selección podemos iniciar el proceso de extracción completo, aunque se debe indicar que los recursos pertenecen al contexto de categorías relacionadas a una entidad, y no los recursos directos relacionados, se consideraría un paso de recomendación de recursos relacionados en base a categorías.

Un requerimiento que se necesita implementar al esquema es determinar el idioma de cada entidad, para poder seleccionar entidades basados en ello, de esta manera se puede evitar duplicidad o colapso al momento de encontrar elementos en un proceso extractivo al considerar un sinnúmero de idiomas que pueden representar un mismo concepto, si seleccionamos el idioma, en nuestro caso Inglés, se puede utilizar solo ese contexto, para ello se ha implementado a todo el proceso:

```
String s2 = "PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"
    + "\n"
    + "SELECT  ?uri ?label\n"
    + "WHERE\n"
    + "  { ?uri rdfs:label ?label\n"
    + "    FILTER (?uri = <"
    + q
    + ">)"
    + "  }\n";
```

Figura 24: Extracción idioma para selección de entidades. – Autor: Pablo Malla

En el proceso extractivo para poder visualizar de mejor manera los conceptos de entidades recuperadas se requiere también devolver el abstracts de cada concepto con el fin de presentar al usuario final información más específica y concreta, para ellos se ha implementado:

```
String s2 = "PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"
    + "\n"
    + "SELECT  ?abstract\n"
    + "WHERE\n"
    + "  { \n"
    + " <"
    + q
    + "> <http://dbpedia.org/ontology/abstract> ?abstract ."
    + "  }\n";
```

Figura 25: Consulta extracción Abstracts de entidades – Autor: Pablo Malla

Finalmente, el encontrar categorías de cada entidad permite conocer la estructura a la que puede pertenecer una entidad extraída, mostrado en la figura 26.

```
String s2 = "PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"
    + "PREFIX dcterms: <http://purl.org/dc/terms/>\n"
    + "PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>\n"
    + "SELECT  ?recurso (str(?definicion ) as ?abstract )(str(?label) as ?titulo) \n"
    + " WHERE\n"
    + " {"
    + "  ?recurso dcterms:subject <"+q+ ">. \n"
    + "  ?recurso dbpedia-owl:abstract ?definicion. \n"
    + "  ?recurso rdfs:label ?label. \n"
    + "    FILTER (langMatches(lang(?label), \""+ l + "\")   && langMatches(lang(?definicion), \"" + l + "\")) \n"
    + "  }LIMIT " + c +"\n";
```

Figura 26: Extracción categorías de entidades recuperadas –
Autor: Pablo Malla

3.6 Fase implementación del servicio en base al análisis de entidades recuperadas.

A continuación se aplica el servicio de mapeo de datos y la primera extracción de recursos relacionados, los cuales nos permitirán partir y comenzar el proceso jerárquico de extracción.

El servicio recibe un atributo “tag” en el cuál puede contener una keyword o palabra a ser procesada, una etiqueta que establece el idioma y un contexto, el cual puede establecer los resultados que se requieren obtener.

El proceso de búsqueda requiere que se rellenen 3 elementos:

- ?tag = es la palabra clave o elemento que requiere ser encontrado
- lang= el idioma en el cual devolverá resultados (español o inglés)
- contextstr= el tipo de contexto en los que se centrará la búsqueda para devolver resultados

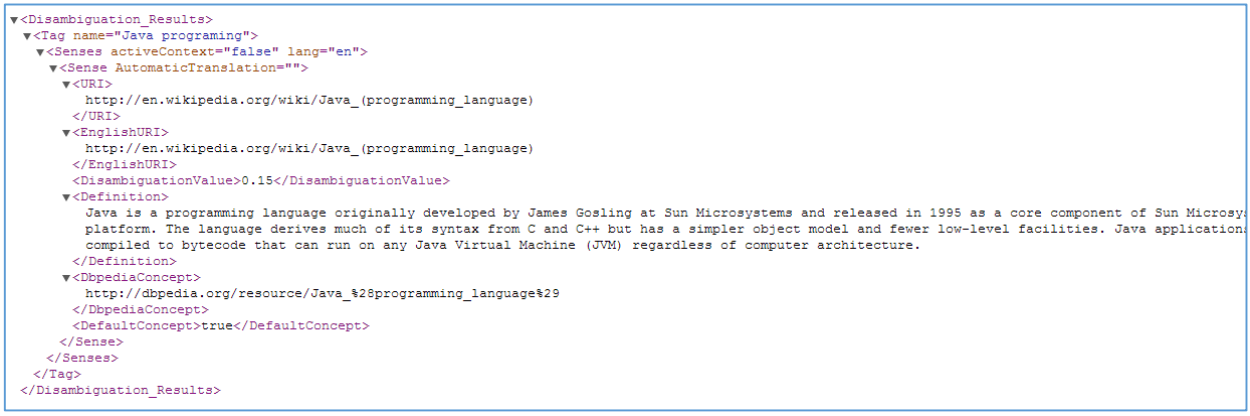
Los resultados que devuelve son:

- Tag: El keyword o palabra clave que requiere ser encontrado.
- URI: La dirección de páginas de Wikipedia que contiene una similaridad con el tag o keyword ingresado.
- English URI: devuelve la uri correspondiente al idioma que ha sido establecido al momento de realizar la consulta.

- DisambiguationValue: determina el valor de precisión en la desambiguación del recurso que ha sido obtenido.
- Definition: Devuelve una definición de cada uno de los resultados encontrados, para cada elemento recupera un descripción.

3.7 Extracción relaciones de recursos por categoría.

La figura 27 nos muestra un ejemplo de un descubrimiento de una entidad relacionada, para el caso de la búsqueda “java Programing” podemos ver que el resultado nos devuelve una URI del recurso, la URI del idioma, en este caso Inglés, un valor de desambiguación, la definición del recurso y el concepto de DBpedia.



```
▼<Disambiguation_Results>
 ▼<Tag name="Java programing">
  ▼<Senses activeContext="false" lang="en">
   ▼<Sense AutomaticTranslation="">
    ▼<URI>
       http://en.wikipedia.org/wiki/Java_(programming_language)
     </URI>
    ▼<EnglishURI>
       http://en.wikipedia.org/wiki/Java_(programming_language)
     </EnglishURI>
     <DisambiguationValue>0.15</DisambiguationValue>
    ▼<Definition>
       Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsy
       platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java application
       compiled to bytecode that can run on any Java Virtual Machine (JVM) regardless of computer architecture.
     </Definition>
    ▼<DbpediaConcept>
       http://dbpedia.org/resource/Java_%28programming_language%29
     </DbpediaConcept>
     <DefaultConcept>true</DefaultConcept>
   </Sense>
  </Senses>
 </Tag>
</Disambiguation_Results>
```


Figura 27: Estructura inicial de una entidad básica
Autor: Pablo Malla

Cuando obtenemos una primera estructura, debemos realizar un análisis de su composición, en este caso de los resultados obtenidos se utiliza el valor de desambiguación, el cual nos permitirá escoger, organizar y determinar de la serie de entidades relacionadas devuelvas las específicas.

El valor de desambiguación y el valor del contexto son críticos en este proceso, debido a que ellos determinan la correcta selección de entidades extraídas y dan inicio al proceso continuo de extracción de recursos de una manera explícita.

Este proceso se compone de tres sub procesos, extracción inicial como punto de partida basado en el Keyword o recurso clave inicial de búsqueda, desambiguación de resultados, selección de recursos y procesamiento de entidades (URI’s) recuperadas para un nuevo proceso de extracción.

El nuevo proceso de extracción ya ha sido especificado para cada uno de los keywords trabajados, pero con la extracción de sus URI's relacionadas a sus keywords realizamos un nuevo proceso en base a estos elementos recuperados.

En la figura 28 se puede ver un conjunto de entidades encontradas en base a la relación inicial considerada, el caso de "java programing" de la figura 27, en este caso se extrae nuevamente los elementos descritos anteriormente, además del valor del nivel al que representa este proceso, en el ejemplo se puede revisar que representa un nivel 1 de entidades, si la entidad en cuestión posee más elementos, el árbol de relaciones se extiende en niveles 1, 2 y 3; no se han consideramos más niveles de profundidad debido a que con una mayor extensión se perderá poco a poco la precisión, un nivel más lejano se extiende hacia otros contextos.

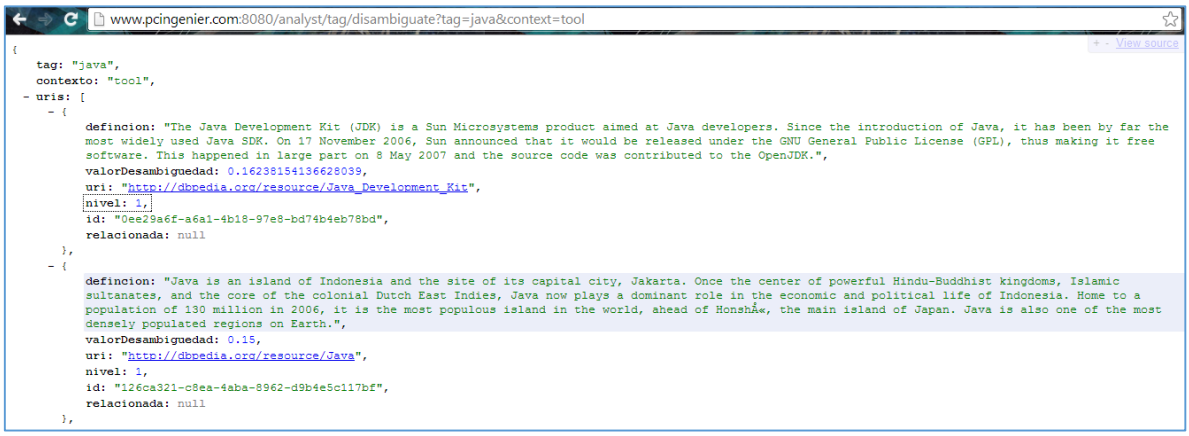


Figura 28: Primera extracción de recursos relacionados en base a Keyword y contexto

Se necesita realizar un mapeo de la estructura devuelta y la utilización de los valores de desambiguación devueltos para la selección de los recursos a ser procesados. Al acceder a las características de cada entidad considerada podemos conocer que elemento se puede considerar como dato clave para su visualización, la definición, la URI y su valor de ambigüedad son los valores críticos de cada entidad.

```java
/***
 * De las urls relacionadas desambigua utilizando diferentes criterios
 **/
@RequestMapping(method = RequestMethod.GET, produces = "application/json", value = "/disambiguate")
public @ResponseBody
ExtractorResultado consultartTagsDisambiguate(final String tag,
                final String context) throws MalformedURLException, IOException,
                JAXBException {

        Disambiguation_Results response = this.consultartTagsRealacionados(tag,
                        context);

        ExtractorResultado extractorResultado = new ExtractorResultado();
        extractorResultado.setContexto(context);
        extractorResultado.setTag(tag);
        extractorResultado.setUris(new ArrayList<UriRelacionada>());

        for (Sense item : response.getTag().getSenses().getSense()) {
                UriRelacionada uri = new UriRelacionada();
                uri.setDefincion(item.getDefinition());
                uri.setValorDesambiguedad(Double.parseDouble(item
                                .getDisambiguationValue()));
                uri.setUri(item.getDbpediaConcept());
                extractorResultado.getUris().add(uri);
        }
```

Figura 29: Código – Proceso inicial extracción de entidades

En la figura 29 se explica el proceso de filtrado que se realiza a cada elemento. Usando el valor disambiguationValue en una primera instancia validamos si entre todos los recursos devueltos su valor es ‘0’, en este caso se puede evidenciar que todos los recursos pertenecen a un mismo concepto, por lo que la selección se daría a los 2 primeros recursos por ser los más óptimos.

```java
/***
 * Aplicar criterios para desambiguar los resultados
 **/
public static ExtractorResultado calcularUrisRelacionadas(ExtractorResultado extractor){
        List<UriRelacionada> uris = new ArrayList<UriRelacionada>();

        if(isTodoCeroUriRelacionada(extractor.getUris())){
                int contador = 0;
                for (UriRelacionada item : extractor.getUris()) {
                        if(item.getValorDesambiguedad() == 0 && contador < 2){
                                uris.add(item);
                        }
                        contador++;
                }
```

Figura 30: Extracción de entidades que cumplen la primera regla de desambiguación

La primera regla de desambiguación nos dicta que en base al análisis del valor de desambiguación de un recurso cercano a 0 es el más preciso tomando en cuenta el Keyword procesado y el valor del contexto. Debemos considerar que no todos los elementos que enviemos a buscar pueden tener o no un recurso en DBPEDIA, por lo tanto, en este caso si devuelve 1 o 2 recursos, serán automáticamente tomados para el posterior análisis. Existen casos en que la recuperación de un recurso no devuelve un recurso sino una categoría, por tal motivo en esos casos se implementó la extracción de categorías para ser considerado.

```
}else if(existRango01(extractor.getUris())){
        for (UriRelacionada item : extractor.getUris()) {
            if(item.getValorDesambiguedad() > 0 && item.getValorDesambiguedad() < 1){
                uris.add(item);
            }
        }
}else{
        int contador = 0;
        for (UriRelacionada item : extractor.getUris()) {
            if(item.getValorDesambiguedad() == 1 && contador < 2){
                uris.add(item);
            }
            contador++;
        }
}

extractor.getUris().clear();
extractor.getUris().addAll(uris);
return extractor;
```

Figura 31: Extracción de entidades que cumplen la segunda regla de desambiguación

Usando el valor disambiguationValue debemos verificar si existe un rango entre 0 y 1, los valores más cercanos a 0 y un valor intermedio de 0,05 son los más precisos. Los valores más cercanos a 1 se alejan de la similaridad entre recursos. Luego de realizar la selección adquiere 2 recursos.

```
/***
 * De cada url relacionada a un tag, obtiene todas las urls relacionadas utilizando el servicio de SINSIN
 **/
@RequestMapping(method = RequestMethod.GET, produces = "application/json", value = "/related")
public @ResponseBody
ExtractorResultado consultartRelacionado(final String tag,
            final String context, ModelMap model) throws MalformedURLException,
            IOException, JAXBException {

        ExtractorResultado extractorResultado = this
                .consultartTagsDisambiguate(tag, context);

        UrlController urlController = new UrlController();

        for (UriRelacionada item : extractorResultado.getUris()) {
            item.setRelacionada(new ArrayList<UriRelacionada>());
            ResultadoUrlTags resultado = urlController
                    .consultarUrlRelacionadas(item.getUri());
            if (resultado.getSystem() != null) {
                for (UriTag itemTag : resultado.getSystem()) {
                    item.getRelacionada().add(
                            new UriRelacionada(itemTag.getDescription(),
                                    itemTag.getRelevance(), itemTag.getUri()));
                }
            }
```

Figura 32: Envío de entidades extraídas al proceso de extracción basado en URI's

Cuando se ha seleccionado URI's en base al proceso de desambiguación se procede a enviar cada una de las URI's obtenidas para su posterior recorrido y extracción de URI's relacionadas.

