UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TITULACIÓN DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Servicios web para extracción de entidades desde contenido html. Piloto en sitios con recursos abiertos OCW

TRABAJO DE FIN DE TITULACIÓN

AUTOR: Zaruma Sozoranga, Jhonny Alonso

DIRECTOR: Piedra Pullaguari, Nelson Oswaldo, Ing.

LOJA – ECUADOR

2014

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

Ingeniero.

Nelson Oswaldo Piedra Pullaguari.

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de fin de titulación: Servicios web para extracción de entidades desde contenido html. Piloto en sitios con recursos abiertos OCW, realizado por Zaruma Sozoranga Jhonny Alonso, ha sido orientado y revisado durante su ejecución, por se aprueba la presentación del mismo.

Loja, noviembre de 2014

f) ……..

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo Zaruma Sozoranga Jhonny Alonso declaro ser autor (a) del presente trabajo de fin de titulación: Servicios web para extracción de entidades desde contenido html. Piloto en sitios con recursos abiertos OCW, de la Titulación de Ingeniero en Sistemas Informáticos y Computación, siendo Nelson Oswaldo Piedra Pullaguari director 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. 67 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 que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f. ……

Autor: Zaruma Sozoranga Jhonny Alonso

Cédula: 1104811482

DEDICATORIA

Esta tesis va dedicada a mis padres y a mi hermana, que siempre estuvieron brindándome su amor y paciencia, y su apoyo incondicional.

A mis amigos por su motivación, especialmente a Jholena, mi negra, que fue como mi hermana y aunque que ya no está físicamente con nosotros, yo sé que desde el cielo siempre me estará cuidando.

AGRADECIMIENTO

A mis padres que son mi ejemplo.
A mi hermana por estar siempre conmigo.
A Jholena que siempre creyó en mí, sé que desde el cielo seguirás apoyándome.
A mis amigos con quienes he compartido alegrías y tristezas.
A mi tutor de tesis Ing. Nelson Piedra y todo el laboratorio de Tecnologías Avanzadas en la Web y SBC por los conocimientos compartidos.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Diseño e implementación de una aplicación y tres servicios web para la extracción de entidades a partir de contenido HTML, alojados en un del Laboratorio De Tecnologías Avanzadas en la Web y SBC de la UTPL e implementados mediante Python.

Con este trabajo se busca facilitar la extracción de información clave dentro del contenido HTML y el enriquecimiento del mismo. Se desarrollaron tres servicios web para este fin: uno para descomponer el texto con la finalidad de etiquetar las palabras, un segundo para procesar las palabras etiquetadas y posteriormente extraer entidades y el contexto del que fueron tomadas, y un tercero para desambiguar y enlazar con la Dbpedia para enriquecer el contenido; los tres servicios han sido descritos con anotaciones semánticas, para que puedan permitir la interoperabilidad entre los servicios existentes.

Al implementarse el piloto de este proyecto en sitios con recursos abiertos OCW, esta investigación constituye un referente para futuros proyectos que se desarrollen a partir de la extracción de entidades y el enriquecimiento de contenido.

PALABRAS CLAVES: Extracción de entidades, Web Semántica, Anotación semántica, OCW, Dbpedia.

ABSTRACT

Design and implementation of an application and three Web services for extracting features from HTML content, housed in a SBC Advanced Technologies Laboratory Web and UTPL and implemented using Python.

This work is to facilitate the extraction of key information within the HTML content and enrich it. Three web services for this purpose were developed: one to break the text in order to label the words a second to process the tagged words and then extract entities and the context from which they were taken, and a third to disambiguate and join the dbpedia to enrich the content; the three services are described with semantic annotations, so they can enable interoperability between existing services.

By implementing the pilot project in open source OCW sites, this research provides a benchmark for future projects developed from entity extraction and enrichment of content.

KEYWORDS: Extraction of entities, Semantic Web, Semantic annotation, OCW, Dbpedia.

INTRODUCCIÓN

La web conforma un mundo de datos, información y conocimiento casi siempre nos encontramos con la dificultad de encontrar la información que realmente necesitamos. Existen muchos buscadores y algoritmos de búsqueda, pero falta mucho por recorrer para poder llegar a automatizar búsqueda y recuperación de información mediante búsquedas inteligentes.

Por lo tanto la asignación de metadatos y etiquetas es de gran importancia si se quiere tener una búsqueda inteligente en la web semántica. Con esto se encuentra la necesidad de una herramienta de extracción de meta información de calidad.

El presente trabajo se enfoca en el procesamiento de los recursos educativos abiertos OCW (Open Course Ware por sus siglas en inglés) que son una de las iniciativas educativas muy importante, ya que son libre acceso a una gran diversidad de recursos y materiales de cursos universitarios de forma gratuita e ilimitada.

En la sección de Marco Teórico se describe lo que es La web de datos, la web semántica como también que son los metadatos, las anotaciones semánticas, rdf, dbpedia, sparql; se define también lo que son los OCW, lo que es el procesamiento de lenguaje natural, también se describe las arquitecturas de servicios web más utilizadas. Y finalmente se describe los programas y herramientas utilizadas.

En la sección Problemática se describe en detalle el problema a resolverse que dio pie a la realización este proyecto.

En la sección Solución se detalla la aproximación de lo que se va a realizar para dar resolución al problema encontrado. Así como también se describe los componentes creados para la resolución. Y finalmente una descripción a detalle de los Servicios Web creados.

Al final se presenta una discusión, trabajos a futuro y las conclusiones del proyecto realizado.

OBJETIVOS

1.1. General

- Implementar Servicios Web descritos semánticamente que faciliten la extracción entidades a partir del contenido HTML para el enriquecimiento de dicho contenido.

1.2. Específicos

- Diseñar un vocabulario para describir semánticamente los Servicios Web que una vez implementada permitan que estos servicios se auto describan.
- Desarrollar Servicios Web para Tokenización, Extracción de entidades, Desambiguación y Enlace con LOD-Cloud.
- Desarrollar una aplicación cliente, para integración de los Servicio Web y que permita la visualización de resultados.

MARCO TEÓRICO

2.1. La web

La Web se define simplemente como el universo de información accesible desde la red global. Se trata de un espacio abstracto con el cual las personas pueden interactuar, actualmente está poblado por páginas interconectadas que contienen texto, imágenes, animaciones y videos. Su existencia marca el final de una era de incompatibilidades frustrantes y debilitantes entre sistemas informáticos. (Berners-Lee, 1996)

El objetivo de la web es ser un espacio de información compartida a través del cual las personas (y las máquinas) puedan comunicarse. (Berners-Lee, 1996)

2.1.1. Web de datos.

La Web de Datos o Linked Data permite pasar de una Web en la que los recursos son documentos HTML (en la que el usuario humano es el destinatario de la información publicada), a una Web de Datos Enlazados que están expresados en RDF.

En la actualidad la Web de Datos cuenta con información vinculada sobre organizaciones, autores, áreas de conocimiento, licencias, países, tipo de medios, etc. (Piedra, Tovar, López, Chicaiza, & Martinez, 2011)

Tim Berners-Lee presentó en TED 2009 (Technology Entertainment and Design) una conferencia, en la que redefinió los principios de Linked Open Data (Datos abiertos enlazados) presentándolos como tres reglas que se resumen en:

- Asignar a todas las cosas conceptuales nombres que comienzan con http
- Obtener información importante de retorno a partir de los nombres; y
- La información obtenida debe contener relaciones.

2.1.2. Web semántica.

El creador del concepto, Tim Berners-Lee, define la web semántica de la siguiente manera: “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” (Vallez, 2009)

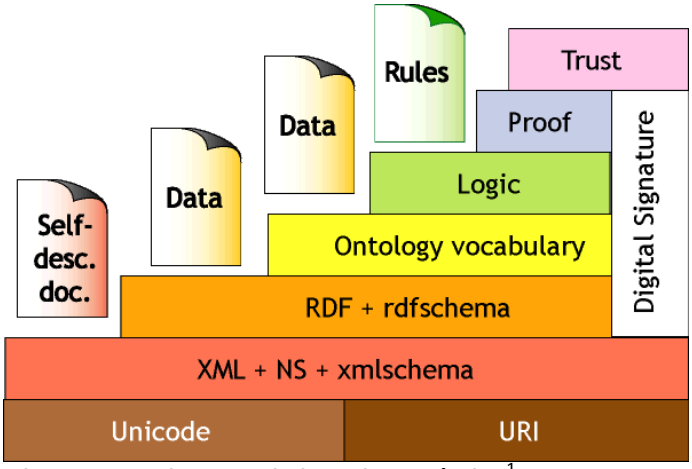

Figura 1 Arquitectura de la web semántica[1]

Unicode - URI: Unicode es un sistema de codificación que asigna un número único para identificar cada carácter sin importar la plataforma, programa ni idioma. Este es compatible con la mayoría de sistemas operativos y con todos los exploradores actuales, además es un requerimiento para estándares modernos como XML.

- **URI** proporciona un nombre para identificar de manera única a los distintos recursos de la Web.
- **XML + NS + xmlschema:** En esta capa se integran tres tecnologías que hacen posible la comunicación entre agentes. XML ofrece un formato común para intercambiar documentos de una forma estructurada, como árboles de etiquetas con atributos.
- **XML Schema** es uno de estos lenguajes para definir su estructura, donde se describen de antemano las estructuras y tipos de datos utilizados.
- **NS** proporciona un método para cualificar elementos y atributos de nombres usados en documentos XML asociándolos con espacios de nombre identificados por referencias URI's.
- **RDF + RDFSchema**: Es un lenguaje simple mediante el cual definimos sentencias en un formato con tres elementos: sujeto, predicado y objeto
- **RDF Schema** provee un vocabulario definido sobre RDF que permite el modelo de objetos con una semántica claramente definida. Esta capa no solo ofrece descripción de los datos, sino también cierta información semántica. Ambos corresponden a las anotaciones de la información llamados metadatos.
- **OWL**: Es uno de los lenguajes de ontologías más extendidos por la Web Semántica. Este estándar W3C fue diseñado para ser compatible con

[1] Tomado de http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

estándares web existentes. Ontology Web Language añade más vocabulario para describir propiedades, clases, relaciones entre clases, cardinalidad, igualdad, características de propiedades, clases enumeradas, etc.

- **Logic, Proof, Trust, Digital Signature:** Las capas Logic (Lógica) y Proof (Pruebas) son encargadas de aplicar reglas de inferencia con sus pruebas respectivas. En la capa Trust (Confianza) encontramos agentes que realizan un análisis completo y comprobación de las fuentes de información de la Web Semántica. Finalmente Digital Signatura (Firma Digital) garantiza que la información ofrecida proviene de sitios confiables.

La visión de la Web Semántica, defendida por Sir Tim Berners-Lee, está construida entorno al concepto de la "Web de Datos" (o LinkedData), que significa pasar de una Web actualmente centrada en Documentos a una Web centrada en Datos. En esta visión la Web, los datos y sus relaciones son fundamentales. (Piedra, Tovar, López, Chicaiza, & Martinez, 2011)

Un objetivo de la Web semántica es crear un sistema de agentes inteligentes que puedan hacer deducciones de una manera automatizada con la información que está en la Web. Este objetivo más que una realidad es una utopía incluso a medio plazo. Por otro lado, los desarrollos que se han realizado gracias a este nuevo paradigma han dado lugar a nuevos servicios ajustados con éxito en la actual Web. Como por ejemplo se ha logrado construir diferentes estándares para poder representar y procesar la información de una manera más sofisticada. Estos estándares que han permitido presentar los metadatos en un formato más lógico y controlados (por ejemplo ontologías) para que sean procesados por programas informáticos. Estos formatos ya son utilizados de manera generalizada, como por ejemplo XML, RDF, SKOS-Core y OWL. (Vallez, Rovira, Codina, & Pedraza, 2012)

2.1.3. Metadatos y anotación semántica.

Un elemento fundamental de la Web semántica son los metadatos, en otras palabras, información que nos describe el contenido de los documentos a los que está ligado y nos representa de una manera explícita el significado de estos.

La anotación semántica hecha con metadatos ofrece contenido semántico a los documentos para logra que las máquinas interpreten la información.

Las herramientas de anotación permiten convertir en metadatos el contenido semántico extraído de las páginas web. Existen herramientas de anotación dirigidas a los autores,

ayudan a incorporar los metadatos dentro o fuera de las propias páginas web, siguiendo los estándares (xml, rdf). Y Las aplicaciones de las herramientas de anotación externa permiten asociar meta información a páginas web, pero esta no se almacena dentro de la misma página sino que se guardada de forma externa en un repositorio. (Vallez, Rovira, Codina, & Pedraza, 2012)

Existen diferentes aproximaciones para realizar la anotación semántica, pero se pueden agrupar en tres grandes categorías. (Vallez, Rovira, Codina, & Pedraza, 2012)

- El primer modelo se basa en la anotación lingüística, el objetivo es etiquetar los textos a partir de los diferentes niveles de la lengua. Resulta de gran interés la identificación de los términos y saber cómo estos se relacionan entre sí porque esta información puede incidir en el valor de un término como palabra clave. Este sistema es muy costoso computacionalmente.
- La segunda aproximación se basa en las ontologías, son utilizadas como recurso principal para extraer las conexiones entre los términos y representar su significado.
- La tercera aproximación propone el uso de un lenguaje controlado, este es un modelo que está directamente vinculado con la asignación de metadatos y la anotación semántica.

2.1.4. Rdf.

Marco de Descripción de Recursos (Resource Description Framework), en sus inicios fue diseñado como modelo de datos para metadatos, este modelo es similar a otros modelos como el de entidad-relación o diagrama de clases, la idea es hacer afirmaciones de los recursos en forma de triples como son conocidas en terminología RDF que son sujeto-predicado-objeto:

- **Sujeto**: es el recurso
- **Predicado**: propiedad que describe los rasgos del recurso y sirve de relación entre el sujeto y
- el **Objeto:** que puede ser un valor u otra entidad.

El uso de este modelo simple permite que los datos estructurados y semi-estructurados puedan ser mesclados, expuestos y compartidos. (Working Group RDF, 2014)

Ejemplo:

Los recursos esta identificados por un identificador de recurso (URI).

Considerando una simple oración:

Ora Lassila is the creator of the resource http: //www.w3.org/Home/Lassila

La oración tiene las siguientes partes:

Subject (Resource)	http://www.w3.org/Home/Lassila
Predicate (Property)	Creator
Object (literal)	"Ora Lassila"

Figura 2 partes de la oración en RDF2[2]

Representándolo gráficamente quedaría:

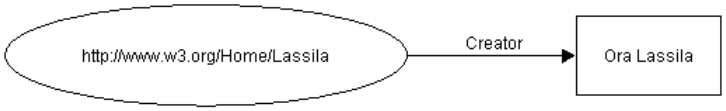

Figura 3 Representación gráfica RDF2[2]

En estos diagramas, los nodos (dibujados como óvalos) representan recursos y las líneas representan las propiedades con nombre. Los nodos que representan los literales de cadena se dibujan como rectángulos.

2.1.5. Sparql.

SPARQL (Protocol and RDF Query Language) es un lenguaje estandarizado para consulta y de recuperación basado en RDF, que esta normalizado por DAWG del Word Wide Web Consortium (W3C). Algunas características son: (W3C, 2008)

- Extraer información de URIs, literales y nodos vacíos
- Obtener subgrafos RDF y construir nuevos grafo a partir de la respuesta del query.

Con SPARQL se pueden expresar consultas para diversas fuentes de datos que estén almacenados como RDF o definidos mediante vistas RDF. También se pueden consultar patrones obligatorios y opcionales de un grafo, con sus conjunciones y disyunciones. Los resultados de las consultas realizadas con SPARQL pueden dar un conjunto de resultados así como también grafos RDF. (W3C, 2008)

2.1.5.1. Consultas simples.[3]

La mayoría de las consultas SPARQL contiene un patrón de grafo básico, estos patrones son similares a las tripletas RDF, con la diferencia que cada sujeto, predicado y objeto pueden ser una variable. (W3C, 2008)

[2] Tomado de http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
[3] Todos los ejemplos de las consultas están tomadas de Tomado (W3C, Sparql query Basicpatterns)

2.1.5.1.1. Ejemplos

2.1.5.1.1.1. Consulta simple

La consulta consiste de dos partes: la condición SELECT define las variables que aparecerán en el resultado de la consulta, y la condición WHERE provee el patrón de grafo básico para la concordancia con el grafo de datos.

Consulta:

```
SELECT *
WHERE
{
  <http://dbpedia.org/resource/Loja,_Ecuador>       <http://dbpedia.org/property/officialName>
?Nombre
}
```

Resultado:

Nombre
"Loja"@en

Consulta:

```
SELECT *
WHERE
{
?Provincias                                    <http://dbpedia.org/property/subdivisionType>
<http://dbpedia.org/resource/Provinces_of_Ecuador>.
   ?Provincias <http://dbpedia.org/property/officialName>  ?Nombre
}
```

Resultado:

Provincias	Nombre
:Loja,_Ecuador	"Loja"@en
:Manta	"San Pablo de Manta"@en
…	…

2.1.5.1.1.2. Concordancias de literales RDF

Concordancias de literales de idioma:

Las etiquetas de idioma en SPARQL se expresan usando @ y la etiqueta de idioma. La siguiente consulta no tiene solución porque "Loja" no es el mismo literal RDF que "Loja"@en:

SELECT ?s WHERE { ?s ?p "Loja" }

Por otro lado la siguiente consulta si, donde la variable s se relación, porque el idioma se especifica.

SELECT ?s WHERE { ?s ?p "Loja"@en }

Concordancias de literales con tipos numéricos:

Los enteros de una consulta SPARQL indican un literal RDF tipado con el tipo de datos xsd:integer.

Por ejemplo:

21 es una forma abreviada de "21 "^^<http://www.w3.org/2001/XMLSchema#integer>.

La consulta quedaría de esta manera:

SELECT ?v WHERE { ?v ?p 21 }

2.1.5.2. Sintaxis sparql.[4]

2.1.5.2.1. Sintaxis de expresiones rdf

2.1.5.2.1.1. Sintaxis para IRI

Los IRIs son tipo de URIs y son compatibles con URIs y URLs.

Los términos RDF tienen referencias RDF URI en cambio que los términos de SPARQL tienen IRIs.

Nombres prefijados: La palabra clave PREFIX enlaza una etiqueta de prefijo con un IRI. Un nombre con prefijo consta de una etiqueta y una parte local, separados por dos puntos ":".

Aquí tenemos algunos ejemplos de las diferentes formas describir la misma dirección IRI:

- <http://example.org/provinica/Loja>
  - PREFIX prov: <http://example.org/provinica/>
  - prov:Loja

[4] Todos los ejemplos de las consultas están tomadas de Tomado (W3C, Sparql Query SparqlSyntax)

2.1.5.2.1.2. Sintaxis para literales

La sintaxis para los literales es una cadena de caracteres entre comillas simples o dobles, con una etiqueta de idioma (antecedida por@) o un tipo de datos IRI o nombre prefijado (antecedido por ^^) opcionales.

Para los facilitar los números se pueden escribirse como normalmente se hace. Los valores del tipo xsd:boolean pueden también escribirse como true o false.

Ejemplos de sintaxis de literales SPARQL:

- 'Loja'@en con la etiqueta de idioma "en"
- "xyz"^^<http://example.org/ns/userDatatype>
- "abc"^^appNS:appDataType
- '"The librarian said, "Perhaps you would enjoy 'War and Peace'."'"
- 5, igual a "5"^^xsd:integer
- 1.5, igual a "1.5"^^xsd:decimal
- 1.500, igual a "1.300"^^xsd:decimal
- 1.0e5, igual a "1.0e5"^^xsd:double
- true, igual a "true"^^xsd:boolean
- false, igual a "false"^^xsd:boolean

2.1.5.2.1.3. Sintaxis para variables

Las variables son prefijadas con "?" o "$";y estas no forman parte del nombre de la variable. Entonces, $abc y ?abc son la misma variable.

2.1.5.2.2. Sintaxis para patrones de tripleta

Los Patrones de Tripleta se escriben como una lista, separada por espacios, de sujeto, predicado y objeto; hay formas abreviadas para escribir algunas construcciones de patrones comunes de tripleta.

Ejemplos:

- PREFIX dc: <http://purl.org/dc/elements/1.1/>
  SELECT ?title
  WHERE { <http://example.org/provinica/Loja> dc:title ?titulo }
- PREFIX dc: <http://purl.org/dc/elements/1.1/>
  PREFIX : <http://example.org/provinica/>
  SELECT $ titulo
  WHERE { :Loja dc:title $ titulo }

2.1.5.2.2.1. Listas de Predicado-Objeto

Los patrones de tripleta con un sujeto común pueden expresarse así:

- ?x foaf:name ?nombre ;
  foaf:mbox ?mail .
- ?x foaf:name ?name .
  ?x foaf:mbox ? mail.

2.1.5.2.2.2. Listas de objetos

Si los patrones de tripleta comparten tanto el sujeto como el predicado:

- ?x foaf:nick "Loja" , "Cuenca" .
  ?x foaf:nick " Loja " .
  ?x foaf:nick " Cuenca " .
- ?x foaf:name ?nombre ; foaf:nick " Loja " , " Cuenca " .
  ?x foaf:name ?nombre .
  ?x foaf:nick "Loja" .
  ?x foaf:nick " Cuenca " .

2.1.6. Dbpedia.

La Dbpedia es un proyecto para la extracción de datos de Wikipedia para proponer una versión Web semántica. Este proyecto es realizado por la Universidad de Leipzig, Universidad Libre de Berlín y la compañía OpenLink Software. Dbpedia permite hacer consultas más sofisticadas contra la Wikipedia, así como también enlazas diferentes conjuntos de datos de la Web con los de la Wikipedia. La información esta almacenada en RDF y se puede hacer consultas atreves de SPARQL. (DBpedia, About, 2012)

2.1.6.1. El Dataset de Dbpedia.

La versión en Inglés de la base de conocimiento Dbpedia describe actualmente 4,0 millones de cosas , de los cuales 3,22 millones se clasifican en una ontología, incluyendo 832.000 personas, 639.000 plazas (incluidos 427 000 lugares poblados), 372.000 obras de creación (incluyendo 116.000 álbumes de música, 78.000 películas y 18.500 juegos de video), 209.000 organizaciones (entre ellas 49.000 empresas y 45.000 instituciones educativas), 226.000 especies y 5.600 enfermedades. (DBpedia, Datasets, 2013)

El conjunto de datos completo Dbpedia cuenta con las etiquetas y los resúmenes de 12,6 millones de cosas únicas en 119 idiomas diferentes; 24,6 millones de enlaces a las imágenes y los 27,6 millones de enlaces a páginas web externas; 45,0 millones de enlaces externos a otros conjuntos de datos RDF, 67,0 millones de enlaces a las categorías de Wikipedia, y 41,2 millones de categorías YAGO. El conjunto de datos consta de 2,46 mil millones de piezas de información (RDF triplica) fuera de las cuales 470 millones fueron extraídos de la edición de Inglés de Wikipedia, 1,98 millones fueron extraídos de ediciones

en otros idiomas, y unos 45 millones son enlaces a bases de datos externas. (DBpedia, Datasets, 2013)

2.2. Recursos educativos abiertos ocw

2.2.1. Oer (open educational resources).

OER (Open Educational Resources) o REA (Recursos Educativos Abiertos) son cualquier tipo de materiales educativos que se encuentran en el dominio público o introducidos con una licencia abierta. La naturaleza de estos materiales abiertos significa que cualquier persona legalmente y libremente puede copiarlos, usarlos, adaptarlos y compartirlos. Los OERs tienen una variedad de libros de texto como: planes de estudio, notas de clase, tareas, exámenes, proyectos, audio, vídeo y animación. (UNESCO, 2005)

OER nació como un concepto con gran potencial para poder sobrellevar la transformación en la educación. EL valor educativo se basa en la idea de utilizar los recursos como un método de aprendizaje, su poder se basa en la facilidad con la que los recursos ya digitalizados, se comparten a través de Internet.

