UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

TITULACIÓN DE INGENIERÍA EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Servicios Web para Extracción de Entidades desde Contenido HTML. Piloto en sitios con recursos abiertos OCW

Jhonny Zaruma

Tutor: Ing. Nelson Piedra

2014

ÍNDICE GENERAL

1. INTRODUCCIÓN

La web conforma un mudo 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.

2. OBJETIVOS

2.1. GENERAL

- Desarrollar Servicios Web que extraigan entidades a partir del contenido HTML, el piloto se ejecutará sobre contenidos OCW.

2.2. ESPECÍFICOS

- Creación de WS para Tokenización
- Creación de WS para Extracción de entidades
- Creación de WS para Desambiguación y Limpieza
- Creación de WS para Enlace con LOD-Cloud
- Creación de App Cliente, para integración de los WS

3. MARCO TEORICO

3.1. 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 máquinas) puedan comunicarse. (Berners-Lee, 1996)

3.1.1.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).

3.1.2.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*" (La Web Semántica y las Tecnologías del Lenguaje Humano)

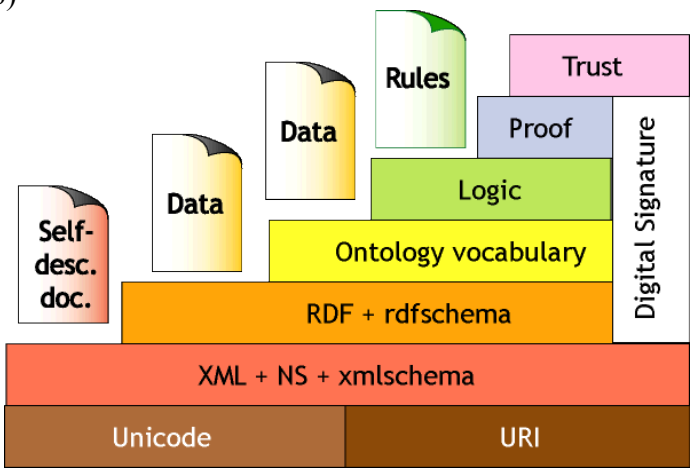


Figura 1 Arquitectura de la web semántica[1]

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

- **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 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 centra 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 esta 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 logrado construir diferentes estándares para poder representar y procesar la información de una manera mas sofisticada. Estos estándares que han permitido presentar los metadatos en un formato mas 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)

3.1.3.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 echa 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 del herramientas de anotación externa permiten asociar metainformación a páginas web, pero esta no se almacena dentro de la misma página sino que se guardada de forma externa en un repositorio.

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

- 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.

3.1.4.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. (W3C, RDF)

Ejemplo:

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

Considerando una simple oración en ingles:

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 oracion en RDF[2]

Representándolo gráficamente quedaría:

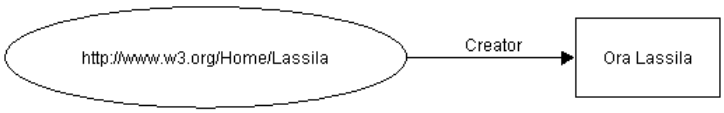


Figura 3 Representacion grafia RDF[2]

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

3.1.5.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: http://en.wikipedia.org/wiki/SPARQL

Extaer información de URIs, literales y nodos vacios

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 alamcenados como RDF o definidos mediante vistas RDF. También se pueden consultar patrones obligatorios y opcionales

[2] Tomado de http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

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, SPARQL Query Language for RDF)

3.1.5.1. 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, Sparql query Basicpatterns)

EJEMPLOS

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 gráfo de datos. En este caso existe un único patrón de tripleta con una sola variable (?title).