Como se visualiza en la figura 33, se ha logrado encontrar más entidades relacionadas derivadas de cada una de las entidades de cada nivel, en este proceso de descubrimiento de entidades se recuperan de igual manera los datos de definición, valor de des ambigüedad, id de la URI, la URI del recurso y el nivel al que representa cada entidad, podemos ver que en el ejemplo se han encontrado entidades de nivel 1, nivel 2 y nivel 3.

Figura 33: Nivel de extracción profundo de entidades relacionadas

3.8 Descripción de una recomendación y método de modelamiento.

Para cada iteración en la extracción de relaciones entre recursos, es requerido en cada proceso una nueva ejecución de desambiguación para reorganizar los resultados en bases concisas y el árbol de relaciones no se expanda fuera de su dominio principal.

El proceso de extracción por niveles nos asegura que se explota correctamente todo el árbol, creciendo de manera horizontal y devolviendo en cada ramificación lo relacionado a un recurso y termina cuando ya no encuentra aristas interconectadas, podemos finalmente graficar estos resultados y almacenarlos para su posterior uso, categorización, recomendación, o procesado de resultados que puedan servir para otros procesos que exijan data concreta a un tema determinado.

La correcta visualización de resultados puede resaltar los nodos relevantes, dar ideas sobre grupos que se encuentran cohesionados entre si dentro de la red o simplemente la capacidad de mostrar las grandes cantidades de información de forma vistosa.

El fin de este proceso es preparar la extracción de entidades relacionadas en base a un conjunto de bases candidatas, un proceso de desambiguación inicial para cada entidad inicial, para en base a ese grupo de entidades padre, realizar el proceso extractivo de entidades relacionadas con la aplicación de los algoritmos previamente explicados.

```java
public @ResponseBody
ExtractorResultado consultartRelacionado(final String tag,
        final String context, final int nivel, final String l) throws MalformedURLException,
        IOException, JAXBException {

    ExtractorResultado extractorResultado = this
            .consultartTagsDisambiguate(tag, context, l);

    UrlController urlController = new UrlController();

    List<UriRelacionada> itemsNivel = new ArrayList<UriRelacionada>();
    itemsNivel.addAll(extractorResultado.getUris());

    int contador = 1;
    while (contador  <= nivel) {
        List<UriRelacionada> itemsNivelAfectado = new ArrayList<UriRelacionada>();
        for (UriRelacionada uriNivel : itemsNivel) {
            if(uriNivel.getNivel() == contador){
                itemsNivelAfectado.add(uriNivel);
            }
        }

        for (UriRelacionada uriAfectada : itemsNivelAfectado) {
            uriAfectada.setRelacionada(new ArrayList<UriRelacionada>());
            if(!uriAfectada.getUri().isEmpty()){
                ResultadoUrlTags resultado = urlController
                        .consultarUrlRelacionadas(uriAfectada.getUri());
                if (resultado.getSystem() != null) {
                    for (UriTag itemTag : resultado.getSystem()) {
                        UriRelacionada uriNueva = new UriRelacionada(itemTag.getDescription(),
                                itemTag.getRelevance(), itemTag.getUri(), contador+1);
                        uriNueva.setLabel(itemTag.getLabel());
                        uriAfectada.getRelacionada().add(uriNueva);
                        itemsNivel.add(uriNueva);
```

Figura 34: Código de extracción profundo de entidades relacionadas

Al momento de recuperar entidades iniciales se puede integrar el proceso de extracción que implica a los 3 algoritmos de recorrido, rankeo y contexto, eso implica el recuperar entidades relacionadas en base a un contexto exclusivo y de esta manera evitar ambigüedades.

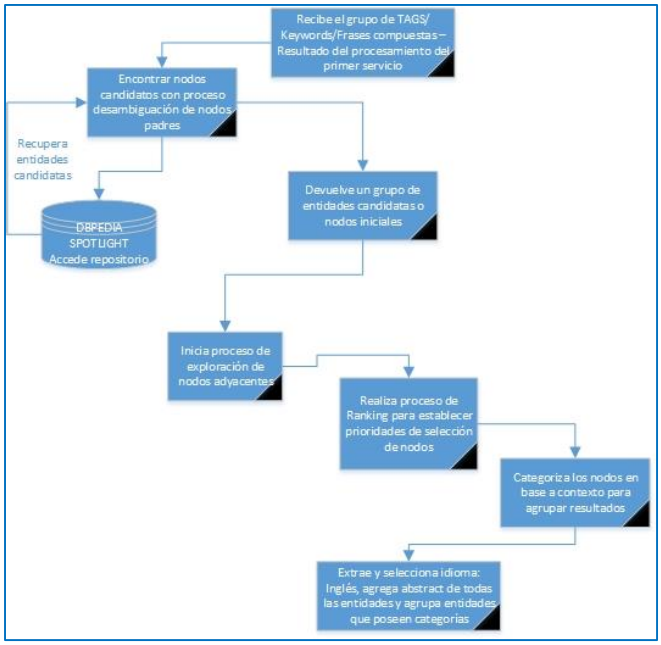


Figura 35: Flujo final de proceso extractivo – Extracción entidades candidatas y relacionadas – Autor Pablo Malla

Con la integración de éste proceso extractivo de entidades, se logra recuperar los elementos que finalmente podrán ser mostrados y manipulados para un usuario final que quiera recuperar y conocer la red de relaciones que puede existir en base a un grupo de tags o recursos que considere.

Cada servicio recibe las siguientes características, las cuales serán explicadas a detalle en los apartados siguientes:

Servicio	Recibe	Devuelve	Se Integra
Pre procesamiento	Grupo de Tags/Keywords o frases compuestas ingresado por el usuario	Paquete organizados de recursos pre-procesado textualmente	Recuperación entidades candidatas
Recuperación entidades candidatas	Paquete organizados de recursos pre-procesado textualmente	Grupo de URI'S candidatos recuperados en base al paquete de búsqueda inicial.	Extracción entidades relacionadas
Extracción entidades relacionadas	Grupo de URI'S candidatos	Red de relaciones o entidades relacionas completa en base a las entidades ingresadas por el usuario inicial	Graficador - Recomendador
Graficador	Red de entidades relacionadas extraída	Visualización gráfica de toda la red de relaciones extraída	
Recomendador	Red de entidades relacionadas extraída	Estructura organizada de los recursos recuperados, determinando en base a un índice de colores las relaciones entre entidades	

3.9 Conclusiones y discusión del capítulo III.

El reconocimiento de patrones de relación es el núcleo principal del proceso de extracción, con la aplicación de los 3 algoritmos de recorrido, reconocimiento y contextualización se puede conocer que entidades rodean a cada nodo, y también que extensión de conocimiento posee una entidad, si consideramos estos resultados

podremos saber qué entidad carece de extensión y se puede explotar aún más, se puede conocer qué entidad contiene a su alrededor conocimientos y se la puede considerar como crítica para una investigación.

Se considera el uso de los recursos presentes en la Wikipedia en inglés para el proceso de encontrar recursos relacionados por lo tanto el proceso devuelve entidades con su descripción en inglés.

Cuando se ha logrado descubrir un conjunto de entidades relacionadas en base a patrones ya se puede considerar cual sería el siguiente paso para su aprovechamiento, como la visualización, el almacenamiento o generar información externa sobre la existencia de una entidad y su red de relaciones para posteriormente establecer aún más trabajo sobre una entidad, abriendo una gran oportunidad de explotar los recursos, aumentar su dimensión o tomar decisiones en base a su complejidad ya descubierta.

CAPITULO IV
CONSTRUCCIÓN DE COMPONENTE WEB QUE IDENTIFIQUE ENTIDADES RELACIONADAS

4.1 Resumen.

Al plantearse la creación de servicios independientes se ha logrado cumplir correctamente con el flujo inicial planteado ya que cada componente está interconectado y sus resultados cumplen con el requerimiento solicitado y asociado de componente posterior, el proceso de pre-procesamiento envía los recursos al siguiente servicio de extracción de URI's a primer nivel, el cual envía las URI's al siguiente proceso de recorrido y finalmente la extracción y recomendación de resultados.

El presente capítulo se centra en la descripción de los componentes, su integración y la forma que afectan el flujo de trabajo en correlación con cada uno de los servicios.

Este capítulo describe la integración de los procesos implicados desde el ingreso de la lista de entidades por parte del usuario, la selección de entidades del paquete generado para el proceso de extracción inicial, visualización de resultados del primer proceso extractivo, la visualización de resultados del segundo proceso de extracción de entidades relacionadas, la visualización dela gráfica obtenida y la recomendación de los recursos en base a la intensidad de relaciones que contiene, todo es un servicio general que los integra para facilitar la ejecución e interpretación de los datos recuperados.

4.2 Herramientas aplicadas para construcción de componente.

Al obtener una idea completa de los procesos que son requeridos construir y tratar al momento de trabajar con entidades, se puede enfocar en las tecnologías que se pueden implementar para lograr el objetivo de encontrar recursos relacionados, la creación de módulos como servicios recaen en la aplicación de herramientas de construcción como JBoss[45] 7 y Spring 3[46], bajo un lenguaje de java.

La construcción bajo un patrón de MVC (modelo, vista, controlador), y para acceder, manipular y exponer servicios se ha visto recomendable utilizar REST[47], finalmente y en

[45] http://www.jboss.org/
[46] https://spring.io/
[47] http://asiermarques.com/2013/conceptos-sobre-apis-rest/

base al análisis de nuestra necesidad de visualizar los recursos finales ha sido necesario aplicar bootstrap[48], jquery[49], thymeleaf [50]como interfaz gráfica.

- BootStrap, es un aplicativo cuya aplicación en el presente trabajo se debe a que se permite acoplar para trabajar bajo el modelo vista controlador y porque se ha considerado el acceso al aplicativo desde un dispositivo móvil.
- Thymeleaf es una librería Java que implementa un motor de plantillas de XML/XHTML/HTML5 que puede ser expandible a otros formatos, y su uso nos permite utilizar un desarrollo web así como entornos no web.

4.3 Esquema del flujo de proceso.

Para el trabajo con el flujo completo fue necesario esquematizar todas las pantallas que comprenderán el aplicativo.

Los prototipos han sido diseñados de manera que un usuario pueda utilizar cada servicio de la forma más simple posible, la figura ; los proceso extractivos disponibles en la actualidad requieren de una manipulación manual y de un conocimiento de nivel medio para poder encadenar las extracciones e interpretar los resultados, por lo que con la implementación del flujo en forma de aplicativo se pretende solventar esos pasos.

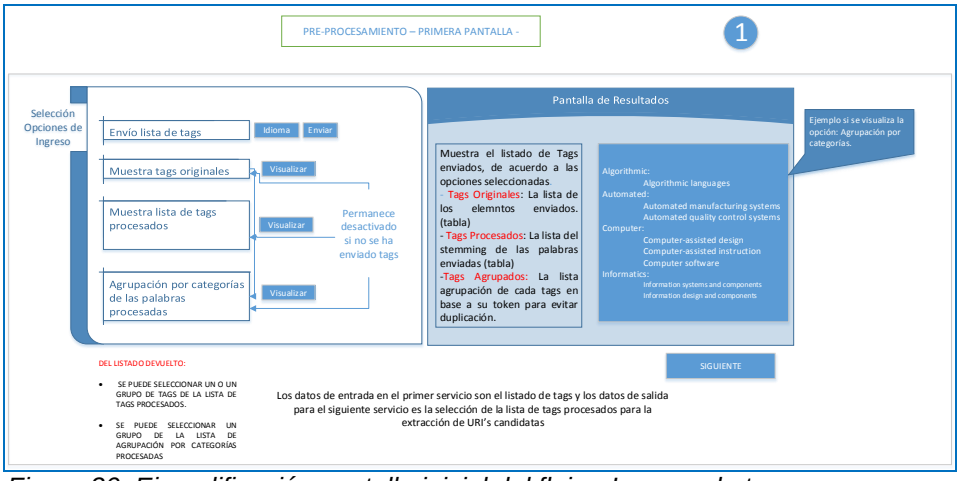


Figura 36: Ejemplificación pantalla inicial del flujo - Ingreso de tags.
Autor: Pablo Malla P.

El primer flujo de procesos establece la preparación de la data a ser trabajada, con el envío de una lista de tags podemos determinar qué conjunto de datos van a ser

[48] http://getbootstrap.com/2.3.2/
[49] https://jquery.com/
[50] http://www.thymeleaf.org/

utilizados, que elementos van a pasar por el primer proceso de procesado y organización en conjunto de datos.

Este conjunto será procesado con la técnica de stemming y lemmatización con el fin de preparar los recursos específicos para una siguiente extracción, y así evitar perder el dominio o el contexto de lo que queremos encontrar. Esta primera estructura facilita el envio de los tags del usuario.

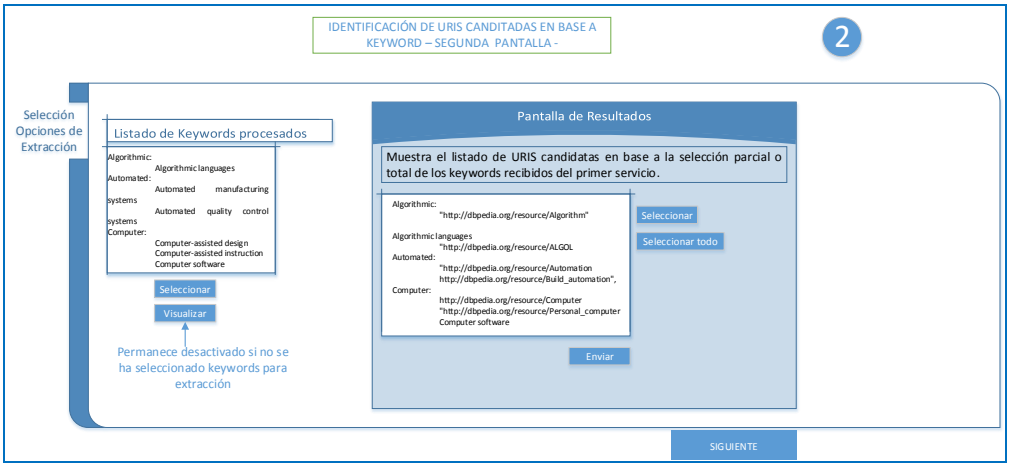


Figura 37: Selección de tags procesados para la posterior extracción de URI's candidatas

Una vez que contamos con el conjunto de tags o keywords relacionados, podemos incluso una vez más validar que recurso, utilizaremos en base a por ejemplo, él envió de un tag individual, el uso de tags divididos de una frase compuesta (análisis de texto) o el uso de un tag considerado como una entidad al ser una frase compuesta y completa.

