**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

*La Universidad Católica de Loja*

**ÁREA TÉCNICA**

TITULACIONES DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

**Desarrollo de Servicios Web para el proceso de Enlace y Enriquecimiento de Datos   Enlazados**

TRABAJO DE FIN DE TITULACIÓN

**Autor:** Montaño Sozoranga, Wilmer Fabricio

**Director:** Piedra Pullaguari, Nelson Oswaldo, Ing

Loja - Ecuador

2014

Contenido

[Indice de FIguras 4](#_Toc399864517)

[Índice de tablas 5](#_Toc399864518)

[CAPITULO I: MARCO TEÓRICO 6](#_Toc399864519)

[1.1. Datos Enlazados 7](#_Toc399864520)

[1.1.1. Introducción 7](#_Toc399864521)

[1.1.2. Principios de Datos Enlazados 7](#_Toc399864522)

[1.1.3. Tecnologías 8](#_Toc399864523)

[1.1.3.1. URI 8](#_Toc399864524)

[1.1.3.2. RDF 11](#_Toc399864525)

[1.1.3.3. SPARQL Query Language for RDF 19](#_Toc399864526)

[1.1.4. Acerca de DBpedia 20](#_Toc399864527)

[1.1.4.1. Framework extracción 20](#_Toc399864528)

[1.1.4.2. DBpedia Dataset 21](#_Toc399864529)

[1.1.4.3. Acceso a DBpedia Dataset 21](#_Toc399864530)

[1.2. Procesamiento de Lenguaje Natural (PLN) 22](#_Toc399864531)

[1.2.1. Introducción 22](#_Toc399864532)

[1.2.2. Part of Speech Tagger 23](#_Toc399864533)

[1.2.3. Chunking 23](#_Toc399864534)

[1.2.4. Desambiguación 24](#_Toc399864535)

[1.2.4.1. Métodos basados en el conociendo. 24](#_Toc399864536)

[1.2.4.2. Métodos supervisados 25](#_Toc399864537)

[1.2.4.3. Métodos semi-supervisados 25](#_Toc399864538)

[1.2.4.4. Métodos sin supervisión 26](#_Toc399864539)

[1.3. Servicios Web 26](#_Toc399864540)

[1.3.1. Introducción 26](#_Toc399864541)

[1.3.2. Tipos de Web Services 26](#_Toc399864542)

[1.3.2.1. SOAP AND THE WS-\* STACK 26](#_Toc399864543)

[1.3.2.2. REST 27](#_Toc399864544)

[1.3.2.3. Hipermedia 28](#_Toc399864545)

[1.3.2.4. Recursos y representaciones 28](#_Toc399864546)

[1.3.2.5. URI y relación con los recursos 29](#_Toc399864547)

[1.4. Trabajos relacionados 29](#_Toc399864548)

[CAPITULO 2: PROBLEMÁTICA 30](#_Toc399864549)

[2.1. Estado actual 31](#_Toc399864550)

[2.2. Justificación 31](#_Toc399864551)

[2.3. Objetivo General. 32](#_Toc399864552)

[2.4. Objetivos Específicos 32](#_Toc399864553)

[CAPITULO 3: Solución 33](#_Toc399864554)

[3.1. Propuesta 34](#_Toc399864556)

[3.2. Metodología 35](#_Toc399864557)

[3.2.1. Fases de desarrollo 35](#_Toc399864558)

[3.3. Desarrollo 36](#_Toc399864559)

[3.3.1. Análisis de requerimientos 36](#_Toc399864560)

[3.3.1.1. Requerimientos 36](#_Toc399864561)

[3.3.1.2. Modelo de Dominio 36](#_Toc399864562)

[3.3.1.3. Modelo de caso de Uso 37](#_Toc399864563)

[3.3.2. Análisis y diseño preliminar 38](#_Toc399864564)

[3.3.2.1. Descripción de casos de uso 38](#_Toc399864565)

[3.3.3. Diseño 43](#_Toc399864566)

[3.3.3.1. Arquitectura 43](#_Toc399864567)

[3.3.3.2. Módulos 44](#_Toc399864568)

[3.3.3.3. Diagrama de secuencia 47](#_Toc399864569)

[3.3.4. Implementación 50](#_Toc399864570)

[CAPITULO 4 51](#_Toc399864571)

[4.1. Validación 51](#_Toc399864573)

[4.1.1. Comparación con servicios similares 51](#_Toc399864574)

[4.2. Pruebas 51](#_Toc399864575)

[CAPITULO 5 51](#_Toc399864576)

[5.1. Conclusiones y Recomendaciones 51](#_Toc399864578)

[Bibliografía 52](#_Toc399864579)

[Anexos 54](#_Toc399864580)

Indice de FIguras

[Figura 1: Relación entre URI, URL y URN. Fuente: propio 11](#_Toc393116457)

[Figura 2: Estructura de la URI (Scheme URI). Fuente: (Albahari & Albahari, 2012) 11](#_Toc393116458)

[Figura 3. RDF 1.0 y 1.1 formatos de serialización. 13](#_Toc393116459)

[Figura 4: RDF con dos nodos (Subject y Object) y una conexión entre ellos (Predicado). 13](#_Toc393116460)

[Figura 5: Representación gráfica RDF 22](#_Toc393116461)

[Figura 6. Arquitectura de provisión de Datos de Dbpedai. 23](#_Toc393116462)

[Figura 7: Ejemplo POS Tag y Chunking 25](#_Toc393116463)

[Figura 8. Lógica propuesta para la Aplicación; 35](#_Toc393116464)

[Figura 9. Arquitectura. 36](#_Toc393116465)

[Figura 10. Diagrama de secuencias del módulo de Etiquetado 39](#_Toc393116466)

[Figura 11. Diagrama de secuencias del módulo de Extracción de Entidades y Keywords 40](#_Toc393116467)

[Figura 12. Diagrama de secuencias módulo de Desambiguación y Enlace 41](#_Toc393116468)

Índice de tablas

[Tabla 1. Ejemplos de Prefijos de Espacios de Nombres e IRIs 14](#_Toc393117110)

[Tabla 3. Ejemplo N-triple 15](#_Toc393117111)

[Tabla 4. Ejemplo Turtle 16](#_Toc393117112)

[Tabla 5. Ejemplo TriG 16](#_Toc393117113)

[Tabla 6. Ejemplo N-Quads 17](#_Toc393117114)

[Tabla 7. Ejemplo JSON-LD 18](#_Toc393117115)

[Tabla 8. Ejemplo RDFa 18](#_Toc393117116)

[Tabla 9. Ejemplo RDF/XML 19](#_Toc393117117)

[Tabla 10. Resultado consulta SPARQL 21](#_Toc393117118)

CAPITULO I: MARCO TEÓRICO

* 1. Datos Enlazados
     1. Introducción

En sus inicios la web en su primera versión 1.0, donde web era rígida en cuanto a la entrega de información, además de poco actualizada, convertía al visitante de un sitio web un simple lector, restringido de cualquier interacción. Se puede decir que la web no era más que paginas enlazadas mediante hipervínculos.

La web que siempre está creciendo y evolucionado, alcanza su versión conocida como la web 2.0 en donde usuario juega el papel más importante, es quien evalúa, puede calificar, compartir, rectificar, alimentar y subir su propia información a la web. Esto producto de le aparición de nuevas tecnologías y estandarización[[1]](#footnote-1).

Los datos relacionados llegan para dar forma a la siguiente versiona de la web, la web semántica de W3C[[2]](#footnote-2) nos dice : “Linked Data se refiere a la utilización delas mejores prácticas para publicación, estructuración de los datos en la web, de tal forma que puedan ser enlazados entre sí, utilizando tecnología propias de web semántica como RDF, OCW, SPARQL, etc.”

Se refiere en si a la estructura de la de la siguiente generación de la web, como es la web semántica, que en si busca que la información que se en publica en internet pueda no solo ser entendida por seres humanos sino también por la maquinas que navegan en la web. En donde a partir de un dato podemos descubrir otros datos por sus relaciones.

* + 1. Principios de Datos Enlazados

Tim Berners Lee en su publicación Linked Data - Design Issues (Berners-Lee, Linked Data - Design Issues, 2006)describe cuatro reglas base para la publicación de datos enlazados:

1. Usar URIs como nombre de las cosas
2. Usar URIs HTTP par que esas cosas puedan ser referenciadas
3. Representar los datos en RDF y SPARQL como lenguaje de consulta
4. Incluir enlaces hacia otra cosas, para descubrir más cosas

La utilizar de estas reglas para la publicación de datos, nos permite que estos por las características propias de las tecnologías sobre las cuales se construyen como:

* Las cosas que nombremos por URIs son inequívocas y estos serán recursos.
* Los detalles o atributos y las relaciones de los datos van a estar descritos y estructurados en formato RDF
* Podremos acceder o realizar consultas sobre estos mediante SPARQL
* La cosas que publiquemos estarán relacionados
  + 1. Tecnologías
       1. URI

URI (Uniform Resource Identifier) ha sido desarrollado por el [IETF](http://www.ietf.org/)[[3]](#footnote-3) y pretende crear un sistema mundial para identificar recursos de todo tipo en la web: documentos, imágenes, programas, servicios, correos electrónicos, etc. Este método combina URNs y URLs, esto es, nombres/direcciones. Se trata de identificar los documentos mediante una secuencia de sintaxis controlada que identifica cada documento de una forma única. (...). Los URIs hacen posible encontrar los recursos bajo una gran variedad de esquemas definidos y métodos de acceso tales como HTTP, FTP, Gopher, news, telnet o correos electrónicos localizables siempre de la misma manera, ya que a un mismo documento se puede acceder desde distintos protocolos. Ya se han establecido una serie de schemes o esquemas direccionados. Los esquemas definidos URI coinciden con los protocolos más usados de Internet. (Lapuente, 2013)

En (Berners-Lee, 2005) el RFC[[4]](#footnote-4) que trata sobre este tema nos dice lo siguiente: “Un identificador uniforme de recursos (URI) proporciona un medio simple y extensible para la identificación de un recurso”. Es base a los términos que lo conforman se explica como:

“Uniforme: Uniformidad ofrece varios beneficios. Permite diferentes tipos de identificadores de recursos que se utilizarán en el mismo contexto, aun cuando los mecanismos utilizados para acceder a esos recursos pueden ser diferentes. Permite la interpretación semántica uniforme de convenciones sintácticas comunes a través de diferentes tipos de identificadores de recursos (…).

Recurso: Esta especificación no limita el alcance de lo que podría ser un recurso; más bien, el término "recurso" se utiliza en un sentido general de lo que pudiera ser identificado por un URI. Ejemplos conocidos incluyen un documento electrónico, una imagen, una fuente de información con un propósito consistente (por ejemplo, "parte meteorológico de hoy para Los Ángeles"), un servicio (por ejemplo, una puerta de enlace HTTP a SMS), y una colección de otros recursos. Un recurso no es necesariamente accesible a través de Internet; por ejemplo, los seres humanos, las empresas y los libros encuadernados en una biblioteca también pueden ser recurso (…).

Identificador: Un identificador encarna la información necesaria para distinguir lo que se identificó a partir de todas las otras cosas dentro de su ámbito de aplicación de la identificación. Nuestro uso de los términos "identificar" y "identificación" se refieren a este fin de distinguir un recurso de todos los demás recursos, independientemente de cómo se logra ese propósito (ejemplos, nombre, dirección, o el contexto). Estos términos no deben confundirse con la presunción de que un identificador define o encarna la identidad de lo que se hace referencia, aunque esto puede ser el caso de algunos identificadores. Tampoco debe asumirse que un sistema que utiliza los URI tendrá acceso al recurso identificado: en muchos casos, los URI se utilizan para referirse a los recursos sin ninguna intención de que se puede acceder. Del mismo modo, "un" recurso identificado podría no ser singular en la naturaleza (ejemplo, un recurso puede ser un conjunto con nombre o una asignación que varía con el tiempo).” (Berners-Lee, 2005)

Suele existir confusión entre URI, URN y URL, que las describimos a continuación de acuerdo a (Berners-Lee, 2005): “El término "Uniform Resource Locator" (URL[[5]](#footnote-5)) se refiere al subconjunto de URIs que, además de la identificación de un recurso, proporcionan los medios para localizar el recurso mediante la descripción de su mecanismo de acceso primario (por ejemplo, su red de "ubicación"). El término "Uniform Resource Name" (URN[[6]](#footnote-6)) se ha utilizado históricamente para referirse tanto a los URI en el marco del esquema de "URN" [RFC2141], que son necesarios para permanecer globalmente único y persistente, incluso cuando el recurso deja de existir o no está disponible, y a cualquier otro URI con las propiedades de un nombre”. La diferencia en grafico de estos conceptos se la puede ver en el Figura 1.

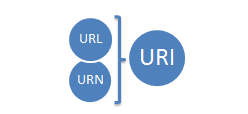


Figura 1: Relación entre URI, URL y URN. Fuente: propio

La estructura de una URI es explicada en el Figura 1 de (Albahari & Albahari, 2012), donde vemos los componentes: Scheme, Authority (Host, Port), PathAndQuery (AbsolutyPath, Query y Fragment), UserInfo.