Consulta:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Resultado:

title
"SPARQL Tutorial

Concordancias múltiples

Datos:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

Consulta:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

Resultado:

[3] Todos los ejemplos de las consultas están tomadas de Tomado (W3C, Sparql query Basicpatterns)

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Concordancias de literales RDF

Datos:

```
@prefix dt:   <http://example.org/datatype#> .
@prefix ns:   <http://example.org/ns#> .
@prefix :     <http://example.org/ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

:x   ns:p    "cat"@en .
:y   ns:p    "42"^^xsd:integer .
```

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 "cat" no es el mismo literal RDF que "cat"@en:

```
SELECT ?v WHERE { ?v ?p "cat" }
```

Por otro lado la siguiente consulta si, donde la variable v se relaciona a :x porque le idioma se especifica.

```
SELECT ?v WHERE { ?v ?p "cat"@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: 42 es una forma abreviada de "42"^^<http://www.w3.org/2001/XMLSchema#integer>.

La consulta quedaría de esta manera:

```
SELECT ?v WHERE { ?v ?p 42 }
```

Construcción de grafos RDF

La consulta SELECT devuelve vínculos de variables. Encambio CONSTRUCT devuelve un grafo RDF. El grafo se construye sobre una plantilla que se usa para generar tripletas RDF basadas en los resultados de la concordancia con el patrón de grafo de la consulta.

Datos:

```
@prefix org:   <http://example.com/ns#> .

_:a  org:employeeName   "Alice" .
_:a  org:employeeId     12345 .

_:b  org:employeeName   "Bob" .
_:b  org:employeeId     67890 .
```

Consulta:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX org:    <http://example.com/ns#>

CONSTRUCT { ?x foaf:name ?name }
WHERE  { ?x org:employeeName ?name }
```

Resultado:

```
@prefix org: <http://example.com/ns#> .
 _:x foaf:name "Alice" .
 _:y foaf:name "Bob" .
```

serializado como RDF/XML

```
<rdf:RDF
   xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
   xmlns:foaf=http://xmlns.com/foaf/0.1/
   >
  <rdf:Description>
   <foaf:name>Alice</foaf:name>
  </rdf:Description>
  <rdf:Description>
   <foaf:name>Bob</foaf:name>
  </rdf:Description>
 </rdf:RDF>
```

3.1.5.2. Sintaxis SPARQL[4]

SINTAXIS DE EXPRESIONES RDF

Sintaxis para IRI

los IRIs son una generalización de los URIs [RFC3986] y son totalmente compatibles con URIs y URLs.

El conjunto de términos RDF incluye referencias RDF URI mientras que los términos de SPARQL contienen IRIs. Las referencias RDF URI que contienen los caracteres "<", ">", '"' (comilla doble), espacio, "{", "}", "|", "\", "^", y "`" no son IRIs. No está definido el comportamiento de una consulta SPARQL frente a las sentencias RDF compuestas de tales referencias URI.

Nombres prefijados: La palabra clave PREFIX asocia una etiqueta de prefijo con un IRI. Un nombre con prefijo está compuesto de una etiqueta de prefijo y una parte local, separados por dos puntos ":". Un nombre con prefijo se correponde (mapped) con un IRI mediante la concatenación del IRI asociado con el prefijo y la parte local. La etiqueta de prefijo y la parte local pueden estar vacías.

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

- <http://example.org/book/book1>
- BASE <http://example.org/book/>
  <book1>
- PREFIX book: <http://example.org/book/>
  book:book1

Sintaxis para literales

La sintaxis general para literales es una cadena de caracteres (entre comillas dobles, "...", o comillas simples, '...'), con una

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

etiqueta de idioma (antecedida por@) o un tipo de datos IRI o nombre prefijado (antecedido por ^^) opcionales.
Para facilitar los enteros pueden escribirse directamente (sin comillas y tipo de datos IRI explícito) y se interpretan como literales del tipo de datos xsd:integer; los decimales con '.' en el número pero sin exponente se interpretan como xsd:decimal; y los números con exponente se interpretan como xsd:double. Los valores del tipo xsd:boolean pueden también escribirse como true o false.
Ejemplos de sintaxis de literales SPARQL:

- "chat"
- 'chat'@fr con la etiqueta de idioma "fr"
- "xyz"^^<http://example.org/ns/userDatatype>
- "abc"^^appNS:appDataType
- '''The librarian said, "Perhaps you would enjoy 'War and Peace'."'''
- 1, igual a "1"^^xsd:integer
- 1.3, igual a "1.3"^^xsd:decimal
- 1.300, igual a "1.300"^^xsd:decimal
- 1.0e6, igual a "1.0e6"^^xsd:double
- true, igual a "true"^^xsd:boolean
- false, igual a "false"^^xsd:boolean

Sintaxis para variables

Las variables de interrrogación en consultas SPARQL tienen alcance global; el uso de un nombre de variable determinado en cualquier lugar de una consulta identifica a la misma variable. Las variables son prefijadas mediante "?" o "$"; la "?" o "$" que no forman parte del nombre de la variable. En una consulta, $abc y ?abc identifican la misma variable.