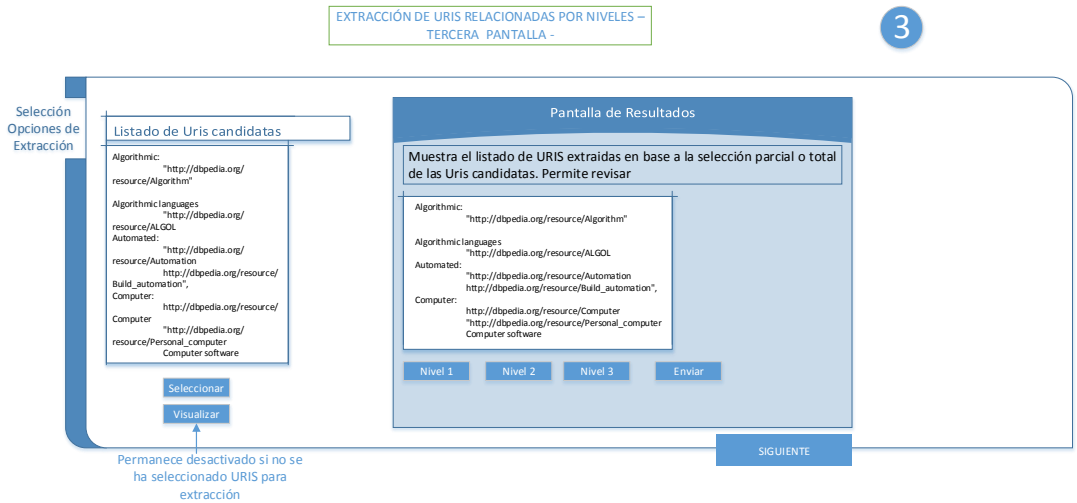


Figura 38: Muestra de resultados del proceso de extracción de recursos relacionados

En este flujo podemos establecer el conjunto final de elementos que nos servirán como punto de partida.

4.4 Integración de servicios.

Con la implementación de los servicios de Stemming, de envío de tags/keywords, de extracción de entidades relacionadas candidatas, de extracción de niveles de relaciones entre entidades, es requerido facilitar el envío y la visualización de los datos con los que se trabaja.

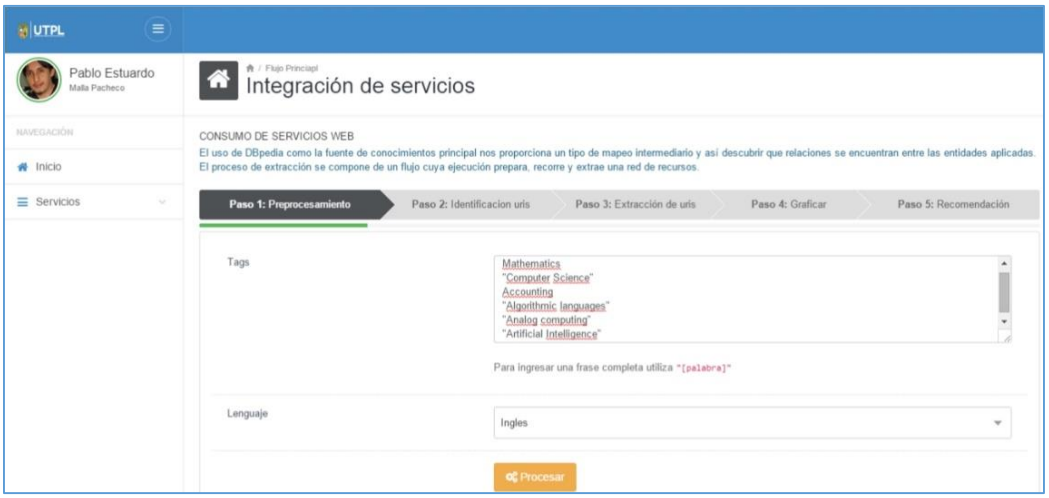


Figura 39: Servicio de envío de tags/keywords para iniciar flujo de extracción

La primera página consta del ingreso de elementos para su posterior procesamiento, en este paso se ha permitido el ingreso en un campo de texto de un conjunto de keywords/tags.

Este conjunto de entidades poseen 2 características especiales, si con el envío del conjunto se establece una frase compuesta, como por ejemplo: Computer Science, se ha establecido que al utilizar los símbolos de comillas (“”), la entidad será considerada como una entidad única, y su representación permitirá la siguiente extracción en base a toda su estructura.

Si en este campo se ingresa un conjunto de términos, cada uno de ellos pasará por el proceso de Stemming y Lematización, esto permitirá ordenar, agrupar y preparar toda la terminología, para el correcto flujo de extracción.

En la parte inferior del servicio se presentarán los resultados de orden y categorización de los términos enviados para su visualización.

Una vez que hemos realizado el proceso de limpieza y organización de la data inicial para su procesado, el servicio web nos permite seleccionar de todo el conjunto de data

ingresada y pre-procesada, que data queremos trabajar y encontrar, de esta manera nos establecemos el conjunto de elementos que vamos a utilizar.

Figura 40: Resultado del pre-procesamiento de tags/keywords

El conjunto de palabras pre-procesadas se muestran en el grupo siguiente al procesamiento del texto ingresado, ahí podemos ver el resultado y si requerimos enviar la información o revisar nuevamente el grupo de elementos a utilizarse.

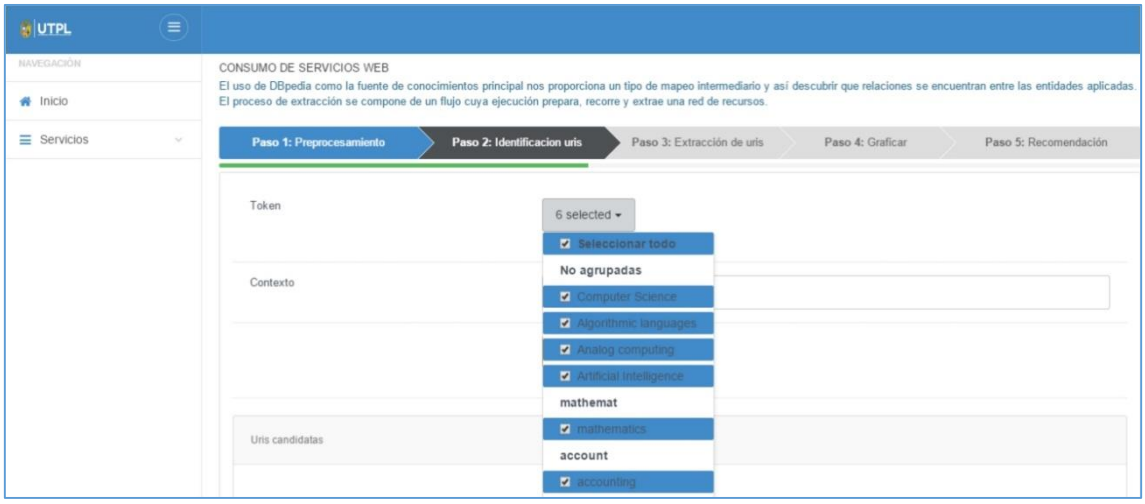


Figura 41: Selección de tags procesados (entidades)

En el combo box podemos determinar qué grupo de entidades utilizaremos, así como las frases compuestas que se considerarán como una entidad y las entidades separadas e frases compuestas ingresadas.

Luego del proceso de selección de los tags, el servicio nos permite procesar el requerimiento para seleccionar un primer conjunto de entidades relacionadas, este primer conjunto ha sido denominado como “URIS CANDIDATAS”, este resultado se basa en el proceso de desambiguación ya descrito en la fase de análisis de entidades.

Figura 42: Resultado de URI's candidatas que cumplen las reglas de desambiguación

Para la visualización de este resultado, se ha determinado organizarlo en base al nombre del tag/keyword/frase compuesta, la etiqueta que la describe en base a su idioma, la definición de la entidad extraída, el valor de ambigüedad que nos permitió encontrarla y seleccionarla, la URI relacionada (el que será nodo inicial), el nivel de extracción al que pertenece y un id.

Figura 43: Resultado de URI's relacionadas, ramificaciones a partir de las candidatas

El siguiente proceso de extracción ya nos muestra todo el grupo exacto de entidades relacionadas en base a un nodo padre. Este proceso devuelve un gran conjunto de elementos debido a que el procedimiento de extracción se realiza hasta 2 niveles de profundidad, lo que nos da la ventaja de descubrir las relaciones más específicas y que conforman. En esta estructura también se extrae nombre del tag/keyword/frase compuesta, la etiqueta que la describe en base a su idioma, la definición de la entidad

extraída, el valor de ambigüedad que nos permitió encontrarla y seleccionarla, la URI relacionada (el que será nodo inicial), el nivel de extracción al que pertenece y un id, similar al procedimiento anterior, se puede determinar así la profundidad y visualización de entidades que orbitan en una dimensión de información o conjunto de datos.

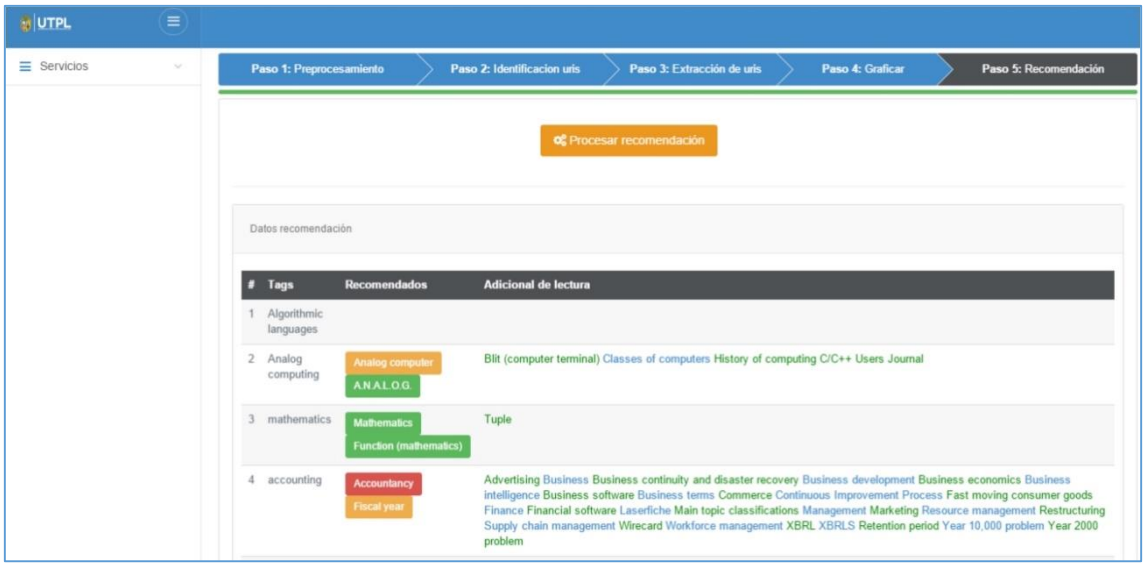


Figura 44: Resultado de URI's relacionadas, ramificaciones a partir de las candidatas

En el proceso final del servicio web de integración de servicios se ha creado una interfaz de categorización y recomendación de los datos devueltos, en la cual en base a una diferenciación de colores, en el cuál el color anaranjado representa el grupo de entidades relacionadas cercanas al contexto original de las entidades, el color verde un contexto bastante simple, es decir que las ramificaciones no llegaron a expandirse de manera extensa, y un color rojo que representa un gran conjunto de entidades relacionadas con el nodo padre y entre ellas, aunque su ramificación sea extensa, es probable que la red de relaciones se divida en varios nodos lo que no determina específicamente el dominio del nodo padre.

Para poder visualizar redes complejas y resultados existen algunas herramientas que nos permiten graficar y mostrar nuestros datos, entre ellas se pueden nombrar: Graphviz[51], Gephi[52], Cytoscape[53], LinkedIn Maps, entre otras. Mediante estas herramientas podemos mejorar aún más los recursos obtenidos y aprovechar correctamente toda la red de información que tenemos disponible.

[51] http://www.graphviz.org/
[52] http://gephi.github.io/
[53] http://www.cytoscape.org/

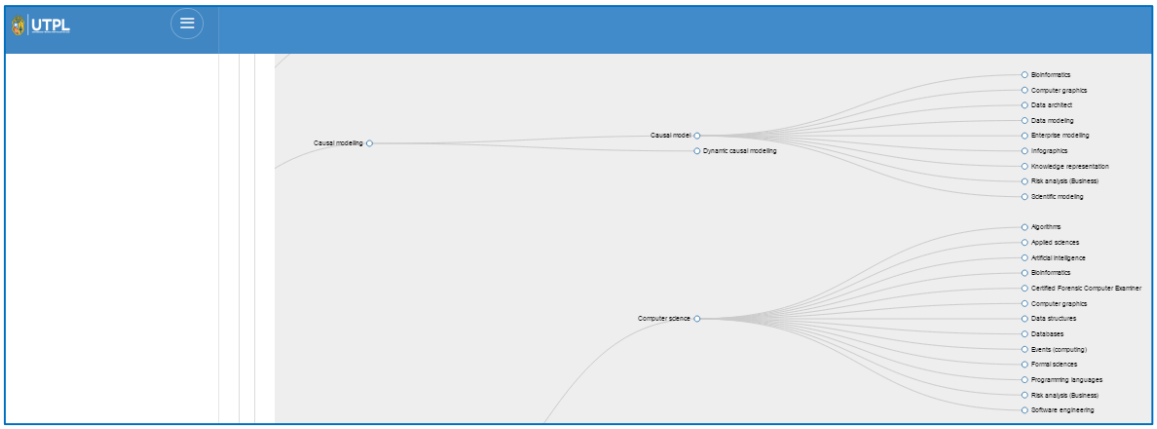


Figura 45: Gráfica recuperación de recursos relacionados

En vista de que una de las metas del network science es obtener recursos e información útil que sea relevante a partir de una red, la visualización de estos resultados se convierte en un factor determinante.

En la visualización de los resultados, cuando el usuario coloca el puntero del ratón sobre un recurso, el aplicativo accede a una parte de la estructura del recurso en donde se mostrará una descripción del recurso extraído, esto permitirá al usuario entender de mejor manera el significado del elemento y debido a que se extrae el nombre o label del recurso, no es necesario salir de la aplicación para discernir sobre la importancia de la entidad.

Se ha creído conveniente la implementación de una pantalla inicial, la explicación de los componentes que conforman el aplicativo, una descripción de los elementos, la función y la finalidad.