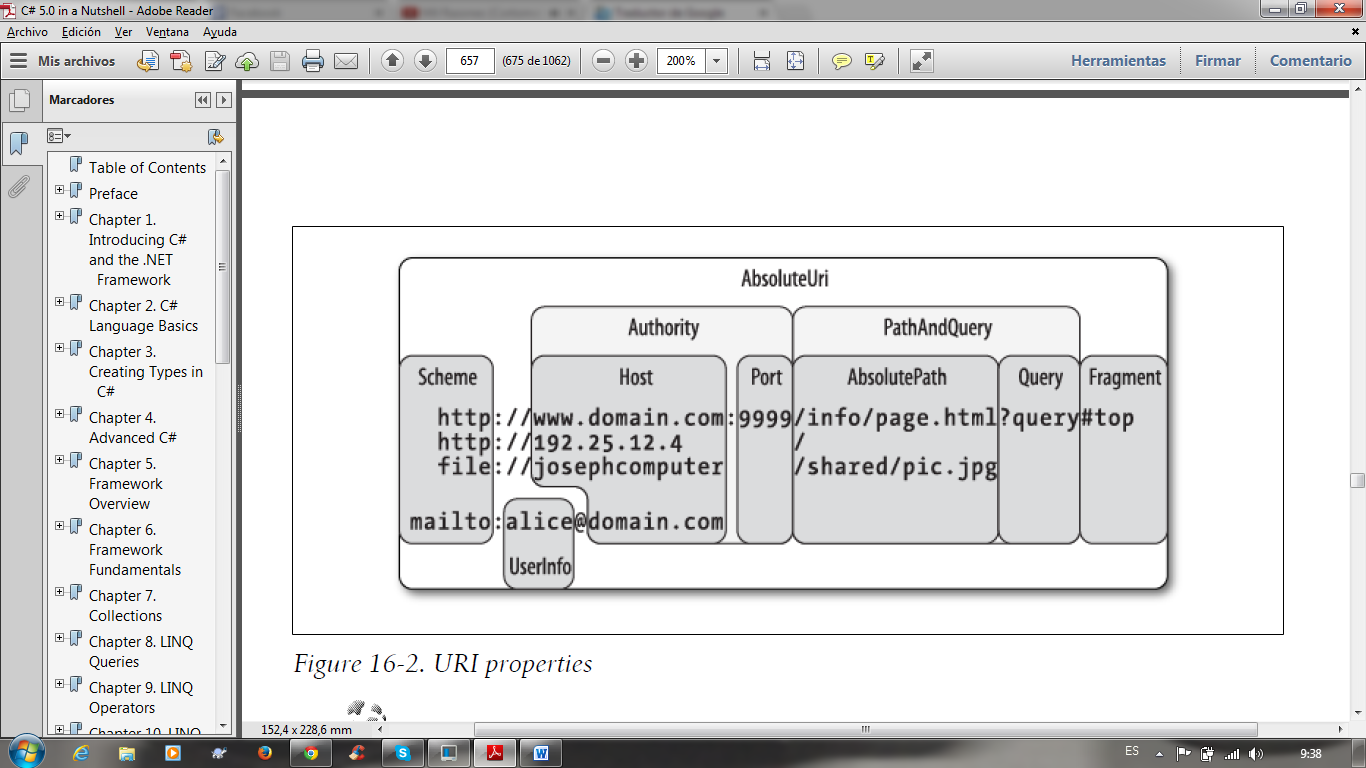


Figura 2: Estructura de la URI (Scheme URI). Fuente: (Albahari & Albahari, 2012)

(Lapuente, 2013) recoge alguna definiciones sobre las URIs de Architecture of the World Wide Web, Volume One especificación publicada por W3C, que exponemos acountinuacion:

* URI: Acrónimo para Uniform Resource Identificer.
* URI aliases: dos o más URIs que son -carácter por carácter-, diferentes, pero que identifican el mismo recurso.
* URI overloading: uso del mismo URI para referirse a más de un recurso en el contexto de los protocolos y formatos de la Web.
* URI ownership: la relación entre el agente que asigna y el URI que es definido por un esquema URI.
* URI persistence: la expectación social que desde hace algún tiempo identifica URI a un recurso particular, este podría continuar indefinidamente para referirse al recurso.
* URI reference: un apunte operacional para un URI.
* Uniform Resource Identifier (URI): un identificador global en el contexto de la World Wide Web.
* Namespace document: el recurso de información identificado por un [namespace](http://www.hipertexto.info/documentos/namespaces.htm) URI en [XML](http://www.hipertexto.info/documentos/xml.htm).
* Link o enlace: una relación entre dos recursos cuando un recurso (representación) se refiera al otro recurso mediante el significado de un URI.
  + - 1. RDF

El Marco de Descripción de Recursos (RDF) es un lenguaje para representar la información acerca de los recursos en la World Wide Web. En particular, se pretende para la representación de metadatos sobre recursos web, como el título, el autor y la fecha de modificación de una página Web, los derechos de autor y la información de licencia de un documento Web, o el calendario de disponibilidad de algún recurso compartido. Sin embargo, al generalizar el concepto de un "recurso de la Web", RDF puede también ser utilizado para representar la información acerca de las cosas que se pueden identificar en la red, incluso cuando no pueden recuperarse directamente en la Web. Los ejemplos incluyen información acerca de productos disponibles de servicios en línea de la compra (por ejemplo, información acerca de las especificaciones, los precios y la disponibilidad), o la descripción de las preferencias del usuario Web para la entrega de información. (McBride, 2004)

Desde la primera publicación de RDF en su versión 1.0 en febrero del 2004 han pasado 10 años para que en la primera mitad del 2014 poder conocer la versión 1.1, las novedades de esta nueva versión y las diferencias entre las versiones podemos encontrarlas en *What’s New in RDF 1.1*[[7]](#footnote-7)*,* entre las cuales se resalta dos, la primera que ahora se utiliza IRIs en lugar de URIs para identificar a los recursos y segundo nuevos formatos se serialización, este último lo visualizamos en la Figura 3.

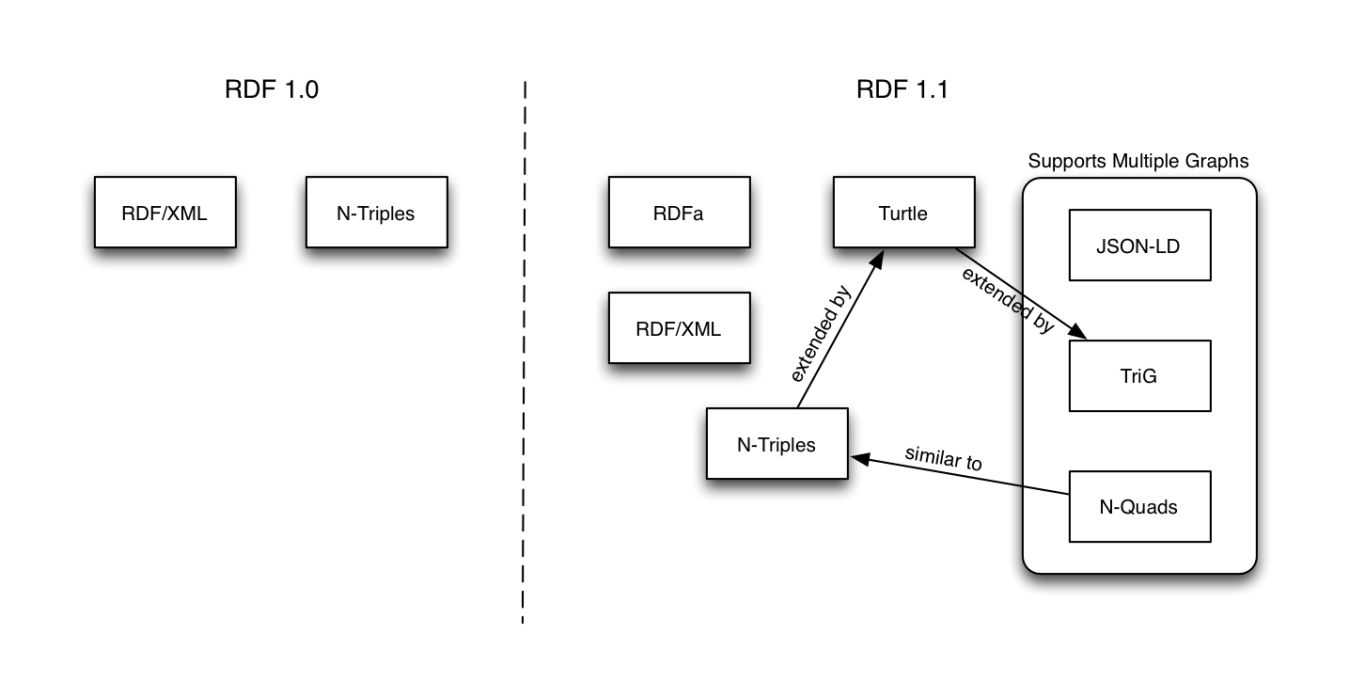


Figura 3. RDF 1.0 y 1.1 formatos de serialización.

Fuente: (Wood, 2014)

* + - * 1. RDF Data Model

La estructura básica de la sintaxis abstracta es un conjunto de triples, cada una consiste de un sujeto, un predicado y un objeto. Un conjunto de triples se llama grafo RDF. Un grafo RDF se puede visualizar como un nodo y el diagrama de arco dirigido, en el que cada uno de triple se representa como un enlace nodo-arco-nodo. (Cyganiak, Wood, & Lanthaler, 2014)

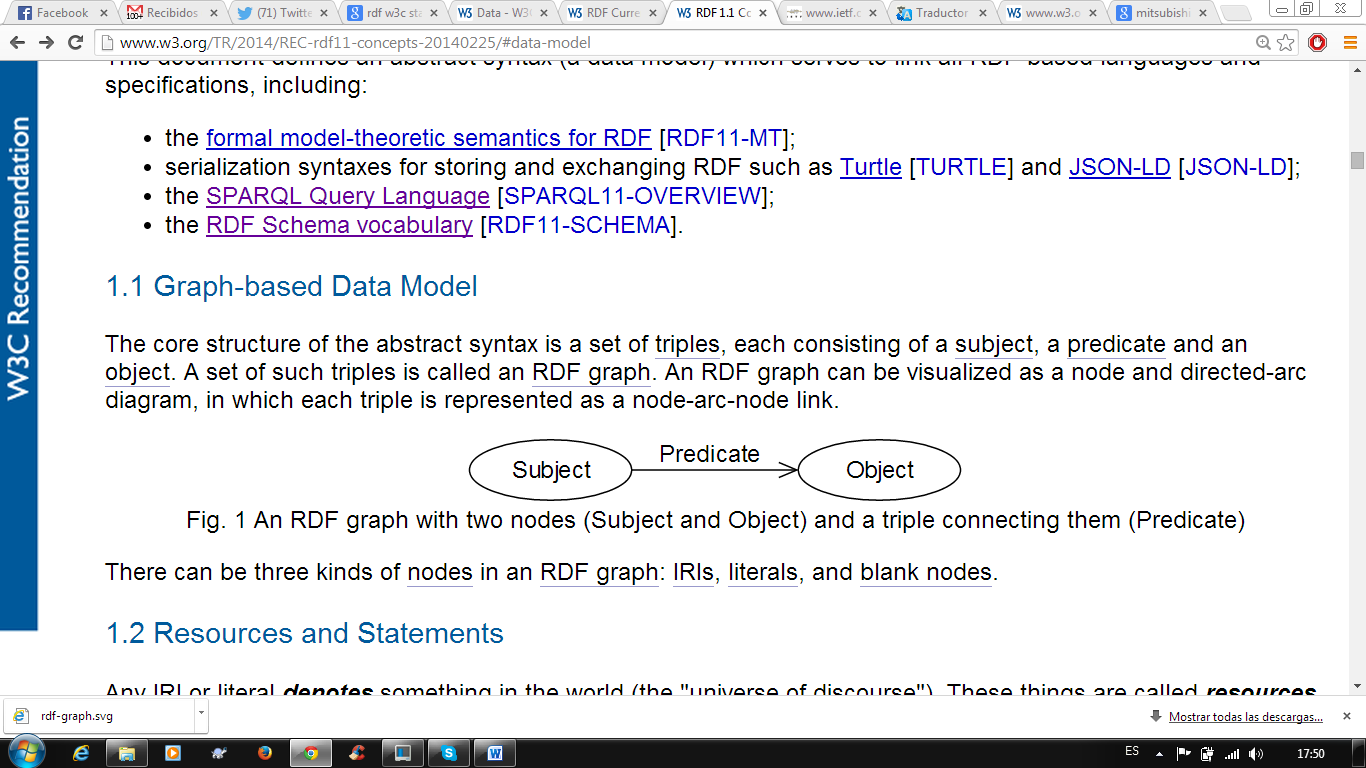


Figura 4: RDF con dos nodos (Subject y Object) y una conexión entre ellos (Predicado).

Fuente: (Cyganiak, Wood, & Lanthaler, 2014)

* + - * 1. RDF Vocabularios y Espacio de Nombres IRIs

Un RDF vocabulario es una colección de IRIs destinados para ser usados en grafos RDF. Por ejemplo, los IRIs documentados en [RDF11-SCHEMA] [[8]](#footnote-8) son el Vocabulario RDF Esquema (RDF Schema Vocabulary). (Cyganiak, Wood, & Lanthaler, 2014)

Una colección de "términos" para un propósito en particular. Los vocabularios pueden ir desde simples el ampliamente usado Schema RDF utilizado, FOAF[[9]](#footnote-9) y Dublin Core Metadata Element Set[[10]](#footnote-10) para vocabularios complejos con miles de términos, tales como los utilizados en la asistencia sanitaria para describir síntomas, enfermedades y tratamientos. Vocabularios juegan un papel muy importante en Linked Data, específicamente para ayudar con la integración de datos. El uso de este término se superpone con la Ontología. (W3C, 2013)

El iris de un vocabulario RDF a menudo comienzan con una subcadena común conocido como un espacio de nombres IRI. Algunos IRIs de espacio de nombres se asocian por convención con un nombre corto conocido como un prefijo de espacio de nombres (namespace prefix). (W3C, 2013)

El término "espacio de nombres" por sí no tiene un significado bien definido en el contexto de la RDF, pero a veces se utiliza de manera informal en el sentido de "espacio de nombres IRI" o "vocabulario RDF”. (W3C, 2013)

Tabla 1. Ejemplos de Prefijos de Espacios de Nombres e IRIs

|  |  |  |
| --- | --- | --- |
| **Namespace prefix** | **Namespace IRI** | **RDF vocabulary** |
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# | The RDF built-in vocabulary [RDF11-SCHEMA][[11]](#footnote-11) |
| rdfs | http://www.w3.org/2000/01/rdf-schema# | The RDF Schema vocabulary [RDF11-SCHEMA][[12]](#footnote-12) |
| xsd | http://www.w3.org/2001/XMLSchema# | The RDF-compatible XSD types[[13]](#footnote-13) |

Fuente: (Cyganiak, Wood, & Lanthaler, 2014)

* + - * 1. Especificación de formatos de serialización de RDF

N-Triples

Tripletas N-Triples son una secuencia de términos RDF que representan al sujeto, predicado y objeto de una Tripleta RDF. Estos pueden estar separados por espacios en blanco (espacios U +0020 o tabulaciones U +0009). Esta secuencia es terminada por un '.' y una nueva línea (opcional al final de un documento). (Carothers & Seaborne, 2014)

N-triple en formato en texto plano para grafos RDF, en la Tabla 2 se ejemplifica su estructura.