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/book/book1> dc:title ?title }
- PREFIX dc: <http://purl.org/dc/elements/1.1/>
  PREFIX : <http://example.org/book/>
  SELECT $title
  WHERE { :book1 dc:title $title }
- BASE <http://example.org/book/>
  PREFIX dc: <http://purl.org/dc/elements/1.1/>
  SELECT $title
  WHERE { <book1> dc:title ?title }

Listas de Predicado-Objeto

Los patrones de tripleta con un sujeto común peden expresarse asi:

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

Listas de objetos

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

- ?x foaf:nick "Alice" , "Alice_" .
- ?x foaf:nick "Alice" .
  ?x foaf:nick "Alice_" .
- ?x foaf:name ?name ; foaf:nick "Alice" , "Alice_" .
- ?x foaf:name ?name .
  ?x foaf:nick "Alice" .
  ?x foaf:nick "Alice_" .

3.1.6.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 mas 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)

3.1.6.1. 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 consistente, 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)

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)

3.2. RECURSOS EDUCATIVOS ABIERTOS OCW

3.2.1.OER (Open Educational Resources)

OER (Open Educational Resources) o REA (Recursos Educativos Abiertos) o 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 copiar, usar, adaptar y re-share ellos. Gama OER de los libros de texto a los planes de estudio, programas de estudio, notas de clase, tareas, exámenes, proyectos, audio, vídeo y animación. (UNESCO)

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:

- Contenido Educativos: 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: parte de legal de recursos, 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 maldición ya que exige mayor habilidad de nivel de búsqueda de información, selección, adaptación y evaluación, tomando en cuenta que las instituciones comparten más contenido 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. (Unesco, 2011)

3.2.2.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 (UPM)**:** 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:

- 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:

- Consulta: Acceso a toda la información que mantiene.
- Colaboración: Puede obtener y brindar información OCW.
- Visibilidad colectiva: Aumentar la posibilidad colaborativa en la red.
- Sostenibilidad: asegurar que el intercambio abierto de recursos educativos tiene un futuro productivo.

3.3. PROCESAMIENTO DEL LENGUAJE NATURAL

El " Procesamiento del Lenguaje Natural " (NLP) 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. (PNL)

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.

3.4. SERVICIOS WEB

Los servicios web(Web Service) 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 diseñados 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)

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. WS soporta

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)

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):

- 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 especifica, 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.

3.4.1.ARQUITECTURA RPC (REMOTE PROCEDURE CALL)

En el paper de 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. Información puede ser transportado de la persona que llama al destinatario de la llamada en los parámetros y puede volver a aparecer en el resultado del procedimiento. (Birrell & Nelson)

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

La idea básica parece simple y elegante, 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 pasados, 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

3.4.2.ARQUITECTURA ORIENTADA A SERVICIOS

Arquitecturas orientadas a servicios se han utilizado durante muchos años. SOA es de una 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)

3.4.3.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)

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)

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.

3.5. PROGRAMAS Y LIBRERÍAS

3.5.1.PYTHON

Python es un lenguaje de programación dinámico y potente que es utilizado en varios dominios de aplicación . A Python se lo compara con Tcl, Perl, Ruby, Scheme o Java. Sus características mas relevantes son (Python):

- 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 te 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 a la de Java y máquinas virtuales. NET.

Python es libre de usar, incluso para los productos comerciales, por su aprobada por OSI licencia de código abierto. (Python)

3.5.1.1. NLTK

NLTK es una plataforma para python que trabajar 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)

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)

Librerías de NLTK utilizadas:

Word_tokenize: es una funcionalidad de NLTK, que sirve para la clasificación de la palabras, lo clasifica mediante etiquetas.