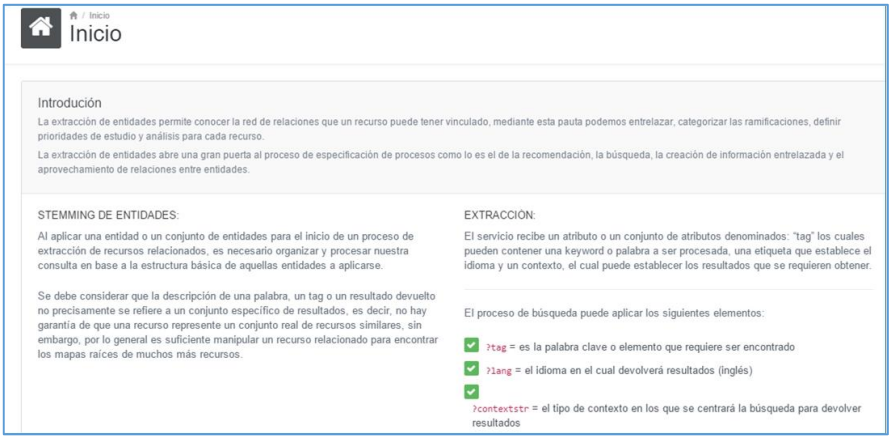


Figura 46. Pantalla de explicación –Wiki- del servicio de extracción de entidades

La visualización de esta descripción facilitará a los usuarios a entender la forma de trabajo del servicio y su acceso no se limitará solo a las personas que conocen de la temática.

4.5 Validación de la propuesta.

Debido al continuo y creciente predominio de las estructuras semánticas en los elementos que conforman el internet, tanto como los documentos científicos, las redes sociales, los repositorios de información, existe un conjunto de oportunidades que muchas herramientas actualmente presentes intentan solventar y cumplir con la extracción de entidades, aprovechar de mejor manera la nueva información, mejora la representación de recursos.

Como uno de los grandes desafíos que se quieren solventar es el de la ausencia de contexto en la información asociada a cada elemento que pudiéramos utilizar, es necesario validar cada uno de los resultados que obtengamos al encontrar entidades y cerciorarnos que cumplan con la correcta dimensión y dominio de trabajo en el que queremos trabajar, incluso porque necesitamos al final una recomendación.

Los procedimientos generales de los extractores de entidades están compuesto de 4 pasos que se pueden establecer como generales estos pasos son:

1. La toma o asignación de un conjunto de muestras o etiquetas para su trabajo.
2. Los cálculos para encontrar vínculos, esto se puede lograr a través de etiquetas, información ya proporcionada por la fuente.
3. Establecimiento y seguimiento de información contextual fuertemente vinculada con la etiqueta y que va en contra de los repositorios de extracción.
4. La aplicación de un procesamiento de lenguaje natural y orden, aplicación de técnicas o algoritmos para descubrir relaciones.

En base a este nuevo enfoque se ha realizado una comparativa con herramientas que realizan un proceso de extracción similar al propuesto en el presente proyecto, que cumplen con este flujo básico.

Es necesario obtener una correcta tasa de precisión en la visualización de relaciones semánticas extraídas, para no perdernos en ramificaciones que lo único que hacen es desviarnos de nuestra correcta dimensión de datos.

La primera herramienta que se ha utilizado para comprar los resultados es Text Razor, cuya descripción nos dice que esta herramienta ofrece un análisis de texto combinando técnicas en el procesamiento de lenguaje natural con tecnología de última generación con una base de conocimiento integral de los hechos de la vida real para ayudar rápidamente descubrir el valor de sus documentos, mensajes de twitter o páginas web.

En la ejemplificación realizada se ha utilizados los mismos datos que he utilizado en el proceso de extracción desarrollado en este trabajo, que son un conjunto de tags iniciales que contienen palabras y frases que hacen referencia a recursos educativos.

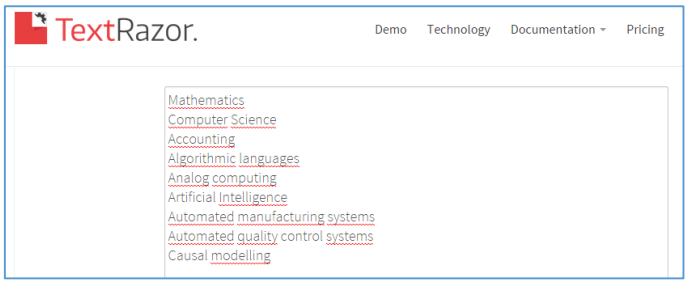


Figura 47. Ingreso de conjunto de tags en pantalla de TextRazor para su posterior análisis

En la primera pantalla podemos evidenciar que el flujo es similar al flujo base de un proceso de extracción, es decir el ingreso de los tags o entidades que requieren ser analizadas para la posterior extracción.

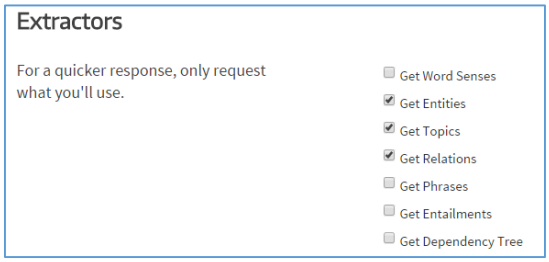


Figura 48: Selección de ingreso de opciones de extracción -

Al ser una herramienta de extracción en base a tags, esta aplicación nos muestra que se puede seleccionar opciones de extracción de las cuales nos centraremos en las más importantes para la comparativa, que es la extracción de entidades y tópicos relacionados.

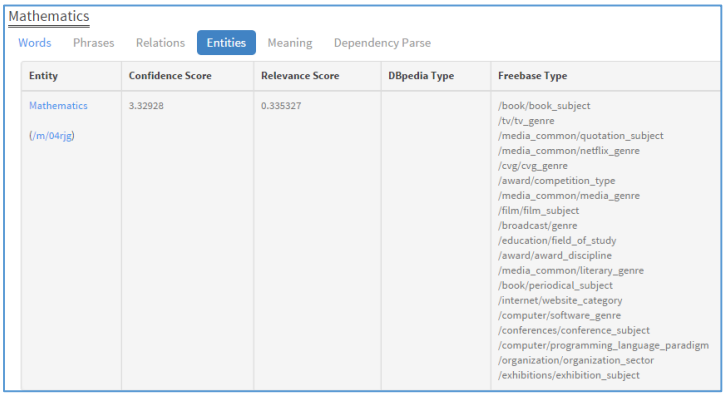


Figura 49: Visualización resultados de extracción (entidades)

En la extracción ya se puede evidenciar una gran diferencia entre la extracción de entidades y tópicos relacionados, en esta extracción podemos evidenciar un mayor

énfasis en la extracción textual y de procesamiento de texto, podemos ver que el proceso de stemming o limpieza de texto tiene mayor relevancia que el proceso de preparación de texto para extracción.

Su mayor diferencia es que si utilizamos una frase completa para encontrar entidades, esta frase es separada en varias partes y toma cada una de las palabras que la conforman en entidades, lo que no permite la extracción de una frase como una sola entidad.

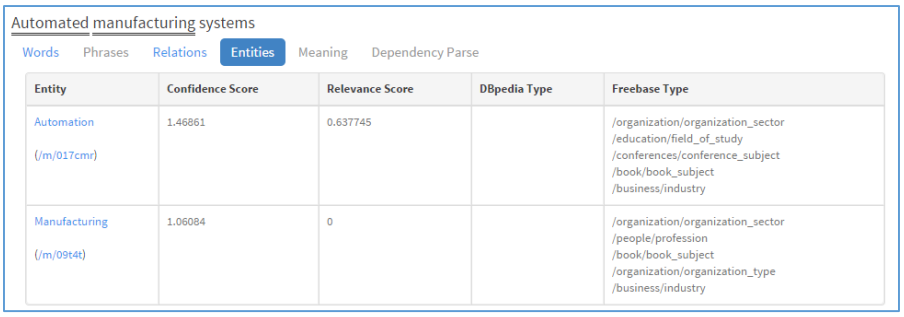


Figura 50: Extracción de recursos (estructura de entidad separada)

En la parte de extracción de tópicos relacionados también se visualiza una gran diferencia, ya que el acceso a la DBPedia es bastante estricto y casi nulo, sin embargo, el uso de Freebase Type tiene bastante relevancia, es más, la mayoría de recursos extraídos pertenecen a esa base de datos.

En la estructura de extracción nos muestra cómo está conformada la entidad, es decir su tipo, un valor de relevancia (Relevance Score) que sería en comparación a nuestro sistema el valor de desambiguación.

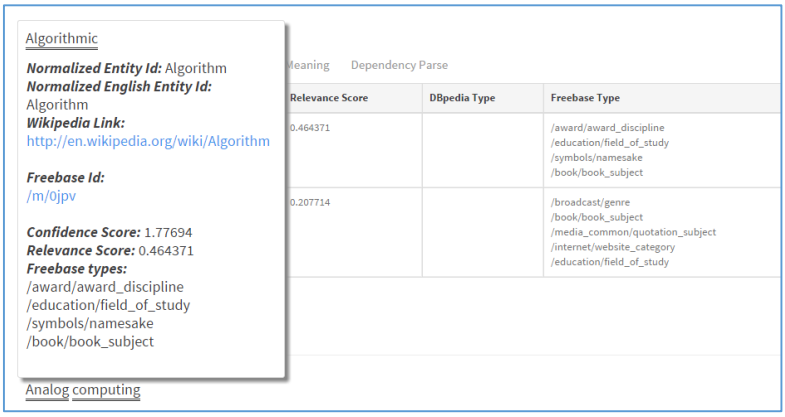


Figura 51: Estructura de la entidad encontrada con TextRazor

Finalmente, nos muestra la estructura de una entidad y sus datos, pero no llega al flujo de recomendación que es requerido en el proceso de tesis, por lo tanto la herramienta

cumple con los flujos de extracción pero no se adaptan conjuntamente para darnos una recomendación de entidades.

La principal validación entre los resultados obtenidos con una sola entidad es la cantidad de recursos relacionados a una sola entrada, si consideramos la red de extensión que se genera se puede evidenciar que de un solo nodo inicial recuperamos más nodos derivados que nos guían hacia nuevos niveles de abstracción de un universo de trabajo.

Con esta información se puede armar un grupo de recursos que representen dicho universo de trabajo y si usamos estos resultados como recomendación podemos establecer que recurso se desenvuelve dentro del rango de relación al nodo padre, o se va expandiendo hacia otros contextos.

Los resultados de los distintos servicios presentan un solo nivel de extracción, un proceso recursivo no es considerado y en el presente proyecto dado el requerimiento de trabajar con grupos de tags se ha determinado una extracción a 3 niveles que se genera en cada flujo realizado.

RESPUESTA DE RECORRIDO DE UNA SOLA ENTIDAD (Accounting)			
"Accountancy",	http://dbpedia.org/resource/Accountancy",	Advertising",	http://dbpedia.org/resource/Advertising",
		"Business"	http://dbpedia.org/resource/Category:Business",
		"Business continuity and disaster recovery",	http://dbpedia.org/resource/Category:Business_continuity_and_disaster_recovery"
		"Business development",	http://dbpedia.org/resource/Business_development
		"Business economics",	http://dbpedia.org/resource/Category:Business_economics",

		"Business terms",	http://dbpedia.org/resource/Category:Business_terms",
		Financial software",	http://dbpedia.org/resource/Category:Financial_software",
"Accountant",	http://dbpedia.org/resource/Accountant",	Occupations",	http://dbpedia.org/resource/Category:Occupations"
		Accountancy qualifications",	http://dbpedia.org/resource/Category:Accountancy_qualifications"
		"Accountants",	http://dbpedia.org/resource/Category:Accountants"

Una diferencia notable es el nivel de extracción de relaciones, es de un solo nivel y las URI´S extraídas no pueden ser reutilizadas para un nuevo proceso de extracción más profundo y preciso.

Otra herramienta que se ha utilizado para la comparativa de nuestras herramientas es un servicio llamado: DBPedia Lookup http: //lookup.dbpedia.org/, un ejemplo de encontrar recurso se visualiza en la figura 52, este servicio cumple una función de extracción muy eficaz utilizando las relaciones de entidades.

El ingreso de un tag, es realizado en el campo “tag” que se encuentra en la estructura de la herramienta y permite concatenación de texto para convertir una frase en palabra, sin embargo la precisión de esta extracción no está ligada a un contexto por lo que muchas de las veces la extracción es muy general

En el proceso de extracción podemos visualizar que los campos extraídos son:

- Label
- URI
- Descripción

Al tener una URI relacionada como factor podemos ya comenzar a tener una idea de los vínculos y una relación concreta, sin embargo debe ser necesario encontrar todo el conjunto de relaciones o las ramificaciones que pueden partir de un nodo padre, por lo tanto para una recomendación sería muy generalizada y requeriría de un nuevo flujo de extracción.

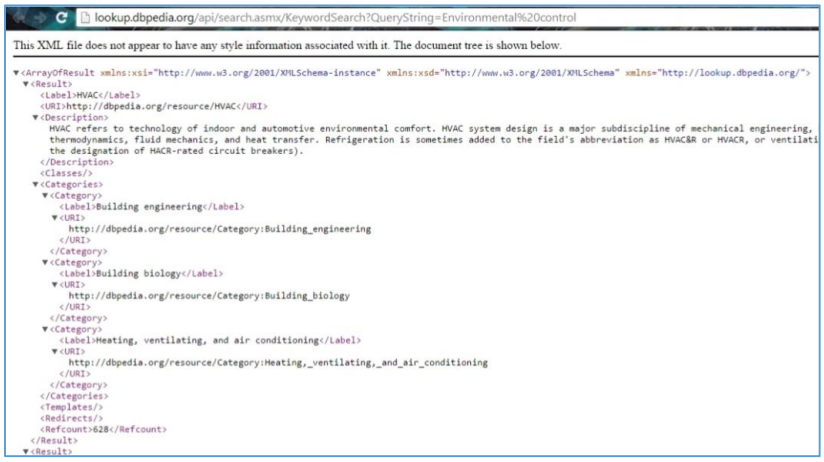


Figura 52. Extracción de entidades mediante Lookup

La característica más compartida de este servicio es que se puede enviar un tag simple o uno compuesto, sin embargo, no es posible agrupar ni clusterizar los resultados lo que al final no podríamos aplicar a una recomendación directa sin que antes no le adaptemos quizás un servicio web y un proceso de categorización.

La herramienta Extract: http: //embed.ly/docs/api/extract/endpoints/1/extract, es otra herramienta que se adapta al flujo básico de extracción por lo que fue incluida en este proceso de comparación.