Tabla 2. Ejemplo N-triple

|  |  |
| --- | --- |
| 01 | <http://example.org/bob#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> . |
| 02 | <http://example.org/bob#me> <http://xmlns.com/foaf/0.1/knows> <http://example.org/alice#me> . |
| 03 | <http://example.org/bob#me> <http://schema.org/birthDate>  "1990-07-04"^^<http://www.w3.org/2001/XMLSchema#date> . |
| 04 | <http://example.org/bob#me> <http://xmlns.com/foaf/0.1/topic\_interest> <http://www.wikidata.org/entity/Q12418> . |
| 05 | <http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/title> "Mona Lisa" . |
| 06 | <http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/creator> <http://dbpedia.org/resource/Leonardo\_da\_Vinci> . |
| 07 | <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619> <http://purl.org/dc/terms/subject> <http://www.wikidata.org/entity/Q12418>. |

Fuente: (Schreiber & Raimond, 2014)

[Turtle](http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/#section-turtle)

Permite a un grafo RDF a ser completamente escrito en un formulario de texto compacto y natural, con las abreviaturas para los patrones y tipos de datos de uso común. Turtle ofrece niveles de compatibilidad con el formato N-Triples, así como con la sintaxis de patrón de tripletas de la Recomendación de la W3C de SPARQL[[14]](#footnote-14). (Beckett, Berners-Lee, Prud'hommeaux, Carothers, & Machina., 2014)

Turtle es una extensión del N-Triples. Además de la sintaxis básica N-Triples, Turtle introduce una serie de atajos sintácticos, como el soporte para prefijos de espacio de nombres, listas y abreviaturas para datos tipo literales. Turtle ofrece una compensación entre la facilidad de la escritura, la facilidad de análisis y facilidad de lectura. (Schreiber & Raimond, 2014)

Tabla 3. Ejemplo Turtle

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07 | <http://example.org/>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  PREFIX schema: <http://schema.org/>  PREFIX dcterms: <http://purl.org/dc/terms/>  PREFIX wd: <http://www.wikidata.org/entity/> |
| 08  09  10  11  12  13 | <bob#me>  a foaf:Person ;  foaf:knows <alice#me> ;  schema:birthDate "1990-07-04"^^xsd:date ;  foaf:topic\_interest wd:Q12418 . |
| 14  15  16  17 | wd:Q12418  dcterms:title "Mona Lisa" ;  dcterms:creator <http://dbpedia.org/resource/Leonardo\_da\_Vinci> . |
| 18  19 | <http://data.europeana.eu/item/04802/243FA8618938F4117025F  17A8B813C5F9AA4D619>  dcterms:subject wd:Q12418 . |

Fuente: (Schreiber & Raimond, 2014)

[TriG](http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/#section-trig)

La sintaxis de la Turtle sólo soporta la especificación de grafos simples sin un medio para "nombrarlos". TriG es una extensión de la Turtle que permite la especificación de múltiples grafos en forma de un conjunto de datos RDF. (Schreiber & Raimond, 2014)

Un documento TriG permite escribir un conjunto de datos RDF en una forma textual compacta. Se consiste de una sucesión de directivas, declaraciones triples, declaraciones de grafos que contienen declaraciones triple-generación y líneas en blanco opcionales. (Bizer & Cyganiak, 2014)

Tabla 4. Ejemplo TriG

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07 | BASE <http://example.org/>  PREFIX foaf: <http://xmlns.com/foaf/0.1/>  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  PREFIX schema: <http://schema.org/>  PREFIX dcterms: <http://purl.org/dc/terms/>  PREFIX wd: <http://www.wikidata.org/entity/> |
| 08  09  10  11  12  13  14  15  16 | GRAPH <http://example.org/bob>  {  <bob#me>  a foaf:Person ;  foaf:knows <alice#me> ;  schema:birthDate "1990-07-04"^^xsd:date ;  foaf:topic\_interest wd:Q12418 .  } |
| 17  18  19  20  21  22  23  24  25  26 | GRAPH <https://www.wikidata.org/wiki/Special:EntityData/Q12418>  {  wd:Q12418  dcterms:title "Mona Lisa" ;  dcterms:creator <http://dbpedia.org/resource/Leonardo\_da\_Vinci> .    <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>  dcterms:subject wd:Q12418 .  } |
| 27  28  29 | <http://example.org/bob>  dcterms:publisher <http://example.org> ;  dcterms:rights <http://creativecommons.org/licenses/by/3.0/> . |

Fuente: (Schreiber & Raimond, 2014)

N-Quads

N-Quads es una simple extensión de N-Triples para permitir el intercambio de RDF datasets. N-Quads le permite a uno agregar un cuarto elemento a una línea, capturando en la gráfica IRI la tripleta descrito en esa línea. (Schreiber & Raimond, 2014)

Tabla 5. Ejemplo N-Quads

|  |  |
| --- | --- |
| O1 | <http://example.org/bob#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> <http://example.org/bob> . |
| 02 | <http://example.org/bob#me> <http://xmlns.com/foaf/0.1/knows> <http://example.org/alice#me> <http://example.org/bob> . |
| 03 | <http://example.org/bob#me> <http://schema.org/birthDate> "1990-07-04"^^<http://www.w3.org/2001/XMLSchema#date> <http://example.org/bob> . |
| 04 | <http://example.org/bob#me> <http://xmlns.com/foaf/0.1/topic\_interest> <http://www.wikidata.org/entity/Q12418> <http://example.org/bob> . |
| 05 | <http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/title> "Mona Lisa" <https://www.wikidata.org/wiki/Special:EntityData/Q12418> . |
| 06 | <http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/creator> <http://dbpedia.org/resource/Leonardo\_da\_Vinci> <https://www.wikidata.org/wiki/Special:EntityData/Q12418> . |
| 07 | <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619> <http://purl.org/dc/terms/subject> <http://www.wikidata.org/entity/Q12418> <https://www.wikidata.org/wiki/Special:EntityData/Q12418> . |
| 08 | <http://example.org/bob> <http://purl.org/dc/terms/publisher> <http://example.org> . |
| 09 | <http://example.org/bob> <http://purl.org/dc/terms/rights> <http://creativecommons.org/licenses/by/3.0/> . |

Fuente: (Schreiber & Raimond, 2014)

JSON-LD (JSON-based RDF syntax);

Proporciona una sintaxis JSON para gráficos y conjuntos de datos RDF. JSON-LD puede ser utilizado para transformar documentos JSON a RDF con cambios mínimos. JSON-LD ofrece identificadores universales de objetos JSON, un mecanismo en el que un documento JSON se puede referir a un objeto descrito en otro documento JSON en otros lugares en la Web, así como el tipo de datos y el lenguaje de manipulación.

Tabla 6. Ejemplo JSON-LD

|  |  |
| --- | --- |
| O1 | { |
| 02 | "@context": "example-context.json", |
| 03 | "@id": "http://example.org/bob#me", |
| 04 | "@type": "Person", |
| 05 | "birthdate": "1990-07-04", |
| 06 | "knows": "http://example.org/alice#me", |
| 07 | "interest": { |
| 08 | "@id": "http://www.wikidata.org/entity/Q12418", |
| 09 | "title": "Mona Lisa", |
| 10 | "subject\_of": "http://data.europeana.eu/item/04802/ 243FA8618938F4117025F17A8B813C5F9AA4D619", |
| 11 | "creator": "http://dbpedia.org/resource/Leonardo\_da\_Vinci" |
| 12 | } |
| 13 | } |

Fuente: (Schreiber & Raimond, 2014)

[RDFa](http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/#section-rdfa)

Es una sintaxis de RDF que se puede utilizar para insertar datos RDF dentro de los documentos HTML y XML. Esto permite, por ejemplo, a los motores de búsqueda agregar estos datos al rastrear la Web y enriquecer los resultados de búsqueda. (Schreiber & Raimond, 2014)

Tabla 7. Ejemplo RDFa

|  |  |
| --- | --- |
| O1 | <body prefix="foaf: http://xmlns.com/foaf/0.1/ |
| 02 | schema: http://schema.org/ |
| 03 | dcterms: http://purl.org/dc/terms/"> |
| 04 | <div resource="http://example.org/bob#me" typeof="foaf:Person"> |
| 05 | <p> |
| 06 | Bob knows <a property="foaf:knows" href="http://example.org/alice#me">Alice</a> |
| 07 | and was born on the <time property="schema:birthDate" datatype="xsd:date">1990-07-04</time>. |
| 08 | </p> |
| 09 | <p> |
| 10 | Bob is interested in <span property="foaf:topic\_interest" |
| 11 | resource="http://www.wikidata.org/entity/Q12418">the Mona Lisa</span>. |
| 12 | </p> |
| 13 | </div> |
| 14 | <div resource="http://www.wikidata.org/entity/Q12418"> |
| 15 | <p> |
| 16 | The <span property="dcterms:title">Mona Lisa</span> was painted by |
| 17 | <a property="dcterms:creator" href="http://dbpedia.org/resource/Leonardo\_da\_Vinci">Leonardo da Vinci</a> |
| 18 | and is the subject of the video |
| 19 | <a href="http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619">'La Joconde à Washington'</a>. |
| 20 | </p> |
| 21 | </div> |
| 22 | <div resource="http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619"> |
| 23 | <link property="dcterms:subject" href="http://www.wikidata.org/entity/Q12418"/> |
| 24 | </div> |
| 25 | </body> |

Fuente: (Schreiber & Raimond, 2014)

[RDF/XML](http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/#section-rdfxml)

Proporciona una sintaxis XML para grafos RDF. Cuando RDF ​​fue desarrollado originalmente en la década de 1990, esta fue su única sintaxis, y algunas personas siguen llamando esta sintaxis "RDF". En 2001, se propuso un precursor de la Tortuga llamado "N3", y poco a poco los otros idiomas que aparecen aquí se han adoptado y normalizado. (Schreiber & Raimond, 2014)

Tabla 8. Ejemplo RDF/XML

|  |  |
| --- | --- |
| O1 | <?xml version="1.0" encoding="utf-8"?> |
| 02 | <rdf:RDF |
| 03 | xmlns:dcterms="http://purl.org/dc/terms/" |
| 04 | xmlns:foaf="http://xmlns.com/foaf/0.1/" |
| 05 | xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" |
| 06 | xmlns:schema="http://schema.org/"> |
| 07 | <rdf:Description rdf:about="http://example.org/bob#me"> |
| 08 | <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/> |
| 09 | <schema:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1990-07-04</schema:birthDate> |
| 10 | <foaf:knows rdf:resource="http://example.org/alice#me"/> |
| 11 | <foaf:topic\_interest rdf:resource="http://www.wikidata.org/entity/Q12418"/> |
| 12 | </rdf:Description> |
| 13 | <rdf:Description rdf:about="http://www.wikidata.org/entity/Q12418"> |
| 14 | <dcterms:title>Mona Lisa</dcterms:title> |
| 15 | <dcterms:creator rdf:resource="http://dbpedia.org/resource/Leonardo\_da\_Vinci"/> |
| 16 | </rdf:Description> |
| 17 | <rdf:Description rdf:about="http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619"> |
| 18 | <dcterms:subject rdf:resource="http://www.wikidata.org/entity/Q12418"/> |
| 19 | </rdf:Description> |
| 20 | </rdf:RDF> |

Fuente: (Schreiber & Raimond, 2014)

* + - 1. SPARQL Query Language for RDF

SPARQL se puede utilizar para expresar consultas que permiten interrogar diversas fuentes de datos, si los datos se almacenan de forma nativa como RDF ​​o son definidos mediante vistas RDF ​​a través de algún sistema middleware. SPARQL contiene las capacidades para la consulta de los patrones obligatorios y opcionales de grafo, junto con sus conjunciones y disyunciones. SPARQL también soporta la ampliación o restricciones del ámbito de las consultas indicando los grafos sobre los que se opera. Los resultados de las consultas SPARQL pueden ser conjuntos de resultados o grafos RDF. (Prud'hommeaux & Seaborne, 2008)

La mayoría de las formas de consulta en SPARQL contienen un conjunto de patrones de tripleta (triple patterns) denominadas patrón de grafo básico. Los patrones de tripleta son similares a las tripletas RDF, excepto que cada sujeto, predicado y objeto puede ser una variable. Un patrón de grafo básico concuerda con un subgrafo de datos RDF cuando los términos RDF (RDF terms) de dicho subgrafo pueden ser sustituidos por las variables y el resultado es un grafo RDF equivalente al subgrafo en cuestión. (Prud'hommeaux & Seaborne, 2008)

A continuación se redacta un ejemplo de una consulta SPARQL, tomado de (Prud'hommeaux & Seaborne, 2008).

Datos:

<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .

Consulta:

SELECT ?title

WHERE

{

<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title.

}

Resultado de la consulta:

Tabla 9. Resultado consulta SPARQL

|  |
| --- |
| **title** |
| "SPARQL Tutorial" |

Fuente: (Prud'hommeaux & Seaborne, 2008).