Se deben tomar en cuenta tres elementos fundamentales de REA/OER:[5]

- Contenido Educativo: Recursos educativos como libros, programas educativos completos, módulos de contenido, publicaciones, etc.
- Herramientas: Software para apoyar la creación, acceso, uso y mejoramiento de contenidos educativos abiertos.
- Recursos de Implementación: licencias de propiedad intelectual que promueva la publicación abierta de principios diseño, adaptaciones del contenido, materiales y técnicas para apoyar al acceso del conocimiento. Los recursos educativos abiertos frecuentemente están distribuidos bajo una licencia Creative Commons.

La tarea de aseguramiento de la calidad se ha visto complicada por la explosión de disposición contenido (tanto abierto como propietario). Esto es tanto una ventaja ya que reduce la probabilidad de necesitar para desarrollar nuevos contenidos, y una desventaja, ya que requiere mayor nivel de habilidades en la búsqueda de información, selección, adaptación y evaluación, tomando en cuenta que las instituciones comparten más contenido

[5] Fuente: Wikipedia (http://es.wikipedia.org/wiki/Open_educational_resources)

educativo en línea, van asegurarse de que le contenido refleje una buena institución y por lo tanto podrá invertir en la mejora de su calidad antes de que este a disposición en los repositorios.
(OER Africa, 2013)

2.2.2. Ocw.

OpenCourseWare (OCW) es una publicación digital abierta y gratuita de materiales educativos de alta calidad. Están clasificados desde cursos completos hasta asignaturas. Mantiene una licencia abierta accesible a cualquiera o cualquier lugar a través de la red. (OCW Consortium, 2012)

OCW es un ejemplo de las iniciativas que desde el 2001 han emergido para promover el acceso libre y sin restricciones al conocimiento.
(UTPL, 2014)

OCW site**:** Es un espacio web que contiene materiales docentes creados por profesores para la formación superior. Las características que distinguen al proyecto OpenCourseWare de iniciativas similares son las siguientes: (UPM)

- Los recursos didácticos publicados en un OCW site se organizan en unidades de “asignaturas” o “cursos”. Con ello se quiere indicar:
  - Los accesos se realizan por asignaturas e incluyen un conjunto significativo de todos los materiales asociados a ella.
  - Los materiales se ofrecen de forma organizada por categorías: programa de la asignatura, lecturas obligatorias, materiales de clase, ejercicios, guía de aprendizaje.
- El profesor o profesores garantizan que el material que publican en el OCW site es original o tiene los derechos, bien directamente por ser propietario o bien a través del tipo de licencia que los soporta, para ser reutilizados en “abierto” sin infringir los “copyrights” de otras personas.
- Son accesibles universalmente a través de la red:
  - Sin limitaciones geográficas.
  - Sin exclusión de usuario, ni necesidad de registrarse o utilizar palabras claves de acceso.
  - No exigen requisitos técnicos más allá de un navegador Web.

Entre los beneficios que nos brinda OCW tenemos:

- “El uso de OCW mejora la calidad de los materiales docentes que ofrecen los profesores, al nutrirse de la experiencia de otros materiales que han sido probados con éxito.
- El OCW incentiva a hacer actividades virtuales.
- El OCW abre ventanas de colaboración con otras instituciones y otros profesionales posicionados en la materia.
- Los alumnos pueden consultar los materiales antes de matricularse con un profesor, lo que les permite tener más herramientas para tomar su decisión.
- Se hace más visible en qué medida la universidad devuelve a la sociedad los recursos que toma de ella, de manera que ésta pueda valerse de éstos directamente.
- Para los gestores se puede saber cómo orientan los profesores su docencia y el aula deja de ser una caja negra en la que no se sabe qué pasa.
- Incrementa la difusión del conocimiento, lo cual es beneficioso en sí mismo.
- Etiquetar los contenidos ha aumentado su visibilidad, incrementando el número de visitas.” (López, Piedra, Sancho, Soto, & Tovar, 2012)

2.3. Procesamiento del lenguaje natural

El " Procesamiento del Lenguaje Natural " (PLN) es una disciplina con una larga trayectoria. Nace en la década de 1960, como un subárea de la Inteligencia Artificial y la Lingüística, con el objeto de estudiar los problemas derivados de la generación y comprensión automática del lenguaje natural. (Mari & Rafael Pedraza, 2012)

El PLN investiga mecanismos eficaces computacionales para la comunicación entre personas o entre personas y máquinas por medio de lenguajes naturales. Trata de diseñar mecanismos para comunicarse que sean eficaces computacionalmente hablando. La lingüística general se relaciona con PNL, ya que esta estudia la estructura general y descubre las leyes universales de funcionalidad de los lenguajes naturales. Estas estructuras y leyes, aunadas a los métodos computacionales forman la lingüística computacional.

La lingüística computacional puede ser considerada como un sinónimo de procesamiento de lenguaje natural, ya que su tarea principal es la construcción de programas que procesen palabras y textos en lenguaje natural. (Bolshakov & Gelbukh, 2004)

Para poder realizar esa tarea, los sistemas de PLN deben tener conocimiento sobre la estructura del lenguaje, y así poder pasar de texto a significado y viceversa.

Niveles de Lenguaje

La lingüística general comprende 5 niveles principales para el análisis de la estructura del lenguaje (Bolshakov & Gelbukh, 2004) que son:

a. **Nivel fonológico**: trata de los sonidos que comprenden el habla, permitiendo formar y distinguir palabras.
b. **Nivel morfológico**: trata sobre la estructura de las palabras y las leyes para formar nuevas palabras a partir de unidades de significado más pequeñas llamadas morfemas.
c. **Nivel sintáctico**: trata como las palabras pueden unirse para construir oraciones y cuál es la función que cada palabra realiza en esa oración.
d. **Nivel semántico**: trata del significado de las palabras y de cómo se unen para dar significado a una oración.
e. **Nivel pragmático**: estudia la intención del hablante al producir oraciones específicas o textos en una situación específica.

2.4. Servicios web

Los Servicios Web (Web Services) son componentes software que permiten intercambiar información y datos entre aplicaciones, mediante el uso de tecnologías Web basadas en estándares y protocolos. Los servicios web se diseñaron para que se pueda acceder por otras aplicaciones. Los servicios web son un conjunto de herramientas que pueden ser usadas en distintas formas. (Alvarado Ruiz, Guamán Eras, & Sigcho Armijos, 2012)

Según el W3C (World Wide Web Consortium) los Servicios Web son aplicaciones de software identificadas por un URI (Uniform Resource Identiflier), cuyos interfaces y vínculos tienen la capacidad de estar bien definidos, descritos y descubiertos como objetos XML. Los servicios web soportan interacciones directas con otros agentes de software usando mensajes de intercambio basados en XML vía protocolos basados en Internet. Se puede perfeccionar esta definición pidiendo que la descripción se haga a través de un documento WSDL (Web Services Description Language) y el protocolo utilizado sea SOAP. (W3C, 2004)

Puesto que cada Servicio Web puede estar implementado en una tecnología heterogénea es necesario cumplir una serie de estándares para hacer posible la comunicación entre ellos. Los más utilizados son los siguientes: (González, 2011)

Web Services Protocol Stack: conjunto de servicios y protocolos de los servicios Web.

- XML: Es un lenguaje de marcas capaz de describir distintos tipos de datos. Es un estándar aceptado y utilizado como medio de descripción de datos.
- SOAP: Es un protocolo de comunicación entre procesos basado en el intercambio de mensajes en formato XML dentro de una red. A su vez SOAP está basado en XML y es completamente independiente de la plataforma y del lenguaje en el que estén implementados los procesos que se comunican.
- WSDL: Es un lenguaje basado en XML que permite describir servicios web (como su nombre indica). Un documento WSDL específica, entre otras cosas, dónde se encuentra el servicio así como las operaciones que pone accesibles a otros servicios.
- UDDI: Es un directorio, basado en XML, en el que las distintas empresas dan de alta servicios web que ponen al servicio de otra empresas.
- WS-Security (Web Service Security): Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los “actores” y la confidencialidad de los mensajes enviados.

2.4.1. Arquitectura rpc (remote procedure call).

En el paper de (Nelson & Andrew, 1984) la idea fue permitir que los programas llamaran a procedimientos localizados en otras máquinas. Cuando un proceso en la máquina A llama a un procedimiento en la máquina B, el proceso llamado en A es suspendido, y la ejecución del procedimiento llamado se lleva a cabo en el B. La Información puede ser transportada de la persona que llama al destinatario de la llamada en los parámetros, y puede volver a aparecer en el resultado del procedimiento.

No hay ningún mensaje que pase visible para el programador. Este método se conoce como Remote Procedure Call, o solo RPC.

La idea básica parece simple y elegante, pero los problemas existen. Para empezar, porque los procedimientos de llamada y la llamador, se ejecutan en máquinas diferentes, que se ejecutan en espacios de direcciones diferentes, lo que provoca complicaciones. Parámetros y resultados también tienen que ser enviados, lo que puede ser complicado, especialmente si las máquinas no son idénticas. Por último, ambas máquinas pueden fallar y cada uno de

los posibles fallos causa diferentes problemas. Sin embargo, la mayoría de estos pueden ser tratados, y RPC es una técnica ampliamente utilizada que subyace a muchos de los sistemas distribuidos.

El RPC sigue los siguientes pasos:

- El procedimiento en el cliente llama al cliente stub de manera normal
- El cliente stub construye un mensaje y llama al sistema operativo local
- El Sistema operativo cliente envía un mensaje para el OS remoto
- El OS remoto da un mensaje al servidor stub
- El servidor stub desempaqueta los parámetros y llama al servidor
- El servidor hace el trabajo y retorna el resultado para el stub
- El servidor stub empaqueta en un mensaje para el OS cliente
- El OS servidor envía el mensaje al OS cliente
- El OS cliente da un mensaje al cliente stub
- El stub desempaqueta el resultado y retorna al cliente" (Nelson & Andrew, 1984) l

2.4.2. Arquitectura orientada a servicios.

Arquitecturas orientadas a servicios se han utilizado durante muchos años. SOA es una de articulación flexible por lo que la distingue otras arquitecturas. Acoplamiento débil significa que el cliente de un servicio es esencialmente independiente del servicio. La forma en que un cliente se comunica con el servicio no depende de la aplicación del servicio. De manera significativa, por lo cual el cliente no tiene que saber mucho sobre el servicio para usarlo. El acoplamiento flexible permite a los servicios ser un documento-orientado. El cliente se comunica con el servicio de acuerdo a una interfaz específica, bien definida, y luego se deja en manos de la implementación del servicio para realizar el procesamiento necesario. Si la implementación del servicio cambia, el cliente se comunica con ella en la misma forma que antes, siempre que la interfaz sigue siendo el mismo.

Sin embargo lo que es relativamente nuevo es la aparición de servicios web basados SOAs. Un servicio web es un servicio que se comunica con los clientes a través de un conjunto de protocolos y tecnologías estándar. Estos estándares de servicios web se ejecutan en plataformas y productos de todos los principales proveedores de software, lo que permite a los clientes y servicios para comunicarse de una manera consistente a través de un amplio espectro de plataformas y entornos operativos. Esta universalidad ha hecho

que los servicios web de la forma más extendida implementen SOA. (SunMicrosystems, 2005)

2.4.3. Arquitectura rest.

REST (Transferencia de estado representacional) es un estilo de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web. El término fue introducido en la tesis doctoral en el año 2000 por Roy Fielding, y ha entrado en uso generalizado en la comunidad de redes. (Fielding, 2000)

REST se refiere estrictamente a una colección de principios de la arquitectura de red que describen cómo los recursos son definidos y tratados. El término se utiliza a menudo en un sentido más amplio para describir cualquier interfaz simple que transmite datos específicos del dominio a través de HTTP y sin capa de mensajería adicionales, tales como SOAP o una sesión de seguimiento a través de cookies HTTP. (Alvarado Ruiz, Guamán Eras, & Sigcho Armijos, 2012)

- Principios de diseño:
- El estado de aplicación y la funcionalidad se dividen en recursos.
- Cada recurso es direccionable únicamente utilizando una sintaxis universal para su uso en enlaces hipermedia
- Todos los recursos comparten una interfaz uniforme para la transferencia de estado entre el cliente y los recursos, constituido por
    - Un conjunto limitado de operaciones bien definidas
    - Un conjunto limitado de tipos de contenido, opcionalmente soporte de código on-deman
- Un protocolo que es:
    - cliente / servidor
    - sin estado
    - Cacheable
    - en capas

La separación de cliente-servidor REST simplifica la implementación de componentes, reduce la complejidad de la semántica de los conectores, mejora la eficacia de la optimización del rendimiento, y aumenta la escalabilidad de los componentes de servidor puros. Limitaciones en capas del sistema permiten a los intermediarios - los proxies,

gateways y servidores de seguridad - que se presentó en varios puntos de la comunicación sin necesidad de cambiar las interfaces entre los componentes, lo que les permite ayudar en la traducción de la comunicación o mejorar el rendimiento a través de gran escala, el almacenamiento en caché compartida.

2.5. Programas y librerías

2.5.1. Python.

Python es un lenguaje de programación dinámico y potente que es utilizado en varios dominios de aplicación. Sus características más relevantes son: (Python, 2002)

- Su sintaxis es muy clara y legible
- Fuerte Capacidad de introspección
- Orientado a objetos
- Expresión natural del código procedimental
- Completamente modular, Soporte para paquetes jerárquicos
- Manejo de errores basado en excepciones
- Los tipos de datos son dinámicos de muy alto nivel
- Bibliotecas estándar y módulos de terceros para prácticamente todas las tareas
- Puede ser integrada en aplicaciones mediante interfaz de script

Python es un lenguaje de programación que permite trabajar con mayor rapidez e integrar sus sistemas con mayor eficacia. También se puede ejecutar en Windows, Linux / Unix, Mac OS X, y ha sido adaptada para las máquinas virtuales de Java y NET.

Python se desarrolla bajo una licencia de código abierto aprobada por OSI, por lo que es de libre uso y distribuible, incluso para uso comercial. (Python, 2002)

2.5.1.1. Nltk.

NLTK es una plataforma para python que trabaja con datos de lenguaje humano. Proporciona interfaces fáciles de usar, junto con un conjunto de bibliotecas de procesamiento de textos para la clasificación, tokenización, derivado, el etiquetado, el análisis y el razonamiento semántico. (Nltk, 2013)

NLTK se lo puede trabajar en Windows, Mac OS X y Linux. NLTK también es código abierto, impulsado por la comunidad.

El público objetivo de NLTK consiste en lingüistas y científicos de computación, y es accesible y desafiante en muchos niveles de las aptitudes computacionales. Y se basa en un lenguaje de programación orientado a objetos apoyándose prototipado rápido y programación literaria. (Loper & Bird, 2002)

Librerías de NLTK utilizadas:
Sent_tokenize: es una funcionalidad de NLTK, lo que hace es segmentar el texto en frases.

Word_tokenize: esta funcionalidad es parecida a la anterior, solo que una vez segmentada en fases, lo que hace word_tokenize es segmentar en palabras.

Pos_tag: es una funcionalidad de NLTK, que sirve para la clasificación de las palabras, procesa la secuencia de las palabras, y les da una etiqueta a cada palabra, dependiendo de la función que desempeña en la frase.

The Penn Treebank POS tagset.

1. CC	Coordinating conjunction	25. TO	*to*
2. CD	Cardinal number	26. UH	Interjection
3. DT	Determiner	27. VB	Verb, base form
4. EX	Existential *there*	28. VBD	Verb, past tense
5. FW	Foreign word	29. VBG	Verb, gerund/present participle
6. IN	Preposition/subordinating conjunction	30. VBN	Verb, past participle
7. JJ	Adjective	31. VBP	Verb, non-3rd ps. sing. present
8. JJR	Adjective, comparative	32. VBZ	Verb, 3rd ps. sing. present
9. JJS	Adjective, superlative	33. WDT	*wh*-determiner
10. LS	List item marker	34. WP	*wh*-pronoun
11. MD	Modal	35. WP$	Possessive *wh*-pronoun
12. NN	Noun, singular or mass	36. WRB	*wh*-adverb
13. NNS	Noun, plural	37. #	Pound sign
14. NNP	Proper noun, singular	38. $	Dollar sign
15. NNPS	Proper noun, plural	39. .	Sentence-final punctuation
16. PDT	Predeterminer	40. ,	Comma
17. POS	Possessive ending	41. :	Colon, semi-colon
18. PRP	Personal pronoun	42. (	Left bracket character
19. PP$	Possessive pronoun	43.)	Right bracket character
20. RB	Adverb	44. "	Straight double quote
21. RBR	Adverb, comparative	45. ‘	Left open single quote
22. RBS	Adverb, superlative	46. “	Left open double quote
23. RP	Particle	47. ’	Right close single quote
24. SYM	Symbol (mathematical or scientific)	48. ”	Right close double quote

Figura 4 Conjunto de etiquetas The Penn Treebank Pos tag[6]

RegexpTagger: sirve para hacer una clasificación y etiquetado de las palabras pero con expresiones regulares en base a patrones. Por ejemplo se puede decir que todas las palabras en inglés que termina en “ed” son verbos en pasado participio, así como también las palabras que terminen en ‘s son nombres posesivos.

[6] Tomado de http://www.nltk.org/book/ch05.html

```
patterns = [
    (r'.*ing$', 'VBG'),               # gerunds
    (r'.*ed$', 'VBD'),                # simple past
    (r'.*es$', 'VBZ'),                # 3rd singular present
    (r'.*ould$', 'MD'),               # modals
    (r'.*\'s$', 'NN$'),               # possessive nouns
    (r'.*s$', 'NNS'),                 # plural nouns
    (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),  # cardinal numbers
    (r'.*', 'NN')                     # nouns (default)
]
```

Figura 5 Ejemplo de expresiones regulares para etiquetado con RegexpTager[5]

Chunk: es un paso preliminar y útil para la extracción de información, este crea árboles de análisis de texto no estructurado con un Chunker. Una vez obtenido un árbol de análisis sintáctico de una oración, se puede hacer la extracción de información más específica, como el reconocimiento de entidades y extracción de relaciones.

Fragmentación es básicamente un proceso de 3 pasos:

- Se etiqueta una sentencia
- Se hace un Chunk de la sentencia etiquetada
- Analizar el árbol obtenido para extraer información

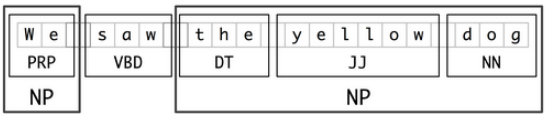

Figura 6 Segmentación y Etiquetado Chunk[7]

2.5.1.2. SparqlWrapper

SparqlWrapper es una librería de python, que sirve para hacer consultas a un end-point de SPARQL. Con esta librería se puede hacer consultas SPARQL, en la figura 7 se ve un ejemplo.

```
1  from SPARQLWrapper import SPARQLWrapper, JSON
2
3  sparql = SPARQLWrapper("http://dbpedia.org/sparql")
4  sparql.setQuery("""
5      PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
6      SELECT ?label
7      WHERE { <http://dbpedia.org/resource/Asturias> rdfs:label ?label }
8  """)
9  sparql.setReturnFormat(JSON)
10 results = sparql.query().convert()
11
12 for result in results["results"]["bindings"]:
13     print(result["label"]["value"])
```

Figura 7 Ejemplo de cómo usar la librería SparqlWrapper[8]

2.5.1.3. Goslate

Goslate es una librería gratis que proporciona a Python un servicio de traducción de Google, funciona haciendo consultas el sitio web del traductor de Google.

[7] Tomado de http://www.nltk.org/book/ch07.html
[8] Tomado de http://rdflib.github.io/sparqlwrapper/

PROBLEMÁTICA

3.1. Problemática actual

Actualmente se puede encontrar una gran cantidad de información bastante detallada, que se encuentra representada en un enorme conjunto de documentos electrónicos que están localizados en distintos lugares y en diferentes formatos.

Por ejemplo, en la web se dispone de una amplia variedad de datos relacionados a un sinfín de temas específicos: medicina, educación, matemáticas, entretenimiento, noticias, entre otros; en lo que concierne a la educación se dispone de bastantes recursos como son Wikis, blogs, tutoriales, presentaciones, papers, noticias, videos, infografías, bases de datos, entre otros.

La educación superior dispone de OER y OCW, estos manejan gran cantidad de recursos anteriormente mencionados, dichos contenidos están presentados en su mayoría en lenguaje natural, es decir, aquel que puede ser entendido y discernido fácilmente por las personas pero no es entendible para las computadoras.

Para que una máquina pueda llevar a cabo la inferencia de conocimiento a partir de unos datos entregados, dichos datos deberán estar definidos en un lenguaje que se conoce como entendible por máquina.

En el caso de los cursos OCW se dispone de una gran cantidad de información que puede ser entregada a un usuario para su consumo, manipulación y almacenamiento, pero existen ciertas preguntas con respecto a todo este conjunto de información:

- ¿Cuál información de todo este conjunto de datos es relevante?
- ¿Cómo seleccionar aquellos cursos que sean más idóneos para la búsqueda planteada?
- ¿Cómo clasificar los datos/recursos/autores/artículos que manejan los cursos OCW?

Esto lleva a plantearse una pregunta más general:

- ¿Qué entidades se puede extraer de un curso OCW?

Puesto que un OCW presenta un conjunto de datos bastante amplio como:

- Organización a la que pertenece
- Autor
- Áreas de conocimiento
- Países

- Contenido, dentro de este existe todavía más datos que pueden ser relevantes:
    - Menciones a papers
    - Autores relacionados
    - URLs, URIs
    - Software
    - Conceptos
    - Organizaciones (universidades, empresas, consorcios)

Con esta amplia variedad de datos se puede crear etiquetas que nos permitirán de alguna forma resumir o detallar el contenido de un curso OCW.

Por ejemplo, se realiza una búsqueda sobre aplicaciones de la Web Semántica en ámbito médico, al llevar a cabo una búsqueda basada en ciertas palabras claves, el resultado de dicha búsqueda sería bastante amplio; en cambio que si cada curso OCW dispone de etiquetas relevantes como: Web Semántica, aplicaciones de Web Semántica, medicina, la búsqueda resultaría más eficaz, al basar la misma en etiquetas o al realizarla mediante palabras claves y luego filtrar la misma mediante las etiquetas.

La extracción de entidades no sólo resultaría en un enriquecimiento de la información disponible del curso, al extraer todos aquellos datos que son relevantes, sino también en una forma de pre-clasificación del curso OCW, mediante la generación de etiquetas a partir del contenido.

En la web toda la información está estrechamente relacionada una con otra, así mismo al extraer entidades de un curso OCW, dichas entidades pueden usarse para interrelacionar un curso con otro.
Por ejemplo, una entidad de tipo autor que ha sido extraída de un curso OCW que habla sobre ontologías biológicas, puede ser usada para relacionar con algún paper, recurso, libro o publicación que dicho autor haya hecho anteriormente.

Esto permite enriquecer o delimitar la información sobre un OCW:

- enriquecer porque se puede relacionar un OCW con otros mediante las distintas entidades extraídas;

- delimitar, porque mediante la generación de etiquetas se puede llevar a cabo una búsqueda más exacta, precisa y eficaz sobre un tema o curso en específico.

SOLUCIÓN

4.1. Aproximación

Se planea realizar la extracción de entidades como: lugares, países, ciudades, áreas de conocimientos, organizaciones académicas, autores.

Las entidades extraídas además servirán como metadatos que representaran el contenido de documentos.

Para lo cual se desarrollaran servicios web cuya función serán: tokenizar, extraer entidades, desambiguar y enlazar con LOD-Cloud, estos servicios web se los desarrollara con anotaciones semánticas, para que puedan permitir la interoperabilidad entre los servicios existentes.