Extract nos indica que está diseñado para proporcionar a los usuarios información importante que se encuentre en cada enlace. Esta variable incluye en su proceso de extracción de un texto del artículo, palabras clave, enlaces relacionados, e incluso los recursos multimedia que pueden existir en cada recurso.

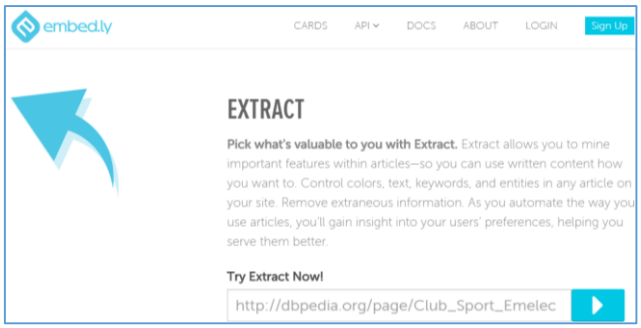


Figura 53. Inicio de flujo de extracción entidades relacionadas mediante Embedly

En la estructura de la aplicación podemos ver que este proceso va directamente al uso de una URL o URI y en base a ella nos devuelve resultados, en este caso sería necesario contar con la uri de la entidad que queremos trabajar, y más no el tag o palabra.

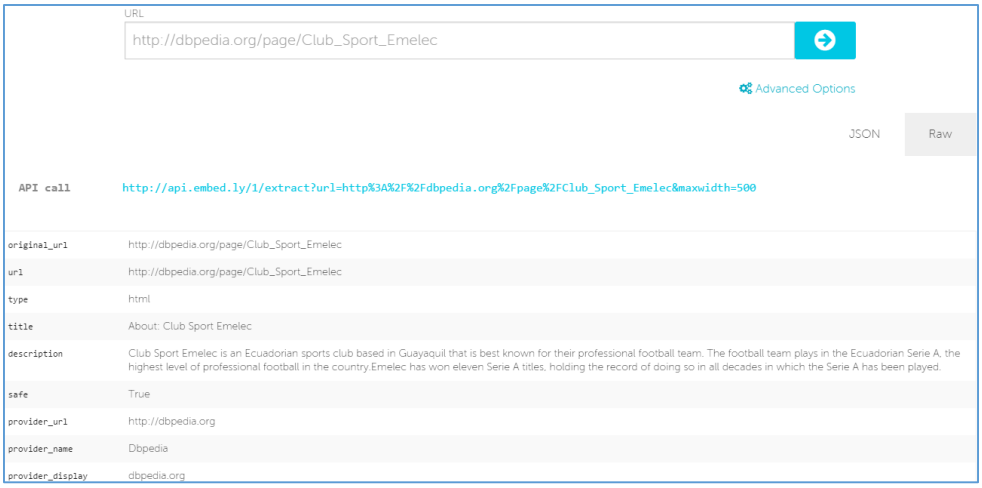


Figura 54. Resultados de la extracción mediante Embedly

En los resultados podemos ver que las variables que nos devuelve y que nos servirían para validar relaciones son:

- Uri
- Type
- Provider

Sin embargo al ser datos generales, no podríamos utilizarlos para deducir la recomendación de un conjunto de recursos, solo adaptándolos a otros procesos podemos armar una recomendación.

4.6 Conclusiones y discusión final capítulo IV.

El integrar el flujo extractivo en un aplicativo se facilita el trabajo de extracción de entidades relacionadas, con el empaquetado y procesado de tags y keywords, se provee de un proceso de ingreso de recursos familiar y fácil, similar a una consulta en web, al mostrar las entidades candidatas relacionadas se da una pauta de la extensión que puede tener una entidad, al encontrar entidades relacionadas y mostrar su estructura llegamos al fin concreto de nuestro proceso que es conocer la extensión de un universo de conocimiento.

Con la obtención de los resultados iniciales podemos ya permitir una visualización a nivel de usuario final que de una idea de cómo está estructurada una entidad, y con esas premisas podemos recomendar recursos para su posterior estudio, para su explotación y crecimiento o para conocer la incidencia que una entidad tiene en el universo de datos actualmente presente en la Web.

La integración de servicios web para la realización de extracción de entidades relacionadas solventa la complejidad en la visualización de recursos recuperados, ya que en la actualidad los procesos extractivos y de minería de datos ha sido de uso exclusivo para personas que trabajan dentro de un ambiente de conocimiento, sin embargo, el fundamento de la web semántica es dar sentido a la información presente en la red y si permitimos que los usuarios que no están inmersos en este sistema entiendan y extraigan sus propios elementos estaremos acercando un paso más a la web del conocimiento compartido.

Si combinamos los resultados obtenidos con los procesos de extracción, más las herramientas de visualización podemos establecer un plan de trabajo que a futuro permita extender, manipular y conocer la red de conocimiento inherente en una entidad.

CONCLUSIONES

El descubrimiento de recursos relacionados nos da la oportunidad de aprovechar la red semántica que vincula dos entidades y de esta forma incrementar de manera significativa la calidad de la información recuperada.

El correcto uso de la información nos dará la pauta ideal para procesar y utilizar cada entidad de manera óptima, lo que nos abre el nuevo campo de la recomendación de datos basándonos en la estructura base de sus relaciones y así podremos saber qué relación tiene una dimensión de información con otra y de esta manera crear un puente o un vínculo mucho más fuerte y preciso entre entidades.

La visualización de resultados permitirá a nivel de procesamiento de datos una forma más fácil de manipular y revisar la información entrelazada y a nivel de usuario la facilidad de utilizar sólo la información precisa de un recurso y también la capacidad de poder navegar y discernir entre los datos que queremos revisar y los que puedan llegar a ser omitidos por alguna razón.

La extracción de entidades abre una gran puerta al proceso de especificación de procesos como lo es el de la recomendación, la búsqueda, la creación de información entrelazada y el aprovechamiento de relaciones entre entidades.

El contar con el conocimiento de una red de relaciones entre recursos disponibles en Web permitirá finalmente comprender e interpretar de mejor manera la información, acceder de manera más óptima a recursos de distinta índole, como nuevos universos de trabajo y de esta manera tener la faculta de explotar la información presente.

Si aplicamos un grupo de tags, como entidades de un caso de estudio como materias de educación, palabras que nos interesan o un conjunto de palabras específicas como datos de organizaciones como UNESCO, podremos conocer cómo se presenta en la actualidad su red de relaciones.

RECOMENDACIONES Y TRABAJOS FUTUROS

El proceso de extracción de entidades permite revisar toda una red de relaciones entre sus datos, las cuales también nos dan una pauta para entrelazar y categorizar cada ramificación del grafo por lo que sería recomendable implementar un proceso a nivel individual de cada grafo generado y categorizar sus ramificaciones, lo que permitiría crear una red de patrones que posteriormente podrían servir para especificar las entidades que la conforman en componentes.

Otra función que se puede aplicar es el enriquecimiento de las nubes de conocimiento extraídas, se puede establecer con enriquecimiento de etiquetas cada conjunto generado, todo esto puede servir para una posterior búsqueda exploratoria o para incrementar los elementos que conforman una dimensión de conocimiento específico.

Se puede implementar la técnica de extracción de relaciones basado en los keywords o entidades que han sido proporcionados inicialmente, pero si se adhiere un nuevo proceso que pueda permitir la extracción dinámica o automática de estas entidades de un conjunto de recursos (por ejemplo de abstracts de artículos, de keywords de una nube en una página web), previo a la extracción de relaciones se podría automatizar estos procesos.

En un trabajo futuro se puede considerar la posibilidad de algunos proyectos, uno de ellos podría ser la implementación de un extractor de tags de un párrafo o un abstract en un contexto determinado, para encontrar entidades relacionadas de un documento o universo de estudio específico considerando también la desambiguación de significados.

Entre los inconvenientes se ha suscitado el cambio repentino e inclusive la desaparición de algunas herramientas que servían de apoyo en los procesos, lo que significó modificar nuevamente la estructura de algún componente, crear el proceso faltante o desaparecido e implementar todos los cambios para adaptar a los componentes.

La detección, correlación y consolidado de entidades en varios idiomas y su respectiva traducción de las descripciones podría permitir generar un grupo mayor y más extenso de elementos extraídos, dándonos la oportunidad de descubrir los distintos significados y recursos de manera global.

BIBLIOGRAFIA

- Vallez, Mari, Rovira, Cristòfol, Codina, Lluís, Pedraza, Rafael (2010). "Procedures for extracting keywords from web pages, based on search engine optimization": http://www.upf.edu/hipertextnet/en/numero8/keywords_extraction.html.

- Flynn, P.; Zhou, L.; Maly, K.; Zeil, S.; Zubair, M. (2007). "Automated template-based metadata extraction architecture". IN: Lecture notes in computer science, 4822, p. 327-336.

- La web semántica y el procesamiento del lenguaje natural - Lluis Codina; Mari Carmen Marcos; Rafael Pedraza-Jimenez (Coords) Vallez, Mari (2009): http://www.upf.edu/hipertextnet/en/numero-8/keywords_extraction.html

- Berners-Lee (2006), Linked Data: http://www.w3.org/DesignIssues/LinkedData.html.

- (Codina, 06) - Exact Matching Concepts from Other Schemes – Sitio Web: http://viaf.org/viaf/sourceID/LC%7Cn+85371948#skos:Concept.

- Miguel A. Martínez Prieto & Javier D. Fernández (2007), “DBpedia: A Nucleus for a Web of Open Data.”

- O’Reilly (What Is Web 2.0): http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=5

- Heath y Bizer (2011) - Linked Data: Evolving the Web into a Global Data Space. Sitio Web: http://linkeddatabook.com/editions/1.0/

- Miguel A. Martínez Prieto & Javier D. Fernández (2013) - DBpedia como núcleo de la Web de Datos “Aprendiendo a nadar en el diluvio de datos” (V). Sitio Web: http://dataweb.infor.uva.es/wp-content/uploads/2012/03/curso5a.pdf

- Pastor Sánchez, Juan Antonio - Diseño de un sistema colaborativo para la creación y gestión de tesauros en Internet basado en SKOS – Sitio Web: http://hdl.handle.net/10803/10914

- Daconta MC, Obrst Leo J, Smith KT. The Semantic Web: A guide to the future of XML, Web Services and Knowledge Management. New York: Wiley, 2003 p. 145.

- Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. J.Web Sem., 6(3):203–217, 2008.

- DBpedia: A Nucleus for a Web of Open Data" (2007) – Sitio Web: http://link.springer.com/chapter/10.1007%2F978-3-540-76298-0_52

- Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Scarlet: Semantic relation discovery by har- vesting online ontologies. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, ESWC, volume 5021 of Lecture Notes in Computer Science, pages 854–858. Springer, 2008. Sitio Web: http://link.springer.com/chapter/10.1007%2F978-3-540-76298-0_52#page-1

- (Shiri, 2014) - Aplicación de los principios Linked Open Data. Sitio Web: http://e-archivo.uc3m.es/bitstream/handle/10016/19674/avila_aplicacion_TFM_2014.pdf?sequence=1

- Manual de SKOS (Simple Knowledge Organization System), 2009. Versión española de Juan Antonio Pastor Sánchez y Francisco Javier Martínez Méndez – Sitio Web: http://skos.um.es/TR/skos-primer/

- Christian Bizer, Freie Universität Berlin,2009 - Linked Data - The Story So Far, Sitio Web: http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf

- Web Semántica, diseño metodológico - SKOS-Core – Sitio Web: http://www.w3.org/2001/sw/Europe/events/200406-esp/trabajo-finalextratesauros/node6.html

- A. Budanitsky, G. Hirst. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Computational Linguistics, ISSN 0891–2017, 32(1), 13—47 (2006). Referenciado en 69

- M. Paolucci, T. Kawamura, T. Payne, K. Sycara. Semantic Matching of Web Services Capabilities. Proceedings of the First International Semantic Web Conference. ISWC 2002. ISBN 3-540-43760-6, Sardinia, Italy, 333–347 (2002). Referenciado en 70
- H Al-Mubaid, H Nguyen. A Cluster-based Approach for Semantic Similarity in the Biomedical Domain. Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE. ISBN 1-4244- 0032-5, New York, USA, 2713–2717 (2006). Referenciado en 70

- Christian Bizer and Andreas Schultz. The berlin sparql benchmark. International Journal On SemanticWeb and InformationSystems, 5(1) ,2009.

- Hien T. Nguyen and Tru H. Cao. Named entity disambiguation on an ontology enriched by wikipedia. InRIVF, pages247–254.IEEE, 2008.

- Marta Sabou, Mathieud'Aquin and Enrico Motta. Using the semantic web as background knowl- edge for ontology mapping. In Pavel Shvaiko, Jérôme Euzenat, Natalya Fridman Noy, Heiner Stuckenschmidt, V. Richard Benjamins, and Michael Uschold, editors, Ontology Matching, vol- ume225ofCEURWorkshopProceedings.CEUR-WS.org, 2006.

- Bases de datos NoSQL y ezscalabilidad horizontal 2012 - Juan Carlos García Candela © Taller Digital de la Universidad de Alicante – Sitio Web: http://www.opiniontecnologica.com/aplicaciones-informaticas/130-bases-de-datos-nosql-y-escalabilidad-horizontal.html

- Jesús Tramullas, Profesor Titular de Documentación Automatizada, publicado el 17 mayo 2011 De base de datos clásica al NoSQL.

- Emil Eifrem, CEO of Neo Technology (Top 5 Reasons to Get Your Graph On - 2012)

- Emil Eifrem, CEO, Neo Technology (Graphs as a New Way of Thinking JANUARY 9, 2013) – Sitio Web: http://allthingsd.com/20130109/graphs-as-a-new-way-of-thinking/?refcat=news

- Martin Hepp, Katharina Siorpaes, and Daniel Bachlechner. Harvesting wiki consensus: Using Wikipedia entries as vocabulary for knowledge management. IEEE Internet Computing, 11(5):54– 65,2007.

- Query a Graph with a Traversal (What is a Graph Database? – 2012). Sitio Web: http://www.neo4j.org/learn/graphdatabase

- Carmona, J., Cervell, S., Màrquez, L., Martí, M., Padró, Pombo, L., Placer, R., Rodríguez, H., Taulé M. & Turmo, J. (1998) An Environment for Morphosyntactic Processing of Unrestricted Spanish text.

- (Novak JD,1984) Gowin DB. Learning How to Learn. New York: Cambridge University Press, 1984.

- Yang, Yiming, y Jan O. Pedersen (2007) - A comparative study on feature selection in text categorization. En "Proceedings of ICML-97, 14th International Conference on Machine Learning" (Douglas H. Fisher, ed.). Morgan Kaufmann Publishers, San Francisco, 1997, 412–420. Sitio Web: http://www.cs.cmu.edu/ yiming/papers.yy/ml97.ps.