* + 1. Acerca de DBpedia

La DBpedia[[15]](#footnote-15) nos da la siguiente definición sobre si misma: “Es un esfuerzo de la comunidad crowd-sourced[[16]](#footnote-16) para extraer información estructurada de Wikipedia[[17]](#footnote-17) y hacer esta información disponible en la web. DBpedia permite que hacer consultas sofisticadas contra Wikipedia.”

El conocimiento extraído de Wikipedia es publicado cumpliendo los estándares de la Web Semántica y las mejores prácticas de Linkend Data.

* + - 1. Framework extracción

Los artículos de Wikipedia consisten sobre todo en texto libre, pero también comprenden diversos tipos de información estructurada en forma de wiki markup[[18]](#footnote-18). Dicha información incluye plantillas infobox, información de categorización, imágenes geo-coordenadas, enlaces a páginas web externas, páginas de desambiguación, redirecciones entre páginas y vínculos a través de las diferentes ediciones lingüísticas de Wikipedia. El marco de la extracción DBpedia extrae esta información estructurada de Wikipedia y lo convierte en una rica base de conocimientos (Lehmann, y otros, 2012)

En la figura 2 se observa el marco de trabajo necesario para lograr que todo el proceso partiendo de la extracción de información desde Wikipedia hasta poder disponer de ella como datos enlazados.

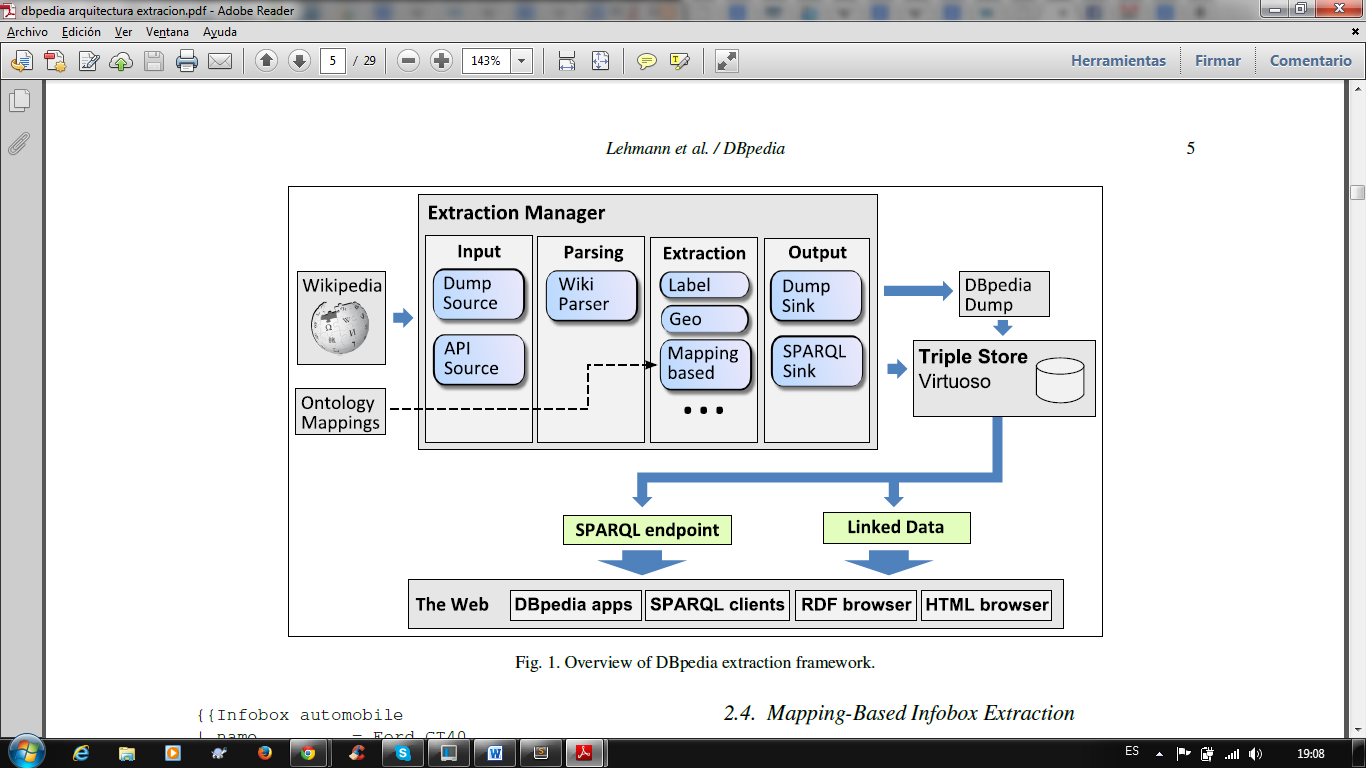


Figura 5: Representación gráfica RDF

Fuente: (Lehmann, y otros, 2012)

* + - 1. DBpedia Dataset

DBpedia se trata de una base de conocimiento (en inglés knowledge base) que se encuentra distribuida 119 idiomas que en total describen 12.6 millones de cosas únicas, 24.6 millones de enlaces a imágenes 27.6 millones enlaces a fuentes externas 45 millo es de enlaces a fuentes externas de datos RDF y 67 millones de enlaces a categorías Wikipedia 42.1 millones a categorías YAGO[[19]](#footnote-19).

Lo cual la establece como una fuente muy basta de información sobre cualquier ámbito de conocimiento, esto gracias a al continuo crecimiento de la Wikipedia, su fuente de información. Pero no esto no quiere decir que única base de conocimiento disponible en la web, podemos encontrar otras como YAGO.

* + - 1. Acceso a DBpedia Dataset

El dataset de Dbpedia se almacena y publica mediante OpenLink Virtuoso. La infraestructura de Virtuoso permite el acceso a los datos RDF de DBpedia a través de SPARQL endpoint, junto al soporte HTTP para cualquier GET estándar de cliente Web para HTML o representación RDF de un recurso Dbpedia. (Bizer, Dbpedia, 2009).

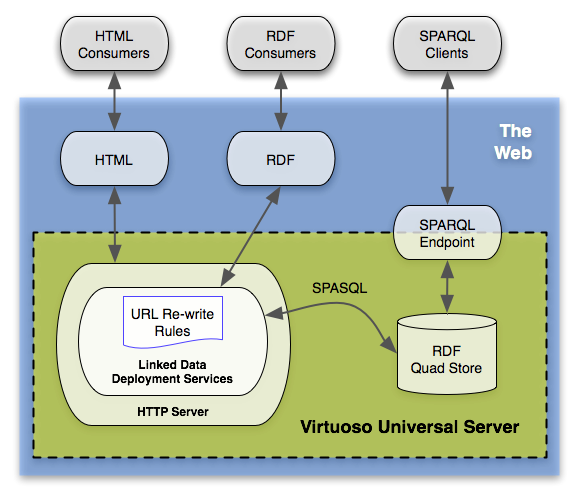


Figura 6. Arquitectura de provisión de Datos de Dbpedai.

Fuente: (Bizer, Dbpedia, 2009)

Se expones dos formas para acceder a la a la dataset de dbpedia:

1. Querying DBpedia: se puede accederé a través del SPARQL endpoint público http://dbpedia.org/sparql proporcionado por OpenLink Virtuoso. Por este método podemos acceder enviando query Sqparl para hacer consultar sobre dataset.
2. Linked Data: se refiere a la aplicación de los principios de datos enlazados revisados en 1.1.2. para nombrar y referenciar los recursos dentro de dataset de dbpedia como por ejemplo: http://dbpedia.org/resource/The\_Lord\_of\_the\_Rings

Además de esta opciones podemos descargar el dataset de dbpedia en diferentes idiomas teniendo en cuenta de que el número de recursos puede cambiar de idioma a idioma puesto que no se trata de una traducción sino de una recopilación de información de Wikipedia la cual se encuentra más extendida en ingles que otros lenguas,

* 1. Procesamiento de Lenguaje Natural (PLN)
     1. Introducción

El procesamiento de lenguaje natural dentro de la ingeniería lingüística comprende la rama se preocupara por entender el lenguaje humano, una tarea que para las personas e inclusive animales es tan natural y cotidiana se vuelve un reto al tratar de interpretarlo mediante procesos computacionales a fin de comprenderlo y poder replicarlo.

La dificultada de la construcción de una aplicación de la ingeniería lingüística variara de acuerdo a objetivo que se persiga, esto explicado por (García, 2005) en donde ejemplifica: “un sistema de generación de cartas personalizado no precisa ningún tratamiento de comprensión, o un sistema de identificación de la lengua (o un detector de errores ortográficos) no necesitan generar lenguaje humano. La mayoría de las aplicaciones incluyen, sin embargo, alguna forma más o menos precisa de comprensión. Así, un sistema de consulta en lenguaje humano a una base de datos precisa un nivel muy alto de comprensión de las expresiones del interlocutor humano para que la respuesta del sistema sea de utilidad. En cambio, en un sistema de traducción o de resumen automáticos se pueden lograr niveles de corrección muy notables con niveles de comprensión bajos. Es decir, no es preciso comprender totalmente una oración para ser capaz de traducirla correctamente.”

* + 1. Part of Speech Tagger

También denominado POS-tagging Nos permite distinguir la función de una palabra en un determinada contexto mediante la asignación de una etiqueta predefinida. (Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P., 1992) Nos dicen que: “Una part of speech tagger es un sistema que usa el contexto para asignar parte de un discurso a una palabra”.

Es te etiquetado de palabra ya permite un primera desambiguación en cuanto a la función de la palabra en un sentencia o contexto. Asi podemos por ejemplo[[20]](#footnote-20) ver que la palabra “*dado”* que si bien es nombre en singular, también puede ser una foram del verbo dar.

Pero antes de poder etiquetar una palabra por su función es necesaria una tokenización del texto que va a analizar, que consiste en separarlo en palabras individuales reconociendo un token para palabra o carácter extraído.

* + 1. Chunking

Text Chunking consiste en dividir un texto en frases de tal manera que palabras sintácticamente relacionadas sean miembros de la misma clase. Estas frases no se superponen es decir que una sola palabra puede ser miembro de un chunk. (Tjong Kim Sang, E. F., & Buchholz, S., 2000)

Este proceso es básico al momento de detectar entidades dentro de un texto, este proceso lo podemos observar en la figura 1 en donde la sentencia, *We saw the yellow dog*, está separada en cuadros en los más pequeños observamos etiquetas de POS Tag y las más grades al nivel de chunking.Una vez la frase ha pasado por el proceso de chunking podemos rescatar dos entidades dentro de la sentencia como *We* y *the yellow dog.*

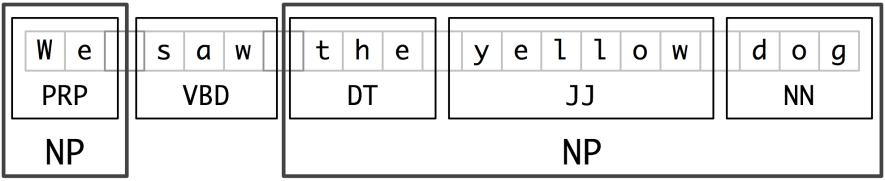


Figura 7: Ejemplo POS Tag y Chunking

Fuente: http://www.nltk.org/book/ch07.html

* + 1. Desambiguación

Este es un fenómeno muy común conocido polisemia que se refiere a cuando una palabra tiene varios significados la desambiguación busca descifrar que significado es el que está activado en determinado contexto se denomina Word Sense Disambiguation (WSD), este problema propio del procesamiento de lenguaje natural (PLN). El descifrar estos distintos significados para los seres humanos es muy común, lo resolvemos de forma cotidiana y pasa casi desapercibida.

* + - 1. Métodos basados en el conociendo.

El método Lesk (Lesk 1986) es el método basado en el diccionario. Se basa en la hipótesis de que las palabras usadas juntas en el texto están relacionadas entre sí y que la relación se puede observar en las definiciones de las palabras y sus sentidos. Dos (o más) palabras son desambiguadas encontrando el par de sentidos diccionario con la palabra mayor superposición en sus definiciones del diccionario.

Diccionarios externos:

*WordNet:*

Es una gran base de datos léxica de inglés. Sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos (synsets), cada una expresando un concepto distinto. Synsets están vinculados entre sí por medio de las relaciones conceptuales semántico y léxico.

*Corpus:*

Corpus es una gran colección de textos. Se trata de un cuerpo de material escrito o hablado sobre la que se basa un análisis lingüístico. Es un conjunto muy amplio de ejemplos reales del uso de la lengua.

*WordNet Domains:*

WordNet Domains es una extensión de WordNet 1.6, donde cada synset tiene asociado uno o varios dominios (categorías semánticas). Estos dominios, están clasificados en una jerarquía con distintos niveles de especialización, cuanto más profundo es el nivel sobre el que nos movemos, mayor es el grado de especialización.

* + - 1. Métodos supervisados

Métodos supervisados ​​se basan en la suposición de que el contexto puede proporcionar evidencia suficiente por sí sola para eliminar la ambigüedad de las palabras. Probablemente cada algoritmo de aprendizaje automático va se ha aplicado a WSD, incluyendo técnicas asociadas tales como la selección de características, la optimización de parámetros, y el aprendizaje conjunto.

* + - 1. Métodos semi-supervisados

El enfoque de bootstrapping comienza a partir de una pequeña cantidad de datos de semillas para cada palabra: cualquiera de ejemplos de entrenamiento manualmente etiquetados - o un pequeño número de reglas de decisión de éxito seguro. Las semillas se utilizan para entrenar un clasificador inicial, utilizando cualquier método supervisado. Este clasificador se utiliza en la parte sin etiqueta del corpus para extraer un conjunto de entrenamiento más grande, en el que sólo se incluyen las clasificaciones más seguras. El proceso se repite, cada nuevo clasificador siendo entrenado en un corpus de entrenamiento sucesivamente mayores, hasta que se consume todo el corpus, o hasta que se alcanza un número máximo dado de iteraciones.