Tag	Meaning	Examples
ADJ	adjective	*new, good, high, special, big, local*
ADV	adverb	*really, already, still, early, now*
CNJ	conjunction	*and, or, but, if, while, although*
DET	determiner	*the, a, some, most, every, no*
EX	existential	*there, there's*
FW	foreign word	*dolce, ersatz, esprit, quo, maitre*
MOD	modal verb	*will, can, would, may, must, should*
N	noun	*year, home, costs, time, education*
NP	proper noun	*Alison, Africa, April, Washington*
NUM	number	*twenty-four, fourth, 1991, 14:24*
PRO	pronoun	*he, their, her, its, my, I, us*
P	preposition	*on, of, at, with, by, into, under*
TO	the word *to*	*to*
UH	interjection	*ah, bang, ha, whee, hmpf, oops*
V	verb	*is, has, get, do, make, see, run*
VD	past tense	*said, took, told, made, asked*
VG	present participle	*making, going, playing, working*
VN	past participle	*given, taken, begun, sung*
WH	*wh* determiner	*who, which, when, what, where, how*

Figura 4 Conjuto de etiquetas[5]
http://www.nltk.org/book/ch05.html

RegexpTagger: sirve para hacer una clasificación y etiquetado de las palabras pero con expresiones regulares en base ha patrones. Por ejemplo se puede decir que todas las palabras en ingles que termina en *ed* son verbos en pasado participio, asi como también las palabras que terminen en *'s* son nombres posesivos.

```
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 expreciones regulares para etiquetado con RegexpTager[5]

Chunck: es un paso preliminar y útil para la extracción de información, sé que 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:

1. Se etiqueta una sentencia
2. Se hace un Chunk de la sentencia etiquetada
3. Analizar el árbol obtenido para extraer información

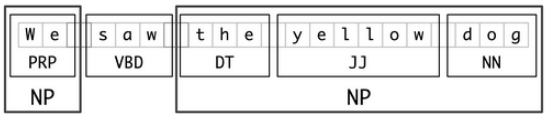


Figura 6 Segmentación y Etiquetado Chunk[6]

[5] Tomado de http://www.nltk.org/book/ch05.html
[6] Tomado de http://www.nltk.org/book/ch07.html

4. PROBLEMÁTICA

4.1. 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 los OER y OCW, estos manejan una gran cantidad de los recursos anteriormente mencionados, dichos contenidos estan 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 conceptos o 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
- Tags
- Contenido, dentro de este existe todavía más datos que pueden ser relevantes:
    - Menciones a papers
    - Autores relacionados

- Emails
- 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, supongamos que se realiza una búsqueda sobre aplicaciones de la Web Semántica en ámbito médico, al llevar a cabo una búsqueda basada en ciertas palabras claves las resultantes de dicha búsqueda sería bastante amplia; 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,* por que mediante la generación de etiquetas se puede llevar a cabo una búsqueda más exacta, precisa y eficaz sobre un tema o curso en específico.

5. SOLUCIÓN

5.1. APROXIMACIÓN

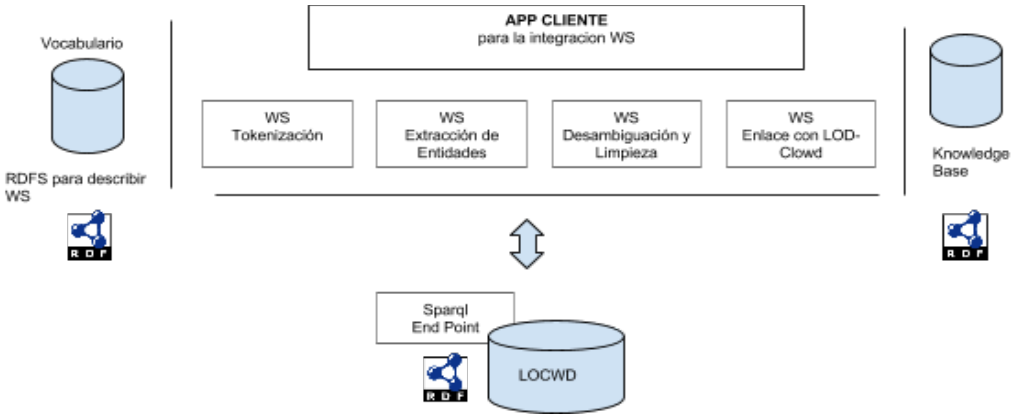
Se planea realizar la extracción de entidades como: lugares, países, ciudades, áreas de conocimientos, organizaciones académicas, autores.