En la siguiente grafica se muestra la estructura mencionada:

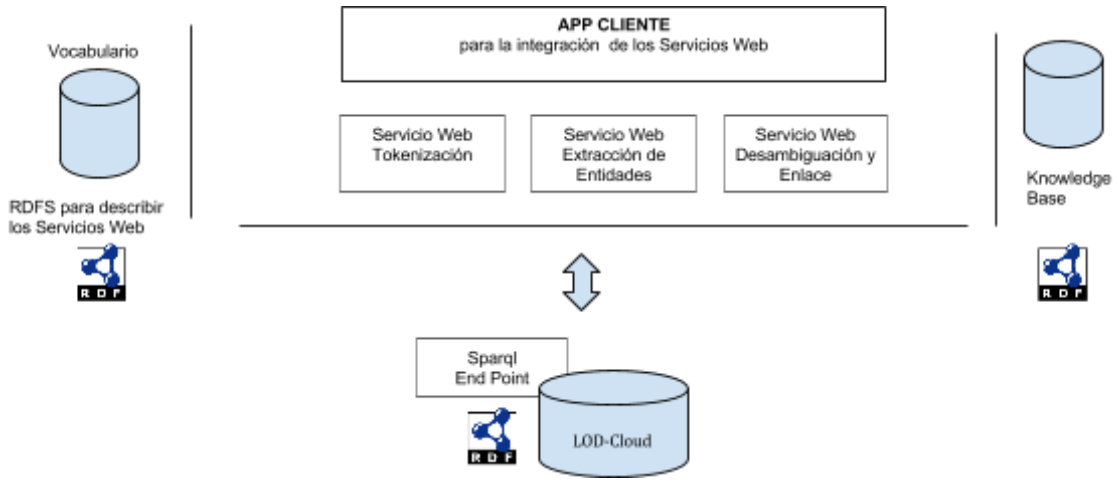

Figura 8 Estructura de los Componentes

4.2. Descripción de componentes.

La solución estará modelada en base a distintos componentes y recursos como son:

4.2.1. Vocabulario.

Se definirá un vocabulario para describir los servicios web semánticamente, puesto que dichos servicios deben llevar anotaciones semánticas para facilitar y optimizar la interoperabilidad entre los servicios existentes, el vocabulario estará definido en RDFs.

4.2.2. Knowledge base (base de conocimiento).

Se necesita definir una data para llevar a cabo la extracción de entidades en base a conceptos que hayamos definido anteriormente; debemos definirla puesto que existen datos que son propios del ámbito universitario; entre estos conceptos tenemos:

a. **Entidades,** entre estas tenemos universidades, organizaciones académicas, empresas, organizaciones gubernamentales.
b. **Autores,** sean conferencistas, autores de publicaciones, sean libros, papers, revistas, artículos, de cursos, de talleres, de toda persona que sea expositora de un tema.
c. **Áreas de conocimiento.**
d. **Lugares,** como son países, ciudades, provincias, localizaciones.
e. **Eventos,** como conferencias, simposios, congresos, encuentros.

4.2.3. Lod-cloud.

Son las siglas de Linked Open Data que es son datos abiertos enlazados contenidos en la nube. Se utilizara los datos contenidos en el Lod-Cloud de Dbpedia para enlazarlos con las entidades extraídas.

4.2.4. Servicio web tokenización.

Se definirá este servicio web para el manejo del texto, sea:

- **Codificación**, mediante esta funcionalidad se definirá el formato estándar con el cual trabajar el texto, sea este uft-8 o asccii, así como se extraerá el idioma en el cual se encuentra el texto introducido.
- **Tokenización**, mediante este servicio se tokenizará cada palabra perteneciente al texto, mediante tokenización de oraciones completas para no perder el sentido ni la sintaxis.
- **Etiquetado**, se establecerá una etiqueta por cada palabra dentro de cada oración del texto introducido, esto con el fin de determinar el tipo de palabra pudiendo ser: un artículo, un verbo, un sustantivo, un adjetivo, un adverbio, un pronombre o nombres propios.

4.2.5. Servicio web extracción de entidades.

Mediante este servicio se analizará y procesará el texto para encontrar aquella información relevante dentro del mismo, como principal enfoque se buscará extraer entidades como autores, organizaciones, lugares, datos informáticos, entre otros.

Dicho servicio utilizará procesamiento de lenguaje natural, puesto que en la actualidad existen librerías, documentación, programas y recursos que nos facilitan el PLN; así mismo se emplearán algoritmos, métodos y técnicas para extraer conocimiento útil de datos de texto no estructurados a través de la identificación de conceptos básicos.

4.2.6. Servicio web desambiguación y enlace.

Este servicio web se encargará de 2 tareas:

a. Desambiguación, se entiende por esto que teniendo varios conceptos relacionados a una palabra específica se definirá un sentido del conjunto de posibilidades predefinidas, de acuerdo al contexto de la palabra.
b. Se enlazará mediante este servicio se creará un enlace entre la data extraída, definida y almacenada con el LOD-Cloud que son servicios en la nube para Linked Open Data, por medio de este enlace la información se enriquecerá, puesto que al estar disponible en la web se crearán relaciones entre las entidades que han sido extraídas y la información que la web tiene sobre la misma entidad.

4.2.7. App cliente.

Es la interfaz que permitirá integrar todos los servicios definidos anteriormente, para hacer más fácil y sencillo el consumo de los mismos.

4.3. Vocabulario para descripción de servicios web.

Para la descripción semántica de los Servicios Web, se a creado una vocabulario reutilizando términos de otros vocabularios.

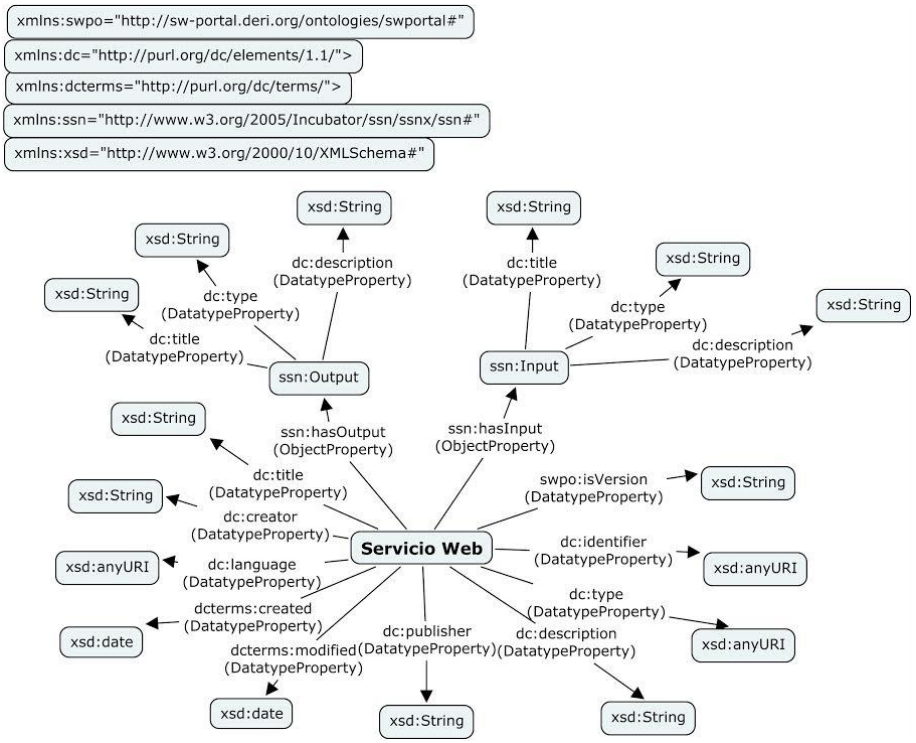

Figura 9 Grafico del Vocabulario para la descripción de los Servicios Web

En la Figura 9 esta descrito gráficamente el vocabulario creado poder anotarlos a los servicios web semánticamente. Para este vocabulario se utilizó varios términos de distintos vocabularios, a continuación se lista los vocabularios reutilizados:

- swpo="http://sw-portal.deri.org/ontologies/swportal#"
- dc="http://purl.org/dc/elements/1.1/">
- dcterms="http://purl.org/dc/terms/">
- ssn="http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#"
- xsd="http://www.w3.org/2000/10/XMLSchema#"

De los vocabularios reutilizados se utilizó ciertos términos, que para mejor entendimiento se los describirá, para dejar claro cuál es su función en este vocabulario:

- dc:title: Se lo utiliza para describir el nombre el título del Servicio Web o de la entrada o salida del Servicio.
- dc:creator: Es para definir el autor que creo el Servicio Web.
- dc:language: Describe el tipo de lenguaje del Servicio Web.
- dcterms:created: Sera para colocar la fecha en que se creó el Servicio Web.
- dcterms:modified: Se lo utiliza para agregar las fechas de las últimas modificaciones del Servicio Web.
- dc:publisher: Identifica quien o donde se lo público.
- dc:description: Sirve para dar una descripción más a detalle del Servicio Web o de la entrada o salida del servicio.
- dc:type: Se lo utiliza tanto para describir el tipo de Servicio Web asi como también el tipo de entrada o salida del Servicio.
- dc:identifier: Es para identificar la dirección donde se encuentra el servicio web.
- swpo:isVersion: Identifica la versión en la que está el Servicio Web.
- ssn:Input: Clase para describir las entradas del Servicio Web
- ssn:Output: Clase para describir las salidas del Servicio Web

En el Anexo 1 se encuentra el vocabulario así como también la descripción de los servicios web en formato rdf/xml.

DESARROLLO

5.1. Servicio web tokenización.

El diseño de este servicio web se realiza la tokenización de texto. Para la elaboración de este servicio se realizó lo siguiente:

- El texto se lo separa en oraciones utilizando la función *sent_tokenize* de la librería NLTK, para luego esas oraciones se la separe en tokens con la misma librería con la función *word_tokenize*.
- Para el etiquetado de los tokens antes mencionados se establece tipos para cada palabra del texto con *RegexpTagger de NLTK* y se etiqueta con la función *pos_tag*.
- Este servicio nos devuelve un json con la lista con los tokens etiquetados.

5.1.1. Arquitectura.

5.1.1.1. Diagrama de secuencia.

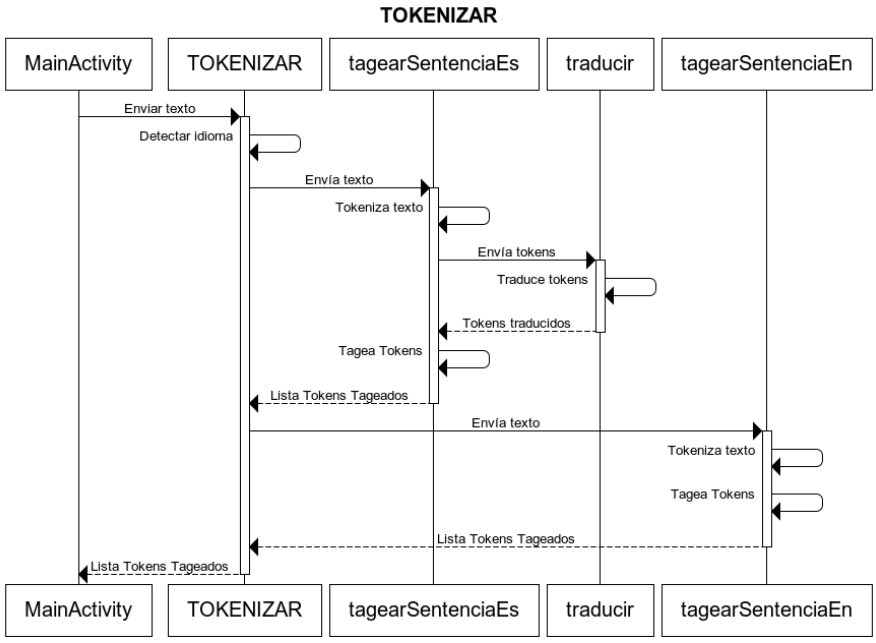

Figura 10 Diagrama de secuencia de la funcionalidad Tokenizar

A continuación se describe su comportamiento:

1. Se llama al servicio web ***tokenizar*** y se le envía el texto.
2. ***tokenizar*** recibe el texto y se procede a detectar el idioma y dependiendo si está en español o inglés se lo envía a las otras funciones.
3. Si el idioma es español se envía el texto a *tagearSentenciaEs*, lo que hace primero este método es dividir el texto en tokens.
4. Luego se lo envía los tokens a la función *traducir* y nos devuelve los tokens traducidos.

5. Con los tokens traducidos la función *tagearSentenciaEs* los etiqueta y devuelve una lista con los token etiquetados.
6. Si en cambio el idioma está en inglés envía el texto a la función *tagearSentenciaEn.*
7. La función *tagearSentenciaEn* con el texto recibido procede a tokenizar y luego a etiquetar los tokens y nos devuelve una lista con los token etiquetados.
8. El servicio Web nos devuelve un json con la lista con los tokens etiquetados.

5.1.2. Implementación.

5.1.2.1. Parámetros.

Como parámetro de entrada se tiene:

- *sentence,* consiste en una variable de tipo carácter o string que contiene el texto ha ser analizado, procesado y etiquetado.

Como parámetro de salida:

- *Json, que contiene* una lista desglosada mediante PLN, en la cual cada palabra se encuentra separada y clasificada mediante una etiqueta que denota su funcionalidad dentro del texto:


```
{
  "tokens": [
    [
      "The",
      "DT"
    ],
    [
      "second",
      "JJ"
    ],
    [
      "half",
      "DT"
    ],
    [
      "of",
      "IN"
    ],
    [
      "the",
      "DT"
    ],
```


Figura 11. Salida del servicio web de tokenización

5.1.2.2. Funciones.

Este servicio utiliza las siguientes funciones:

Tabla 1. Funciones utilizadas en el servicio web tokenizar

FUNCIÓN	PARÁMETROS		DESCRIPCIÓN
tagear Función principal	Entrada	sentencia	El servicio web recibe el parámetro de entrada sentencia y lo envía a esta función que lo recibe como sentencia
	Salida	Lista	En este parámetro esta la lista con las palabras separadas en tokens con sus etiquetas
tagearSentenciaEs	Entrada	tags	Esta variable recibe el contenido para ser procesado, para luego ser tokenizado y etiquetado.
	Salida	tags	Devuelve una lista con los tokens etiquetados.
tagearSentenciaEn	Entrada	tags	Esta variable recibe el contenido para ser procesado, para luego ser tokenizado y etiquetado.
	Salida	tags	Devuelve una lista con los tokens etiquetados.
traducir	Entrada	token	En esta variable recibe el token en español que se va a traducir
	Salida	tokentra	Devuelve el token traducido a ingles

5.1.2.2.1. Tagear

```
def tagear(self,sentencia):
    sentencia= self.encuentraUrl(sentencia)
    sentencia=sentencia.decode('utf-8')
    try:
        idioma=gs.detect(sentencia)
    except Exception, e:
        return []
    lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Albanian","ar"

    idiomaCompleto=lenguajes[idioma]

    if idioma =='es':
        Lista=self.tagearSentenciaEs(sentencia)
    elif idioma == 'en':
        Lista=self.tagearSentenciaEn(sentencia)
    else:
        mensaje='El idioma %s no es soportado'%idiomaCompleto
        print mensaje
        mensaje=gs.translate(mensaje, idioma)
        print mensaje
        Lista=[[mensaje,mensaje]]
    return Lista
```

Figura 12. Script de la función tagear del servicio web tokenizar

- Esta función recibe como entrada el texto ingresado que se almacena en la variable ***sentencia***.
- Se procede a detectar el idioma del texto mediante la librería goslate (gs), para determinar el lenguaje en que se encuentra escrito el texto ingresado.
- Si el texto está en español se lo envía a la función ***tagearSentenciaEs***, y esta función nos devuelve una lista con el texto separados en tokens y etiquetado, para ser asignado a la variable ***Lista.***
- Por otro lado si el texto está en inglés se lo envía a la función ***tagearSentenciaEn***, y esta función nos devuelve una lista con el texto separados en tokens y etiquetado, para ser asignado a la variable ***Lista.***
- Y si el idioma del texto no es ni español, ni inglés, se almacena en la variable ***Lista*** el mensaje de lenguaje no soportado.
- Por ultimo esta función nos da como salida la variable ***Lista***.

5.1.2.2.2. TagearSentenciaEs

```
def tagearSentenciaEs(self,tags):
    tags=word_tokenize(tags)
    patterns = [
    (u'^,$', ','),
    (u'^;$', ';'),
    (u'^:$', ':'),
    (u'^\.$', '.'),
    (u'^-$', '-'),
    (u'^/$', '/'),
    (u'^\($', '('),
    (u'^\)$', ')'),
    (u'^\-$', '-'),
    (u'^(como|de|De|con|Con|En|en|\xe2\x80\x93)$', 'IN'),
    (u'^(A|A|(a|A)nte|(b|B)ajo|(c|C)on|(c|C)ontra|(d|D)esde|(e|E)n|(e|E)ntre|
        (h|H)acia|(h|H)asta|(p|P)ara|(p|P)or|(s|S)egún|(s|S)in|(s|S)obre|(t|T
        )ras)$', 'IN'),
    (u'^(el|El|EL|Los|los|la|La|Las|las|del)$', 'DT'),
    (u'^(éste|Éste|ésta|Ésta|esto|Esto|éstos|Éstos|éstas|Éstas|ése|Ése|ésa|És
    a|eso|Eso|ésos|Ésos|ésas|Ésas|áquel|Áquel|áquella|Áquella|aquello|Aquello
    |áquellos|Áquellos|áquellas|Áquellas)$', 'PRP'),
    (u'^(este|Este|esta|Esta|esto|Esto|estos|Estos|estas|Estas|ese|Ese|esa|Es
    a|eso|Eso|esos|Esos|esas|Esas|aquel|Aquel|aquella|Aquella|aquello|Aquello
    |aquellos|Aquellos|aquellas|Aquellas)$', 'DT'),
    (u'^(yo|Yo|tú|Tú|tu|él|Él|Ella|ella|nosotros|Nosotros)$', 'PRP'),
    (u'^(Así|Asi|asi|así|más|Más)$','RB'),
    (u'^(Así|Asi|asi|así|más|Más)$','RB'),
    (u'^(Y|y|o|O|U|u|ni|Ni|ya|Ya)$','CC'),
    (u'^(véase)$','VR')

    ]
    regexp_tagger = nltk.RegexpTagger(patterns)
    for index,t in enumerate(tags):
        newtag=regexp_tagger.tag(nltk.word_tokenize(t))
        if newtag[0][1]!=None:
            t=newtag[0]
            tags[index]=newtag[0]
        else:
            if index+1<len(tags):
                tTra=self.traducir(t)
            else:
                tTra=self.traducir(t)
            nuevotag=pos_tag(nltk.word_tokenize(tTra))[0]
            nuevotag=(t,nuevotag[1])
            tags[index]=nuevotag
    return tags
```

Figura 13. Script de la función tagearSentenciaEs del servicio web tokenizar

- La función recibe el texto en español, en la variable ***tags***.

- Se descompone el texto en tokens con la librería ***word_tokenice***, esta lista se la almacena en la variable ***tags***.
- En la variable ***patterns*** se definen patrones para el etiquetado, que se los realizo con expresiones regulares.
- Con los patrones definidos en la variable ***patterns***, se utiliza la librería de NLTK ***nltk.RegexpTagger***, la cual clasifica cada palabra encontrada en el texto, con una etiqueta definida en la variable ***patterns***.
- Si existe una palabra que no se haya etiquetado, se envía a la función ***traducir***, con el token traducido a ingles se lo etiqueta con la librería de NLTK ***pos_tag***, y se asigna la etiqueta del token en inglés, al token en español, y se lo guarda en la variable ***tags***, que contienen la lista de tokens con etiquetas.
- Esta función devuelve como parámetro de salida la variable **tags**.

5.1.2.2.3. TagearSentenciaEn

```
def tagearSentenciaEn(self,tags):
        tags=word_tokenize(tags)
        tags=pos_tag(tags)

    return tags
```

Figura 14. Script de la función tagearSentenciaEn del servicio web tokenizar

- La función ***tagearSentenciaEn*** recibe el texto en inglés, en la variable ***tags***.
- Puesto que el texto está en inglés y NLTK está especializado en este idioma, resulta más sencillo llevar a cabo el etiquetado.
- Con ***word_tokenize*** separa las pablas en tokens y con ***pos_tag*** etiqueta estos tokens, y la lista de estos se los asigna el la variable ***tags***.
- Esta función devuelve como parámetro de salida la variable **tags**

5.1.2.2.4. Traducir

En la siguiente figura se encuentra el script de la función traducir.

```
def traducir(self,token):
    #print token
    if len(token)>2:
        tokenTra=gs.translate(token, 'en')
    else:
        tokenTra= gs.translate(token+' '+token+' '+token+' '+token, 'en')
        tokenTra=word_tokenize(tokenTra)
        tokenTra=tokenTra[0]
    tokenTra=word_tokenize(tokenTra)
    tokenTra=tokenTra[len(tokenTra)-1]
    if token.islower()==True:
        tokenTra=tokenTra.lower()
    return tokenTra
```

Figura 15. Script de la función traducir del servicio web tokenizar

- Esta función recibe como parámetro el token en español en la variable ***token***.
- Y con la librería goslate (***gs***) se lo traduce a inglés y se lo almacena en la variable ***tonkenTra***.
- La función devuelve como salida la variable ***tokenTra***.

5.2. Servicio web extracción de entidades

Este servicio web realiza la extracción de entidades. Para la elaboración de este prototipo se realizó lo siguiente:

- El texto introducido se lo envía a descomponer con la función sent_tokenize de la librería NLTK, luego se lo envía a la función principal del servicio anterior (servicio web tokenizar) obteniendo una lista con los tokens etiquetados.
- Utilizando la función *RegexpParser*(función de NLTK) se realiza un conjunto de expresiones regulares para la extracción de las entidades en base a las etiquetas recuperadas.
- Con la función *chunker* en fusión con expresiones regulares realizadas, se extraen las entidades.

5.2.1. Arquitectura

5.2.1.1. Diagrama de secuencia

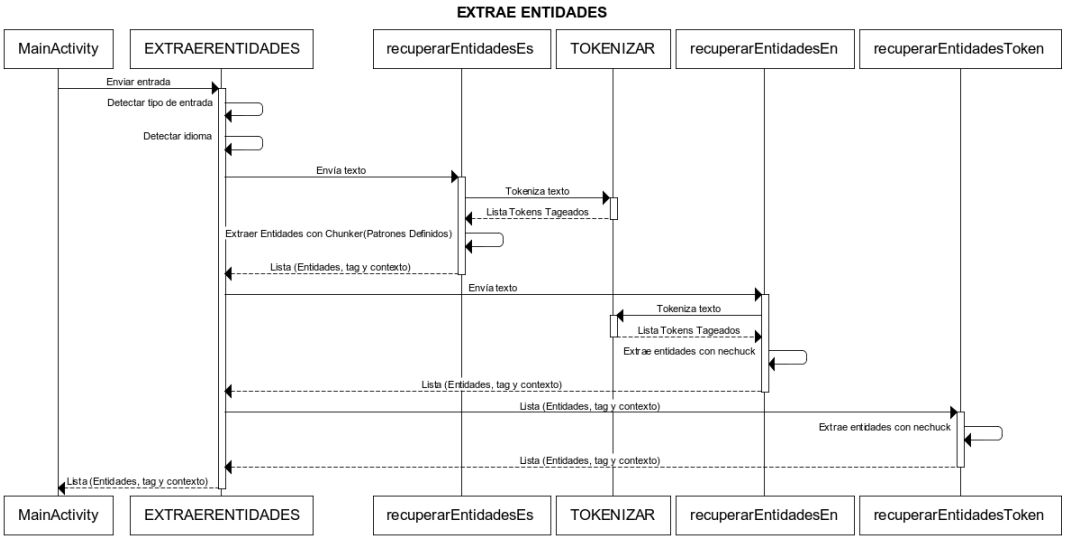

Figura 16 Diagrama de secuencia de la funcionalidad Extrae Entidades

A continuación se describe su comportamiento:

1. Se llama el servicio web ***extraerentidades*** y recibe como parámetro de entrada el texto.

2. Al recibir la entrada se detecta el tipo y si es un texto se detecta el idioma(pueden ser español o inglés)
3. Si el idioma detectado es el español se envía el texto a la función ***recuperarEntidadeEs***.
4. La función ***recuperarEntidadesEs*** recibe el texto y lo envía a la funcionalidad tagear del servicio web ***tokenizar*** la cual nos devuelve una lista de tokens etiquetados.
5. Luego de recibir los la lista de tokens etiquetados se procede a extraer las entidades con chunker una funcionalidad de NLTK definiendo patrones para la extracción. Y devolvemos una lista de las entidades extraídas con respectivas etiquetas y con su contexto de donde se extrajo la entidad.
6. Si el idioma detectado fue el inglés se procede a enviar el texto a la función ***recuperarEntidadesEn***, y esta la envía a la funcionalidad tagear del servicio web ***tokenizar***
7. Luego de recibir la lista con los tokens etiquetados de la funcionalidad, extrae las entidades utilizando la función de NLTK ***nechunk***. Y se devuelve una lista de las entidades extraídas con respectivas etiquetas y con su contexto de donde se extrajo la entidad.
8. El servicio Web devuelve un json con la lista de las entidades extraídas con respectivas etiquetas y con su contexto de donde se extrajo la entidad.