- Green, R. "The Role of Relational Structures in Indexing for the Humanities". Knowledge Organization. Vol. 24, No. 2, 1997, pp. 72-28.

- WebSets: Extracting Sets of Entities from the Web Using Unsupervised Information Extraction: Sitio Web: http://www.cs.cmu.edu/~bbd/wsdm2012.pdf

- Hai Zhuge. Communities and emerging semantics in semantic link network: Discovery and learning. IEEE Trans. on Knowl. and Data Eng., 21(6):785– 799, 2009.)

- Ryen W. White and Resa A. Roth. Exploratory Search: Beyond the Query-Response Paradigm. Synthesis Lectures on Information Concepts, Retrieval, and Services, 1(1):1{98, January 2009.]

- Supachai Tungwongsarn. (2010). System for storage and retrieval of information by computer. Bangkok: Pithak Printing.

- (Brin, 1998; Iria y Ciravegna, 2005; Nguyen et al, 2007; Roth y Yih, 2002).

- Prywes, Noah S., "Online Information Storage and Retrieval" (1968). Technical Reports (CIS). Paper 823. Sitio Web: http://repository.upenn.edu/cis_reports/823

- Lovins [1968] Development of a Stemming Algorithm Sitio Web: http://mt-archive.info/MT-1968-Lovins.pdf

- Bogomolny (Feb/2006) - Distance between strings. Sitio Web: http://www.cut-the-knot.org/do_you_know/...

- M. Gilleland (Feb/2006) - Levenshtein Distance, in Three Flavors, Sitio Web: http://www.merriampark.com/ld.htm#WHATIS

- : Fillmore, C. J. “Types of lexical information”. In: STEINBERG, D.D. and JAKOBOVITS, L.A., eds. Semantics: and interdisciplinary reader in philosophy, linguistics and psychology. Cambridge: Cambridge University Press, 1971

INDICE DE FIGURAS

ANEXOS

[Anexo 1] Realización de la conexión de una nueva base de datos gráfica

Cuando trabajamos bajo estas plataformas podemos obtener mucho más rendimiento en consultas, y como nuestro objetivo principal es el análisis de relaciones nos evitaríamos la inserción de muchos Joins necesarios en un modelo relacional, por lo que convierte al uso de esta base de datos basada en grafos como la opción óptima.

Usar conjuntos de elementos RDF tradicionales en un proceso de almacenamiento y búsqueda no es específicamente lo más óptimo, sin embargo, cuando se necesita calcular las rutas más cortas entre sujetos al azar, la aplicación del análisis de grafos y su uso de base de datos gráficas lo vuelve una proceso efectivo.

Debido a la gran cantidad de datos que se encuentran vinculados de manera libre en la web de datos ha sido requerida la implementación de una herramienta que permita recolectar información y conocimiento centrado en servicios.

Cuando accedemos a un recurso debemos “decodificarlo” en base a su conjunto de datos y componentes que lo conforman para poder encontrar los datos que requerimos y poderlo explotar en las funcionalidades y en los resultados de las herramientas. Una de las formas más factibles de representar esos resultados es permitir la visualización de resultados en nubes de etiquetas o árboles de representación visual.

La exploración y explotación de un recurso mediante un tag de entrada o la utilización de una URI será guiada mediante un flujo de procesos explicados en el trabajo realizado.

Cada una de las fases de este flujo cumple con una tarea específica de exploración y aprovechamiento de los recursos utilizados.

Cuando iniciamos un mecanismo de recorrido podemos establecer una serie de instrucciones para encontrar un dato de una manera más eficiente, por ejemplo, si hablamos de una red social, al acceder seríamos el primer nodo, y al cargarse el recurso se crea una consulta de nodos y nos traería todos los nodos en una capa de x profundidad relacionados con el usuario que acaba de conectarse.

A continuación se explica el proceso de recorrido y establecimiento de una base de datdos gráfica, en este caso con el uso de gremlin para su recorrido.

1. Se establece la creación de un nodo inicial del cual partirán los demás recursos conforme vayan siendo encontrados.

- gremlin> g = new LinkedDataSailGraph(new MemoryStoreSailGraph())

  ==>sailgraph [linkeddatasail]

2. Realización de una ruta hacia una URI de Linked Data, para este ejemplo utilizo la información de Chris Bizer http://data.semanticweb.org/person/christian-bizer.

- gremlin> v = g.v('http://data.semanticweb.org/person/christian-bizer')

  ==>v[http://data.semanticweb.org/person/christian-bizer]

3. Siempre es requerido colocar los “namespaces” por defecto para poder cargar la información rdf, rdfs, owl, xsd and foaf.

- gremlin> g.addDefaultNamespaces()

==>null

- gremlin> g.getNamespaces()

==>rdfs=http://www.w3.org/2000/01/rdf-schema#

==>foaf=http://xmlns.com/foaf/0.1/

==>xsd=http://www.w3.org/2001/XMLSchema#

==>owl=http://www.w3.org/2002/07/owl#

==>rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#

4. A continuación podemos ya comenzar a realizar el recorrido de los nodos y recuperar la información que contienen a base de un parámetro de búsqueda, en este caso realizo una búsqueda específica de que artículos ha escrito esta persona: Chris Bizer.

- gremlin> v.out('foaf:made')

==>v[http://data.semanticweb.org/conference/iswc-aswc/2007/tracks/in-use/papers/715]

==>v[http://data.semanticweb.org/conference/eswc/2007/demo-3]

==>v[http://data.semanticweb.org/conference/iswc/2006/paper-24]

==>v[http://data.semanticweb.org/workshop/LDOW/2008/paper/6]

==>v[http://data.semanticweb.org/workshop/ssws/2008/paper/main/4]

==>v[http://data.semanticweb.org/conference/iswc/2009/paper/research/301]

==>v[http://data.semanticweb.org/conference/iswc/2009/paper/research/306]

==>v[http://data.semanticweb.org/conference/iswc/2010/paper/495]

==>v[http://data.semanticweb.org/conference/iswc/2010/paper/512]

==>v[http://data.semanticweb.org/conference/iswc/2010/paper/519]

==>v[http://data.semanticweb.org/workshop/cold/2010/BizerEtAlPaper]

==>v[http://data.semanticweb.org/workshop/cold/2010/IseleEtAlPaper]

5. Como resultado, se ha obtenido de la URI: http://data.semanticweb.org/person/christian-bizer las ubicaciones de los documentos que esta persona ha escrito, este resultado es un RDF y también está compuesto por RDFstatments. De igual manera este resultado es devuelto con una estructura RDF

<RDF: RDF>

...

<ns1: Person rdf:about="http://data.semanticweb.org/person/christian-bizer">

<RDF: type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>

<ns2: affiliation rdf:resource="http://data.semanticweb.org/organization/freie-universitaet-berlin"/>

<ns2: affiliation rdf:resource="http://data.semanticweb.org/organization/freie-universitaet-berlin"/>

<Rdfs: seeAlso rdf:resource="http://ontoworld.org/wiki/Special:ExportRDF/Chris_Bizer"/>

<Rdfs: seeAlso rdf:resource="http://ontoworld.org/wiki/Special:ExportRDF/Chris_Bizer"/>

<Owl: sameAs rdf:resource="http://ontoworld.org/wiki/Special:URIResolver/Chris_Bizer"/>

...

<Owl: sameAs rdf:resource="http://ontoworld.org/wiki/Special:URIResolver/Chris_Bizer"/>

<ns1: made rdf:resource="http://data.semanticweb.org/conference/iswc-aswc/2007/tracks/in-use/papers/715"/>

<ns1: made rdf:resource="http://data.semanticweb.org/conference/iswc-aswc/2007/tracks/in-use/papers/715"/>

<ns1:mbox_sha1sum>50c02ff93e7d477ace450e3fbddd63d228fb23f3</ns1:mbox_sha1sum>

<ns1:mbox_sha1sum>50c02ff93e7d477ace450e3fbddd63d228fb23f3</ns1:mbox_sha1sum>

<ns1: name>Chris Bizer</ns1: name>

<ns1: name>Chris Bizer</ns1: name>

...

</Rdf: RDF>

6. Otra ejemplificación de recorrido y obtención de resultados sería realizar una búsqueda de los títulos de los artículos que han sido escritos por Chris Bizer.

- gremlin> v.out('foaf:made').out('dcterms:title').value

==>DBpedia: A Nucleus for a Web of Open Data

==>Fresnel: A Browser-Independent Presentation Vocabulary for RDF

==>DBpedia Mobile: A Location-Enabled Linked Data Browser

==>Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints

==>Executing SPARQL Queries over the Web of Linked Data

==>Discovering and Maintaining Links on the Web of Data

==>LDSpider: An open-source crawling framework for the Web of Linked Data

==>Extending SMW+ with a Linked Data Integration Framework

==>Silk - Generating RDF Links while publishing or consuming Linked Data

==>The R2R Framework: Publishing and Discovering Mappings on the Web

==>Silk Server - Adding missing Links while consuming Linked Data

==>DBpedia internationalization - a graphical tool for I18n infobox-to-ontology mappings

==>LDIF - Linked Data Integration Framework

==>LDIF - Linked Data Integration Framework (Systems Paper)

==>LDIF - A Framework for Large-Scale Linked Data Integration

==>Benchmarking the Performance of Linked Data Translation Systems

Utilizando las herramientas de recorrido y de recuperación de recursos que Gremlin nos proporciona, podemos consumir cualquier tipo de recurso de un repositorio o endpoint en donde contengamos información, y de esta manera incluso podemos llegar a realizar recomendaciones específicas en base a un recurso o temática específica.

[Anexo 2] Manipulación y recorrido de una estructura basada en Gremlin

Como hermos revisado, al ser un servicio bastante factible para la manipulación de los datos que pueden estar alojados en una base de datos gráfica, Gremlin posee algunas funciones que permiten maximizar y soportar cualquier base de datos que sea requerida implementar.

El acceso a un repositorio de recursos de entidades como la DBpedia requiere de métodos que permitan su recorrido y la extracción de los elementos que luego necesitan ser manipulados para crear paquetes de resultados que permitan generar un resultado específico como: empaquetado de componentes de una entidad para determinar las relaciones específicas o vínculos que puedan existir con otros recursos al momento de ser comparados.

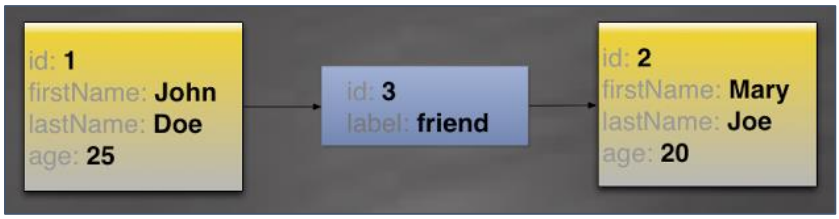
Una de las características principales de Gremlin el soporte que existe con las propiedades de Blueprints, característica que es importante en el desarrollo del proyecto de extracción de entidades relacionadas en recursos OCW ya que es necesario crear los servicios y recursos en un entorno de desarrollo JVM como Groovy y Java en los cuales se basará la creación de los servicios.

Existen bastantes bases de datos que pueden ser soportadas con sus lenguajes personalizados como Cypher en Neo4j, pero existe una inconsistencia si se requiere migrar a otro tipo de bases de datos.

El uso de Gremlin para manipular y atravesar bases de datos es equivalente a el proceso de aprendizaje de SQL para trabajar con una base de datos relacional, si se aprende a trabajar con SQL posteriormente podremos trabajar con bases de datos como My Sql u Oracle, de la misma manera el conocer Gremlin permitirá asegurar el proceso de trabajo cuando se trabaje con Neo4j o Titan o Graph DB o cualquier tipo de base de datos orientada a grafos.

Es necesario conocer los métodos primordiales que serán utilizados para manipular los datos extraídos, estos métodos nos permitirán realizar los procesos de búsqueda, recorrido, análisis y manipulación.

Las búsquedas en Gremlin están orientadas a la lectura de los datos que son extraídos de un repositorio, ya sea un repositorio accedido por una dirección URL o mediante el acceso a un repositorio de entidades.



Ejemplificación relación entre entidades

- Obtener un vértice por su Id: Cuando utilizamos (v) se puede acceder a un vértice en base al Id de un element, cabe recalcar que debe ser (v) minúscula pues (V) mayúscula tiene un uso diferente.
    - g.v(1);
- Obtener todos los vértices con un rango Id: En el caso de que se requiera obtener todos los vertices con un rango determinado, por ejemplo del 1 al 100
    - g.V[1..100]
- La obtención de un atributo específico de cada vértice, por ejemplo si es el primer nombre de un elemento, se puede realizar de la siguiente forma:
    - g.V[1..100].firstName
- Obtener el atributo de un vértice por su Id: Por ejemplo si el atributo de un nombre es “firstName” entonces se puede utilizar el siguiente ejemplo para encontrar el primer nombre de un vértice en base a su id.
    - g.v(1).firstName;
- Obtener un vértice mediante un atributo: si se requiere encontrar un vértice o todos los vértices que tengan “firtName” como “Pablo”
    - g.V('firsName','Pablo');
- Obtener la Id de un vertice mediante un atributo: El atributo id es un atributo que siempre estará disponible para cualquier vértice, de esta manera inclusive si no se agrega un id de atributo, en la base de datos gráfica se asignará un valor único para identificar un vértice.
    - g.V('firsName','Pablo').id;

- Obtener el número de vértices con un valor de atributo: Por ejemplo, si es necesario conocer cuántas personas en una base de datos con el primer nombre “Pablo”

    - g.V('firsName','Pablo').count();

- Obtener los filos de un vértice que tengan una etiqueta determinada: En este proceso se puede evidenciar el proceso real que se realiza mediante un lenguaje gráfico. Por ejemplo, el conocer las relaciones de entrada y salida en un grafo.Un ejemplo sería conocer todos los nodos de salida que estén etiquetados como “friend”

    - g.v(1).outE('friend');

Si fuera necesario concoer los nodos entrants que estén etiquetados con “friend”

    - g.v(1).inE('friend');

Si fuera requerido conocer todos los nodos entrants y salientes que estén etiquetados como “friend”

    - g.v(1).bothE('friend');

- Obtener todos los nodos de un vértice que tengan una etiqueta “friend”. El conteo es una función que puede ser aplicada tanto en vértices como en filos para obtener su número, en lugar de obtenerlos un por uno. Por ejemplo, el conocer cuántos amigos están conectados con un nodo con una id 1

    - g.v(1).outE('friend').count();

- Obtener la etiqueta de todos los nodos filo de un vértice específico: De la misma forma que un vértice se puede obtener los atributos de cualquier nodo filo usando el nombre del atributo.