* + - 1. Métodos sin supervisión

Aprendizaje no supervisado es el mayor desafío para los investigadores WSD. El supuesto subyacente es que los sentidos similares ocurren en contextos similares, y por lo tanto los sentidos puede ser inducido a partir del texto agrupando las ocurrencias de palabras usando alguna medida de similitud de contexto.

* 1. Servicios Web
     1. Introducción

La W3C[[21]](#footnote-21) (World Wide Web Consortium) encarda de estandarización de las tecnologías en la web aborda este tema de la siguiente forma: “Los servicios web proporcionan un medio estándar de interoperabilidad entre las distintas aplicaciones de software, que se ejecuta en una variedad de plataformas y/o marcos de trabajo. Los servicios Web se caracterizan por su gran interoperabilidad y extensibilidad. Se pueden combinar en una forma de acoplamiento flexible con el fin de lograr operaciones complejas. Programas que prestan servicios simples pueden interactuar entre sí con el fin de ofrecer servicios de valor añadido sofisticados.” Los servicios web permiten la colaboración entre aplicaciones independientemente de la plataforma en las que están desarrolladas, utiliza protocolos y normas estandarizadas en la web, además esto permite la reutilización de código, además de disminuir el coste de integración.

* + 1. Tipos de Web Services

Encontramos dos tipos de servicios web de acuerdo con la forma en que se puede implementar abarcado diferentes tecnologías: *RESTful Web Services* y *“Big”[[22]](#footnote-22) Web Services* (o también, The “Big” Web services technology stack, debido a la diversas tecnologías en las que se implementar como: SOAP, WSDL, WS-Addressing, WS-ReliableMessaging, WSSecurity, etc), estos dos tipos son expuesto y comparados en (Pautasso, Zimmermann, & Leymann, 2008)

* + - 1. SOAP AND THE WS-\* STACK

Proporcionar interoperabilidad sin fisuras entre los heterogéneos pilas de tecnología de middleware y el fomento de la articulación flexible de servicio al consumidor (solicitante, cliente) y proveedor de servicios son los principales objetivos de diseño de arquitectura orientada a servicios (SOA) conceptos y tecnologías de servicios Web. (Pautasso, Zimmermann, & Leymann, 2008)

En el plano conceptual, un servicio es un componente de software que se proporciona a través de un endpoint[[23]](#footnote-23) accesible en la red. Consumidores de servicios y proveedores usan mensajes para intercambiar solicitudes e información de respuesta en forma de *documentos self-containing[[24]](#footnote-24)* que hacen muy pocas suposiciones sobre las capacidades tecnológicas del receptor. En particular, no hay noción de una referencia de objeto remoto que requeriría un corredor de objeto para gestionar un espacio distribuido dirección de memoria. En el nivel de la tecnología, SOAP es un lenguaje XML que define una arquitectura de mensajes y formatos de mensaje, por lo tanto, proporcionar un protocolo de procesamiento rudimentario. El documento SOAP define un elemento XML de nivel superior llamada sobre, que contiene un encabezado y un cuerpo. El encabezado SOAP es un contenedor de información de infraestructura extensible de capa de mensajes que se puede utilizar para fines de enrutamiento (por ejemplo, hacer frente) y Calidad de Servicio (QoS) de configuración (por ejemplo, las transacciones, la seguridad, la fiabilidad). El cuerpo contiene la carga útil del mensaje. Esquema XML se usa para describir la estructura del mensaje SOAP, por lo que los motores de jabón en los dos puntos finales pueden Marshall y Resolver referencia el contenido del mensaje y la ruta a la aplicación apropiada. (Pautasso, Zimmermann, & Leymann, 2008)

* + - 1. REST

Transferencia de estado representacional (REST) ​​se introdujo originalmente como un estilo de arquitectura para la construcción de sistemas hipermedia distribuidos a gran escala. Este estilo arquitectónico es una entidad más abstracta, cuyos principios se han utilizado para explicar la excelente escalabilidad del protocolo HTTP 1.0 y también han limitado el diseño de su siguiente versión, HTTP 1.1. Por lo tanto, el término REST muy a menudo se utiliza junto con HTTP. (Pautasso, Zimmermann, & Leymann, 2008)

REST no es un protocolo, un formato de archivo, o un marco de desarrollo. Es un conjunto de restricciones de diseño, la hipermedia como el motor de estado de la aplicación.

El estilo arquitectónico REST se basa en cuatro principios:

*Identificación de recursos a través de URI*. Un servicio web RESTful expone un conjunto de recursos que identifican los objetivos de la interacción con sus clientes. Los recursos son identificados por URI, que proporcionan un espacio de direccionamiento global de los recursos y de descubrimiento de servicios.

*Interfaz uniforme*. Los recursos son manipulados utilizando un conjunto fijo de cuatro crear, leer, actualizar, eliminar operaciones: PUT, GET, POST y DELETE. PUT crea un nuevo recurso, que puede ser luego borrar con DELETE. GET recupera el estado actual de un recurso en alguna representación. POSTE transfiere un nuevo estado sobre un recurso.

*Mensajes de auto-descriptivo*. Recursos están desconectados de su representación para que su contenido se puede acceder en una variedad de formatos (por ejemplo, HTML, XML, texto plano, PDF, JPEG, etc.) Metadatos sobre el recurso está disponible y se utiliza, por ejemplo, para controlar el almacenamiento en caché, detectar errores de transmisión, negociar el formato de representación adecuada, y llevar a cabo la autenticación o controlar el acceso. Interacciones con estado a través de hipervínculos. Cada interacción con un recurso no tiene estado, es decir, los mensajes de solicitud son autónomos.

Interacciones con estado se basan en el concepto de transferencia de estado explícito. Existen varias técnicas para el intercambio de estado, por ejemplo, la reescritura de URI, cookies, y los campos de formulario ocultos. Estado puede ser embebido en los mensajes de respuesta para señalar válidos estados futuros de la interacción.

* + - 1. Hipermedia

(Ruby, 2007)Es una estrategia que nos permite establecer una conexión entre los recursos y describe sus capacidades, La estrategia hipermedia tiene siempre el mismo objetivo. Hipermedia es una manera para que el servidor para decirle al cliente qué HTTP request el cliente podría querer hacer en el futuro. Es un menú, proporcionado por el servidor, desde el que el cliente es libre de elegir. El servidor sabe lo que podría pasar, pero el cliente decide lo que realmente sucede.

* + - 1. Recursos y representaciones

(Leonard Richardson and Mike Amundsen, 2013)Rest denomina recursos a los datos estructurados que son objetos de las interacciones entre métodos de HTTP, y se dice que cualquier cosa que pueda ser almacenado de un computador puede ser un recurso, como documento electrónico, una fila de una base de datos o el resultado de un algoritmo

No solo las cosas almacenadas en un computador pueden ser llamados recursos también pueden ser recursos artículos tangibles como frutas por ejemplos, y es posible representarlo como recursos a través de la web como por ejemplo como un artículo en de venta o una imagen binaria depende de la aplicación así que por eso decimos sobre las representaciones que puede ser cualquier documento legible que contenga información acerca de un recurso.

* + - 1. URI y relación con los recursos

(Leonard Richardson and Mike Amundsen, 2013)Este tecnología de propia de la web que ya hemos revisa en capítulos anterior y que ahora recordamos teniendo en cuenta de que Rest trabaja sobre recursos o representaciones de los mismos u que la forma estandarizada para la identificación los recursos son las URIs, así como la relación existencial que tiene una sobre la otra.

* 1. Trabajos relacionados

# CAPITULO 2: PROBLEMÁTICA

* 1. Estado actual

La documentación dentro del desarrollo de trabajos educativos universitarios es indispensable para la difusión de los avances realizados en las distintas ramas y disciplinas de las ciencias, estas publicaciones se encuentran en texto plano el cual está diseñado para la fácil compresión por parte del usuario humano que acceda a estos, y contienen datos relevantes dentro de sus líneas los cuales se pierden puesto que no son explotados. Y que a pesar que estos documentos de encontrasen almacenados y publicados de tal forma que sean alcanzados por motores de búsquedas, no se puede acceder por este métodos a los datos relevantes que representa recursos disponibles en la Web a los cuales se requiere enlazar.

Se busca que las fuentes de datos que contengan estos recursos “objetivos”, sean estructuradas y publicadas de acuerdo con los principios de Datos Enlazados y permita acceder a Linking Open Data Cloud.

En la estructura de estos documentos existe resumen inicial en el que se explica el tema que se abarca en la publicación, etiquetado en inglés como “abstract” que en español significa “resumen” que es donde se centraran los esfuerzos para descubrir datos, esto a pesar de la existencia otros campos como “keywords” (en español, palabras claves), que exponen los temas que expone los abordan, pero que no son tan descriptivos.

* 1. Justificación

Teniendo en cuanta la tendencia actual de web, la web semántica que se basa en los principios de datos enlazados, los datos toma un factor importante, por lo cual que estos se encuentren “ocultos” dentro del texto no hace posible su enlace e impide la apertura hacia otras fuentes de información.

Esto hace necesario medios que permitan extraer y relacionar estos datos dentro de las publicaciones, de acuerdo a los principio de la web semántica que se encuentra en construcción y que iniciativas como esta ayudan a su expansión.

La información escrita es de fácil compresión para las seres humanos, se puede entender palabras por palabra su significado, aun cuando este puede varias de acurdo al contexto en que se encuentre y a la vez modificando en significado de otras palabras.

Dentro de un texto existen palabras que son más representativas que otras al momento de dar sentido a toda una sentencia o frase, esto puede ocurrir debido a que una palabra o varias palabras, más allá de tener un sentido pueden ser representaciones que entidades existentes en mundo real, como: personas, lugares, eventos, organizaciones etc. o representen entidades abstractas como la Web y diferentes tecnologías existentes, en sí, un texto plano como tal pude estar relacionado con diferentes entidades del mundo real por pedio de las palabras.

Pero el sentido de una frase descansa todas las palabras siendo unas más representativas que otras como ya habíamos visto, aunque no necesariamente estas tengas representaciones, sino los diferentes temas que abarca el texto.

* 1. Objetivo General.

Implementar servicios web para descubrimiento, enlace y enriquecimiento de Datos Enlazados, aplicado a publicaciones universitarias que permitan el enlace con nuevos datos vinculados a LOD Cloud.

* 1. Objetivos Específicos
* Crear Base de Conocimiento que sirva como fuente de información para el descubrimiento de datos.
* Desarrollar Servicios Web para Desambiguación, Enlace, Descubrimiento y Enriquecimiento Datos LOD-Cloud
* Implementar Frontal Piloto para Integración de Servicios Web.

CAPITULO 3: Solución

1. 1. Propuesta

Los datos que se encuentran dentro del texto tanto de las publicaciones como de fuentes en general, se encurtan relacionados con otros temas y fuentes de datos, a los que por medio este proyecto se tratara de acceder, extraer, relacionar y enlazar con fuentes de información abierta como lo es DBpedia (que se basa en los principios de los Datos Enlazados), esto permitirá el enriquecimiento del contenido. A todos estos datos a descubrir los denominaremos

Esta propuesta utiliza Servicios Web Rest lo que permite una independencia de la fuente de origen de texto a ser analizado, con la lógica de la aplicación propuesta y esta a su vez devuelve un resultado en que en este caso será formato Json[[25]](#footnote-25) (JavaScript Object Notation - Notación de Objetos de JavaScript) que es ampliamente conocido y utilizado para el intercambio de datos, hacia el cliente que consume el servicio.

La lógica que se propone es explicada en la Figura XXX, en donde las interacciones inician con el ingreso del texto a ser analizado, en el cual se aplican las diferentes tecnologías revisadas en capítulos 1 de este documento, para obtener como resultado entidades y keywords estructurados en formato JSON, que estarán desambiguadas y enlazadas, de existir un recurso al cual corresponda dentro del DataSet de DBpedia, es decir que no todos las entidades que encontremos dentro del texto de una publicación es referenciado en la DBpedia.

Puesto que nos concentraremos en los *abstracts* de las publicaciones y que esto se redactados en idioma ingles a pesar de que se trate de una publicación en español, se limitara el desarrollo de la soluciona este idioma.

Esta propuesta se surge como solución para el descubrimiento de datos en el texto de los *abstracts* de las publicaciones universitarias, pero debido a la gran cantidad de recursos de diversos temas que se encuentran actualmente disponibles en DBpedia, se puede aplicar a cualquier texto (en idioma ingles) para esto se implementa un interfaz gráfica web donde el usuario puede insertar su texto y ver el resultado.

* 1. Metodología

Para el desarrollo del proyecto de software se propone una metodología de desarrollo en prototipos, con el fin que en el desarrollo del producto de software adicionar funcionalidades al sistema de forma incremental, los cuales se dispondrán en forma de Servicios Web de acuerdo con los objetivos del proyecto.

Por lo cual nos basamos en el “proceso” de desarrollo ICONIX, está entre la complejidad de RUP (Rational Unified Processes) y la simplicidad y pragmatismo de XP (Extreme programming), sin eliminar las tareas de análisis y de diseño que XP no completa. (San Martin Oliva)

* + 1. Fases de desarrollo

Se establece las siguientes fases de desarrollo de acuerdo con metodología ICONIX, con las respectivas tareas y resultados esperados de las iteraciones de estas fases.

|  |  |  |
| --- | --- | --- |
| Fase | Tarea | Resultado |
| Análisis de requerimientos | * Identificar objetos del mundo real que interviene en el proceso * Identificar actores involucrados * Identificar casos de uso del sistema en interacciones con actores identificados | * Requerimientos * Modelo de dominio * Modelo de caso de uso |
| Diseño preliminar | * Describir los casos de uso como un flujo de acciones * Verificar el diseño | * Especificación de casos de uso |
| Diseño | * Especificar comportamiento a través de diagrama de secuencias * Verificar el diseño | * Diagrama de secuencias |
| Implementación | * Escribir código * Realizar pruebas | * Código * Pruebas |

* 1. Desarrollo
     1. Análisis de requerimientos

A partir de las primeras reuniones se determina los requerimientos funcionales con los debe contar la propuesta de software a desarrollar y se inicia con la formalización de estos requerimientos y luego con su necesario análisis.