5.2.2. Implementación

5.2.2.1. Parámetros.

Como parámetro de entrada se tiene:

sentence, esta variable puede ser de tipo carácter o string que almacena el texto a ser analizados y procesados.

Como parámetro de salida:

- *Json, que contiene* una lista desglosada mediante PLN, en la cual se encuentra las entidades extraídas:

```
{
  "entidades": [
    {
      "contexto": "The second half course covers Virgil
Vandals ; major social , economic , political religious
trends Rome provinces .",
      "entidad": [
        [
          "course",
          "NN"
        ]
      ]
    },
    {
      "contexto": "The second half course covers Virgil
Vandals ; major social , economic , political religious
trends Rome provinces .",
      "entidad": [
        [
          "Virgil",
          "NNP"
        ]
      ]
    },
    {
      "contexto": "The second half course covers Virgil
Vandals ; major social , economic , political religious
trends Rome provinces .",
      "entidad": [
        [
          "Vandals",
          "NNS"
        ]
      ]
    },
```


Figura 17 Salida del servicio web de extracción de entidades

5.2.2.2. Funciones.

Esta clase implementa funciones para:

Tabla 2. Funciones utilizadas en el servicio web extracción de entidades

FUNCIÓN	PARÁMETROS		DESCRIPCIÓN
ExtEntidades Función principal	Entrada	Entrada	El servicio web recibe el parámetro de entrada sentencia y lo envía a esta función que lo recibe como Entrada
	Salida	Lista	En este parámetro esta la lista con las la entidades extraídas y su contexto de donde se las extrajo.
recuperarEntidadesEs	Entrada	texto	Esta variable recibe el contenido para ser procesado, y extraer las entidades existentes en el contenido.
	Salida	Lista2	Devuelve una lista con los las entidades extraídas y su contexto.
recuperarEntidadesEn	Entrada	texto	Esta variable recibe el contenido para ser procesado, y extraer las entidades existentes en el contenido.

	Salida	Lista2	Devuelve una lista con las entidades extraídas y su contexto.
recuperarEntidadesToken	Entrada	tokens	En esta variable recibe en una lista con los tokens ya etiquetados.
	Salida	Lista2	Devuelve una lista con las entidades extraídas y su contexto.

5.2.2.2.1. ExtEntidades

```
def ExtEntidades(self,Entrada):
    Lista=[]
    if type(Entrada) is str:
        try:
            gs= goslate.Goslate()
            idioma=gs.detect(Entrada)
        except Exception, e:
            idioma='en'

        lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Albanian",

        idiomaCompleto=lenguajes[idioma]

        if idioma =='es':
            Lista=self.recuperarEntidadesEs(Entrada)
        elif idioma == 'en':
            Lista=self.recuperarEntidadesEn(Entrada)
        else:
            mensaje='El idioma %s no es soportado'%idiomaCompleto
            print mensaje
            mensaje=gs.translate(mensaje, idioma)
            print mensaje
            Lista=[[mensaje,mensaje]]
    elif type(Entrada) is list:
        Lista=self.recuperarEntidadesToken(Entrada)
    else:
        mensaje='Tipo de datos no soportados'
        print mensaje
        Lista=[[mensaje,mensaje]]
    return Lista
```

Figura 18. Script de la función ExtEntidades del servicio web extraerentidades

- Esta función recibe como entrada el texto ingresado o también puede tener como entrada una lista (que contenga tokens etiquetados) que se almacena en la variable ***Entrada***.
- Se detecta si la entrada es una lista o un texto.
- Si es un texto se procede a identificar el idioma del texto mediante la librería goslate (gs), para determinar el lenguaje en que se encuentra el texto ingresado.
- Si el texto está en español se lo envía a la función ***recuperarEntidadesEs,*** a esta función se le envía la variable ***Entrada*** y nos devuelve una lista con las entidades encontradas, con su contexto, y se lo asigna a la variable ***Lista***.
- En cambio si el texto está en ingles se lo envía a la función ***recuperarEntidadesEn,*** a la cual se le envía la variable ***Entrada*** y nos devuelve una lista con las entidades encontradas, con su contexto, y se lo asigna a la variable ***Lista***.

- Y si el idioma del texto no es ni español, ni inglés, se almacena en la variable ***Lista*** el mensaje de lenguaje no soportado.
- Pero si la entrada a esta función no es ni texto, ni una lista, se almacena en la variable ***Lista*** el mensaje de tipo de datos no soportado.
- Por ultimo esta función nos da como salida la variable ***Lista***.

5.2.2.2.2. recuperarEntidadesEs

```
def recuperarEntidadesEs(self,texto):
    chunker = RegexpParser("""
    ENTI:
        {<NNP|NNPS>+<NNP|NNPS>}
        {<NNP|NNPS><IN|DT><NNP|NNPS|NN|NNS>}
        {<NN|NNS>*<NN|NNS>}
        {<NNP|NNPS>}
    ENTIDACOMP:

        {<ENTI|ENTIDACOMP><JJ><IN><ENTI|ENTIDACOMP>}
        {<ENTI|ENTIDACOMP><IN><ENTI|ENTIDACOMP>}
        {<ENTI|ENTIDACOMP><IN><ENTI|ENTIDACOMP><IN><ENTI|ENTIDACOMP>}
    ENTIDACOMP2:
        {<ENTI|ENTIDACOMP><IN><ENTI|ENTIDACOMP>}
    FECHA:
        {<LS|CD><IN><ENTI><DT><LS|CD>}
        {<LS|CD><IN><ENTI>}
        {<ENTI><DT><LS|CD>}
        {<ENTI><LS|CD>}
    """)

    ObjTag = Tokenizar()
    Lista = []
    Lista2 = []
    for sentence in sent_tokenize(texto.decode('utf-8')):
        try:
            tags=ObjTag.tagear(sentence.encode('utf-8'))
        except Exception, e:
            print e
            continue

        tagsentX=word_tokenize(sentence)
        filtered_words = ' '.join(w.encode('utf-8') for w in tagsentX if not
            w in nltk.corpus.stopwords.words('spanish'))
        try:
            parsed = chunker.parse(tags)
        except Exception, e:
            print e
            continue

        for chunk in parsed:
            if hasattr(chunk, 'label'):
                Lista2.append([chunk.leaves(),filtered_words])
                Lista.append (' '.join(c[0] for c in chunk.leaves()))
    return Lista2
```

Figura 19. Script de la función ExtEntidades del servicio web extraerentidades

- La función recibe el texto en español, en la variable ***texto***.
- En la variable ***chunker*** se definen patrones, para la extracción de las entidades.

- Se descompone el texto en oraciones con la librería ***sent_tokenize***, y a esas oraciones se la envía a la función ***tagear*** del servicio web ***tokenizar***, el cual devuelve las oraciones descompuestas en tokens.
- En la variable ***filtered_words*** se guarda las palabras del contexto exceptuando los stopwords (que son las palabras sin significado en el texto como artículos, pronombres, preposiciones)
- Con la utilidad de NLTK y utilizando los patrones para la extracción de entidades de la variable ***chunker*** clasificamos las palabras para encontrar las entidades y lo asignamos a la variable ***parsed***.
- Descomponemos y buscamos las entidades, para guardarlas en la variable ***Lista2***.
- Esta función devuelve la lista de entidades encontradas, con su contexto, que se asignó en la variable ***Lista2***.

5.2.2.2.3. recuperarEntidadesEn

```
def recuperarEntidadesEn(self,texto):
    ObjTag = Tokenizar()
    Lista = []
    Lista2= []

    for sentence in sent_tokenize(texto.decode('utf-8')):
        try:
            tags=ObjTag.tagear(sentence.encode('utf-8'))
        except Exception, e:
            print e

        tagsentX=word_tokenize(sentence)
        filtered_words = ' '.join(w for w in tagsentX if not w in nltk.corpus
            .stopwords.words('english'))

        grammar ="""
            NBAR:
                {<NN.*|JJ>*<NN.*>}  # Nouns and Adjectives, terminated with
                    Nouns
             NP:
                  {<NBAR>}
                  {<NBAR><IN><NBAR>}  # Above, connected with in/of/etc...
            """
        chunker = nltk.RegexpParser(grammar)
        tags=ObjTag.tagear(sentence)
        parsed = chunker.parse(tags)

        for chunk in parsed:
            if hasattr(chunk, 'label'):
                Lista2.append([chunk.leaves(),filtered_words])
                Lista.append (' '.join(c[0] for c in chunk.leaves()))
    return Lista2
```

Figura 20. Script de la función recuperarEntidadesEn del servicio web extraerentidades

- Esta función recibe el texto en inglés, en la variable texto.

- Se descompone el texto en oraciones con la librería ***sent_tokenize***, y a esas oraciones se la envía a la función ***tagear*** del servicio web ***tokenizar***, el cual devuelve las oraciones descompuestas en tokens.
- En la variable ***filtered_words*** se guarda las palabras del contexto exceptuando los stopwords (que son las palabras sin significado en el texto como artículos, pronombres, preposiciones)
- En la variable ***chunker*** se definen patrones, para la extracción de las entidades en inglés.
- Con la utilidad de NLTK y utilizando los patrones para la extracción de entidades de la variable ***chunker*** clasificamos las palabras para encontrar las entidades y lo asignamos a la variable ***parsed***.
- Descomponemos y buscamos las entidades, para guardarlas en la variable ***Lista2***.
- Esta función devuelve la lista de entidades encontradas, con su contexto, que se asignó en la variable ***Lista2***

5.2.2.2.4. recuperarEntidadesToken

```
def recuperarEntidadesToken(self,tokens):
    Lista = []
    Lista2 = []
    sentence= ' '.join(n[0] for n in tokens)
    parsed = ne_chunk(tokens)

    for chunk in parsed:
        if hasattr(chunk, 'node'):
            Lista2.append([chunk.leaves(),sentence])
            Lista.append (' '.join(c[0] for c in chunk.leaves()))
    return Lista2
```

Figura 21. Script de la función recuperarEntidadesToken del servicio web extraerentidades

- Esta función recibe una lista que contenga tokens con sus etiquetas, en la variable ***tokens***.
- En la variable ***sentence*** se guarda las palabras del contexto.
- Con la utilidad de NLTK ne_chunk clasificamos las palabras para encontrar las entidades y lo asignamos a la variable ***parsed***.
- Descomponemos y buscamos las entidades, para guardarlas en la variable ***Lista2***.
- Esta función devuelve la lista de entidades encontradas, con su contexto, que se asignó en la variable ***Lista2.***

5.3. Servicio web desambiguación y enlace.

Este servicio web realiza la extracción de entidades. Para la elaboración de este prototipo se realizó lo siguiente:

- El texto introducido se lo envía a la funcionalidad principal del servicio web anterior (servicio web extraer entidades) obteniendo una lista con las entidades extraídas.
- Utilizando consultas SPARQL se realiza consultas a la Dbpedia para desambiguar y extraer Uris de los recursos para enlazarlos con estos.

5.3.1. Arquitectura.

5.3.1.1. Diagrama de secuencia

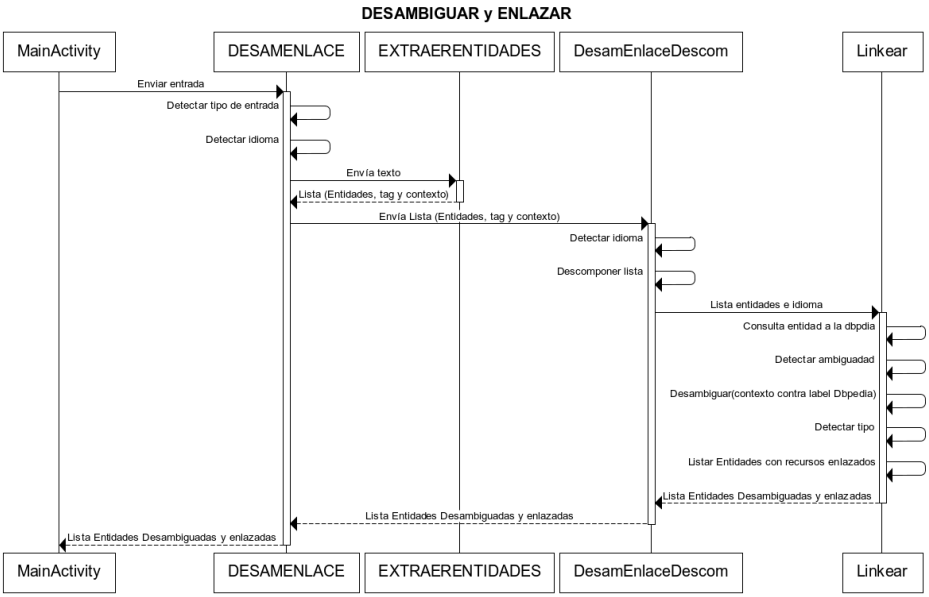

Figura 22 Diagrama de secuencia de la funcionalidad Desambiguar y Enlazar

A continuación se describe su comportamiento:

1. Se llama el servicio web ***desambiguarenlazar*** y, recibe como parámetro de entrada un texto.
2. Al recibir la entrada se detecta el tipo y si es un texto se detecta el idioma (pueden ser español o inglés).
3. Si el idioma detectado es el español o inglés se envía el texto a la funcionalidad **extraerentidades**.
4. La funcionalidad **extraerentidades** del servicio web anterior recibe el texto y nos devuelve una lista de las entidades extraídas con respectivas etiquetas y con su contexto de donde se extrajo la entidad.

5. Una vez teniendo la lista de las entidades se la envía a **DesamEnlaceDescom** y esta función detecta el idioma y descompone la lista para enviarla a la función **linkear**.
6. La función **linkear** recibe la lista y realiza consultas a la Dbpedia de las entidades.
7. Luego detecta si hay ambigüedad en la consulta de la entidad y desambigua con el contexto de la entidad contra los datos de la Dbpedia. Y procede a enlazar los datos con los recursos de la Dbpedia.
8. Posteriormente detecta el tipo de entidad y lista los enlaces a los recursos ordenándolos desacuerdo a la relevancia.
9. Y se devuelve la lista de las entidades Desambiguadas y enlazadas.
10. El servicio web devuelve un json con dicha lista que contiene las entidades desambiguados con los enlaces a los recursos de la Dbpedia.

5.3.2. Implementación.

5.3.2.1. Parámetros.

Como parámetro de entrada se tiene:

- sentence, esta variable puede ser de tipo string que almacena el texto a ser analizados y procesados.

Como parámetro de salida:

- Json, que contendrá una lista desglosada, en la cual se encuentra las entidades extraídas, desambiguadas y enlazadas:

```
{
  "enlace": [
    {
      "enlaces": [
        {
          "enlace": "http://dbpedia.org/resource/Virgil",
          "frec": 3,
          "tipo": "AGT"
        },
        {
          "enlace": "http://dbpedia.org/resource/Virgil,_New_York",
          "frec": 2,
          "tipo": "GEO"
        }
      ],
      "entidad": "Virgil",
      "link": "http://dbpedia.org/resource/Virgil",
      "tipo": "AGT"
    },
    {
      "enlaces": [
        {
          "enlace": "http://dbpedia.org/resource/Provinces",
          "frec": 1,
          "tipo": "Sin Tipo"
        }
      ],
      "entidad": "Provinces",
      "link": "http://dbpedia.org/resource/Provinces",
      "tipo": "Sin Tipo"
    }
  ]
}
```


Figura 23 Salida del servicio web de desambiguar y enlazar

5.3.2.2. Funciones.

Esta clase implementa funciones para:

Tabla 3. Funciones utilizadas en el servicio web desambiguación y enlace

FUNCIÓN	PARÁMETROS		DESCRIPCIÓN
DesamEnlace Función principal	Entrada	Entrada	El servicio web recibe el parámetro de entrada sentencia y lo envía a esta función que lo recibe en la variable Entrada
	Salida	Lista	En este parámetro esta la lista con las la entidades extraídas, desambiguadas y con un enlace a un recurso de la Dbpedia.
DesamEnlaceDescom	Entrada	Entrada	Esta variable recibe la lista con las entidades extraídas con su contexto.
	Salida	DesamEnl	Devuelve una lista con los las entidades desambiguadas y enlazadas.
Linkear	Entrada	entidad	Esta variable recibe una entidad de la lista de entidades.
		contexto	En esta variable, se recibe el contexto de la entidad.

		idioma	El idioma en el que esta la entidad extraída, se almacena en esta variable.
	Salida	LisDesyEnlase	Devuelve la entidad desambiguada y enlazada a un recurso de la Dbpedia, así como también una lista de otros posibles enlaces a los recursos de la Dbpedia.
SelecionaTipo	Entrada	listaTipos	Esta variable recibe una lista con los posibles enlaces a la Dbpedia de una determinada entidad.
		contexto	En esta variable recibe el contexto de una determinada entidad.
	Salida	ListaOrdenado	Devuelve una lista con los enlaces a la Dbpedia, ordenados según la relevancia que tengan con la entidad y su contexto.
eliminasignos	Entrada	text	Recibe un carácter o string
	Salida	text	Devuelve el carácter o string sin signos de puntuación.
TipoEntidad	Entrada	tiposent	Esta variable una lista con enlaces a la dbpedia con relación al tipo de entidad.
	Salida	TipoFinal	Devuelve un string con el nombre del tipo de entidad que se determinó luego de procesarlo.
ConsuDbpediaExtraLabel	Entrada	link	Esta variable recibe un link de una entidad de la Dbpedia para hacer la consulta.
		predicado	Recibe un string con el predicado que se desea hacer la consulta SPARQL (ejemplo rdfs:comment)
	Salida	sparql	Json de la consulta SPARQL.
ConsuDbpedia3	Entrada	link	Esta variable recibe un link de una entidad de la Dbpedia para hacer la consulta.
		predicado	Recibe un string con el predicado que se desea hacer la consulta SPARQL (ejemplo rdfs:comment)
	Salida	sparql	Json de la consulta SPARQL.
ConsuDbpedia2	Entrada	link	Esta variable recibe un link de una entidad de la Dbpedia para hacer la consulta.
		predicado	Recibe un string con el predicado que se desea hacer la consulta SPARQL (ejemplo rdfs:comment)
	Salida	sparql	Json de la consulta SPARQL.

ConsuDbpedia	Entrada	entidades	Esta variable recibe la entidad que se desea buscar en la Dbpediapara la consulta SPARQL.
		predicado	Recibe un string con el predicado que se desea hacer la consulta SPARQL (ejemplo rdfs:comment)
		idioma	Recibe en esta variable el idioma en el que esta la entidad.
	Salida	sparql	Json de la consulta SPARQL.

5.3.2.2.1. DesamEnlace

```
def DesamEnlace(self,Entrada):
    DesamEnl=[]
    if type(Entrada) is str:
        try:
            gs= goslate.Goslate()
            idioma=gs.detect(Entrada)
        except Exception, e:
            idioma='en'

        lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Albanian",

        idiomaCompleto=lenguajes[idioma]

        if idioma =='es' or idioma == 'en':
            Entrada=ObjExt.ExtEntidades(Entrada)
            DesamEnl=self.DesamEnlaceDescom(Entrada)
        else:
            mensaje='El idioma %s no es soportado'%idiomaCompleto
            mensaje=gs.translate(mensaje, idioma)
            DesamEnl=[[mensaje,mensaje]]

    elif type(Entrada) is list:
        DesamEnl=self.DesamEnlaceDescom(Entrada)
    else:
        mensaje='Tipo de datos no soportados'
        DesamEnl=[[mensaje,mensaje]]

    return DesamEnl
```

Figura 24. Script de la función DesamEnlacedel servicio web desmbiguarenlazar

- Esta función recibe como entrada el texto ingresado que se almacena en la variable ***Entrada***.
- Si la entrada es de in tipo string, se precede a detectar el idioma.
- Para detectar el idioma del texto, selo hace mediante la librería goslate (gs), para determinar el lenguaje en que se encuentra escrito el texto ingresado.
- Si el texto está en español o inglés de lo envía a la funcionalidad ***ExtEntidades*** del servicio web extraer entidades, esta función nos devuelve una lista con las entidades encontradas y su contexto, y lo almacenamos en la variable ***Entrada***.

- Luego enviamos la variable ***Entrada*** que contiene la lista de entidades a la función ***DesamElnaceDescom***, que nos devuelve una lista con las entidades desambiguadas y con enlace a recurso de la Dbpedia.
- La lista que devuelve la función ***DesamElnaceDescom***, se la almacena en la variable ***DesamEnl***
- Por otro lado si el texto no está ni en español ni en inglés, se almacena en la variable ***DesamEnl*** el mensaje de lenguaje no soportado.
- Si la entrada contiene ya una lista con entidades y su contexto, se lo envía directamente a la función ***DesamElnaceDescom*** y se almacena en la variable DesamEnl el resultado.
- Y si la entrada no es ni string, ni la lista antes mencionada, se almacena en la variable ***DesamEnl*** el mensaje de tipo de datos no soportado.
- Por ultimo esta función nos da como salida la variable ***DesamEnl***.

5.3.2.2.2. DesamEnlaceDescom

```
def DesamEnlaceDescom(self,Entrada):
    DesamEnl=[]
    paraidioma=' '.join(entidad[1] for entidad in Entrada)

    gs = goslate.Goslate()
    idioma=gs.detect(paraidioma)
    for entidad in Entrada:
        entidades=' '.join(c[0] for c in entidad[0])

        contexto= entidad[1]
        try:
            link=self.Linkear(entidades.capitalize(),contexto,idioma)
        except Exception, e:
            print e
        DesamEnl.append(link)

    return DesamEnl
```

Figura 25. Script de la función DesamEnlaceDescom del servicio web desmbiguarenlaza

- Esta función recibe una lista con las entidades ya extraídas y con su contexto, en la variable ***Entrada***.
- En la variable ***paraidioma*** almacenamos el contexto de las entidades, para luego detectar el idioma con la librería goslate (gs) y almacenar el idioma en la variable ***idoma***.
- Se separa la lista de entidades, que está en la variable ***Entrada*** y se envía cada entidad con su contexto a la función ***Linkear***, así como también se le envía el ***contexto*** y el ***idioma***.
- El resultado de la función ***Linkear***, se lo almacena en la variable ***link***, y se lo agrega a la lista llamada ***DesamEnl***.

- Por ultimo esta función nos da como salida la variable ***DesamEnl***.