    - g.v(1).outE().label;

- Obtener el número de todos los nodos filos salientes de un vértice. Si es requerido conocer cuántos nodos filos están conectados con un nodo de id 1.

    - g.v(1).outE().count();

- Obtener los primeros nombres de todas las personas que están conectadas con el atributo relación de un amigo específico. Con esta forma se puede devolver los primeros nombres de todas las personas que son amigos de un vértice con id 1.

    - g.v(1).outE('friend').inV().firstName;

- Obtener la edad de todos los amigos (un nodo específico). Por ejemplo si se requiere sacar las edades de todas las personas que están conectadas a un nodo (persona con apellido “Malla”), mediante la relación de amigo. Con esta combinación se demuestra que un punto de inicio puede ser una propiedad y que cualquier atributo conectado con un vértice puede ser devuelto como resultado.

    - g.V('lastName','Malla').outE('friend').inV().age;

- Obtener todos los amigos con una edad de 25 años. Con este ejemplo se puede obtener los primeros nombres de todos los amigos de un vértice con id 1 que estén conectados por la relación amigo.

    - g.v(1).outE('friend').inV().has('age',25).firstName;

- Encontrar todas las personas con una edad superior a 25 años. Un ejemplo sencillo, devolver personas con un rango de edad específico.

    - g.V.filter{it.age>25}.firstName

- Devolver como resultado todas las personas con una edad mayor a 25 años y menor a 35 años

    - g.V.and(_().has("age",T.gt,25),_().has("age",T.lt,35));

    o también puede ser realizada utilizando una método de intervalo

    - g.V.interval("age", 25, 35);

- Devolver todas las personas que contengan una dirección de correo electrónico.

    - g.V.hasNot('email',null)

    O de otra forma podemos devolver de manera sencilla un resultado opuesto a lo que se necesita, en el ejemplo del correo electrónico aplicando la función hasNot

- g.V.has('email', null)

- Obtener resultados únicos con una búsqueda compleja. Esto puede ser aplicado con la función dedup

    - g.V('lastName','Malla').outE('friend').inV().dedup();

- Obtener el número de resultados únicos con una búsqueda compleja. Esto puede ser realizado con la función Count.

    - g.V('lastName','Malla').outE('friend').inV().dedup().count();

[Anexo 3] Manipulación gráfica de búsquedas en Gremlin

Gremlin permite sentencias que permiten manipular el grafo generado o los elementos que conforman una estructura

- Crear un nuevo vértice
  - g.addVertex([firstName:'Pablo',lastName:'Malla',age:'26']);

  g.commit();
- Crear 2 nuevos vértices y una relación (con una etiqueta “friend”) entre ellos.

En este proceso se requiere múltiples líneas. Hay que tener encuentra como las variables (pm y dc) son definidas simplemente cuando se les designa un valor en Gremlin.

  - pm=g.addVertex([firstName:'Pablo',lastName:'Malla',age:'26']);
  - dc=g.addVertex([firstName:'Diana',lastName:'Cueva',age:'24']);
  - g.addEdge(pm,dc,'friend');

  g.commit();
- Crear una relación entre 2 vértices ya existentes con la id 1 y 2
  - g.addEdge(g.v(1),g.v(2),'coworker');

  g.commit();
- Remover todos los vértices de un grafo
  - g.V.each{g.removeVertex(it)}

  g.commit();
- Remover todos los filos de un grafo
  - g.E.each{g.removeEdge(it)}

  g.commit();
- Remover todos los vértices con “firstName = Pablo
  - g.V('firstName','Pablo').each{g.removeVertex(it)}

  g.commit();
- Remover un vertice con un id 1
  - g.removeVertex(g.v(1));

  g.commit();
- Remover todas las aristas con id 1
  - g.removeEdge(g.e(1));

  g.commit();

- Crear índices usando Gremlin. Para indexar un grafo con un campo específico se puede crear un campo (myfield)
    - g.createKeyIndex("myfield",Vertex.class);

[Anexo 4] Diferencias entre la utilización de Gremlin y Cypher en el proceso de recorrido y manipulación de datos. [Anexo 3]

- El proceso de atravesar un grafo simple es mucho más eficiente cuando se aplica Gremlin
- Las búsquedas en Gremlin son entre 30-50% más rápidas en recorridos simples
- Cypher es un lenguaje ideal para recorridos complejos donde se requiere un resultado de vuelta
- Cypher es un lenguaje ideal para la generación de reportes
- Gremlin es un lenguaje de consulta recomendado para cuando se requieran recorridos sencillos en dónde no se requieran proyecciones
- Cypher tiene un modelo de proyección de tabla intrínseco
- Cuando se requiere recuperar la estructura de datos simples podemos aplicar Gremlin
- Cypher es factible para cuando se requiere realizar combinaciones externas que Gremlin
- La tabla de proyección de Gremlin es bastante robusta, sin embargo al realizar combinaciones externas podemos encontrarnos con algunos inconvenientes.

La diferencia fundamental entre Cypher y Gremlin es que Cypher posee un lenguaje completo declarativo (sin regreso) y Gremlin realiza un proceso de manipulación de datos a través de canalizaciones, set de iteradores que trabajan en la parte superior de la API de Java de Neo4j

Cypher tiene una estructura similar a SPARQL o SQL, utiliza sentencias declarativas y una más eficiente descripción de los procesos que se requieren realizar y no precisamente la forma en que se requiere que los resultados se muestren. Debido a esta característica se muestra adecuado como para ser aplicado mediante comunicación remota, estandarización y optimización.

Si es requerido utilizar y explotar estas características de Cypher es requerido solventar algunas necesidades:

- Optimización en el proceso de consultas y los planes de ejecución

- Establecer una sintaxis para expresar las necesidades que de otra forma sería muy complejo obtenerlas, como la ubicación de un camino, restricciones o patrones.

- Solventar el bajo nivel de apalancamiento, API's de bajo rendimiento y backends que no requieran un cambio a nivel de usuario.

Cypher aplica en la actualidad el núcleo de JavaBeans y se basa en las herramientas de las API's de Blueprints que sería la API subyacente de Gremlin/Tubes) lo que aún no la convierte en un proceso más rápido que la realización de expresiones programadas imperativas bien elaboradas.

Si se aplican las estadísticas de consultas, los metadatos de indexación y la optimización de la aplicación en una tienda basada en patrones de uso, podemos establecer un cambio dinámico en el proceso de trabajo en relación a consultas o procesos realizados con Cypher.

Debido a que no será posible hacer frente a todos los posibles casos de uso en la aplicación al momento de solicitar un proceso ya sea de consulta, recomendación, visualización, se creará al igual que en SQL funciones definidas por el usuario contra un API de rendimiento.

[Anexo 5] Análisis de herramientas Grafos y Relaciones entre entidades

Andreas Colleger (2012 -Intro to Neo4j) nos explica que: La manera de trabajar en una base de datos gráfica es el de trabajar y almacenar las relaciones que existen entre los datos. Si tratamos como ejemplo un sitio como el de LinkedIn, podemos ver que trabaja a través de conexiones entre personas y no de manera lineal, contiene elementos unidos bajo relaciones derivadas ya que cada dato puede contener uniones de tipo: “A es colega de B” y a su vez “B es proveedor de C” y dónde “C es cliente de A”. [R.9]

Una base gráfica es compatible cuando se quiere declarar límites de profundidad por lo que sería ideal utilizarlo conjuntamente con una base de grafos.

A diferencia de un sistema RDBSM dónde toda una tabla debe ser recorrida, con Neo4j podemos establecer límites y tomar acciones basadas en el estado actual de un sujeto. Cada ruta es una declaración predefinida.

Duane Nickull - Sr. Data Architect & SOA Consultant en su artículo (Getting started with Neo4J April 25, 2012) [R.10] nos explica el proceso en el que trabaja Neo4j y una reseña de cómo se puede llegar a implementar en los diferentes ambientes de trabajo actualmente disponibles:

Cuando un nodo requiere un punto específico de inicio cada vez que es convocado, podemos indexarlo e iniciar directamente con él. Un índice es un simple punto de inicio contextual, cada índice puede ubicar directamente un nodo, una relación o una propiedad, en lugar de decir:

- Select * from tables where * equals “Duane Nickull”

Al usar una base de datos gráfica, podemos:

- “convocar a Duane Nockull” directamente.

Si intentamos modelar un sistema de transportes contaremos con mapas gráficos de las estaciones, los nodos y paradas, para poder trabajar con elementos de esta naturaleza podemos utilizar Neo4j y combinarlo con RDF para poder obtener resultados más concisos.

Encontrando una rota más corta de un nodo a otro se puede aplicar una clase de filtro - predicado. Podemos utilizar Neo4j en Java o Jruby, Scala, etc., crear una base de

datos en forma de una base integrada, utilizar SQLite on Rails, analizar la carga de nodos y las relaciones entre ellos y a continuación con un sencillo programa de Java encontrar la ruta más corta entre los nodos.

Al trabajar con SPARQL existen caminos de propiedad pero no se puede hacer el camino más corto.

Neo4j es una base de datos gráfica de primer nivel y esta optimizada para un trabajo bajo entornos Java y existe uniones embebidas Python y un interfaz REST.

Al trabajar bajo un ambiente PYTHON podríamos encontrarnos con los siguientes problemas:

- Enlaces de Python no son muy fáciles de instalar.
- La interfaz de REST es útil pero tiende a ser lenta y se vuelve impotente al momento de importar grandes cantidades de gráficos.
- El trabajar con Python para abrir archivos RDF y almacenarlos en Neo4j se pueden presentar algunos problemas de rendimiento inclusive si se trabaja localmente.

Existe una fuerte necesidad de que cada dato se encuentre fuertemente interconectado ya que cada relación puede determinar información más detallada y para conseguir aquello no hay nada mejor que la implementación de una base de datos basada en grafos.

Si nos referimos al rendimiento de grandes conjuntos de datos se puede trabajar con una base de datos relacional pero cuando surge la necesidad de acceder rápidamente a esos datos la opción de una base de datos basada en grafos se vuelve la solución óptima.

Como ya nos referimos un grafo puede reducirse a nodos, relaciones y propiedades que tiene asignados en forma clave/valor. La mayoría de las soluciones existentes que modelan bases de datos como grafos aplican esta abstracción de elementos pares clave/valor para poder funcionar y al no contener un estándar establecido para el uso de base de datos mediante grafos impide el traspaso de información entre distintos sistemas por ello se ha creado un proyecto que intenta solventar este inconveniente.

El proyecto Tinkerpop Blueprints (http://www.tinkerpop.com/ - 2012) [R.11] tiene como objetivo establecer un conjunto de interfaces trabajar conjuntamente con objetos

abstractos Grafo, Arista, Nodo y Propiedad, y de esta manera permitir que las aplicaciones puedan fácilmente migrar sus datos hacia una nueva implementación.

- Gremlin: Es un lenguaje de programación abierto escrito en Java y basado en grafos el cual permite realizar consultas, análisis y manipulaciones a los datos. Tiene una gran capacidad de modelar hipergrafos dentro de su propio lenguaje y una excelente característica que le permite adaptarse o implementarse en otros frameworks de grafos como: Neo4j, Sesame Sail, Quad Store o inclusive permite la creación de uno propio.
- Rexster: Crea una capa RESTful sobre cualquier grafo Blueprint y lo expone como servidor web e incluso es capaz de ejecutarse sobre varios sistemas a la vez. Quizás los esfuerzos dedicados a neo4j-rest-client deberían desviarse en este sentido, ya que Rexster proporcionaría potencialmente compatibilidad para todos los sistemas de bases de datos en grafo que se acojan a las especificaciones Blueprint.
- Pipes: que es un framework de flujo de datos con de/multiplexado y filtrado en los «canales» hacia o desde la entrada y la salida.
- Blueprints: Trabaja como una propiedad de interfaz gráfica. Cuando una base de datos gráfica aplica una interfaz Blueprints, ésta automáticamente es compatible con las aplicaciones que han sido habilitadas.
- Frames: los cuales exponen los elementos de un gráfico Blueprints como objetos Java. En lugar de escribir software en términos de vértices y aristas, con esta herramienta, el software está escrito en términos de los objetos de dominio y sus relaciones entre sí.
- Furnace: Es un paquete de propiedades gráficas de algoritmos el cual proporciona implementación de algoritmos de análisis gráficos estándares que se pueden aplicar a los gráficos de una propiedad de manera significativa.

Conclusión:

El utilizar las características inherentes de una base de datos basada en gráficos resulta ideal para el desarrollo de proyectos que exijan resultados concretos y especializados basados en las relaciones que sus datos puedan tener entre sí.

En lugar de realizar múltiples sub-consultas a nuestros datos para luego establecer relaciones por nuestra cuenta, el aplicar una base de datos gráfica nos permitirá ordenar rápidamente los datos bajo la marcha sin vernos obligados a diseñar un esquema de datos sin contar con las relaciones que puedan existir y a futuro

descubrir requerimientos que para ser cubiertos necesiten de un cambio global y radical de nuestra base de datos.

Esto significa que toda la información que se encuentre almacenada en una base de datos gráfica pueda ser organizada y mostrada rápidamente bajo distintas premisas o cualquier otra subcategoría llamando a los elementos individuales que la conforman una base de datos que mediante su indexación permitan establecer aquellas consultas especiales y devolvernos un resultado único.

"Si usted es capaz de esbozar su dominio en una pizarra, podrá traducir esto en un modelo de datos para su base de datos gráfica y podrá tomar esa pizarra y todo lo que ha dibujado como cajones en nodos y cada flecha como relaciones", expresó Emil Eifrem, CEO, Neo Technology en su artículo (Springtime for Graph Databases March 1, 2013)

[Anexo 6] Creación de una conexión y recorrido en una base de datos gráfica

Se ha tratado últimamente si las bases de datos NOSQL se adaptan y son la mejor opción para trabajar con entidades OCW a diferencia de las bases de datos relacionales o viceversa. El consenso al que se ha llegado determina que su uso depende del contexto en el que se requiera aplicar el uso de una base de datos.

La aplicación de un modelo de base de datos NOSQL se puede aplicar sustancialmente cuando es requerida una base de datos de grafos, por ejemplo cuando podemos utilizar elementos como recursos OCW para el proceso de análisis y recomendación de recursos académicos.

Con Neo4j se puede aspirar a crear aplicaciones que permitan manipular elementos y obtener resultados deducibles como es el caso de: las redes sociales, la realización de recomendaciones, y entre sus fortalezas podemos encontrar la facilidad de utilización de algoritmos gráficos como por ejemplo, camino más corto, conectividad, n grado de relaciones, etc.

Aunque también se pueden encontrar algún tipo de debilidades como la necesidad de que atravesar todo el gráfico para lograr una respuesta definitiva y tal vez, en ciertos casos, no hay facilidad de implementación de clúster

Existen normas ya disponibles, tales como SPARQL, un estándar que nos permitirá realizar consultas RDF o datos de tupla. De esta manera podríamos adaptar entidades y bases de datos gráficas.