* + - 1. Requerimientos

Los requerimientos descritos a continuación definen el comportamiento del sistema para obtener los resultados esperados.[[26]](#footnote-26) Estos requerimientos forman la base del desarrollo y se complementa con requerimientos no funcionales que los cuales serán implementados con la finalidad de agregar calidad al producto frente al usuario que lo va a utilizar.

|  |  |  |
| --- | --- | --- |
| Código | Requerimiento | Descripción |
| REQ001 | Extraer entidades y palabra relevantes | Descubrir datos relevantes en el texto, a quien se describe y las palabra relevantes que lo acompañan |
| REQ002 | Enlazar entidades y palabra relevantes con LOD Cloud | Se enlazara los términos encontrados en caso de que sea posible con la LOD Cloud |
| REQ003 | Desambiguar entidades y palabra relevantes | Se determinara el sentido con que las palabras estas siendo usadas en caso de que estas sean ambiguas. |
| REQ004 | Levantar servicios REST separados para los procesos relevantes. | Para que los procesos relevantes dentro del sistema pueda ser consumidos de forma individual y así reutilizados se levantaran servicios separados |
| REQ005 | Frontal UI Web | Construir una interfaz web que permita visualizar el comportamiento del sistema, es decir, la integración de los servicios y su funcionamiento individual |

Los requerimientos redactados establecen las tecnologías necesarias para el desarrollo del producto de software así como las áreas de conocimiento con las que esta ligados siendo el procesamiento de lenguaje natural (PLN) uno de los puntos más fuertes a resolver junto el levantamiento de servicios web y la construcción de un cliente web.

* + - 1. Modelo de Dominio

Uno de los ámbitos más importantes a resolver para el desarrollo de este sistema es el tratamiento de texto que es enviado por un usuario a través de un cliente, que será la entrada y base del procesamiento para descubrir datos relevantes dentro de este.

Se ha procedido a separar en servicios web distintos los procesos de relevancia del producto de software, en vista de que este uno de los requerimientos (REQ004), los servicios web que serán levantados perteneces a fases importantes dentro de las técnicas de procesamiento de lenguaje natural (PLN) aplicadas al texto, así como funciones dadas por los requerimientos, lo cual se observa en la figura XXX.,



* + - 1. Modelo de caso de Uso

El comportamiento de los objetos del mundo real, los componentes del sistema a desarrollar y las interacciones que entre estos se puedes realizar son dados por los casos de uso aplicables al sistema dadas las funciones que incorporara basados en el análisis de los requisitos, estas interacciones son visualizados en figura XXX,



* + 1. Análisis y diseño preliminar
       1. Especificación de casos de uso

Una vez determinadas los casos de uso que resolverá el software se presenta a continuación es especificaciones de cada uno.

* + - * 1. Tokenización en Sentencias

|  |  |  |
| --- | --- | --- |
| **Número** | **ECS-01** | |
| **Nombre** | Tokenización en Sentencias | |
| **Actores** | Usuario, Cliente | |
| **Descripción** | Divide el texto de entrada en sentencias cortas separadas por un punto y parte, la salida es una lista de estas sentencias. | |
| **Precondición** | * Ingreso texto como parámetro de la aplicación * Texto ha sido validado y procesado * Texto segmentado en sentencias | |
| **Secuencia**  **Normal** | Paso | Acción |
| 1 | Entrada del texto validado, como parámetro para el servicio web. **SA1** |
| 2 | Verifica el número de sentencias que comenten al texto, que estén separadas por un punto seguido (.) |
| 3 | Divide cada una teniendo en cuenta la terminación con punto (.) estructura las sentencias dentro de una lista. **SA1** |
| 4 | Estructura la lista de elementos formato JSON. |
| 5 | Devuelve el JSON resultante. |
| **Poscondición** | * El texto dividido en sentencias. | |
| **Secuencia alternativo** | **SA1 el número de sentencias es 1**  Se estructura una lista de un solo elemento con la sentencia. | |
| **Prioridad** | Media | |
| **Requerimientos Especiales** | Idioma de texto ingles | |
| **Asunciones y**  **Dependencias** |  | |
| **Notas adicionales** |  | |

* + - * 1. Tokenización en palabras

|  |  |  |
| --- | --- | --- |
| **Número** | **ECS-02** | |
| **Nombre** | Tokenización en palabras | |
| **Actores** | Cliente, Servicio Web | |
| **Descripción** | Divide cada sentencia en palabras validas, tokens. | |
| **Precondición** | * Ingreso texto como parámetro de la aplicación * Texto ha sido validado y procesado | |
| **Secuencia Normal** | Paso | Acción |
| 1 | Entrada del texto validado, como parámetro para el servicio web. |
| 2 | Segmentación del texto en sentencias. **ECS-01** |
| 3 | Se recorre la lista de sentencias segmentadas. |
| 4 | Se divide palabra por palabra de la sentencia en una lista, se obtiene una lista de listas. |
| 5 | Se estructura en formato JSON |
| 6 | Retorna el JSON con las sentencias divididas en “tokens” |
| **Postcondición** | * Texto tokenizado por sentencias y estos a la vez tokenizados en palabras | |
| **Secuencia alternativo** |  | |
| **Prioridad** | Baja | |
| **Requerimientos Especiales** | Del funcionamiento del Servicio web de Tokenización en Sentencias | |
| **Asunciones y**  **Dependencias** |  | |
| **Notas adicionales** |  | |

* + - * 1. Etiquetado

|  |  |  |
| --- | --- | --- |
| **Número** | **ECS-03** | |
| **Nombre** | Etiquetado | |
| **Actores** | Cliente, Servicio Web | |
| **Descripción** | Este servicio permite la tokenización de cada palabra y etiquetación de las mismas de acuerdo a la función que cumplen en el contexto que se encuentra, para hacerlo se apoya en el servicio web de tokenización en sentencias | |
| **Precondición** | * Ingreso texto como parámetro de la aplicación * Texto ha sido validado y procesado | |
| **Secuencia Normal** | Paso | Acción |
| 1 | Entrada del texto validado, como parámetro para el servicio web. |
| 2 | Tokenización del texto en sentencias. **ECS-01** |
| 3 | Recorrido de la lista de sentencias |
| 4 | Etiquetado de las palabras que conforman cada sentencia |
| 5 | Estructura y retorna data en JSON |
|  |  |
| **Poscondición** | * Texto tokenizado a nivel de palabras y etiquetado. | |
| **Secuencia alternativo** |  | |
| **Prioridad** | Alta | |
| **Requerimientos Especiales** |  | |
| **Asunciones y**  **Dependencias** |  | |
| **Notas adicionales** | Depende del funcionamiento del servicio web de Etiquetado en Sentencias (ECS-01) | |

* + - * 1. Extracción de Entidades

|  |  |  |
| --- | --- | --- |
| **Número** | **ECS-04** | |
| **Nombre** | Extracción de Entidades | |
| **Actores** | Cliente, Servicio Web | |
| **Descripción** | Permite reconocer y extraer, las entidades y palabras relevantes o claves (keywords) que se encuentran dentro del texto, para lograr se apoya en el servicio web de Etiquetado (y en los que este a su vez , servicio web de tokenización de sentencias) | |
| **Precondición** | * Ingreso texto como parámetro de la aplicación * Texto ha sido validado y procesado | |
| **Secuencia Normal** | Paso | Acción |
| 1 | Entrada del texto validado, como parámetro para el servicio web. |
| 2 | Tokenización del texto en sentencias. **ECS-01** |
| 3 | Tokenización y Etiquetado de palabra **ECS-03** |
| 4 | Reconocimiento de estructuras de Entidades y Keywords |
| 5 | Extracción de Entidades y Keywords |
| 6 | Estructuración de retorno de resultado en formato JSON |
| **Poscondición** | * Entidades y palabra importantes que las acompañan reconocidos y extraídos | |
| **Secuencia alternativo** |  | |
| **Prioridad** | Alta | |
| **Requerimientos Especiales** |  | |
| **Asunciones y**  **Dependencias** |  | |
| **Notas adicionales** | Este servicio depende del funcionamiento del servicio web de Tokenización en Entidades (ECS-01) y Servicio web de Etiquetado (ECS-03) | |

* + - * 1. Desambiguación y Enlace

|  |  |  |
| --- | --- | --- |
| **Número** | **ECS-05** | |
| **Nombre** | Desambiguación y Enlace | |
| **Actores** | Cliente, Servicio Web | |
| **Descripción** | Enlaza las entidades y palabras relevantes (keywords) hacia LOD Cloud, más específicamente DBpedia, esto de existir un recurso al cual vincular, en caso de que una entidad o keyword tuviese más de uno posible recurso al cual enlazar, se realizara un proceso de desambiguación y luego de enlace. | |
| **Precondición** | * Ingreso texto como parámetro de la aplicación * Texto ha sido validado y procesado | |
| **Secuencia Normal** | Paso | Acción |
| 1 | Entrada del texto validado, como parámetro para el servicio web. |
| 2 | Tokenización del texto en sentencias. **ECS-01** |
| 3 | Tokenización y Etiquetado de palabra. **ECS-03** |
| 4 | Extracción de Entidades y keywords. **ECS-04** |
| 5 | Consulta de recursos a DBpedia. |
| 6 | Consulta de “Abstract” de recurso a DBpedia |
| 7 | Verificar si existen Entidades o keywords ambiguas |
| 8 | Desambiguar Entidades y keywords ambiguos. **SA1** |
| 9 | Estructurara resultado |
| 10 | Retornar resultado |
| **Poscondición** | * Entidades y palabra importantes que las acompañan reconocidos y extraídos | |
| **Secuencia alternativo** | **SA1 Entidades y keywords no ambiguos**  Se enlaza con los recursos únicos encontrados a las entidades y keywords del texto. | |
| **Prioridad** | Alta | |
| **Requerimientos Especiales** |  | |
| **Notas adicionales** | Este servicio depende de los servicios web de tokenización en sentencias (ECS-01), etiquetado (ECS-03), extracción de entidades (ECS-04). | |

* + 1. Diseño
       1. Arquitectura

Después de haber realizado el análisis y posterior investigación sobre los puntos más relevantes dentro del desarrollo, se propone la siguiente propuesta lógica para el funcionamiento de los componentes a desarrollar y en base a los requerimientos funcionales, casos uso a los que va a responder el software y tecnologías disponibles que se pueden implementar para la resolución de la problemática planteada.



Figura 8. Lógica propuesta para la Aplicación;

Fuente: (Propio)

La distribución de los compones implementados se resume en la gráfica XXX, haciendo una división de los componentes de la aplicación que interactúan para dar solución a la problemática. Las capas que componen la arquitectura de aplicación se detallan a continuación.



Figura 9. Arquitectura.

Fuente: (Propio)

* + - 1. Módulos
         1. Servidor

**Validar texto**

Que el texto sea legible para el sistema en funciones siguientes a fin de evitar errores. Comprobar que el texto contenga caracteres a fin de evitar trabajar sobre texto vacío. Es origen obligatorio por el cual los todos los módulos deben pasar para la comprobación de los requisitos del texto que ingresa como parámetro para ser procesado.

**Tokenización en sentencias**

Dividir el texto ingresado en sentencias u oraciones, generalmente separados por un punto (.) significando el final de esta. Este componente forma parte de los procesos relevantes dentro de procesamiento de lenguaje natural siendo el inicio de estos procesos.

**Tokenización en palabras**

Una vez divididos el texto en sentencias, se realiza en mismo proceso para las palabras que lo conforman así como los signos de computación, obteniendo un “token” por cada palabra o carácter reconocido. Parte del procesamiento de lenguaje natural.

**Etiquetado**

Este componente realiza etiquetado de las palabras, de acuerdo al contexto dentro de la sentencia en la que se encuentra, tomando una función específica como verbo, sustantivo, etc. la etiquetación de palabras (Part fo Speech, por su nombre en inglés) forma parte del procesamiento de lenguaje natural.

**Extracción de entidades**

Permite el reconociendo y extracción de entidades y palabra claves dentro del contexto de una oración, para lo cual realiza un reconocimiento de los etiquetas de las palabras y un análisis de su estructura para determinar que palabra es una entidad o una palabra relevante, este análisis puede extraer una palabra o un conjunto de estas formando una entidad o palabras claves. Se basa en componentes posteriores para poder realizar sus operaciones, es decir, necesita de un texto que haya sido tokenizado y etiquetado.

**Extracción recursos DBpedia**

Una vez que los entidades y palabra claves hayan sido extraídos de las sentencias. Se realizan consultas con estos términos hacia el servidor local de DBpedia, para encontrar los recursos que se denominen igual que estos términos, así como las descripciones rápidas de estos recursos.

**Desambiguación de recursos**

A través del algoritmo de Lesk se analiza el contexto del término y las descripciones de los recursos de DBpedia, para determinar que cual de estos concuerda mejor con el uso que se le está dando al término en la sentencia.

**Servicios web**

Los principales proceso descritos se levantara como servicios web, para que puedan ser consumidos y analizados de forma individual. Asi se tendrá servicio diferentes de:

* Tokenización en sentencias
* Tokenización en palabras
* Etiquetado
* Extracción de entidades
* Desambiguación y enlace

Las consultas de recursos a DBpedia y la desambiguación entre estos se los unido en un servicio, puesto que la extracción por sí sola no se considera como un resultado relevante dentro de los procesos del sistema.