5.3.2.2.3. Linkear

```
def Linkear(self,entidad,contexto,idioma):
    linkear=[]
    try:
        results = self.ConsuDbpedia(entidad,'rdfs:label','en')
        results2= self.ConsuDbpedia(entidad,'rdfs:label','es')
        results["results"]["bindings"].extend(results2["results"]["bindings"]
            )
    except Exception, e:
        print e
        return [entidad,linkear]
    UrlyTipo=[]
    bandera=False
    cont=0
    for result in results["results"]["bindings"]:
        link=(result["label"]["value"])
        if 'Category' in link:
            continue
        try:
            results2 = self.ConsuDbpedia2(link,'rdf:type')
        except Exception, e:
            continue
        results2 = results2["results"]["bindings"]
        if results2 ==[]:
            results3=self.ConsuDbpedia3(link,'dbpedia-
                owl:wikiPageDisambiguates')
            results3=results3["results"]["bindings"]
            if results3!=[]:
                for result3 in results3:
                    link=(result3["label"]["value"])
                    if 'Category:' in link:
                        continue

                    UrlyTipo.append(link)
                break
            else:
                TipoFinal=self.TipoEntidad(results2)
                linkear=[[1,link,TipoFinal]]
                bandera=True
        else:
            for result2 in results["results"]["bindings"]:
                link=(result2["label"]["value"])
                if 'Category:' in link:
                        continue
                UrlyTipo.append(link)

    if bandera==False:
        UrlyTipo=list(set(UrlyTipo))
        linkear=self.SelecionaTipo(UrlyTipo,contexto)
    LisDesyEnlace=[entidad,linkear]
    return LisDesyEnlace
```

Figura 26. Script de la función Linkear del servicio web desmbiguarenlazar

- La función recibe como parámetro de entrada ***entidad*** que almacena la entidad a ser procesada, ***contexto*** que almacena el contexto de la entidad e idioma que almacena el ***idioma*** en el que está el contexto y la entidad.

- Se hace consultas a la Dbpedia, enviando la entidad a la función ConsuDbpedia, enviando también el predicado rdfs:label, y el idioma.
- La función ***ConsuDbpedia*** devuelve un Json con los enlaces a la Dbpedia que contiene similitud a la entidad, y se lo amacena en la variable ***results*** y ***results2***, para combinarlo en uno solo que sería ***results.***
- Se recorre todos los enlaces a la Dbpedia contenidos en la variable ***results***["results"]["bindings"], y cada enlace se lo almacena en la variable ***link.***
- Luego con la función ***ConsuDbpedia2*** se determina qué tipo de recurso es el que está apuntando el enlace almacenado en la variable ***link***.
- Si no se tiene un tipo de recurso determinado, es posible que sea un recurso ambiguo, por lo que se utiliza la función ***ConsuDbpedia3***, enviando el predicado dbpedia-owl:wikiPageDisambiguates para saber si tiene ese atributo el recurso al que apunta el enlace; y si lo tienen se agrega el resultado de la función (una lista de enlaces que apuntan a recursos similares) a una variable tipo lista llamada ***UrlyTipo***.
- Y si no tiene el predicado dbpedia-owl:wikiPageDisambiguates, se sigue recorriendo los enlaces que se guardan en la variable link y agregándolos a la variable tipo lista llamado ***UrlyTipo.***
- Al final se envía la variable ***UrlyTipo***, con el ***contexto*** de la entidad que se recibió, a la función ***SelecionaTipo***, para desambiguar, y se lo almacena en la variable ***linkear***.
- Esta función tiene como salida la ***variableLisDesyEnlace***, que contiene la ***entidad*** y los enlaces a la Dbpedia desambiguados que se encuentra en la variable ***linkear***.

5.3.2.2.4. SelecionaTipo

```
def SelecionaTipo(self,listaTipos,contexto):
    listafrec=[]
    for link in listaTipos:
        try:
            LabelsExtraidos=self.ConsuDbpediaExtraLabel(link,'rdfs:comment')
        except Exception, e:
            continue

        LabelsExtraidos=LabelsExtraidos["results"]["bindings"]
        labels=(' '.join(self.eliminasignos(c["label"]["value"]) for c in
            LabelsExtraidos)).lower().split(' ')
        labelsSinDuplicados=list(set(labels))
        count=0

        for label in labelsSinDuplicados:
            if label in self.eliminasignos(contexto.lower()).split(' ') and
                label!='':
                count=count+1
        results4 = self.ConsuDbpedia2(link,'rdf:type')
        results4 = results4["results"]["bindings"]
        tipo=self.TipoEntidad(results4)
        listafrec.append([count,link,tipo])
    ListaOrdenado= sorted(listafrec,key=lambda x:x[0], reverse=True)
    return ListaOrdenado
```

Figura 27. Script de la función SelecionaTipo del servicio web desmbiguarenlazar

- A este método llegan como paramentos de entrada la lista de los recursos de la entidad ambigua y el contexto de dicha entidad.
- Para proceder a determinar cuál es el recurso valido o más cercano, para esto nos servimos de otra consulta enviando a la función ***ConsuDbpediaExtraLabel*** para extraer los rdfs:comement de los recursos, y almacenado el resultado en la variable ***LabelsExtraidos***.
- Se procede a hacer una comparación con el ***contexto*** de la entidad y los rdfs:comement de los recursos almacenado en la variable ***LabelsExtraidos,*** y de esta manera ordenarlos en una lista de acuerdo al que tenga más concordancia en dicha comparación.
- El método devuelve la lista ordenada de los recursos encontrados y las entidades que sealmacena en la variable ***ListaOrdenado***..

5.3.2.2.5. Eliminasignos

```
def eliminasignos(self,text ):
        return re.sub('[%s]' % re.escape(string.punctuation), '', text)
```

Figura 28. Script de la función Eliminasignos del servicio web desmbiguarenlazar

- Esta función recibe un parámetro tipo string y lo amacena en la variable text.
- La función devuelve el texto o string, eliminado los signos de puntuación.

5.3.2.2.6. TipoEntidad

```
def TipoEntidad(self,tiposent):
    TipoFinal='Sin Tipo'
    tipoMeclado=''
    for result in tiposent:
        Tipo=(result["label"]["value"])
        TipoEx= Tipo.split('/')
        TipoEx=TipoEx[len(TipoEx)-1]
        if tipoMeclado=='':
            tipoMeclado=TipoEx
        else:
            tipoMeclado=tipoMeclado+'-'+TipoEx
        if TipoEx in ['City','Place','PopulatedPlace', 'Settlement', 'Country
            ']:
            TipoFinal='GEO'
        if TipoEx in ['Agent']:
            TipoFinal='AGT'
        elif TipoEx in ['Organisation']:
            TipoFinal='ORG'
        elif TipoEx in ['Person']:
            TipoFinal='PER'

    if tipoMeclado!='' and TipoFinal=='Sin Tipo':
        TipoFinal=tipoMeclado
    return TipoFinal
```

Figura 29. Script de la función TipoEntidad del servicio web desmbiguarenlazar

- Esta función recibe una lista con los tipos de un determinado recurso de la Dbpedia, en la variable ***tiposent.***
- Se recorre cada link para determinar a qué tipo pertenece, puede ser Geo (geográfico), Agt (agente), Org (organización), Per (persona), el tipo se lo almacena en la variable ***TipoFinal***.
- Esta función tiene como salida la variable ***TipoFinal***.

5.3.2.2.7. ConsuDbpediaExtraLabel

```
def ConsuDbpediaExtraLabel(self,link,predicado):
    sparql = SPARQLWrapper("http://apolo.utpl.edu.ec/vtitanio/sparql")
    sparql.setQuery("""
    SELECT *
    WHERE {
    <%s> %s ?label
    }
    """%(link,predicado))
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()
```

Figura 30. Script de la función ConsuDbpediaExtraLabel del servicio web desmbiguarenlazar

- Esta función recibe como parámetro de entrada un enlace a un recurso, y un predicado para realizarla consulta SPARQL.
- Con la librería ***SPARQLWrapper*** se realiza la consulta usando como sujeto el enlace almacenado en la variable ***link,*** y como predicado la variable ***predicado***.

- La salida de esta función es el Json resultante de la consulta.

5.3.2.2.8. ConsuDbpedia3

```
def ConsuDbpedia3(self,linkear,predicado):
    sparql = SPARQLWrapper("http://apolo.utpl.edu.ec/vtitanio/sparql")
    sparql.setQuery("""
    SELECT *
    WHERE {
    <%s> %s ?label
    FILTER regex(str(?label), "http://dbpedia.org/resource/", "i")
    }
    """%(link,predicado))
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()
```

Figura 31. Script de la función ConsuDbpedia3 del servicio web desmbiguarenlazar

- Esta función recibe como parámetro de entrada un enlace a un recurso, y un predicado para realizarla consulta SPARQL.
- Con la librería ***SPARQLWrapper*** se realiza la consulta usando como sujeto el enlace almacenado en la variable ***link,*** y como predicado la variable ***predicado***.
- También la consulta tiene un filtro para que el resultado o el objeto contenga http://dbpedia.org/resource/
- La salida de esta función es el Json resultante de la consulta.

5.3.2.2.9. ConsuDbpedia2

```
def ConsuDbpedia2(self,link,predicado):
    sparql = SPARQLWrapper("http://apolo.utpl.edu.ec/vtitanio/sparql")
    sparql.setQuery("""
    SELECT *
    WHERE {
    <%s> %s ?label
    FILTER regex(str(?label), "http://dbpedia.org/ontology/", "i")
    }
    """%(link,predicado))
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()
```

Figura 32. Script de la función ConsuDbpedia2 del servicio web desmbiguarenlazar

- Esta función recibe como parámetro de entrada un enlace a un recurso, y un predicado para realizarla consulta SPARQL.
- Con la librería ***SPARQLWrapper*** se realiza la consulta usando como sujeto el enlace almacenado en la variable ***link,*** y como predicado la variable ***predicado***.
- También la consulta tiene un filtro para que el resultado o el objeto contenga http://dbpedia.org/ontology/
- La salida de esta función es el Json resultante de la consulta.

5.3.2.2.10. ConsuDbpedia

```
def ConsuDbpedia(self,entidades,predicado,idioma):
    #sparql = SPARQLWrapper("http://dbpedia.org/sparql")
    sparql = SPARQLWrapper("http://apolo.utpl.edu.ec/vtitanio/sparql")
    sparql.setQuery("""
    SELECT *
    WHERE {
    {?label %s  "%s"@%s}
    UNION{?label foaf:name "%s"@%s}
    FILTER regex(str(?label), "http://dbpedia.org/resource/", "i")
    }
    """%(predicado,entidades,idioma,entidades,idioma))

    sparql.setReturnFormat(JSON)
    return sparql.query().convert()
```

Figura 33. Script de la función ConsuDbpedia del servicio web desmbiguarenlazar

- La función recibe como parámetro de entrada la entidad a buscar almacenada en la variable ***entidades***, el ***predicado*** a usarse, y el ***idioma*** en el que esta la entidad.
- Con la librería ***SPARQLWrapper*** se realiza la consulta usando como predicado la variable ***predicado,*** y buscando la entidad almacenado en la variable ***entidades***, con su determinado ***idioma***.
- Se hace una unión para también encontrar con el predicado foaf:name, así como también un filtro para que el resultado contenga el texto http://dbpedia.org/resource/
- La salida de esta función es el Json resultante de la consulta.

5.4. App Cliente.

La app cliente es el frontal para la integración de los Servicios Web. Para la elaboración de este frontal o app cliente se realizó lo siguiente:

- El frontal se lo hizo con JavaScript y Html, consumiendo los Servicios Web realizados.
- Este frontal envía un texto a los Servicios Web y presenta en tablas los resultados así como también tiene un botón para poder descargar los resultados en un archivo json.

5.4.1. Implementación.

5.4.1.1. Parámetros.

Como parámetro de entrada se tiene:

- Una caja de texto que se puede introducir texto.

- Un combo box para seleccionar los diferentes servicios (tokenizar, extraer entidades o desambiguar y enlazar)

Como parámetro de salida:

- El resultado se presenta en una tabla, también se puede descargar el resultado en un archivo json.

En la siguiente figura se muestra el frontal o App cliente:

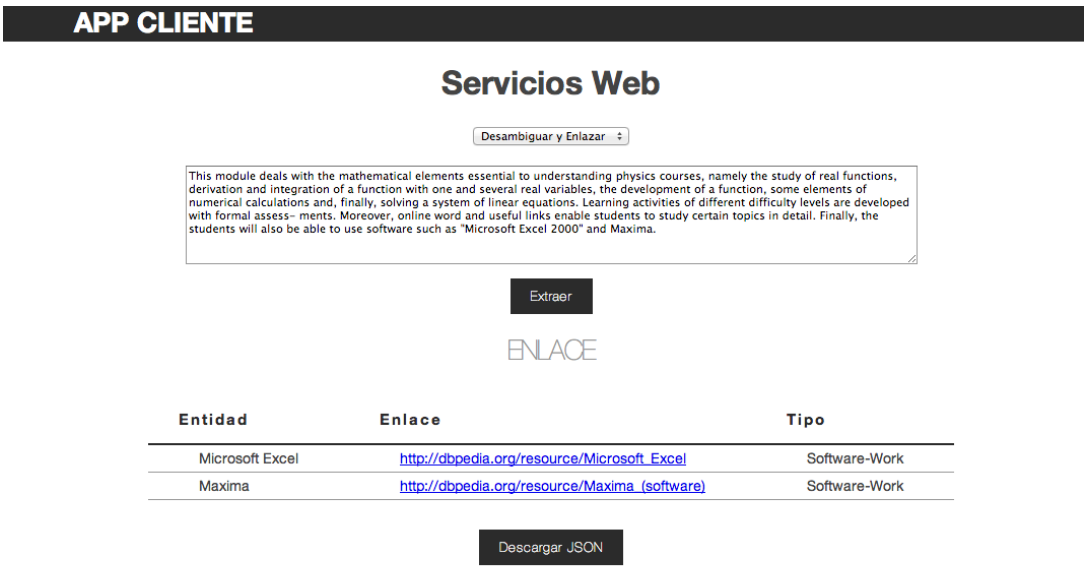

Figura 34 App cliente

DESCRIPCIÓN DE LOS SERVICIOS WEB

6.1. Apis rest

Las APIs REST que a continuación se describen nos proporcionan funciones para tokenizar, extraer entidades, desambiguar y enlazarlas las entidades a recursos de la Dbpedia, a partir de un texto.

6.1.1. Api del servicio web tokenizar

Nos devuelve una lista con el texto de entrada descompuesta en tokens, con sus respectivas etiquetas que identifican a cada token.

6.1.1.1. Url del servicio

http://taw02.utpl.edu.ec/anavisoers/tokenizar

6.1.1.2. Entradas del servicio

Los parámetros de entrada de este servicio se lo describen en la siguiente tabla:

Tabla 4. Parámetros de entrada del servicio web tokenizar

Parámetro	Descripción
sentence	Se envía en este parámetro el texto que se desea procesar. **Ejemplo**: An analysis of historical structures is presented themed sections based around construction materials. Structures from all periods of history are analyzed. The goal of the class is to provide an understanding of the preservation of historic structures for all students.

6.1.1.3. Salidas del servicio

El tipo de respuesta o salida es un Json. En la siguiente tabla se describe los atributos que contiene esta salida.

Tabla 5. Atributos del Json salida del servicio web tokenizar

Atributo	Descripción
tokens	Contiene una lista de tuplas con este formato [tokens,etiquetas]

6.1.1.4. Ejemplo de llamada

http://taw02.utpl.edu.ec/anavisoers/tokenizar?sentence=An analysis of historical structures is presented themed sections based around construction materials.

6.1.1.1. Ejemplo del resultado


```
{
  "tokens": [
    ["An","DT"],
    ["analysis","NN"],
    ["of","IN"],
    ["historical","JJ"],
    ["structures","NNS"],
    ["is","VBZ"],
    ["presented","VBN"],
    ["themed","VBN"],
    ["sections","NNS"],
    ["based","VBN"],
    ["around","IN"],
    ["construction","NN"],
    ["materials","NNS"],
    [".","."]
  ]
}
```


6.1.2. API del Servicio web extracción de entidades

Este Api procesa la entrada de texto, para devolver una lista con las entidades extraídas y su contexto.

6.1.2.1. Url del servico

http://taw02.utpl.edu.ec/anavisoers/extraerentidades

6.1.2.2. Entradas del servicio

Los parámetros de entrada de este servicio se lo describe en la siguiente tabla:

Tabla 6. Parámetros de entrada del servicio web extraer entidades

Parámetro	Descripción
sentence	Se envía en este parámetro el texto que se desea procesar. **Ejemplo**: The conquest of Italy; Roman expansion: Pyrrhus, Punic Wars and provinces; classes, courts, and the Roman revolution; Augustus and the formation of empire.

6.1.2.3. Salidas del servicio

El tipo de respuesta o salida es un Json. En la siguiente tabla se describe los atributos que contiene esta salida.

Tabla 7. Atributos del Json salida del servicio web extraer entidades

Atributo	Descripción

entidades	Contiene una lista que contiene los atributos contexto y entidad
contexto	Este atributo contiene el contexto en donde se la encontró a la entidad.
entidad	Contiene una lista de tuplas, con las entidad con este formato [entidad,etiquetas].

6.1.2.4. Ejemplo de llamada

http://taw02.utpl.edu.ec/anavisoers/extraerentidades?sentence=The conquest of Italy; Roman expansion: Pyrrhus, Punic Wars and provinces; classes, courts, and the Roman revolution; Augustus and the formation of empire.

6.1.2.5. Ejemplo del resultado


```
{
  "entidades": [
    {
      "contexto": "The conquest Italy ; Roman expansion : Pyrrhus , Punic Wars provinces ;
classes , courts , Roman revolution ; Augustus formation empire .",
      "entidad": [
        [
          "Italy",
          "NNP"
        ]
      ]
    },
    {
      "contexto": "The conquest Italy ; Roman expansion : Pyrrhus , Punic Wars provinces ;
classes , courts , Roman revolution ; Augustus formation empire .",
      "entidad": [
        [
          "Roman",
          "NNP"
        ],
        [
          "expansion",
          "NN"
        ]
      ]
    },
    .
    .
    .
    {
      "contexto": "The conquest Italy ; Roman expansion : Pyrrhus , Punic Wars provinces ;
classes , courts , Roman revolution ; Augustus formation empire .",
      "entidad": [
        [
          "Roman",
          "NNP"
        ],
        [
          "revolution",
          "NN"
        ]
      ]
    }
  ]
}
```

6.1.3. API del Servicio web desambiguación y enlace

Este Api procesa la entrada de texto, para devolver una lista con las entidades extraídas, desambiguadas y enlazadas a un recurso de la Dbpedia.

6.1.3.1. Url del servico

http://taw02.utpl.edu.ec/anavisoers/desambiguarenlazar

6.1.3.2. Entradas del servicio

Los parámetros de entrada de este servicio se lo describe en la siguiente tabla:

Tabla 8. Parámetros de entrada del servicio web desambiguar y enlazar

Parámetro	**Descripción**
sentence	Se envía en este parámetro el texto que se desea procesar. **Ejemplo**: The conquest of Italy; Roman expansion: Pyrrhus, Punic Wars and provinces; classes, courts, and the Roman revolution; Augustus and the formation of empire.

6.1.3.3. Salidas del servicio

El tipo de respuesta o salida es un Json. En la siguiente tabla se describe los atributos que contiene esta salida.

Tabla 9. Atributos del Json salida del servicio web desambiguar y enlazar

Atributo	**Descripción**
enlace	Contiene una lista que contienes a los atributos enlaces, entidad, link y tipo
enlaces	Este atributo contiene una lista con los posibles enlaces para la entidad. Esta lista de entidades tiene atributos como enlace (que será el link), frecuencia y tipo.
entidad	Contiene un string con la entidad.
link	Contiene un enlace a la Dbpedia
tipo	Este atributo contiene el tipo de recurso al que hace referencia el atributo enlace o link.
frecuencia	Este atributo está dentro de la lista de atributo enlaces, y se lo utiliza para identificar la relevancia del enlace.

6.1.3.4. Ejemplo de llamada

http://taw02.utpl.edu.ec/anavisoers/desambiguarenlazar?sentence=The conquest of Italy; Roman expansion: Pyrrhus, Punic Wars and provinces; classes, courts, and the Roman revolution; Augustus and the formation of empire.

6.1.3.5. Ejemplo del resultado


```
{
  "enlace": [
    {
      "enlaces": [
        {
          "enlace": "http://dbpedia.org/resource/Italy",
          "frec": 2,
          "tipo": "GEO"
        }
      ],
      "entidad": "Italy",
      "link": "http://dbpedia.org/resource/Italy",
      "tipo": "GEO"
    },
    …
    {
      "enlaces": [
        {
          "enlace": "http://dbpedia.org/resource/Wars_of_Augustus",
          "frec": 5,
          "tipo": "Sin Tipo"
        },
        {
          "enlace": "http://dbpedia.org/resource/Augustus",
          "frec": 4,
          "tipo": "AGT"
        },
        …
        {
          "enlace": "http://dbpedia.org/resource/Arch_of_Augustus",
          "frec": 0,
          "tipo": "Sin Tipo"
        }
      ],
      "entidad": "Augustus",
      "link": "http://dbpedia.org/resource/Wars_of_Augustus",
      "tipo": "Sin Tipo"
    },
    …
    {
      "enlaces": [
        {
          "enlace": "http://dbpedia.org/resource/Empire",
          "frec": 1,
          "tipo": "Sin Tipo"
        }
      ],
      "entidad": "Empire",
      "link": "http://dbpedia.org/resource/Empire",
      "tipo": "Sin Tipo"
    }
  ]
}
```

VALIDACIÓN Y PRUEBAS

Se llevó a cabo algunas pruebas sobre los Servicios Web, entre las mismas tenemos:

7.1. Validación.

Se las llevara a cabo sobre todos los Servicios Web, así como también al app Cliente:

- Servicio Web Tokenizar
- Servicio Web Extraer Entidades
- Servicio Web Desambiguar y Enlazar
- App Cliente

La prueba que se llevara a cabo será la Prueba de Funcionalidad.

7.1.1. Prueba de funcionalidad

7.1.1.1. Objetivo

El objetivo de esta prueba será comprobar un desempeño correcto de los servicios, lo que se busca verificar es:

- Los servicios Web funcionan de manera adecuada
- No presenta errores en ejecución
- No se paralizan los servicios
- Responde como fue programada, devuelve el resultado lo que cada servicio debe devolver.

7.1.1.2. Escenario

Las pruebas se llevaran a cabo en un servidor local.

7.1.1.3. Pruebas sobre el servicio web tokenizar

Tabla 10 Pruebas sobre el Servicio Web Tokenizar

TOKENIZAR					
Entradas	**Resultados esperados**	**Resultados Obtenidos**	**Error (S/N)**	**Observaciones**	**Escenarios-Condiciones**
texto vacía	Json/Resultado vacío	Json/Resultado vacío	N	El resultado se presenta en una ventana de interfaz para el usuario, en caso de descargar el JSON estará vacío	Ninguno
texto consistente de únicamente un espacio en blanco	Json/Resultado vacío	Json/Resultado vacío	N	El resultado se presenta en una ventana de interfaz para el usuario, en caso de descargar el JSON estará vacío	Ninguno

texto consistente de únicamente un signo de puntuación o carácter especial como: #, ", ', &, y <,	Json/Resultado vacío	Json/Resultado vacío	N	Se presenta una lista de resultados con el texto tokenizado y la etiqueta del mismo. En caso del JSON será una lista de variables con los mismos campos.	Ninguno
texto consistente de varios signos de puntuación o caracteres especiales como: #, ", ', &, y <, o también solo números	JSON con texto tokenizado	JSON con texto tokenizado	N	Se presenta una lista de resultados con el texto tokenizado y la etiqueta del mismo. En caso del JSON será una lista de variables con los mismos campos.	Existan varias cadenas conformadas por caracteres separadas por un espacio en blanco
Texto conformado por palabras, caracteres o números.	JSON con texto tokenizado	JSON con texto tokenizado	N	Se presenta una lista de resultados con el texto tokenizdo y la etiqueta del mismo. En caso del JSON será una lista de variables con los mismos campos.	* Conjunto de palabras separadas por espacios. * Caso más probable.
Texto en inglés o español con algunos palabras en otro idioma	JSON con texto tokenizado	JSON con texto tokenizado	N	Se etiquetan las palabras en idioma extranjero como Palabra Extranjera	* Mayoría del texto escrito en inglés/español con ciertas palabras en otro idioma.
Combinación de palabras en inglés o español.	JSON con texto tokenizado	JSON con texto tokenizado		Se presenta una lista de resultados con el texto tokenizdo y la etiqueta del mismo. En caso del JSON será una lista de variables con los mismos campos.	* Conjunto de palabras en español/inglés separadas por espacios. null
Texto conformado totalmente palabras en idioma distinto a inglés o español	Mensaje de notificación sobre idioma no soportado	Mensaje de notificación sobre idioma no soportado		La funcionalidad está implementada para soportar solo idioma inglés o español.	* Conjunto de palabras en otro idioma separadas por espacios. null

7.1.1.1. Pruebas sobre el servicio web extraer entidades

Tabla 11 Pruebas sobre el Servicio Web Extraer Entidades

EXTRAER ENTIDADES					
Entradas	**Resultados esperados**	**Resultados Obtenidos**	**Error (S/N)**	**Observaciones**	**Escenarios-Condiciones**

texto vacía	Json/Resultado vacío	Json/Resultado vacío	N	El resultado será vacio puesto que no existen cadenas de las cuales extraer entidades, se presentará una ventana de interfaz para el usuario, en caso de descargar el JSON estará vacío.	* Ninguno. * Para extraer entidades se necesita que el texto tenga significado y una estructura correcta y ordenada.
texto consistente de únicamente un espacio en blanco			N		
texto consistente de únicamente un signo de puntuación o carácter especial como: #, ", ', &, y <,			N		
texto consistente de varios signos de puntuación o carácteres especiales como: #, ", ', &, y <, o también solo números			N		
Texto conformado por palabras, caracteres o números.	JSON con lista entidades	JSON con lista entidades	N	Se presenta una lista de entidades relevantes con el contexto donde se encuentra la misma. En el caso del JSON se presenta: * entidades * etiquetas * contexto	El texto debe estar correctamente estructurado y tener sentido
Texto en inglés o español con algunos palabras en otro idioma	JSON con lista entidades	JSON con lista entidades	N		
Combinación de palabras en inglés o español.	JSON con lista entidades	JSON con lista entidades	N		
Texto conformado totalmente por palabras en idioma distinto a inglés o español	Mensaje de notificación sobre idioma no soportado	Mensaje de notificación sobre idioma no soportado	N	Mensaje de Notificación sobre idioma tanto en el interfaz como en el JSON.	El texto debe estar en un idioma no soportado por la aplicación.