Al ser almacenada la información en forma de grafos podemos ofrecer un API que sea capaz de permitir consultas a través de un recorrido de los grafos. Éste modelo se aplica de forma ideal para almacenar, explotar y devolver la información en entornos sociales, debido a que es muy útil el guardar las relaciones entre las entidades comunes mediante los arcos que se forman en los grafos.

En Graph Databases: The New Way to Access Super Fast Social Data [R13], se describe el uso de las bases de datos de grafos, estableciendo que su campo de uso es bastante extenso, entre algunos usos tenemos:

- Social networking y recomendaciones
- Google’s Knowledge Graph, Facebook's Social Graph, Twitter’s Interest Graph.
- Cloud management

- Master data management
- Gestión de contenidos, seguridad y control de acceso
- Geoespacial. Busqueda de caminos mínimos, elementos cercanos
- Bioinformática

La herramienta más factible para realizar el trabajo y mantenimiento de bases de datos de grafos es Neo4j.

El uso verdadero de una base de datos gráfica es permitir a las personas el crear un mapa de relaciones gráficas entre los elementos que están intentando trabajar.

El proceso de explorar una base de datos gráfica se fundamenta en sus relaciones, entre más conexiones sean establecidas, mucha más eficiente será la respuesta a solicitudes y mucho más precisos los resultados.

La implementación para el almacenamiento y uso de recursos RDF da la factibilidad de agregar, analizar y convertir los elementos que han ingresado a una estructura de nodo, mediante esta forma puede ser trabajado como nodo a nodo, analizar sus atributos, propiedades y las conexiones o relaciones que pueden existir entre ellos.

Gracias a esta característica se puede llegar a establecer valores específicos, sentencias de búsqueda, creación de relaciones en tiempo real que puedan necesitarse, analizar patrones o estructuras relacionadas entre los nodos que se vayan creando y permitir el crecimiento del mapa de datos.

La tecnología de una base mediante grafos brinda una potente opción al momento de iniciar la creación de una interfaz de usuario a nivel de desarrollador y de usuario final, lo que facilita el trabajo de visualización de los datos con los que se trabaja.

Con esta herramienta y gracias a sus múltiples características y posibilidades que tiene para manipular todo tipo de elementos, como las entidades OCW, podemos explotar la información y crear soluciones parciales que se adapten a nuestros requerimientos.

Al contar con la característica de adaptar la metadata de un elemento podemos ir creando las relaciones y los caminos que nos permitan llegar a una recomendación precisa de un recurso.

Por ejemplo:

- Un recurso tiene relación con otros recursos.
- Se puede recomendar un recurso similar al buscado.
- Un recurso tiene mayor prioridad que otro.
- Un recurso puede derivar otro tipo de elemento útil.

Al trabajar bajo un esquema de base de datos basado en grafos nos resultaría mucho más permitir a los usuarios realizar consultas de manera más sencilla y al mismo tiempo devolver resultados más precisos. Un ejemplo sería: ¿Cuáles son los artículos más valorados por los usuarios sobre una temática determinada?:

Si se tiene determinado un conjunto de relaciones específicas, una ruta planificada, sería muy fácil recomendarle los mejores artículos en función a las relaciones pasadas y de esta manera estaríamos retro alimentando y fortaleciendo el recurso OCW que es consultado para una consulta posterior.

Al trabajar con una estructura gráfica debemos entender que las relaciones que utiliza con unidireccionales, es decir, se mueven de un nodo a otro. Se puede realizar la búsqueda de relaciones indicando el sentido de las mimas, para ellos es necesario considerar la dirección es definida desde el nodo que estamos utilizando o partiendo.

Cuando realizamos aplicaciones que utilizan base de datos de grafos no nos enfocamos tanto en las búsquedas complejas basados en los valores de un nodo, sino que se utilizan y recorren las relaciones haciendo búsquedas simples por las relaciones entre nodos.

Al crear un motor de recomendación permitiremos a los usuarios encontrar artículos nuevos e interesantes dentro de un conjunto de recursos. Existen varios tipos de algoritmos de recomendación y mediante un gráfico podemos llegar a cumplir el propósito general de evaluar dichos algoritmos.

Podemos construir un motor de recomendación de recursos OCW utilizando una gráfica y basarnos en el uso de recursos, la creación de una gráfica Neo4j y el recorrido del grafo el lenguaje Gremlin – lenguaje script de grafos, y Cypher – Consultas “a la” SQL.

[Anexo 7] Ejercicio pre-procesamiento de texto

- Grupo de entidades

tropical moist broadleaf forests
tropical wet forest
tropical forest
equatorial rainforest
manigua
mixed deciduous forest
moist evergreen forest
subtropical moist broadleaf forest
subtropical moist forest
tropical and subtropical moist broadleaf forests
tropical evergreen forest
tropical moist deciduous forests
tropical moist evergreen forest
tropical moist forest
tropical moist forests
tropical or subtropical moist broadleaf forest
indonesian rainforest
indonesian rainforests
tropical forests
tropical moist broadleaf forest
tropical and subtropical moist broadleaf forest
tropical rain forest

<Cantidad>23</cantidad>

<Palabrasentrada>

<e>tropical moist broadleaf forests</e>

<e>tropical wet forest</e>

<e>tropical forest</e>

<e>equatorial rainforest</e>

<e>manigua</e>

<e>mixed deciduous forest</e>

<e>moist evergreen forest</e>

<e>subtropical moist broadleaf forest</e>

<e>subtropical moist forest</e>

<e>tsmf</e>

<e>tropical & subtropical moist broadleaf forests</e>

<e>tropical evergreen forest</e>

<e>tropical moist deciduous forests</e>

<e>tropical moist evergreen forest</e>

<e>tropical moist forest</e>

<e>tropical moist forests</e>

<e>tropical or subtropical moist broadleaf forest</e>

<e>indonesian rainforest</e>

<e>indonesian rainforests</e>

<e>tropical forests</e>

<e>tropical moist broadleaf forest</e>

<e>tropical and subtropical moist broadleaf forest</e>

<e>tropical rain forest</e>

- Grupo de palabras individuales para procesar estructura

</Palabrasentrada>

<Palabrasprocesadas>

<e>tropical</e>

<e>moist</e>

<e>broadleaf</e>

<e>forests</e>

<e>tropical</e>

<e>wet</e>

<e>forest</e>

<e>tropical</e>

<e>forest</e>

<e>equatorial</e>

<e>rainforest</e>

<e>manigua</e>

<e>mixed</e>

<e>deciduous</e>

</Palabrasprocesadas>

<Palabrassteam>

<e>

<Findin>

<e>indonesian rainforest</e>

<e>indonesian rainforests</e>

</Findin>

<Frecuencia>2</frecuencia>

<Token>indonesian</token>

</e>

<e>

- Agrupación número 1 (contiene tropical – subtropical)

La agrupación es realizada debido a que analiza la palabra tropical como la palabra base para agrupación y omite las demás palabras.

<Findin>

<e>tropical moist broadleaf forests</e>

<e>tropical wet forest</e>

<e>tropical forest</e>

<e>subtropical moist broadleaf forest</e>

<e>subtropical moist forest</e>

<e>tropical & subtropical moist broadleaf forests</e>

<e>tropical evergreen forest</e>

<e>tropical moist deciduous forests</e>

<e>tropical moist evergreen forest</e>

<e>tropical moist forest</e>

<e>tropical moist forests</e>

<e>tropical or subtropical moist broadleaf forest</e>

<e>tropical forests</e>

<e>tropical moist broadleaf forest</e>

<e>tropical and subtropical moist broadleaf forest</e>

<e>tropical rain forest</e>

</Findin>

<Frecuencia>14</frecuencia>

<Token>tropic</token>

</e>

<e>

- Agrupación número 2 (contiene tropical – subtropical)

La agrupación es realizada debido a que analiza la palabra subtropical como la palabra base para agrupación y omite las demás palabras. Se puede verificar que se repiten los elementos que ya contienen la palabra tropical debido a que conforman el grupo de palabras donde se encuentra subtropical.

<Findin>

<e>tropical moist broadleaf forests</e>

<e>tropical wet forest</e>

<e>tropical forest</e>

<e>equatorial rainforest</e>

<e>mixed deciduous forest</e>

<e>moist evergreen forest</e>

<e>subtropical moist broadleaf forest</e>

<e>subtropical moist forest</e>

<e>tropical & subtropical moist broadleaf forests</e>

<e>tropical evergreen forest</e>

<e>tropical moist deciduous forests</e>

<e>tropical moist evergreen forest</e>

<e>tropical moist forest</e>

<e>tropical moist forests</e>

<e>tropical or subtropical moist broadleaf forest</e>

<e>indonesian rainforest</e>

<e>indonesian rainforests</e>

<e>tropical forests</e>

<e>tropical moist broadleaf forest</e>

<e>tropical and subtropical moist broadleaf forest</e>

<e>tropical rain forest</e>

</Findin>

<Frecuencia>18</frecuencia>

<Token>forest</token>

</e>

- Agrupación número 3 (contiene tropical – subtropical)

La agrupación es realizada debido a que analiza la palabra moist como la palabra base para agrupación y omite las demás palabras. Se puede verificar que se repiten los elementos que ya contienen la palabra tropical y subtropical debido a que conforman el grupo de palabras donde se encuentra moist.

<e>

<Findin>

```
      <e>subtropical moist broadleaf forest</e>
      <e>subtropical moist forest</e>
      <e>tropical &amp;amp; subtropical moist broadleaf forests</e>
      <e>tropical or subtropical moist broadleaf forest</e>
      <e>tropical and subtropical moist broadleaf forest</e>
    </Findin>
    <Frecuencia>5</frecuencia>
    <Token>subtrop</token>
  </e>
  <e>
    <Findin>
      <e>equatorial rainforest</e>
      <e>indonesian rainforest</e>
      <e>indonesian rainforests</e>
    </Findin>
    <Frecuencia>3</frecuencia>
    <Token>rainforest</token>
  </e>
  <e>
    <Findin>
      <e>mixed deciduous forest</e>
```

```
    </Findin>
    <Frecuencia>1</frecuencia>
    <Token>mix</token>
  </e>
  <e>
    <Findin>
      <e>mixed deciduous forest</e>
      <e>tropical moist deciduous forests</e>
    </Findin>
    <Frecuencia>2</frecuencia>
    <Token>decidu</token>
  </e>
  <e>
    <Findin>
      <e>manigua</e>
    </Findin>
    <Frecuencia>1</frecuencia>
    <Token>manigua</token>
  </e>
  <e>
    <Findin>
```

```
      <e>equatorial rainforest</e>
    </Findin>
    <Frecuencia>1</frecuencia>
    <Token>equatori</token>
  </e>
  <e>
    <Findin>
      <e>tropical moist broadleaf forests</e>
      <e>subtropical moist broadleaf forest</e>
      <e>tropical &amp;amp; subtropical moist broadleaf forests</e>
      <e>tropical or subtropical moist broadleaf forest</e>
      <e>tropical moist broadleaf forest</e>
      <e>tropical and subtropical moist broadleaf forest</e>
    </Findin>
    <Frecuencia>6</frecuencia>
    <Token>broadleaf</token>
  </e>
  <e>
    <Findin>
      <e>tsmf</e>
    </Findin>
```

```
    <Frecuencia>1</frecuencia>
    <Token>tsmf</token>
  </e>
  <e>
    <Findin>
      <e>tropical moist broadleaf forests</e>
      <e>moist evergreen forest</e>
      <e>subtropical moist broadleaf forest</e>
      <e>subtropical moist forest</e>
      <e>tropical &amp;amp; subtropical moist broadleaf forests</e>
      <e>tropical moist deciduous forests</e>
      <e>tropical moist evergreen forest</e>
      <e>tropical moist forest</e>
      <e>tropical moist forests</e>
      <e>tropical or subtropical moist broadleaf forest</e>
      <e>tropical moist broadleaf forest</e>
      <e>tropical and subtropical moist broadleaf forest</e>
    </Findin>
    <Frecuencia>12</frecuencia>
    <Token>moist</token>
  </e>
```

<e>
<Findin>
<e>tropical wet forest</e>
</Findin>
<Frecuencia>1</frecuencia>
<Token>wet</token>
</e>
<e>
<Findin>
<e>moist evergreen forest</e>
<e>tropical evergreen forest</e>
<e>tropical moist evergreen forest</e>
</Findin>
<Frecuencia>3</frecuencia>
<Token>evergreen</token>
</e>
<e>
<Findin>
<e>equatorial rainforest</e>
<e>indonesian rainforest</e>
<e>indonesian rainforests</e>

<e>tropical rain forest</e>

</Findin>

<Frecuencia>1</frecuencia>

<Token>rain</token>

</e>

</Palabrassteam>

</Response>

- Resultado pretendido cuando se aplica el Algoritmo de Leverhstein:

Debido al que proceso de Leverhstein compara strings o cadenas de texto, en este caso cada una de los datos enviados, es requerida devolver en el mejor de los casos los elementos más similares agrupados para evitar la duplicidad de elementos en los grupos generados por el primer proceso de stemming.

- Grupo 1 (agrupa los resultados más similares, en este caso hay recursos que poseen palabras similares (tropical - forest))

<e>tropical wet forest</e>

<e>tropical forest</e>

<e>tropical evergreen forest</e>

<e>tropical forests</e>

<e>tropical rain forest</e>

- Grupo 2 (agrupa los resultados más similares, en este caso hay recursos que poseen palabras que general similaridad: tropical, subtropica, moist broadleft, forets)

<e>subtropical moist broadleaf forest</e>

<e>tropical & subtropical moist broadleaf forests</e>

<e>tropical moist broadleaf forest</e>

<e>tropical moist broadleaf forests</e>

<e>tropical and subtropical moist broadleaf forest</e>

<e>tropical moist deciduous forests</e>

<e>tropical moist evergreen forest</e>

<e>tropical moist forest</e>

<e>tropical moist forests</e>

<e>subtropical moist forest</e>

- Grupo 3 (agrupa los resultados más similares, en este caso hay recursos que poseen2 palabras similares)

<e>tropical moist deciduous forests</e>

<e>tropical moist evergreen forest</e>

<e>tropical moist forest</e>

<e>tropical moist forests</e>

- Grupo 2 (agrupa los resultados más similares, en este caso hay recursos que poseen palabras similares)

<e>equatorial rainforest</e>

<e>manigua</e>

<e>mixed deciduous forest</e>

<e>moist evergreen forest</e>

- Grupo 2 (agrupa los resultados más similares, en este caso hay recursos que poseen 2 palabras similares)

<e>indonesian rainforest</e>