* + - * 1. Servidor DataSet Dbpedia Local (SPARQL EndPoint)

El poder obtener accesos a los recursos de la DBpedia es vital para dar desarrollar la propuesta, y debido a que la conexión con la DBpedia es un obstáculo puesto que estas no siempre está disponible o su retorno de resultados son lentos, se decidió implementar un servidor local propio que contenga, los recursos y atributos de estos, necesarios para poder ser encontrados, desambiguados y referenciados.

DBpedia utiliza Virtutoso OpenLink para almacenar y servir los datos que contiene, por lo cual se decidió utilizar esta misma tecnología para el servidor local utilizando su versión Open Source, como almacén de datos y poder replicar el comportamiento de la misma en la web.

Archivos necesarios para implementar los recursos:

* Label de recursos:
  + labels\_en.nt.bz2
* Datos personales de los recursos tipos Persona:
  + persondata\_en.nt.bz2
* Resúmenes Corto de los recursos
  + short\_abstracts\_en.nt.bz2
* Links de Desambiguación de Wikipedia
  + disambiguations\_en.nt.bz2
* Redirecciones entre Recursos
  + redirects\_en.nt.bz2

El tamaño total de las importaciones es de 3.8 GB, de espacio en disco.

* + - * 1. Cliente

La vista es un Interfaz Web que permite al usuario la facilidad de la comunicación entre la aplicación y el usuario. Instruye al usuario sobre el uso de la herramienta, identificando con facilidad los parámetros necesarios y en especial la facilidad de la presentación de los resultados. Permite la integración de los compontes del sistema en un entorno amigable para el usuario.

Prototipo de interfaz

* + - 1. Diagrama de secuencia

Al observar el modelado de las interacciones entre los componentes del sistema se evidencia una clara dependencia e interoperabilidad entre los estos, iniciando siempre con el módulo validación del texto a ser procesado.







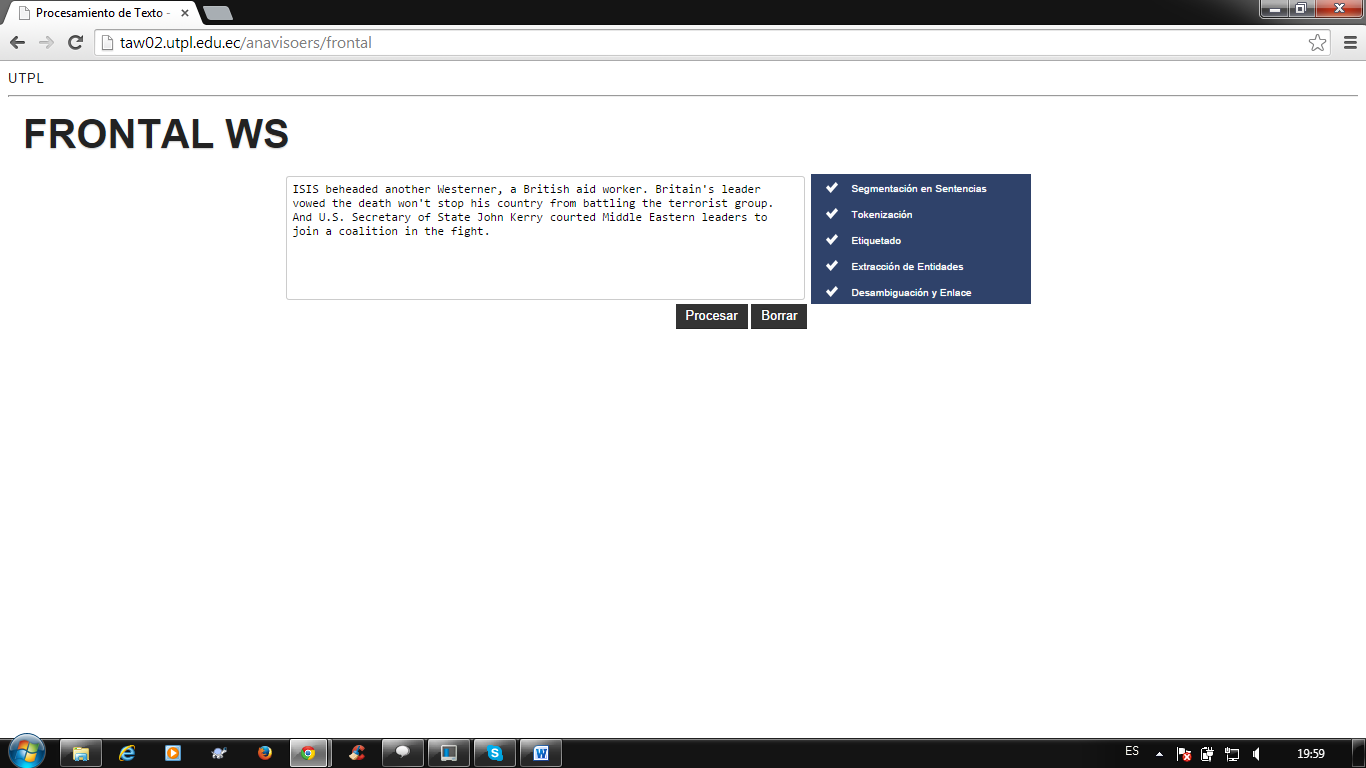


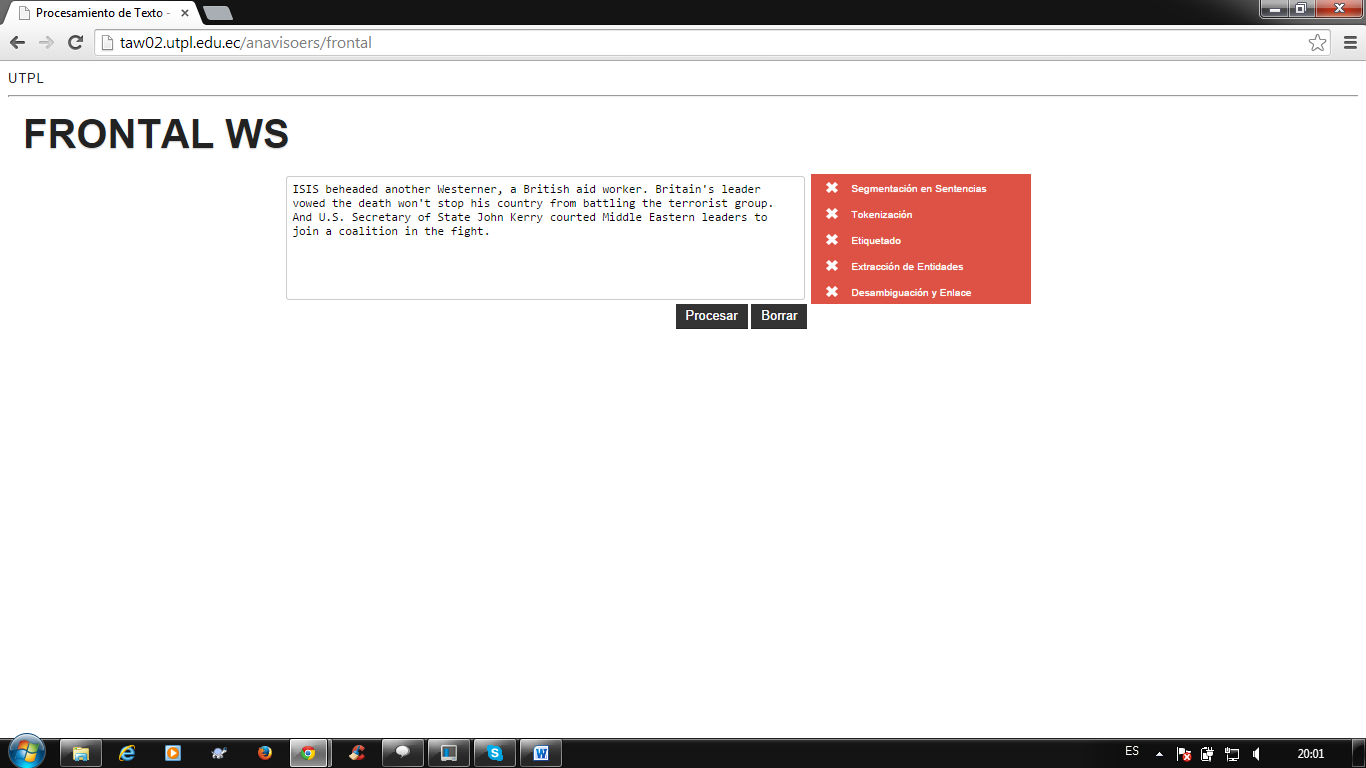


Figura 10. Diagrama de secuencias del módulo de Etiquetado

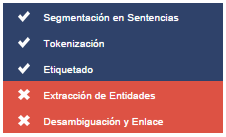
Fuente: (Propio)

* + 1. Implementación
       1. Servidor
       2. Servidor DataSet DBpedia Local
       3. Cliente