7.1.1.1. Pruebas sobre el servicio web desambiguar enlazar

Tabla 12 Pruebas sobre el Servicio Web Desambiguar y Enlazar

DESAMBIGUAR-ENLAZAR					
Entradas	**Resultados esperados**	**Resultados Obtenidos**	**Error (S/N)**	**Observaciones**	**Escenarios-Condiciones**
texto vacía	Json/Resultado vacío	Json/Resultado vacío	N	La ventana de resultados estará vacia puesto que no existen cadenas para procesar, en caso de descargar el JSON	* Ninguno. * Para poder desambiguar y enlazar se debe primero extraer
texto consistente de únicamente un espacio en blanco			N		

texto consistente de únicamente un signo de puntuación o carácter especial como: #, ", ', &, y <,			N	estará vacio.	entidades y necesita que el texto tenga significado y una estructura correcta y ordenada.
texto consistente de varios signos de puntuación o carácteres especiales como: #, ", ', &, y <, o también solo números			N		
Texto conformado por palabras, caracteres o números.	JSON con lista entidades desambiguadas y elnazadas a un recurso de la dpedia	JSON con lista entidades desambiguadas y elnazadas a un recurso de la dpedia	N	Se presenta una lista de entidades relevantes desambiguadas con un enlace a un recurso de la dbpedia y con una lista de enlaces de la dbpedia que tambien podrian tatar la misma entidad. En el caso del JSON se presenta: * entidades * enlace * lista de enlaces posibes	El texto debe estar correctamente estructurado y tener sentido
Texto en inglés o español con algunos palabras en otro idioma			N		
Combinación de palabras en inglés o español.			N		
Texto conformado totalmente por palabras en idioma distinto a inglés o español	Mensaje de notificación sobre idioma no soportado	Mensaje de notificación sobre idioma no soportado	N	Mensaje de Notificación sobre idioma tanto en el interfaz como en el JSON.	El texto debe estar en un idioma no soportado por la aplicación.

7.1.2. Pruebas adicionales

7.1.2.1. Objetivo

El objetivo de esta prueba será compara el desempeño de los servicios creados contra otros servicios similares, para la comparación se utilizara los servicios Spotlight (http://dbpedia-spotlight.github.io/demo) de Dbpedia y Textalitics (http://textalytics.com/core/demo-test).

Lo que se busca verificar es la capacidad de cada servicio, así como la efectividad y precisión de sus procesos.

7.1.2.2. Escenario

Las pruebas se llevaran a cabo mediante el uso de 10 frases que se las utilizara en los servicios creados, en Spotlight y en Textalitics.

7.1.2.3. Pruebas sobre los servicios

Para realizar las pruebas se seleccionaron 10 frases, 5 en inglés y 5 en español, las frases fueron tomadas de contenidos de páginas Ocw.

Para poder comparar y medir la precisión se crearon distintas pruebas, así como distintas métricas:

Tokenización

- N Tokens, los tokens extraídos con el servicio.
- # palabras, las palabras que existen en la frase evaluada.
- % exactitud, cociente entre las palabras totales y los tokens extraídos

Extracción entidades

- Entidades, conceptos extraídos mediante la aplicación.
- Entidades Correctas, entidades extraídas correctamente a partir de un análisis humano.
- % extracción, cociente entre las entidades extraídas y las entidades correctas.

Desambiguar y enlazar

- Entidades, conceptos extraídos mediante la aplicación
- Enlaces, número de entidades enlazadas.
- % enlace, cociente entre las entidades y las entidades enlazadas.

Las tablas de las pruebas se las puede ver en el Anexo 5.

A continuación se presenta una estadística de todas las pruebas realizadas.

Tabla 13 Comparación entre servicios con texto en inglés

Con texto en Inglés			
Servicios	**Tokenizar**	**Extraer Entidades**	**Desambiguar y Enlazar**
http://taw02.utpl.edu.ec/anavisoers/interfaz	100,0%	96,2%	72,2%
http://dbpedia-spotlight.github.io/demo	0,0%	91,6%	98,2%
http://textalytics.com/core/demo-test	0,0%	100,0%	2,5%

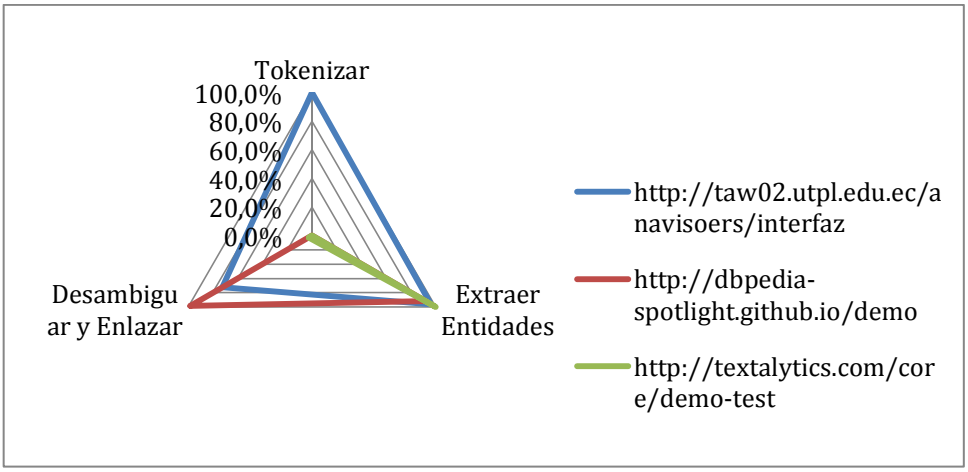

Figura 35 Comparación entre servicios con texto en inglés.

Tabla 14 Comparación entre servicios con texto en español.

Con texto en español			
Servicios	**Tokenizar**	**Extraer Entidades**	**Desambiguar y Enlazar**
http://taw02.utpl.edu.ec/anavisoers/interfaz	100,0%	97,8%	42,1%
http://dbpedia-spotlight.github.io/demo	0,0%	50,5%	100,0%
http://textalytics.com/core/demo-test	0,0%	100,0%	0,0%

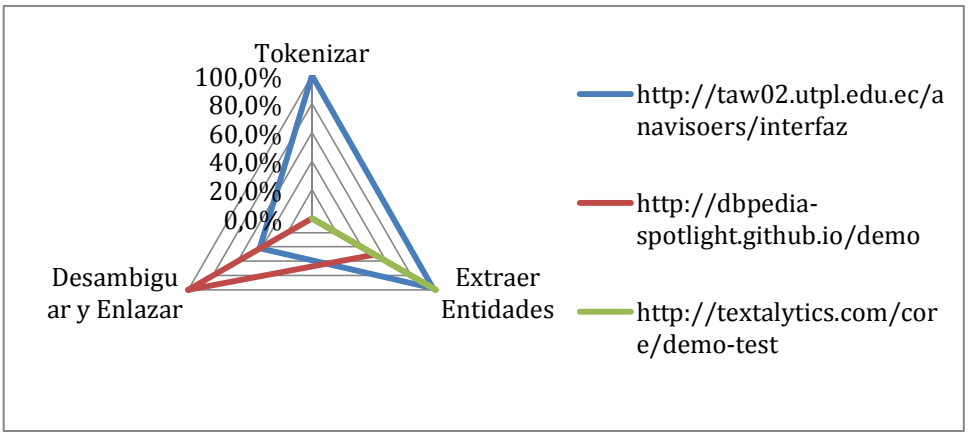

Figura 36 Comparación entre servicios con texto en español.

DISCUSIÓN FINAL

Como respuesta a la problemática planteada en los capítulos iniciales se han establecido como solución la implementación de 3 servicios web que permitan llevar a cabo un procesamiento de lenguaje natural en lo que respecta a:

- Tokenización y etiquetado
- Extracción de entidades
- Desambiguación y enlace

Cada uno de estos servicios nos permite obtener distintas abstracciones a partir de un mismo texto, y mientras más complejo es cada uno de ellos más depende de los anteriores, por ejemplo: para realizar la extracción de entidades se debe tokenizar y etiquetar todo el texto y así mismo para la desambiguación y enlace se debe llevar a cabo una extracción de entidades.

Para medir la precisión de los Servicios Web se los comparo con otros servicios de PLN para tener una métrica referencial con respecto a los mismos, para ello se crearon distintas pruebas, se las puede observar en la tabla 13 y figura 45 que se realizó con texto en inglés y en la tabla 14 y figura 46 con texto en español.

Estas estadísticas fueron determinadas a partir de las distintas pruebas que se realizaron sobre los servicios.

Se estableció y utilizo como base de conocimiento a la Dbpedia ya que contiene mucha información ya relacionada semánticamente, ayudando en el proceso de desambiguar y enlazar.

Aun no existe una librería que nos permita realizar una extracción de entidades 100% efectiva y clara sobre texto ingresado en distintos idiomas, existe limitación por esta parte de las tecnologías actuales, en el caso del idioma español la librería NLTK no tiene módulos especializados que realizan un procesamiento de lenguaje natural bastante preciso.
La creación de un vocabulario para la descripción semántica de los servicios web, es de gran ayuda ya que sirve para que se puedan auto describir estos servicios, así como también puedan interactuar con otros servicio, púes se conoce su información que está estructurada semánticamente para que sea de fácil entendimiento para un humano como para una máquina.

Al brindar un servicio web simple como es la tokenización y taggeo se puede potenciar el análisis y procesamiento de texto a partir de otras aplicaciones o sistemas, los 2 analizadores que se evaluaron juntamente con el sistema no ofrecían dicho servicio, sino que directamente ofrecían la extracción de entidades o la desambiguación; es importante un Servicio web de tokenización puesto que dicho proceso permite etiquetar cada palabra de un texto o del contenido de una página web, según la función que desempeña dentro de su contexto, permitiendo de esta manera obtener un significado de las mismas y posteriormente encontrar y extraer las entidades localizadas en ese contenido.

Mediante la creación del servicio web extracción de entidades, se logró encontrar las entidades relevantes que componen un texto o el contenido de una página web, para así etiquetar de mejor manera ese contenido, con las entidades extraídas, esto ayudara a clasificar mejor dicha información, como también se facilitara búsquedas futuras de ese contenido.

Al tener el servicio de desambiguar y enlazar, se dará más relevancia a las entidades encontradas ya que también se enlaza la entidad con un recurso de la Dbpedia, si dicha entidad se encuentra en la base de conocimientos (Dbpedia). La entidad será desambiguada con la base de conocimientos, ya que el recurso será escogido de entre muchos similares pero el que tenga más relación con el contexto de la entidad encontrada será el que se asignara a la entidad. Esto ayuda a enriquecer las entidades y a su vez el contendió, ya que con estas relaciones se podrán establecer conexiones entre los contenidos relacionados. Con la ayuda de esto se podrían realizar búsquedas más exactas y precisas sobre un tema.

CONCLUSIONES

Luego de haber implementado cada una de las funcionalidades del sistema, realizadas las respectivas pruebas de verificación y corregidos los errores presentados, se puede concluir:

- La estructuración del conocimiento a partir de una vocabulario permite obtener una gran ventaja al analizar la información y más aún al crear un vocabulario para la descripción semántica de los servicios web, se establece un marco de trabajo y un modelo estándar para proyectos referentes al enriquecimiento de contenido y la generación de meta data útil en la descripción de la información de los Servicios.
- Construir un sistema mediante servicios web que se pueden usar entre sí, amplía la gama de posibilidades para desarrollar futuros proyectos a partir de los mismos; puesto que al implementar un servicio simple el mismo puede ser consumido por uno o más servicios y de esta manera facilitar la implementación de métodos más complejos para el análisis y procesamiento de información.
- Mediante procesamiento de lenguaje natural (PNL) se puede enriquecer la información que se encuentra en texto, ya que mediante los proceso de extracción de entidades se puede adicionar información sobre la ya existente, por ejemplo, en un texto se pueden encontrar conceptos, entidades, personas, países, entre otros, que amplían la información de ese contenido.
- Al momento de extraer una entidad no es relevante solamente las funciones o etiquetas de la palabra o palabras que conforman esa entidad, sino también es importante el contexto en el cual se encuentra la misma y mediante el cual se puede determinar la relevancia de la entidad extraída dentro del contenido y posteriormente establecer enlaces con la Dbpedia.
- Con la visualización, que nos permite tener la aplicación cliente, facilita darse cuenta de las entidades existentes en un texto, y que estas entidades están relacionadas con otros datos o información, en este caso recursos de la Dbpedia, para llegar así a una web semántica ya que esta base de datos esta interconectada con mucha información ordenada y relacionada en la web.

RECOMENDACIONES

- Para llevar a cabo PLN sobre un texto o contenido web Python ofrece una librería bastante extensa y compleja, que además puede ser integrada con otras librerías para extender ciertas funcionalidades o mejorar la precisión de las mismas.

- Realizar una investigación a profundidad para la elección de vocabulario rdf, con el fin de reutilizar el vocabulario más adecuado, adaptable a nuestras exigencias, y sin dejar de lado la calidad de las fuentes.

- Es muy importante crear un entorno virtual para trabajar y desarrollar con Python, así se puede hacer pruebas y manipular las librerías sin causar ningún daño al sistema operativo.

- Se debe tener un conocimiento previo sobre la programación de lenguaje natural antes de utilizar la herramienta NLTK.

BIBLIOGRAFÍA

Alvarado Ruiz, P. A., Guamán Eras, D. E., & Sigcho Armijos, J. P. (2012). *Aplicación de tecnologías móviles para la búsqueda de recursos educativos abiertos.* Recuperado el 20 de 01 de 2014, de Bibliotec UTPL: http://dspace.utpl.edu.ec/bitstream/123456789/4938/1/Pablo%20Antonio%20Alvarado%20Ruiz.pdf

Berners-Lee, T. (agosto de 1996). *www.w3c.org.* Recuperado el 13 de 02 de 2014, de Actas de la V Conferencia Internatonal World Wide Web: www.w3c.org

Birrell, A., & Nelson, B. (1 de Febrero de 1984). Recuperado el 19 de Febrero de 2012, de http://nd.edu/~dthain/courses/cse598z/fall2004/papers/birrell-rpc.pdf

Blank, I. (Mayo de 2005). Recuperado el 15 de Septiembre de 2012, de http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf

Bolshakov, I., & Gelbukh, A. (2004). Computational Linguistics. Models, Resources, Applications. *Ciencia de la Computacion Primera Edición*.

DBpedia. (06 de 08 de 2012). *About.* Recuperado el 05 de 05 de 2014, de DBpedia: http://wiki.dbpedia.org/About

DBpedia. (17 de 09 de 2013). *Datasets.* Recuperado el 05 de 05 de 2014, de DBpedia Datasets: http://wiki.dbpedia.org/Datasets

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Recuperado el 02 de 2014, de https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

González, M. (2011). *ESTUDIO DE ARQUITECTURAS DE REDES ORIENTADAS A SERVICIO.* Recuperado el 19 de 01 de 2014, de UpCommons: http://upcommons.upc.edu/pfc/bitstream/2099.1/12312/1/ESTUDIO_DE_ARQUITECTURAS_DE_REDES_ORIENTADAS_A_SERVICIO.pdf

Google Inc. (14 de Agosto de 2012). *Android Developers*. Recuperado el 03 de Septiembre de 2012, de http://developer.android.com/reference/org/apache/http/package-summary.html

Loper, E., & Bird, S. (17 de 05 de 2002). *NLTK: The Natural Language Toolkit.* Recuperado el 21 de 01 de 2014, de Cornell University: http://arxiv.org/pdf/cs/0205028v1.pdf

Mari , V., & Rafael Pedraza, P. (05 de 06 de 2012). *PNL*. Recuperado el 23 de 02 de 2014, de El Procesamiento del Lenguaje Natural en la Recuperación de Información Textual y áreas afines: http://www.upf.edu/hipertextnet/numero-5/pln.html

Mednieks, Z., Dornin, L., Meike, G., & Nakamura, M. (2011). *Programming Android.* California: O'Reilly.

Murphy, M. (2010). *Beginning Android 2.* Califronia: O'Reilly.

Nelson, B., & Andrew, B. (1984). *RCP*. Recuperado el 19 de 01 de 2014, de Implementing Remote Procedure Calls: http://www.cs.princeton.edu/courses/archive/fall03/cs518/papers/rpc.pdf

Neuburg, M. (2011). *Programming iOS4.* California: O'Reill.

Nltk. (2013). *Nltk.* Recuperado el 27 de 01 de 2014, de NLTK: http://nltk.org/

OCW Consortium. (2012). *OpenCourseWare Consortium*. Recuperado el 7 de 02 de 2014, de http://www.ocwconsortium.org/

OCWConsortium. (2012). *OCWConsortium*. Recuperado el 03 de Octubre de 2012, de http://www.ocwconsortium.org/en/courses

Piedra , N., Chicaiza, J., & Lopez, J. (2012). *Combining Linked Data and Mobiles Devices to improve access to OCW.* Recuperado el 18 de 02 de 2014, de Mendeley: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6201202&contentType=Conference+Publications&searchWithin%3Dpiedra%2C+n%26queryText%3Dpiedra

Piedra , N., Chicaiza, J., & Lopez, J. (2012). *Combining Linked Data and Mobiles Devices to improve access to OCW.* Recuperado el 25 de Julio de 2012, de Mendeley: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6201202&contentType=Conference+Publications&searchWithin%3Dpiedra%2C+n%26queryText%3Dpiedra

Piedra, N. (25 de Enero de 2012). *slideshare.* Recuperado el 03 de Octubre de 2012, de http://www.slideshare.net/emadridnet/2012-01-20-upm-emadrid-etovar-upm-npiedra-utpl-linked-data-repositorios-ocw

Piedra, N., Tovar, E., López, J., Chicaiza, J., & Martinez, O. (4 de abril de 2011). *www.ocw.org*. Recuperado el 18 de 02 de 2014, de http://conference.ocwconsortium.org/index.php/2011/cambridge/paper/view/162

Python. (2002). *About Python.* Recuperado el 19 de 01 de 2014, de Python: http://www.python.org/about/

San Martin Oliva, C. R. (s.f.). *portalhuarpe.* Recuperado el 25 de Julio de 2012, de http://www.portalhuarpe.com.ar/Seminario09/archivos/MetodologiaICONIX.pdf

SunMicrosystems, I. (04 de 2005). *RESTfulWeb Services*. Recuperado el 03 de 02 de 2014, de http://docs.huihoo.com/glassfish/v3/820-4867.pdf

SunMicrosystems, Inc. (01 de Abril de 2005). Recuperado el 18 de Junio de 2012, de http://docs.huihoo.com/glassfish/v3/820-4867.pdf

Tovar, E., Piedra, N., Chicaiza, J., Lopez , J., & Martínez, O. (2012). Development and promotion of OERs. Outcomes of an international research project under OpenCourseWare model. . *Journal of Universal Computer Science*.

Turner, J. (2011). *Developing Enterprise iOS Applications.* California: O'Reilly.

UNESCO. (2005). *UNESCO*. Recuperado el 20 de 02 de 2014, de What are Open Educational Resources (OERs)?: http://www.unesco.org/new/en/communication-and-information/access-to-knowledge/open-educational-resources/what-are-open-educational-resources-oers/

Unesco. (2011). *A Basic Guide to Open Educational Resources.* Recuperado el 22 de 8 de 2013, de www.unesco.org/education

Universia. (2012). *Universia*. Recuperado el 03 de Octubre de 2012, de http://ocw.universia.net/es/buscar-por-areas.php?ord=A

UPM. (s.f.). *¿Qué es OCW?* Recuperado el 20 de 01 de 2014, de OpenCourseWare de la Universidad Politécnica de Madrid: http://ocw.upm.es/bfque-es-ocw

UTPL. (20 de 01 de 2014). *OpenCourseWare UTPL.* Recuperado el 21 de 02 de 2014, de UTPL OCW: http://ocw.utpl.edu.ec/

Vallez, M. (2009). *La Web Semántica y las Tecnologías del Lenguaje Humano.* Recuperado el 22 de 11 de 2013, de e-Lis: http://eprints.rclis.org/15586/1/La%20Web%20Sem%C3%A1ntica%20y%20las%20Tecnolog%C3%ADas%20del%20Lenguaje%20Humano%20-%20Preprint.pdf

Vallez, M., Rovira, C., Codina, L., & Pedraza, R. (05 de 06 de 2012). *Procedimientos para la extracción de palabras clave de páginas web basados en criterios de posicionamiento en buscadores.* Recuperado el 21 de 01 de 2014, de UPF: http://www.upf.edu/hipertextnet/numero-8/extraccion_keywords.html

W3C. (11 de 02 de 2004). *Web Services Architecture.* Recuperado el 19 de 01 de 2014, de W3C: http://www.w3.org/TR/ws-arch/

W3C. (15 de 01 de 2008). *Sparql query Basicpatterns.* (P. Eric, & S. Andy, Edits.) Recuperado el 06 de 05 de 2014, de W3C: http://www.w3.org/TR/rdf-sparql-query/#basicpatterns

W3C. (01 de 15 de 2008). *SPARQL Query Language for RDF.* Recuperado el 28 de 04 de 2014, de RDF sparql-query: http://www.w3.org/TR/rdf-sparql-query/

W3C. (15 de 01 de 2008). *Sparql Query SparqlSyntax.* Recuperado el 07 de 05 de 2014, de W3C: http://www.w3.org/TR/rdf-sparql-query/#sparqlSyntax

Working Group RDF. (25 de 02 de 2014). *RDF.* Recuperado el 25 de 04 de 2014, de RDF: http://www.w3.org/RDF/

Zeman, E. (03 de Noviembre de 2011). *informationweek*. Recuperado el 19 de Junio de 2012, de http://www.informationweek.com.mx/movilidad/android-4-0-vs-ios-5-cara-a-cara/

ANEXO 1: ONTOLOGÍA PARA LA DESCRIPCIÓN DE LOS SERVICIOS WEB

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY dcterms "http://purl.org/dc/terms/" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY sw "http://taw02.utpl.edu.ec/anavisoers/" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY swpo "http://sw-portal.deri.org/ontologies/swportal#" >
    <!ENTITY ssn "http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#" >
]>
<rdf:RDF xmlns="http://taw02.utpl.edu.ec/anavisoers#"
     xml:base="http://taw02.utpl.edu.ec/anavisoers"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:swpo="http://sw-portal.deri.org/ontologies/swportal#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:ssn="http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:sw="http://taw02.utpl.edu.ec/anavisoers/"
     xmlns:dcterms="http://purl.org/dc/terms/">
    <owl:Ontology rdf:about="http://taw02.utpl.edu.ec/anavisoers"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Annotation properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://purl.org/dc/elements/1.1/contributor -->
```

```
<owl:AnnotationProperty rdf:about="&dc;contributor"/>
<!-- http://purl.org/dc/elements/1.1/date -->
<owl:AnnotationProperty rdf:about="&dc;date"/>
<!-- http://purl.org/dc/elements/1.1/description -->
<owl:AnnotationProperty rdf:about="&dc;description"/>
<!-- http://purl.org/dc/elements/1.1/identifier -->
<owl:AnnotationProperty rdf:about="&dc;identifier"/>
<!-- http://purl.org/dc/elements/1.1/language -->
<owl:AnnotationProperty rdf:about="&dc;language"/>
<!-- http://purl.org/dc/elements/1.1/publisher -->
<owl:AnnotationProperty rdf:about="&dc;publisher"/>

<!-- http://purl.org/dc/elements/1.1/source -->
<owl:AnnotationProperty rdf:about="&dc;source"/>
<!-- http://purl.org/dc/elements/1.1/title -->
<owl:AnnotationProperty rdf:about="&dc;title"/>
<!-- http://purl.org/dc/elements/1.1/type -->

<owl:AnnotationProperty rdf:about="&dc;type"/>
<!-- http://purl.org/dc/terms/created -->
<owl:AnnotationProperty rdf:about="&dcterms;created"/>
<!-- http://purl.org/dc/terms/modified -->
<owl:AnnotationProperty rdf:about="&dcterms;modified"/>
<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Object Properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#hasInput -->
<owl:ObjectProperty rdf:about="&ssn;hasInput">
    <rdfs:domain rdf:resource="http://utpl.edu.ec/WebServices#WebService"/>
    <rdfs:range rdf:resource="&ssn;Input"/>
</owl:ObjectProperty>
```

```
<!-- http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#hasOutput -->
<owl:ObjectProperty rdf:about="&ssn;hasOutput">
    <rdfs:domain rdf:resource="http://utpl.edu.ec/WebServices#WebService"/>
    <rdfs:range rdf:resource="&ssn;Output"/>
</owl:ObjectProperty>
<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->
<!-- http://purl.org/dc/elements/1.1/creator -->
<owl:DatatypeProperty rdf:about="&dc;creator"/>
<!-- http://purl.org/dc/elements/1.1/description -->
<owl:DatatypeProperty rdf:about="&dc;description"/>
<!-- http://purl.org/dc/elements/1.1/identifier -->
<owl:DatatypeProperty rdf:about="&dc;identifier"/>


<!-- http://purl.org/dc/elements/1.1/language -->
<owl:DatatypeProperty rdf:about="&dc;language"/>
<!-- http://purl.org/dc/elements/1.1/publisher -->
<owl:DatatypeProperty rdf:about="&dc;publisher"/>
<!-- http://purl.org/dc/elements/1.1/title -->
<owl:DatatypeProperty rdf:about="&dc;title"/>
<!-- http://purl.org/dc/elements/1.1/type -->
<owl:DatatypeProperty rdf:about="&dc;type"/>
<!-- http://purl.org/dc/terms/created -->
<owl:DatatypeProperty rdf:about="&dcterms;created"/>
<!-- http://purl.org/dc/terms/modified -->
<owl:DatatypeProperty rdf:about="&dcterms;modified"/>
```

```
<!-- http://sw-portal.deri.org/ontologies/swportal#isVersion -->
<owl:DatatypeProperty rdf:about="&swpo;isVersion"/>

<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://utpl.edu.ec/WebServices#WebService -->
<owl:Class rdf:about="http://utpl.edu.ec/WebServices#WebService"/>
<!-- http://www.w3.org/2002/07/owl#Thing -->
<owl:Class rdf:about="&owl;Thing"/>
<!-- http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#Input -->
<owl:Class rdf:about="&ssn;Input"/>



<!-- http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#Output -->
<owl:Class rdf:about="&ssn;Output"/>
<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://taw02.utpl.edu.ec/anavisoers/desambiguarenlazar -->
<owl:NamedIndividual rdf:about="&sw;desambiguarenlazar">
    <rdf:type rdf:resource="http://utpl.edu.ec/WebServices#WebService"/>
    <dc:creator rdf:datatype="&xsd;dateTime">03-05-2014</dc:creator>
```

<dcterms:modified rdf:datatype="&xsd;dateTime">03-05-2014</dcterms:modified>
<dcterms:modified rdf:datatype="&xsd;dateTime">04-07-2014</dcterms:modified>
<swpo:isVersion rdf:datatype="&xsd;string">1.0</swpo:isVersion>
<dc:creator rdf:datatype="&xsd;string">Jhonny Zaruma</dc:creator>
<dc:description rdf:datatype="&xsd;string">Servicio Web que desambigua y enlaza con la dbpeda a partir de entidades que son extraídas del texto que se le envié a este servicio, y este servicio devuelve como resultado un json con una lista de las entidades con un enlace a la Dbpedia</dc:description>
<dc:publisher rdf:datatype="&xsd;string">TAW-SBC</dc:publisher>
<dc:title rdf:datatype="&xsd;string">Web Service Desambiguar y Enlazar</dc:title>
<dc:language rdf:datatype="&xsd;anyURI">http://dbpedia.org/resource/English_language</dc:language>
<dc:language rdf:datatype="&xsd;anyURI">http://es.dbpedia.org/resource/Idioma_español</dc:language>
<dc:identifier rdf:datatype="&xsd;anyURI">http://taw02.utpl.edu.ec/anavisoers/desambiguarenlazar</dc:identifier>
<ssn:hasInput rdf:resource="http://utpl.edu.ec/WebServices#IWSD01"/>
<ssn:hasOutput rdf:resource="http://utpl.edu.ec/WebServices#OWSD01"/>
</owl:NamedIndividual>

<!-- http://taw02.utpl.edu.ec/anavisoers/extraerentidades -->

<owl:NamedIndividual rdf:about="&sw;extraerentidades">
<rdf:type rdf:resource="http://utpl.edu.ec/WebServices#WebService"/>
<dcterms:created rdf:datatype="&xsd;dateTime">03-05-2014</dcterms:created>
<dcterms:modified rdf:datatype="&xsd;dateTime">03-05-2014</dcterms:modified>
<dcterms:modified rdf:datatype="&xsd;dateTime">04-07-2014</dcterms:modified>
<swpo:isVersion rdf:datatype="&xsd;string">1.0</swpo:isVersion>
<dc:creator rdf:datatype="&xsd;string">Jhonny Zaruma</dc:creator>
<dc:description rdf:datatype="&xsd;string">Servicio web que extrae entidades a partir de un texto que se le envie, devuelve un json con una lista de entidades con etiquetas y contexto.</dc:description>
<dc:publisher rdf:datatype="&xsd;string">TAW-SBC</dc:publisher>
<dc:title rdf:datatype="&xsd;string">Web Service Extrae Entidades</dc:title>

<dc:language
rdf:datatype="&xsd;anyURI">http://dbpedia.org/resource/English_language</dc:language>
<dc:type
rdf:datatype="&xsd;anyURI">http://dbpedia.org/resource/Web_service</dc:type>
<dc:language
rdf:datatype="&xsd;anyURI">http://es.dbpedia.org/resource/Idioma_español</dc:language>
<dc:identifier
rdf:datatype="&xsd;dateTime">http://taw02.utpl.edu.ec/anavisoers/extraerentidades</dc:iden
tifier>
<ssn:hasInput rdf:resource="http://utpl.edu.ec/WebServices#IWSE01"/>
<ssn:hasOutput rdf:resource="http://utpl.edu.ec/WebServices#OWSE01"/>
</owl:NamedIndividual>
<!-- http://taw02.utpl.edu.ec/anavisoers/tokenizar -->

<owl:NamedIndividual rdf:about="&sw;tokenizar">
<rdf:type rdf:resource="http://utpl.edu.ec/WebServices#WebService"/>
<dcterms:created rdf:datatype="&xsd;dateTime">03-05-2014</dcterms:created>
<dcterms:modified rdf:datatype="&xsd;dateTime">03-05-2014</dcterms:modified>
<dcterms:modified rdf:datatype="&xsd;dateTime">04-07-2014</dcterms:modified>
<swpo:isVersion rdf:datatype="&xsd;string">1.0</swpo:isVersion>
<dc:creator rdf:datatype="&xsd;string">Jhonny Zaruma</dc:creator>
<dc:description rdf:datatype="&xsd;dateTime">Servicio Web que tokeniza y etiqueta el
texto que se le envia, y de vuelve un json con una lista de los tokens
tageados</dc:description>
<dc:publisher rdf:datatype="&xsd;string">TAW-SBC</dc:publisher>
<dc:title rdf:datatype="&xsd;string">Web Service Tokenizar</dc:title>
<dc:language
rdf:datatype="&xsd;anyURI">http://dbpedia.org/resource/English_language</dc:language>
<dc:type
rdf:datatype="&xsd;anyURI">http://dbpedia.org/resource/Web_service</dc:type>
<dc:language
rdf:datatype="&xsd;anyURI">http://es.dbpedia.org/resource/Idioma_español</dc:language>
<dc:identifier
rdf:datatype="&xsd;anyURI">http://taw02.utpl.edu.ec/anavisoers/tokenizar</dc:identifier>
<ssn:hasInput rdf:resource="http://utpl.edu.ec/WebServices#IWST01"/>
<ssn:hasOutput rdf:resource="http://utpl.edu.ec/WebServices#OWST01"/>

```
    </owl:NamedIndividual>

    <!-- http://utpl.edu.ec/WebServices#IWSD01 -->

    <owl:NamedIndividual rdf:about="http://utpl.edu.ec/WebServices#IWSD01">
        <rdf:type rdf:resource="&ssn;Input"/>
        <dc:description rdf:datatype="&xsd;short">La entrada puede ser un texto o
string</dc:description>
        <dc:type rdf:datatype="&xsd;string">String</dc:type>
        <dc:title rdf:datatype="&xsd;anyURI">sentence</dc:title>
    </owl:NamedIndividual>

    <!-- http://utpl.edu.ec/WebServices#IWSE01 -->

    <owl:NamedIndividual rdf:about="http://utpl.edu.ec/WebServices#IWSE01">
        <rdf:type rdf:resource="&ssn;Input"/>
        <dc:description rdf:datatype="&xsd;string">La entrada puede ser un texto o
string</dc:description>
        <dc:type rdf:datatype="&xsd;string">String</dc:type>
        <dc:title rdf:datatype="&xsd;string">sentence</dc:title>
    </owl:NamedIndividual>

    <!-- http://utpl.edu.ec/WebServices#IWST01 -->

    <owl:NamedIndividual rdf:about="http://utpl.edu.ec/WebServices#IWST01">
        <rdf:type rdf:resource="&ssn;Input"/>
        <dc:description rdf:datatype="&xsd;string">La entrada puede ser un texto o
string</dc:description>
        <dc:type rdf:datatype="&xsd;string">String</dc:type>
        <dc:title rdf:datatype="&xsd;string">sentence</dc:title>
    </owl:NamedIndividual>