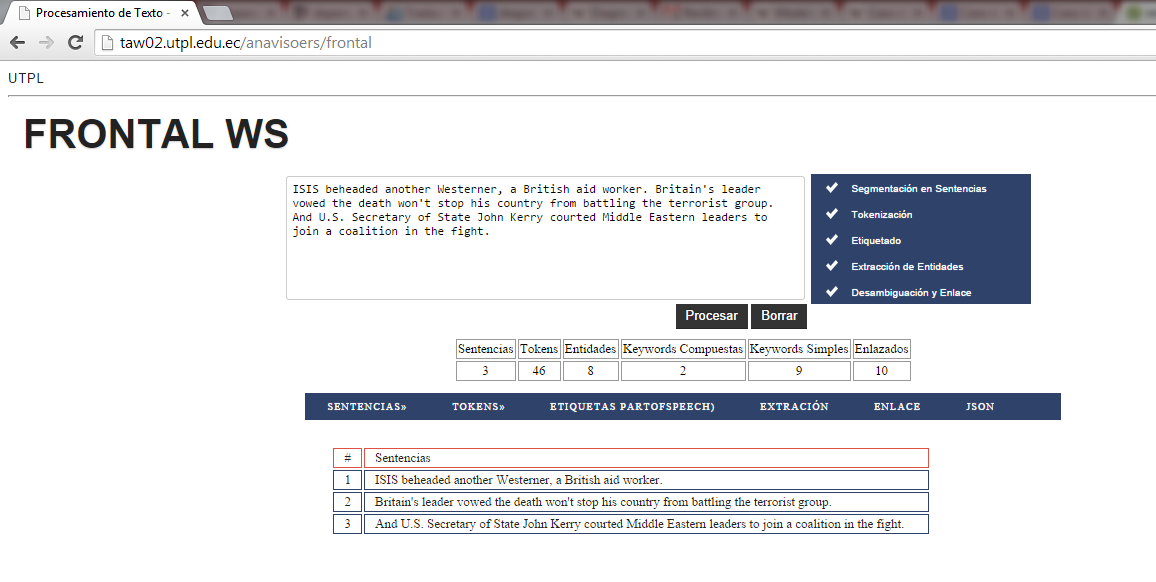






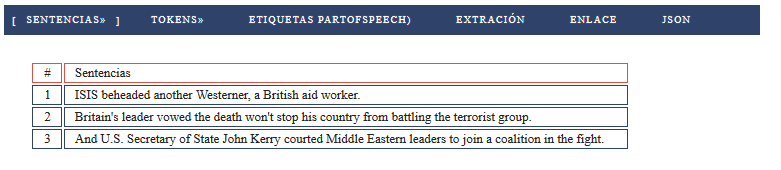


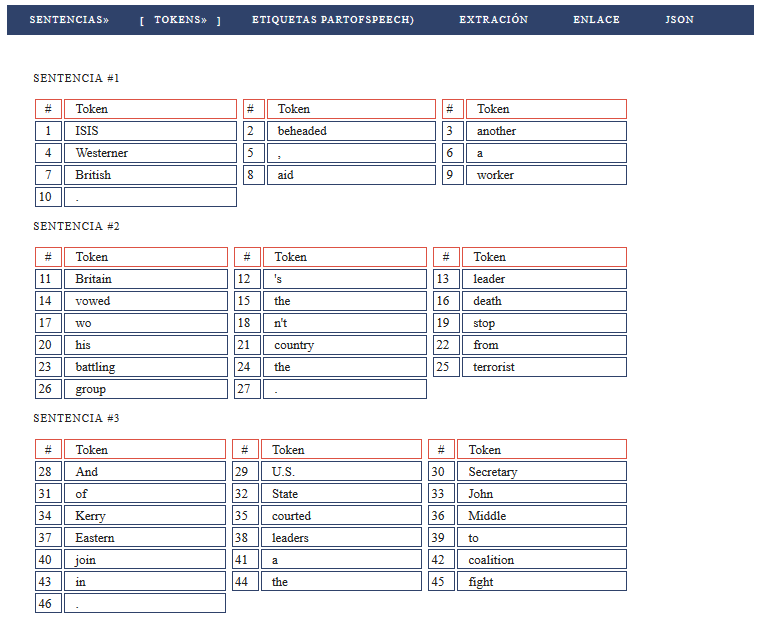


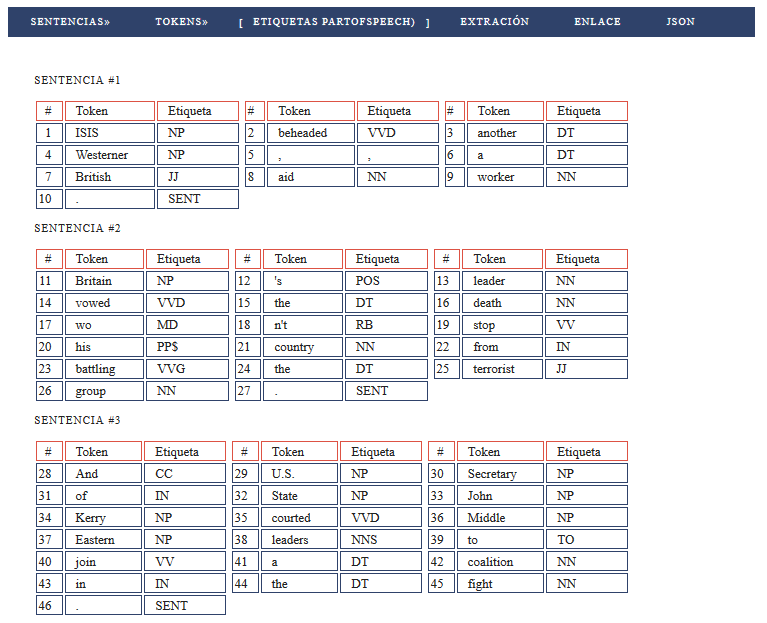


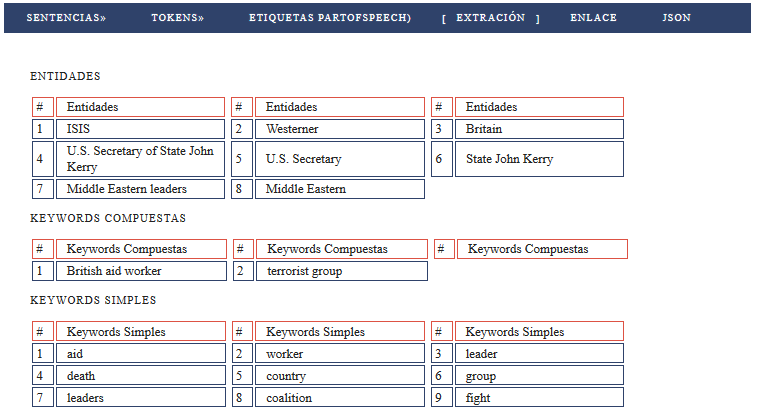


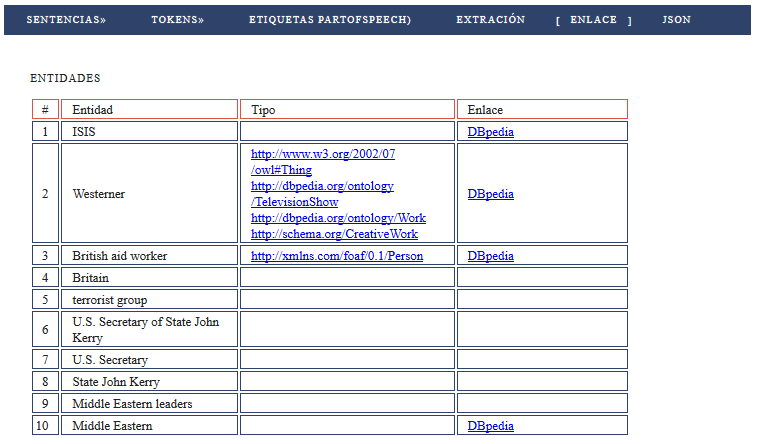












* + - 1. Resumen de prototipos

**Prototipo 1**

Descripción

En inicio de se trató de utilizar herramientas abiertas disponibles en la web para el proceso de enlace y desambiguación de los elementos trascedentes en un texto, así se consumió el servicio ofrecido por DBpedia, spotlight el cual produce el resultado esperado pero con problemas de conexión debido a que el servidor no se encontraba siempre disponible, para lo cual se construyó un cliente REST en python. A partir de este intento y conforme se fue avanzando en la investigación de soluciones para los requerimientos presentados, se decidió desarrollar un propuesta de software propia.

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidad especifica | Aprendizaje | Herramienta | Tiempo |
| Desambiguación y enlace | Consumir servicios libres existentes para procesos de  Tiempo excesivos de estará para espera respuesta | DBpedia Spotlight | 3 días |
| Construcción de cliente para servicios | Conexión con servidores REST | Python | 2 días |

**Prototipo 2**

Descripción

A través de Procesamiento de Lenguaje Natural (PLN), se trabaja con el texto de entrada para obtener las entidades de quien se habla en las oraciones, así como las palabras claves que los acompañan. Este procesamiento se lo realiza especializado en el idioma ingles utilizando la librería Natural Language Toolkit (NLTK) de Python.

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidad especifica | Aprendizaje | Herramienta | Tiempo |
| Tokenización en sentencias | Procesamiento de lenguaje natural o PLN | NLTK (Natural Language Toolkit) librería de Python. | 2 semanas |
| Tokenización en palabras | PLN | NLTK | 2 horas |
| Etiquetado (Part of speech) en idioma ingles | PLN | NLTK | 2 horas |
| Extracción de entidades y palabras claves idioma ingles | PLN | NLTK | 1 día |

**Prototipo 3**

Descripción

Todos los esfuerzos realizados se concentran en el idioma inglés, a través de NLTK para su procesamiento, para en otros idiomas como español por ejemplo no se puede realizar el mismo proceso que en el idioma ingles debido a lo diferente de su estructura. Para poder trabajar en otros idiomas se debe partir de un etiquetado (Part of Speech) propio del idioma, por lo cual se decidió trabajar con TreeTagger, que permite etiquetado en diferentes idiomas.

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidad especifica | Aprendizaje | Herramienta | Tiempo |
| Etiquetado (Part of speech) en idioma ingles | PLN | TreeTagger | 5 días |
| Extracción de entidades y palabras claves idioma ingles | PLN | NLTK | 2 día |

**Prototipo 4**

Descripción

Una vez que se ha logrado extraer los datos importantes de un texto, procederemos a enlazar esto con recursos disponibles en BDpedia.org, para esto se realizan consultas Sparql para obtener los recursos que coinciden con los datos extraídos. Una vez obtenidos los recurso de DBpedia, pueden existir más de una recurso que pueda coincidir para un dato del texto, este caso decimos que este término es ambiguo y es necesario romper esta ambigüedad para enlazarlo con un solo recurso de DBpedia, para lo cual utilizamos el algoritmo de Lesk introducido por  Michael E. Lesk en 1986.

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidad especifica | Aprendizaje | Herramienta | Tiempo |
| Enlazar entidades y palabras claves encontrados con recursos similares en DBpedia | Sparql | Python  Sparql | 4 días |
| Desambiguación entre los recursos encontrados | Algoritmo de desambiguación de Lesk | Python  Sparql | 5 días |

**Prototipo 5**

Descripción

Se levantan servicio Rest para ofrecer la propuesta y una interfaz para que pueda interactuar con usuarios.

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidad especifica | Aprendizaje | Herramienta | Tiempo |
| Servicio Web REST | Levantar servicios Web Rest | REST  Python | 2 semanas |
| Interfaz de usuario (Web) prototipo inicial | Construcción de cliente Web JavaScript | JavaScript  HTML | 3 días |

**Prototipo 6**

Descripción

Se levantan servicios diferenciados por los procesos realizado de esta forma el resultado de un servicio o es la entrada de toro servicio y así puedes ser consumido por otros servicios externos a esta propuesta.

Para poder visualizar el resultado de la interacción de los servicios Web de forma agradable para usuarios se mejora la interfaz gráfica en donde se diferencia el resultado de cada servicio.

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidad especifica | Aprendizaje | Herramienta | Tiempo |
| Servicios web diferenciados por Procesos | Levantar servicios Web Rest | REST  Python | 2 semanas |
| Interfaz de usuario final | Mejoramiento de interfaz Web | Css  JavaScript | 7 días |

CAPITULO 4

1. 1. Validación
      1. Comparación con servicios similares
   2. Pruebas

CAPITULO 5

1. 1. Conclusiones y Recomendaciones

# Bibliografía

Albahari, J., & Albahari, B. (2012). *C# 5.0 IN A NUTSHELL.* O’Reilly.

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G., & Machina., L. (25 de 02 de 2014). *RDF 1.1 Turtle.* Obtenido de W3C Recommendation: http://www.w3.org/TR/2014/REC-turtle-20140225/

Berners-Lee, T. (01 de 2005). *Uniform Resource Identifier (URI): Generic Syntax.* Recuperado el 24 de 06 de 2014, de http://tools.ietf.org/html/rfc3986

Berners-Lee, T. (23 de Julio de 2006). Linked Data - Design Issues.

Bizer, C. (09 de 11 de 2009). *Dbpedia*. Recuperado el 10 de 06 de 2014, de The DBpedia Data Provision Architecture: http://wiki.dbpedia.org/Architecture

Bizer, C., & Cyganiak, R. (25 de 02 de 2014). *RDF 1.1 TriG.* Obtenido de W3C Recommendation: http://www.w3.org/TR/2014/REC-trig-20140225/

Carothers, G., & Seaborne, A. (25 de 02 de 2014). *RDF 1.1 N-Triples.* Recuperado el 25 de 06 de 2014, de W3C Recomendations: http://www.w3.org/TR/2014/REC-n-triples-20140225/

Clark, K. G., Feigenbaum, L., & Torres, E. (01 de 15 de 2008). *SPARQL Protocol for RDF.* Recuperado el 24 de 06 de 2014, de W3C Recommendation 15 January 2008: http://www.w3.org/TR/rdf-sparql-protocol/

Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (Marzo de 1992). A Practical Part of Speech Tagger.

Cyganiak, R., Wood, D., & Lanthaler, M. (25 de 02 de 2014). *RDF 1.1 Concepts and Abstract Syntax.* Recuperado el 25 de 06 de 2014, de W3C Recommendation: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

Lapuente, M. J. (08 de 12 de 2013). *HIPERTEXTO: EL NUEVO CONCEPTO DE DOCUMENTO EN LA CULTURA DE LA .* Recuperado el 24 de 06 de 2014, de Tesis doctoral. Universidad Complutense de Madrid.: http://www.hipertexto.info/

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., y otros. (2012). *DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia*. Obtenido de http://semantic-web-journal.net/system/files/swj499.pdf

McBride, B. (10 de 02 de 2004). *W3C Recommendation.* Recuperado el 26 de 06 de 2014, de RDF Primer: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

Miller, E. (Mayo de 1998). *An Introduction to the Resource Description Framework*. Recuperado el 27 de 06 de 2014, de http://www.dlib.org/dlib/may98/miller/05miller.html

Miller, E. (1998). Wiley Online Library. *Bulletin of the American Society for Information Science and Technology*, 15-19.

Pautasso, C., Zimmermann, O., & Leymann, F. (Abril de 2008). RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision.

Prud'hommeaux, E., & Seaborne, A. (15 de 01 de 2008). *SPARQL Lenguaje de consulta para RDF.* Recuperado el 25 de 06 de 2014, de Recomendación del W3C de 15 de enero de 2008 : http://skos.um.es/TR/rdf-sparql-query/

Richardson, L., & Ruby, S. (2007). RESTful Web Services. En L. Richardson, & S. Ruby, *RESTful Web Services* (pág. 299). O'Reilly.

Ruby, L. R. (2007). *RESTful Web Services.*

Sandeep Chatterjee, j. W. (2004). *Developing Enterprise Web Services: An Architect's Guide.* Person Education Inc.

Schreiber, G., & Raimond, Y. (24 de 06 de 2014). *RDF 1.1 Primer.* Obtenido de W3C Working Group: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. (2007). *In The semantic web.*

Tjong Kim Sang, E. F., & Buchholz, S. (Septiembre de 2000). Introduction to the CoNLL-2000 shared task: Chunking.

W3C. (01 de 2005). *Uniform Resource Identifier (URI): Generic Syntax.* Recuperado el 22 de 02 de 2014, de January 2005

W3C. (2013). *W3C.* Obtenido de http://www.w3.org/standards/semanticweb/ontology

Wood, D. (25 de 02 de 2014). *What’s New in RDF 1.1.* Obtenido de W3C: http://www.w3.org/TR/2014/NOTE-rdf11-new-20140225/

Anexos

1. http://www.evolutionoftheweb.com [↑](#footnote-ref-1)
2. <http://www.w3.org/> [↑](#footnote-ref-2)
3. http://www.ietf.org/ [↑](#footnote-ref-3)
4. http://en.wikipedia.org/wiki/Request\_for\_Comments [↑](#footnote-ref-4)
5. http://www.ietf.org/rfc/rfc1738.txt [↑](#footnote-ref-5)
6. http://www.ietf.org/rfc/rfc3406.txt [↑](#footnote-ref-6)
7. [↑](#footnote-ref-7)
8. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/ [↑](#footnote-ref-8)
9. http://www.foaf-project.org/ [↑](#footnote-ref-9)
10. http://dublincore.org/documents/dces/ [↑](#footnote-ref-10)
11. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#bib-RDF11-SCHEMA [↑](#footnote-ref-11)
12. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#bib-RDF11-SCHEMA [↑](#footnote-ref-12)
13. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#dfn-rdf-compatible-xsd-types [↑](#footnote-ref-13)
14. http://www.w3.org/TR/sparql11-query/ [↑](#footnote-ref-14)
15. http://dbpedia.org/About [↑](#footnote-ref-15)
16. http://es.wikipedia.org/wiki/Crowdsourcing [↑](#footnote-ref-16)
17. <http://www.wikipedia.org/> [↑](#footnote-ref-17)
18. http://en.wikipedia.org/wiki/Help:Wiki\_markup [↑](#footnote-ref-18)
19. http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/ [↑](#footnote-ref-19)
20. http://es.wikipedia.org/wiki/Ambig%C3%BCedad [↑](#footnote-ref-20)
21. http://www.w3.org/ [↑](#footnote-ref-21)
22. Nombrado así en (Richardson & Ruby, 2007) [↑](#footnote-ref-22)
23. http://www.w3.org/TR/ws-gloss/#endpoint [↑](#footnote-ref-23)
24. http://www.thefreedictionary.com/self-contained [↑](#footnote-ref-24)
25. http://json.org/json-es.html [↑](#footnote-ref-25)
26. Ver en anexos documento de especificación de requerimientos [↑](#footnote-ref-26)