    <!-- http://utpl.edu.ec/WebServices#OWSD01 -->

    <owl:NamedIndividual rdf:about="http://utpl.edu.ec/WebServices#OWSD01">
        <rdf:type rdf:resource="&ssn;Output"/>
```

<dc:type rdf:datatype="&xsd;string">Json</dc:type>
<dc:title rdf:datatype="&xsd;string">enlace</dc:title>
<dc:description rdf:datatype="&xsd;string">servicio devuelve como resultado un json con una lista de las entidades con un enlace a la Dbpedia.
Ejemplo:
{
"enlace": [
["United States",[[1,"http://dbpedia.org/resource/United_States","GEO"]]],
["European Center",[]],

["CERN",[[1,"http://dbpedia.org/resource/CERN","ORG"]]]
]
}</dc:description>
</owl:NamedIndividual>

<!-- http://utpl.edu.ec/WebServices#OWSE01 -->

<owl:NamedIndividual rdf:about="http://utpl.edu.ec/WebServices#OWSE01">
<rdf:type rdf:resource="&ssn;Output"/>
<dc:type rdf:datatype="&xsd;string">Json</dc:type>
<dc:description rdf:datatype="&xsd;string">devuelve un json con una lista de entidades con etiquetas y contexto.

Ejemplo:
{
"entidades": [
[[["United","NNP"],["States", "NNPS"]],"One year ago , several hours cities across ..."],
[[["European", "NNP"],["Center","NNP"]],"One year ago , several hours cities..."],

[[["Nuclear","NNP"],["Research","NNP"]]," One year ago , several hours cities ..."]
]

}</dc:description>
        <dc:title rdf:datatype="&xsd;string">entidades</dc:title>
    </owl:NamedIndividual>

    <!-- http://utpl.edu.ec/WebServices#OWST01 -->

    <owl:NamedIndividual rdf:about="http://utpl.edu.ec/WebServices#OWST01">
        <rdf:type rdf:resource="&ssn;Output"/>
        <dc:type rdf:datatype="&xsd;string">Json</dc:type>
        <dc:description rdf:datatype="&xsd;string">Salida es un json con una lista de los tokens tageados
Ejemplo:
{
  "tokens": [
    ["cities", "NNS"],
    ["across", "IN"],
    ["the", "DT"],
    ["United", "NNP"],
    ["States", "NNPS"]
  ]
}</dc:description>
        <dc:title rdf:datatype="&xsd;string">tokens</dc:title>
    </owl:NamedIndividual>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->

Especificación de Caso de Uso (ECS)
Tokenizar

Versión [1.0.0]

Información del Documento

TÍTULO:	Especificación de Caso de Uso (ECS)
SUBTÍTULO:	Tokenizar
VERSIÓN:	[1.0.0]
ARCHIVO:	01 - Tokenizar
AUTOR:	Jhonny Alonso Zaruma
ESTADO:	Borrador

Lista de Cambios

VERSIÓN	FECHA	AUTOR	DESCRIPCIÓN
1.0.0	2014-05-13	JAZ	Emisión Inicial

Firmas y Aprobaciones

ELABORADO POR:	Jhonny Alonso Zaruma Analista - Desarrollador		
FECHA:	2014-05-13	Firma:	
REVISADO POR:	Nelson Piedra		

Especificación de Caso de Uso (ECS)

TOKENIZAR

Especificación de Casos de Uso

Número:	01	
Nombre:	Tokenizar	
Actores:	Usuario, Servicio Web	
Descripción:	Su función es Separar por tokens el texto que se le envié, y dichos tokens también etiquetarlos.	
Precondiciones:	El usuario a enviado el texto al WebService	
Pos condiciones:	Recibe una lista con los tokens tageados	
Flujo Normal:	**Actor:**	**Sistema:**
	1. Acede al Servicio web y envía una entrada(texto)	2. El sistema recibe el texto 3. Detecta el idioma **FA 1** 4. Si el idioma del texto esta en español lo traduce, tokeniza y etiqueta 5. Si el idioma del texto esta inglés lo tokeniza y etiqueta 6. El Servicio web devuelve una lista con los tokens etiquetados.
Sub Flujo		
Flujo Alternativo:	**FA 1 Idioma ingresado no soportado** Si el idioma del texto que se envía no es español ni inglés nos devuelve una lista vacía.	
Excepciones:		
Prioridad:	Alta	

Glosario

Actor	Usuario
WebService	Servicio web

Especificación de Caso de Uso (ECS)
Extraer Entidades

Versión [1.0.0]

Información del Documento

TÍTULO:	Especificación de Caso de Uso (ECS)
SUBTÍTULO:	Extraer Entidades
VERSIÓN:	[1.0.0]
ARCHIVO:	02 – Extraer Entidades
AUTOR:	Jhonny Alonso Zaruma
ESTADO:	Borrador

Lista de Cambios

VERSIÓN	FECHA	AUTOR	DESCRIPCIÓN
1.0.0	2014-05-13	JAZ	Emisión Inicial

Firmas y Aprobaciones

ELABORADO POR:	Jhonny Alonso Zaruma Analista - Desarrollador		
FECHA:	2014-05-13	Firma:	
REVISADO POR:	Nelson Piedra		

Especificación de Caso de Uso (ECS)

EXTRAER ENTIDADES

Especificación de Casos de Uso

<table>
<tr><td>Número:</td><td colspan="2">02</td></tr>
<tr><td>Nombre:</td><td colspan="2">Extraer Entidades</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario, Servicio web</td></tr>
<tr><td>Descripción:</td><td colspan="2">Su función es extraer entidades del texto que se le envié o la lista de tokens etiquetados que se le envié.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">El usuario a enviado el texto o la lista con los tokens etiquetados al Servicio web</td></tr>
<tr><td>Pos condiciones:</td><td colspan="2">Recibe una lista de las entidades extraídas con respectivos tags y con su contexto de donde se extrajo la entidad</td></tr>
<tr><td rowspan="2">Flujo Normal:</td><td>Actor:</td><td>Sistema:</td></tr>
<tr><td>1. Acede al Webservice y envía una entrada(texto o lista)</td><td>2. El sistema recibe la entrada
3. Detecta el tipo de entrada FA 1
4. Detecta el idioma FA 2
5. Si la entrada es texto y el idioma esta en español lo va a tokenizar FA 3. Luego extrae entidades definiendo patrones con el método Chunker de NLTK, y se crea una lista con las entidades extraídas.
6. Si la entrada es texto y el idioma esta en inglés se lo envía tokenizar FA 3, y se procede a extraer entidades con nechunk de NLTK
7. Si la entrada es una lista(que contiene tokens tageados) se hace la extracción de la entidades con nechunk
8. El Servicio web devuelve</td></tr>
</table>

		una lista con las entidades , los tags y con el contexto de donde se la extrajo.
Sub Flujo		
Flujo Alternativo:	**FA 1 Tipo de entrada no soportado** Si el tipo de entrada no es un texto ni tampoco una lista con tokens tageados, el Webservice nos devolverá una lista vacía. **FA 2 Idioma ingresado no soportado** Si el idioma del texto que se envía no es español ni inglés nos devuelve una lista vacía. **FA 3 Enviar texto a tokenizar** Se envía el texto al webservice Tokenizar y nos devuelve una lista con los tokens tageados.	
Excepciones:		
Prioridad:	Alta	
Referencias Cruzadas:		
Requerimientos Especiales:		
Asunciones y Dependencias:		
Notas adicionales:		

Glosario

Actor	Usuario
WebService	Servicio web
OCW	OpenCourseWare

Especificación de Caso de Uso (ECS)
Desambiguar y Enlazar

Versión [1.0.0]

Información del Documento

TÍTULO:	Especificación de Caso de Uso (ECS)
SUBTÍTULO:	Desambiguar y Enlazar
VERSIÓN:	[1.0.0]
ARCHIVO:	01 - Tokenizar
AUTOR:	Jhonny Alonso Zaruma
ESTADO:	Borrador

Lista de Cambios

VERSIÓN	FECHA	AUTOR	DESCRIPCIÓN
1.0.0	2014-05-13	JAZ	Emisión Inicial

Firmas y Aprobaciones

ELABORADO POR:	Jhonny Alonso Zaruma Analistas - Desarrolladores		
FECHA:	2014-05-13	Firma:	
REVISADO POR:	Nelson Piedra		

Especificación de Caso de Uso (ECS)

DESAMBIGUAR Y ENLAZAR

Especificación de Casos de Uso

<table>
<tr><td>Número:</td><td colspan="2">03</td></tr>
<tr><td>Nombre:</td><td colspan="2">Desambiguar y Enlazar</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario, WebService</td></tr>
<tr><td>Descripción:</td><td colspan="2">Su función es Desambiguar y Enlazar con recursos de la Dbpedia las entidades recibidas en un tipo de lista o el texto recibido.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">El usuario a enviado el texto o la lista de las entidades extraídas con respectivos tags y con su contexto de donde se extrajo la entidad al WebService</td></tr>
<tr><td>Pos condiciones:</td><td colspan="2">Recibe una lista que contiene las entidades desambiguados con los enlaces a los recursos de la Dbpedia.</td></tr>
<tr><td rowspan="2">Flujo Normal:</td><td>Actor:</td><td>Sistema:</td></tr>
<tr><td>1. Acede al Webservice y envía una entrada(texto o lista)</td><td>2. El sistema recibe la entrada
3. Detecta el tipo de entrada FA 1
4. Detecta el idioma FA 2
5. Si la entrada es texto y el idioma esta en español o inglés lo envía a tokenizar FA 3. Y contenemos una lisa con las entidades, tags y contexto .
6. Si la entrada es una lista que contiene entidades con tags y contexto, se procede a detectar el idioma FA 2 , luego se descompone la lista y mediante consultas SPARQL a la Dbpedia se desambigua y se enlaza con recursos de la Dbpedia. Para crear una lista con las entidades desambiguadas y enlazadas.
7. El Webservice devuelve una lista con las entidades desambiguadas y enlazadas a los recursos de la Dbpedia.</td></tr>
</table>

Sub Flujo	
Flujo Alternativo:	**FA 1 Tipo de entrada no soportado** Si el tipo de entrada no es un texto ni tampoco una lista con las entidades sus tokens y contexto, el Webservice nos devolverá una lista vacía. **FA 2 Idioma ingresado no soportado** Si el idioma del texto que se envía no es español ni inglés nos devuelve una lista vacía. **FA 3 Enviar texto a tokenizar** Se envía el texto al webservice Tokenizar y nos devuelve una lista con los tokens tageados.
Excepciones:	
Prioridad:	Alta
Referencias Cruzadas:	
Requerimientos Especiales:	
Asunciones y Dependencias:	
Notas adicionales:	

Glosario

Actor	Usuario
WebService	Servicio web

ANEXO 3: DIAGRAMAS DE SECUENCIA

Secuencia 01 - Tokenizar

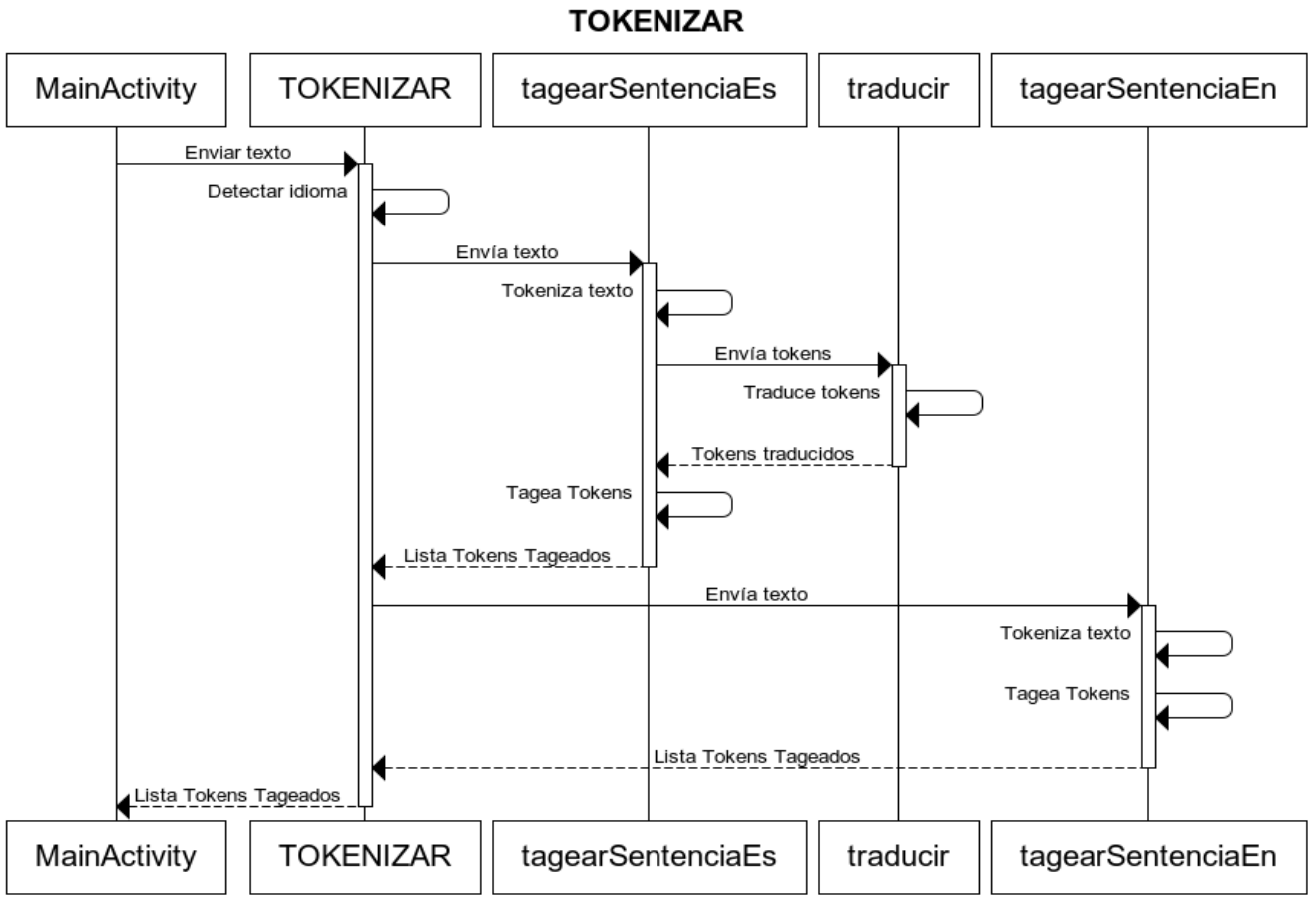

Figura 37 Diagrama de Secuencia Tokenizar

Secuencia 02 – Extraer Entidades

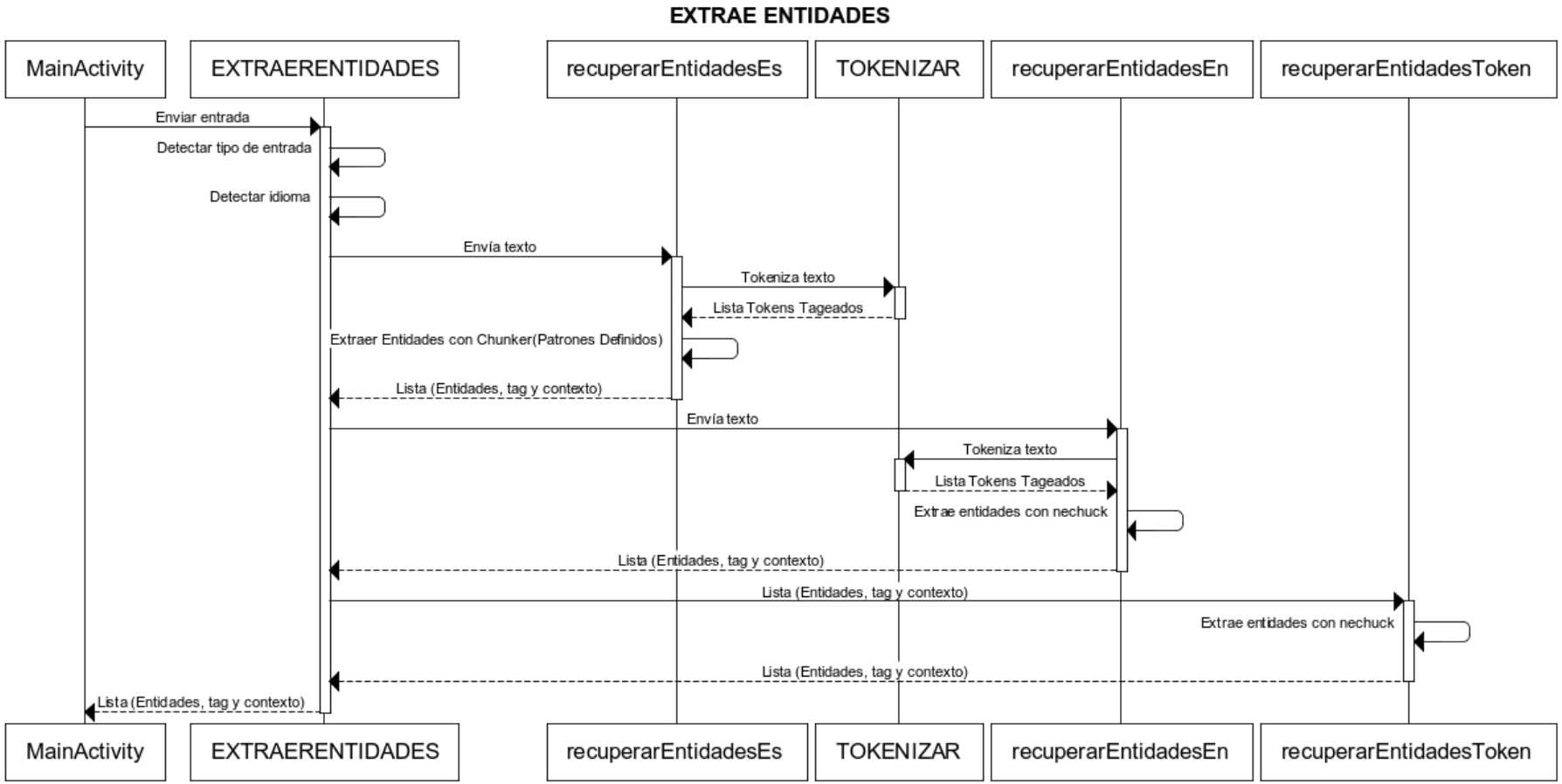

Figura 38 Diagrama de Secuencia Extrae Entidades

Secuencia 03 – Desambiguar y Enlazar

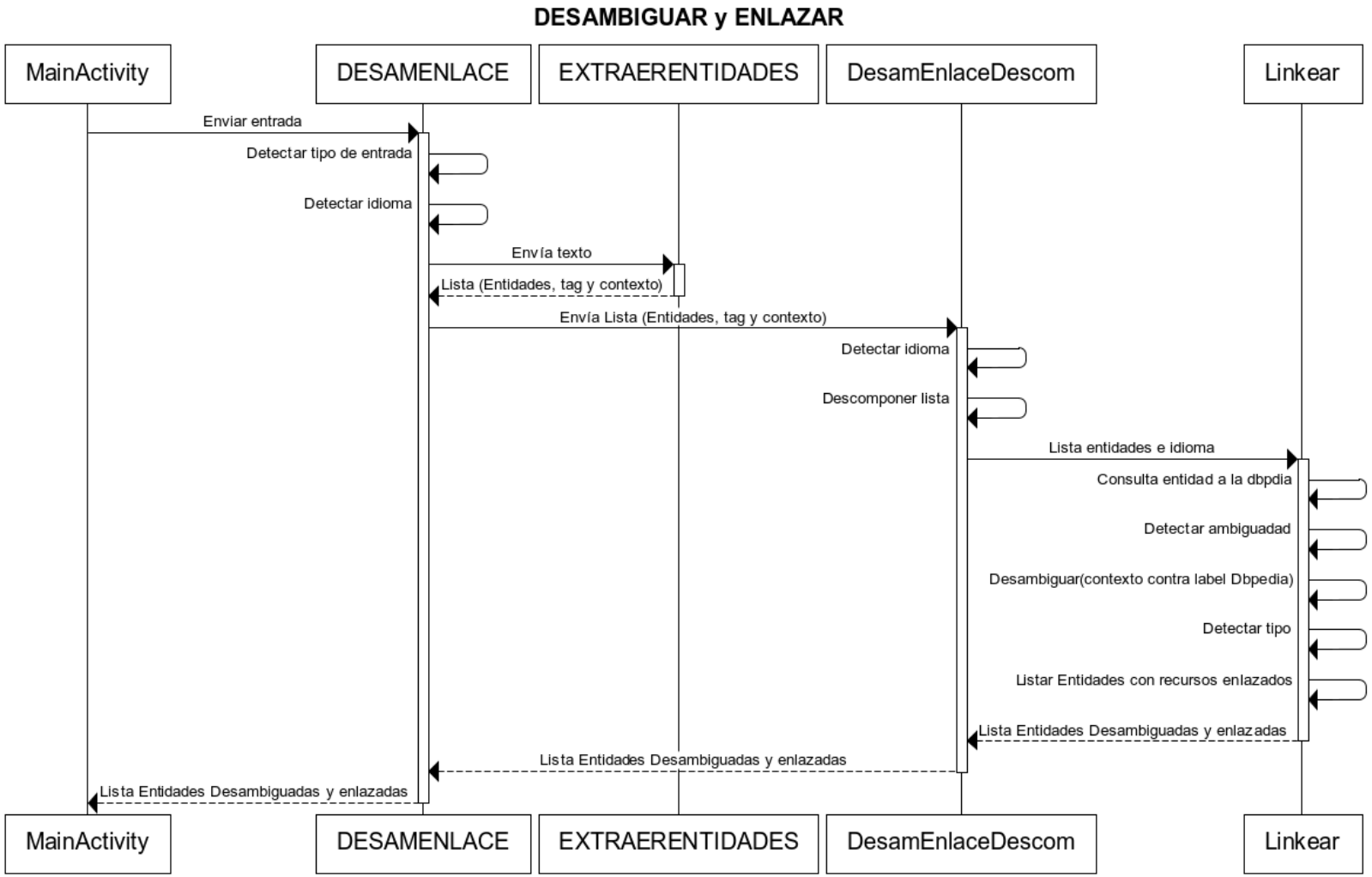

Figura 39 Diagrama de Secuencia Desambiguar y Enlazar

ANEXO 4: CASOS DE PRUEBA

Tokenizar

Caso de prueba:	Tokenizar
Propósito	Descomponer un texto ingresado en palabras separadas y con significado, con el fin de posteriormente manipular este conjunto de datos obtenidos para otras funcionalidades y procesos.
Prerrequisitos:	Disponer de una conexión de red

Datos de Prueba:	*Texto ingresado* = {
	texto vacío
	texto consistente de únicamente un espacio en blanco
	texto consistente de únicamente un signo de puntuación o carácter especial como: #, ", ', &, y <,
	texto consistente de varios signos de puntuación o carácteres especiales como: #, ", ', &, y <, o también solo números
	texto conformado por palabras, caracteres o números.
	texto en inglés o español con algunos palabras en otro idioma
	combinación de palabras en inglés o español.
	texto conformado totalmente por palabras en idioma distinto a inglés o español
	}

Pasos:	1. Entrar a la página 2. Ingresar cadena de texto 3. Seleccionar opción Tokenizar 4. Clic en Procesar 5. Visualizar texto tokenizado 6. Descargar JSON si se desea
Criterios de paso:	Resultados vacíos si el texto es vacío. Lista de palabras tokenizadas con sus respectivas etiquetas Mensaje de advertencia sobre idioma diferente español o inglés.
Criterios de fallo:	Texto sin etiquetas. Texto mal taggeado o mal separado.

Extraer Entidades

Caso de prueba:	Extraer entidades
Propósito	Extraer entidades relevantes a partir del texto ingresado y proporcionar un JSOn para ser consumido o procesado posteriormente
Prerrequisitos:	Disponer de una conexión de red

Datos de Prueba:	*Texto ingresado* = {
	texto consistente de únicamente un espacio en blanco
	texto consistente de únicamente un signo de puntuación o carácter especial como: #, ", ', &, y <,
	texto consistente de varios signos de puntuación o carácteres especiales como: #, ", ', &, y <, o también solo números
	texto conformado por palabras, caracteres o números.
	texto en inglés o español con algunos palabras en otro idioma
	combinación de palabras en inglés o español.
	texto conformado totalmente por palabras en idioma distinto a inglés o español
	}

Pasos:	1. Entrar a la página 2. Ingresar cadena de texto 3. Seleccionar opción Extraer entidades 4. Clic en Procesar 5. Visualizar lista entidades 6. Descargar JSON si se desea
Criterios de paso:	Resultados vacíos si el texto es vacío. Lista de entidades extraídas con el contexto de las mismas Mensaje de advertencia sobre idioma diferente español o inglés.
Criterios de fallo:	Lista de entidades sin contexto

Desambiguar y Enlazar

Caso de prueba:	Desambiguar y enlazar
Propósito	Al desambiguar el texto se especifica el contexto en el cual se encuentra una entidad para detallar con más precisión las etiquetas de la misma así como su significado dentro del contexto. Al enlazar se relaciona la entidad con un contexto específico.
Prerrequisitos:	Disponer de una conexión de red.

Datos de Prueba:	*Texto ingresado* = {
	texto vacío
	texto consistente de únicamente un espacio en blanco
	texto consistente de únicamente un signo de puntuación o carácter especial como: #, ", ', &, y <,
	texto consistente de varios signos de puntuación o carácteres especiales como: #, ", ', &, y <, o también solo números
	texto conformado por palabras, caracteres o números.
	texto en inglés o español con algunos palabras en otro idioma
	combinación de palabras en inglés o español.
	texto conformado totalmente por palabras en idioma distinto a inglés o español
	}
	}

Pasos:	1. Entrar a la página 2. Ingresar cadena de texto 3. Seleccionar opción Desambiguar y enlazar 4. Clic en Procesar 5. Visualizar lista entidades enlazadas 6. Descargar JSON si se desea
Criterios de paso:	Resultados vacíos si el texto es vacío. Presentación de una lista de entidades enlazadas y con su tipo de contexto. Mensaje de advertencia sobre idioma diferente español o inglés.
Criterios de fallo:	

ANEXO 5: PRUEBAS ADICIONALES

Tabla 15 Fases en inglés utilizadas para pruebas de los Servicios Web

N	Frase	Entidades	Enlaces Creados	Entidades Correctas	Porcentaje de extracción	Página
1	This course elaborates the history of Rome from its humble beginnings to the fifth century A.D. The first half of the course covers Kingship to Republican form; the conquest of Italy; Roman expansion: Pyrrhus, Punic Wars and provinces; classes, courts, and the Roman revolution; Augustus and the formation of empire. The second half of the course covers Virgil to the Vandals; major social, economic, political and religious trends at Rome and in the provinces. Emphasis is placed on the use of primary sources in translation.	29	25	29	100,00%	http://ocw.mit.edu/courses/history/21h-302-the-ancient-world-rome-spring-2005/
2	This course provides a brief introduction to the field of biocatalysis in the context of process design. Fundamental topics include why and when one may choose to use biological systems for chemical conversion, considerations for using free enzymes versus whole cells, and issues related to design and development of bioconversion processes. Biological and engineering problems are discussed as well as how one may arrive at both biological and engineering solutions.	17	8	16	94,12%	http://ocw.mit.edu/courses/chemical-engineering/10-492-2-integrated-chemical-engineering-topics-i-introduction-to-biocatalysis-fall-2004/

3	This class provides a general introduction to the diverse roles of microorganisms in natural and artificial environments. It will cover topics including: cellular architecture, energetics, and growth; evolution and gene flow; population and community dynamics; water and soil microbiology; biogeochemical cycling; and microorganisms in biodeterioration and bioremediation.	19	14	19	100,00%	http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-89-environmental-microbiology-fall-2004/
4	An analysis of historical structures is presented themed sections based around construction materials. Structures from all periods of history are analyzed. The goal of the class is to provide an understanding of the preservation of historic structures for all students.	13	11	13	100,00%	http://ocw.mit.edu/courses/architecture/4-448-analysis-of-historic-structures-fall-2004/
5	This course is devoted to the theory of Lie Groups with emphasis on its connections with Differential Geometry. The text for this class is Differential Geometry, Lie Groups and Symmetric Spaces by Sigurdur Helgason (American Mathematical Society, 2001). Much of the course material is based on Chapter I (first half) and Chapter II of the text. The text however develops basic Riemannian Geometry, Complex Manifolds, as well as a detailed theory of Semisimple Lie Groups and Symmetric Spaces.	23	16	20	86,96%	http://ocw.mit.edu/courses/mathematics/18-755-introduction-to-lie-groups-fall-2004/

Tabla 16 Fases en español utilizadas para pruebas de los Servicios Web

N	Frase	Entidades	Enlaces Creados	Entidades Correctas	Porcentaje de extracción	Página

1	La asignatura Organización y Actividad de las Administraciones Públicas pretende proporcionar una visión de conjunto sobre la organización de las Administraciones Públicas, estudiando la potestad organizativa que rige la Organización General del Estado, así como las autonómicas y locales. Además, se estudiará la competencia de cada una de ellas, así como sus regímenes jurídicos, los procedimientos y actos administrativos y el control sobre sus actuaciones.	18	6	18	100,00%	http://ocw.uc3m.es/derecho-administrativo/organizacion-y-actividad-de-las-administraciones-publicas-2013
2	El Aprendizaje Automático es un componente importante en campos tales como el análisis de datos o la minería de datos. Sin dejar de lado una introducción básica, esta asignatura pretende ser un complemento a otras que describan los fundamentos del aprendizaje automático, mediante el desarrollo de temas menos tratados, tales como los clasificadores basados en prototipos, el aprendizaje de funciones distancia, el aprendizaje sensible a la distribución y al coste, curvas ROC y de coste, o la inducción de fórmulas matemáticas mediante computación evolutiva.	20	9	20	100,00%	http://ocw.uc3m.es/ingenieria-informatica/aprendizaje-automatico-para-el-analisis-de-datos-2013

3	Se trata de una asignatura con un marcado carácter práctico que aborda una de las tecnologías que mayor repercusión han tenido en el desarrollo de la sociedad de la información al permitir la transmisión de cantidades masivas de información a grandes distancias utilizando la luz. Como dato relevante de este impacto, en 2009 se concedió el Premio Nobel de Física al investigador que demostró la capacidad de las fibras ópticas como medio de transmisión. En la asignatura se abordan no sólo las características del medio sino también de toda la tecnología fotonica que hace posible su funcionamiento. Siendo esta una de las 5 tecnologías que la Comisión Europea ha definido como habilitadores del futuro desarrollo de tecnologías competitivas dentro de su programa marco de investigación. Además, se abordan las prestaciones y se desarrollan enlaces reales.	27	8	24	88,89%	http://ocw.uc3m.es/tecnologia-electronica/dispositivos-y-medios-de-transmision-opticos
4	Tema 1: DEFINICIÓN DEL CONCEPTO DE IDEA Y PROYECTO Idea definición y concepto. Idea o pensamiento, aplicable al proyecto. Tema 2: TEORÍA DEL DISEÑO Definición de diseño. Relación del diseño con la idea. Estructura del diseño. Objetivos principales. Fases en el diseño. Metodología del diseño.	20	14	20	100,00%	http://ocw.unican.es/ensenanzas-tecnicas/proyectos-mineros-energeticos/programa
5	El objetivo de esta asignatura es que los alumnos adquieran conocimientos básicos de las redes de comunicaciones en general y de Internet en particular, haciendo énfasis en el análisis de procedimientos específicos de las redes de acceso y redes de medio compartido	9	4	9	100,00%	http://ocw.uc3m.es/ingenieria-telematica/arquitectura-de-redes-de-acceso-y-medio-compartido

Tabla 17 Pruebas con la frase 1 en inglés

Frase 1 en Inglés				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	100	100	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa función		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	29	29	100,00%
http://dbpedia-spotlight.github.io/demo		34	31	91,18%
http://textalytics.com/core/demo-test		24	24	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	29	25	86,21%
http://dbpedia-spotlight.github.io/demo		34	31	91,18%
http://textalytics.com/core/demo-test		24	3	12,50%

Tabla 18 Pruebas con la frase 2 en inglés

Frase 2 en Inglés				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	75	75	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa función		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	17	16	94,12%
http://dbpedia-spotlight.github.io/demo		19	18	94,74%
http://textalytics.com/core/demo-test		17	17	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	17	8	47,06%
http://dbpedia-spotlight.github.io/demo		19	19	100,00%
http://textalytics.com/core/demo-test		17	0	0,00%

Tabla 19 Pruebas con la frase 3 en inglés

Frase 3 en Inglés				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	57	57	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa función		
http://textalytics.com/core/demo-test				
		Entidades	Entidades **Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	19	19	100,00%
http://dbpedia-spotlight.github.io/demo		21	19	90,48%
http://textalytics.com/core/demo-test		13	13	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	19	14	73,68%
http://dbpedia-spotlight.github.io/demo		21	21	100,00%
http://textalytics.com/core/demo-test		13	0	0,00%

Tabla 20 Pruebas con la frase 4 en inglés

Frase 4 en Inglés				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	43	43	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa función		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	13	13	100,00%
http://dbpedia-spotlight.github.io/demo		19	18	94,74%
http://textalytics.com/core/demo-test		9	9	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y**	13	11	84,62%

http://dbpedia-spotlight.github.io/demo	**Enlazar**	19	19	100,00%
http://textalytics.com/core/demo-test		9	0	0,00%

Tabla 21 Pruebas con la frase 5 en inglés

Frase 5 en Inglés				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	90	90	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa funcion		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	23	20	86,96%
http://dbpedia-spotlight.github.io/demo		30	26	86,67%
http://textalytics.com/core/demo-test		21	21	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	23	16	69,57%
http://dbpedia-spotlight.github.io/demo		30	30	100,00%
http://textalytics.com/core/demo-test		21	0	0,00%

Tabla 22 Pruebas con la frase 1 en español

Frase 1 en Español				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	72	72	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa funcion		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	18	18	100,00%
http://dbpedia-spotlight.github.io/demo		40	14	35,00%
http://textalytics.com/core/demo-test		10	10	100,00%
		Entidades	**Enlaces**	**% enlace**

http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	18	6	33,33%
http://dbpedia-spotlight.github.io/demo		40	40	100,00%
http://textalytics.com/core/demo-test		10	0	0,00%

Tabla 23 Pruebas con la frase 2 en español

Frase 2 en Español				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	93	93	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa funcion		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	18	18	100,00%
http://dbpedia-spotlight.github.io/demo		53	24	45,28%
http://textalytics.com/core/demo-test		19	19	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	18	6	33,33%
http://dbpedia-spotlight.github.io/demo		53	53	100,00%
http://textalytics.com/core/demo-test		19	0	0,00%

Tabla 24 Pruebas con la frase 3 en español

Frase 3 en Español				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	142	142	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa funcion		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	27	24	88,89%
http://dbpedia-spotlight.github.io/demo		81	34	41,98%
http://textalytics.com/core/demo-test		22	22	100,00%

		Entidades	Enlaces	% enlace
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	27	8	29,63%
http://dbpedia-spotlight.github.io/demo		81	81	100,00%
http://textalytics.com/core/demo-test		22	0	0,00%

Tabla 25 Pruebas con la frase 4 en español

Frase 4 en Español				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	58	58	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa funcion		
http://textalytics.com/core/demo-test				
		Entidades	**Entidades Correctas**	**% extracción**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	20	20	100,00%
http://dbpedia-spotlight.github.io/demo		23	18	78,26%
http://textalytics.com/core/demo-test		11	11	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	20	14	70,00%
http://dbpedia-spotlight.github.io/demo		23	23	100,00%
http://textalytics.com/core/demo-test		11	0	0,00%

Tabla 26 Pruebas con la frase 5 en español

Frase 5 en Español				
Servicio	**Funcionalidad**	**N Tokens**	**# palabras**	**% exactitud**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Tokenizar**	43	43	100,00%
http://dbpedia-spotlight.github.io/demo		no implementa funcion		
http://textalytics.com/core/demo-test				

		Entidades	Entidades Correctas	% extracción
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Extraer Entidades**	9	9	100,00%
http://dbpedia-spotlight.github.io/demo		23	12	52,17%
http://textalytics.com/core/demo-test		6	6	100,00%
		Entidades	**Enlaces**	**% enlace**
http://taw02.utpl.edu.ec/anavisoers/interfaz	**Desambiguar y Enlazar**	9	4	44,44%
http://dbpedia-spotlight.github.io/demo		23	23	100,00%
http://textalytics.com/core/demo-test		6	0	0,00%