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-Clowd, estos servicios web se los desarrollara con anotaciones semánticas, para que puedan permitir la interoperabilidad entre los servicios existentes.

Finalmente la información obtenida será almacenada repositorios que permitan tener permanecía de la información.

En la siguiente grafica se muestra la estructura mencionada:



5.2. DESCRIPCIÓN DE COMPONENTES.

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

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.
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. **Metadata,** como son URIs, URLs, emails, números telefónicos.
d. Áreas de conocimiento .
e. **Lugares,** como son países, ciudades, provincias, localizaciones.
f. **Eventos,** como conferencias, simposios, congresos, encuentros .

3. **LOCWD**, son las siglas de Linked OpenCourseWare Data que es un vocabulario que implementa datos enlazados en contenidos abiertos educacionales, dicho vocabulario reutiliza un conjunto de vocabularios RDF. Cada vocabulario incluye un conjunto de términos y clases que son comunes a nuestro dominio particular de conocimiento. (Piedra , Chicaiza, & Lopez, Combining Linked Data and Mobiles Devices to improve access to OCW)
4. **WS Tokenización,** se definirá este servicio web para el manejo del texto, sea:
    a. 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.
    b. Tokenización, mediante este servicio se tokenizará cada palabra perteneciente al texto, mediante tokenización de oraciones completas para no perder el sentido ni la sintaxis.
    c. Taggeo, 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.
5. **WS Extracción de Entidades,** mediante este servicio se analizará y procesará el texto para encontrar aquellas 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.
6. **WS Desambiguación y Limpieza,** 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. Limpieza, mediante esta tarea se eliminarán aquellos caracteres o palabras que puedan alterar el sentido de la entidad extraída.

7. **WS Enlace con LOD-Clowd:** 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.
8. **APP Cliente:** es la interfaz que permitirá integrar todos los servicios definidos anteriormente, para hacer más fácil y sencillo el consumo de los mismos.

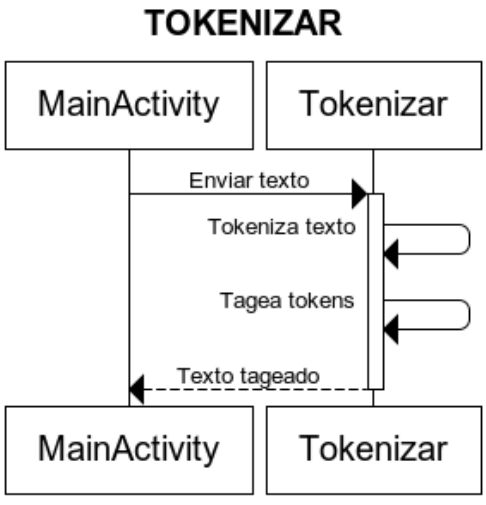
5.3. PROTOTIPOS

5.3.1. WS TOKENIZACIÓN

El diseño de este prototipo se realiza la tokenización de texto. Para la elaboración de este prototipo se realizo 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 tageo de los tokens antes mencionados se establece etiquetas para cada palabra del texto con *RegexpTagger de NLTK* y se tagea con la función *pos_tag*.
- Este prototipo nos devuelve una lista con los tokens tageados

ARQUITECTURA
Diagrama de secuencia



IMPLEMENTACIÓN

Parámetros.

Como parámetro de entrada se tiene :

- *Sentencia,* consiste en una variable de tipo carácter o string que almacena el texto a ser analizado, procesado y etiquetado.

Como parámetro de salida:

- *Lista tokens,* es 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:

```
('El', 'DT')
('Comit\xc3\xa9', 'NN')
('del', 'DT')
('Patrimonio', 'NNP')
('Mundial', 'NN')
('reunido', 'VBD')
('en', 'IN')
('Phnom', 'NNP')
('Penh', 'NNP')
```

Funciones.

Esta clase implementa funciones para:

- Tokenizar texto
- Taggear texto tokenizado

Para acceder al servicio de esta clase se lo hace mediante la función:

```
def tagear(self,sentencia):
    idioma=gs.detect(sentencia)
    lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Albanian","ar":"Arabic","
    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=[]
    return Lista
```

la misma recibe como parámetro el texto ingresado que se almacena en la variable `sentencia`.

Se procede a detectar el idioma del texto mediante la función

```
def tagear(self,sentencia):
    idioma=gs.detect(sentencia)
    lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Albanian","ar":"Arabic",'
    idiomaCompleto=lenguajes[idioma]
```

de esta manera se determina el lenguaje en que se encuentra escrito el texto ingresado para posteriormente llevar a cabo la tokenización y taggeo del mismo.

Pueden darse 3 casos dentro de la tokenización y taggeo:

A. Idioma Español.

Si el idioma está en español se procede a llamar a la función:

```
def tagearSentenciaEs(self,tags):
    tags=word_tokenize(tags)
    patterns = [
    (r'^@\w+', 'NNP'),
    (r'^\d+$', 'CD'),
    (r'.*ing$', 'VBG'),
    (r'.*ment$', 'NN'),
    (r'.*ful$', 'JJ'),
    (r'la', 'IN'),
    (r'La', 'IN'),
    (r'de', 'IN'),
    (r'el', 'IN'),
    #(r'.*', 'NN')
    ]
    patterns = [
    (r'^,$', ','),
    (r'^;$', ';'),
    (r'^:$', ':'),
    (r'^\.$', '.'),
    (r'^-$', '-'),
    (r'^/$', '/'),
    (r'^\($', '('),
    (r'^\)$', ')'),
    (r'^\-$', '-'),
    (r'^(de|De|con|Con|En|en|\xe2\x80\x93)$', 'IN'),
    (r'^(A|A|(a|A)nte|(b|B)ajo|(c|C)on|(c|C)ontra|(d|D)esde|(e|E)n|(e|E)ntre|(h|H)acia|(h|H
    (r'^(el|El|EL|Los|los|la|La|Las|las|del)$', 'DT'),
    (r'^(éste|Éste|ésta|Ésta|esto|Esto|éstos|Éstos|éstas|Éstas|ése|Ése|ésa|Ésa|eso|Eso|ésos
    (r'^(este|Este|esta|Esta|esto|Esto|estos|Estos|estas|Estas|ese|Ese|esa|Esa|eso|Eso|esos
    (r'^(yo|Yo|tú|Tú|tu|él|Él|Ella|ella|nosotros|Nosotros)$', 'PRP'),
    (r'^(Así|Asi|asi|así|más|Más)$','RB'),
    (r'^(Así|Asi|asi|así|más|Más)$','RB'),
    (r'^(Y|y|o|O|U|u|ni|Ni|ya|Ya)$','CC'),
    (r'^(véase)$','VR'),
    (r'^(admitir|afectar|estar|apuntar|permitir|responder|aparecer|aplicar|discutir|arregla

    ]
    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,tags[index+1])
            else:
                tTra=self.traducir(t,tags[index])
            nuevotag=pos_tag(nltk.word_tokenize(tTra))[0]
            nuevotag=(t,nuevotag[1])
            tags[index]=nuevotag
    return tags
```

esta función recibe el texto ingresado y lo tokeniza mediante el método:

```
tags=word_tokenize(tags)
```

se define patrones para el etiquetado mediante NLTK (en la variable ***patterns***), dicho etiquetado se realiza mediante el método:

```
regexp_tagger = nltk.RegexpTagger(patterns)
```

en el cual se clasifica cada palabra encontrada en el texto ingresado, con una etiqueta que se encuentra definida según la variable ***patterns***.

Luego de este método, si existen palabras que no han podido ser etiquetadas, las mismas serán traducidas mediante la función:

```
def traducir(self,token, token2):
    #print token
    if len(token)>2:
        tokenTra=gs.translate(token, 'en')
    else:
        #tokenTra= gs.translate(token+' '+token2, 'en')
        tokenTra= gs.translate(token+' '+token+' '+token+' '+token, 'en')
        #print tokenTra
        tokenTra=word_tokenize(tokenTra)
        tokenTra=tokenTra[0]
    tokenTra=word_tokenize(tokenTra)
    tokenTra=tokenTra[len(tokenTra)-1]
    #print tokenTra
    if token.islower()==True:
        tokenTra=tokenTra.lower()
    return tokenTra
```

con esta función se reciben los tokens, si se encuentran en español se los traduce en inglés y devuelve la misma mediante la variable ***tokenTra***, una vez que se regresa esta palabra a la función principal se procede a tokenizarla mediante el corpus en inglés de NLTK, y se almacena la misma palabra en español con el token extraído de su versión traducida.

Luego de todos estos procesos se regresa una lista (la variable ***tags***) que contiene todas las palabras del texto separadas y clasificadas según etiquetas o tags.

B. Idioma inglés.

En este caso se llama al método:

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

y puesto que el texto está en inglés y el corpus del NLTK está especificado completamente en este idioma, resulta más sencillo y simple llevar a cabo el taggeo de las palabras; igual que el método anterior se devuelve una lista de tags.

C. Idioma no encontrado.

Se presentará un mensaje en el cual se especifica que el idioma no se encuentra soportado por la clase.

5.3.2.WS EXTRACCIÓN DE ENTIDADES

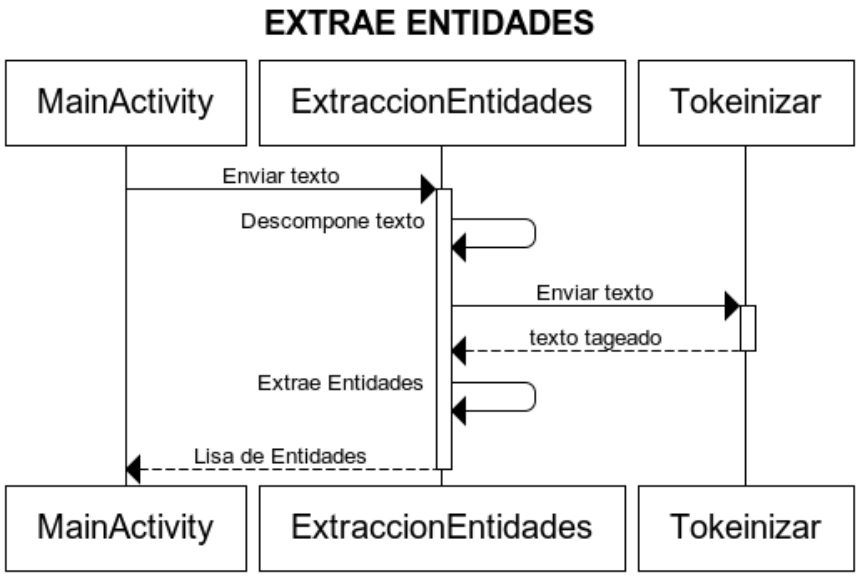
El diseño de este prototipo se realiza la extracción de entidades. Para la elaboración de este prototipo se realizo lo siguiente:

- El texto introducido se lo envía a descomponer con la función sent_tokenize de la librería NLTK, luego se lo envía al prototipo anterior(WS tokenización) obteniendo una lista con los tokens tageados.
- Utilizando la función *RegexpParser*(función de NLTK) se realiza un conjunto de expresiones regulares para la extracción de las entidades en base a los tags recuperados.

- Con la función *chunker* en fusión con exprecines regulares realizadas, se extraen las entidades.

- **ARQUITECTURA**

Diagrama de secuencia



- **IMPLEMENTACIÓN**

Parámetros.

Como parámetro de entrada se tiene :

- *Entrada,* esta variable puede ser de tipo string o list que almacena el texto o una lista de tokens tageados a ser analizados y procesados .

Como parámetro de salida:

- *Lista entidades,* es una lista desglosada mediante PLN, en la cual se encuentra las entidades extraídas:

```
[[('Jap\xc3\xb3n', 'NNP')], 'El volc\xc3\xa1n Fuji, 13 de Enero del 2014
[[('Educaci\xc3\xb3n', 'NNP')], 'El volc\xc3\xa1n Fuji, 13 de Enero del 2
[[('Unesco', 'NNP')], 'El volc\xc3\xa1n Fuji, 13 de Enero del 2014 en  es
[[('Patrimonio', 'NNP'), ('Mundial', 'NN')], 'El Comit\xc3\xa9 del Patrim
[[('Phnom', 'NNP'), ('Penh', 'NNP')], 'El Comit\xc3\xa9 del Patrimonio Mu
[[('Fujiyama', 'NNP')], 'Los japoneses consideran sagrado el Fujiyama (mo
[[('Fuyi', 'NNP')], 'Los japoneses consideran sagrado el Fujiyama (monte
[[('Unesco', 'NNP')], 'Los expertos de la Unesco tambi\xc3\xa9n inscribie
[[('Honghe', 'NNP'), ('Hani', 'NNP')], 'Los expertos de la Unesco tambi\x
[[('China', 'NNP')], 'Los expertos de la Unesco tambi\xc3\xa9n inscribier
[[('Parque', 'NNP'), ('Nacional', 'NNP'), ('Sehlabathebe', 'NNP')], 'Los
[[('Lesoto', 'NNP')], 'Los expertos de la Unesco tambi\xc3\xa9n inscribie
[[('Hongye', 'NNP'), ('Hani', 'NNP'), ('cubren', 'NN'), ('16.603', 'CD')]
```

Funciones.

Esta clase implementa funciones para:

- Recuperar Entidades a partir de:
  - Texto
- Lista de tokens de entidades extraidas con sus respectivos tag

Para acceder al servicio de esta clase se lo hace mediante la función:

```
def ExtEntidades(self,Entrada):
    if type(Entrada) is str:
        gs= goslate.Goslate()
        idioma=gs.detect(Entrada)
        lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Al
        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=[]
    elif type(Entrada) is list:
        Lista=self.recuperarEntidadesToken(Entrada)
    else:
        mensaje='Tipo de datos no soportados'
        print mensaje
        Lista=[]
    return Lista
```

la misma recibe como parámetro el texto ingresado que se almacena en la variable `Entrada`.

Se procede a detectar el tipo de entrada mediante la función

```
    if type(Entrada) is str:
```

```
    elif type(Entrada) is list:
```

Pueden darce 3 casos de tipo de entrada:

1. **Entrada tipo String**

   Si el tipo es String se procede a detectar el idioma mediante:

   ```
   gs= goslate.Goslate()
   idioma=gs.detect(Entrada)
   lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Al
   idiomaCompleto=lenguajes[idioma]
   ```

   de esta manera se determina el lenguaje en que se encuentra escrito el texto ingresado para posteriormente llevar a cabo la tokenización y taggeo del mismo.

   Pueden darse 3 casos dentro de la extracción de entidades:

   A. **Idioma Español.**

      Si el idioma del string de entrada esta en español llama al siguiente método:

```
def recuperarEntidadesEs(self,texto):
    chunker = RegexpParser("""
    ENTI:
        {<NNP|NNPS>+<NNP|NNPS|NN|NNS>}|
        {<NN|NNS>+<NN|NNS><JJ>}
        {<NNP|NNPS><IN|DT><NNP|NNPS|NN|NNS>}
        {<NN|NNS><JJ>|<JJ><NN|NNS>}
        {<NNP|NNPS>}
    ENTIDACOMP:
        {<DT><NN|NNS><ENTI>}
        {<DT><NN|NNS><IN><ENTI>}
        {<ENTI>(<IN>|<IN><DT>)<ENTI|NN|NNS>}
        {<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):
        tags=ObjTag.tagear(sentence)
        parsed = chunker.parse(tags)
        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
```

Esta función recibe el texto y lo envía a la case de Tokenizacion para ser procesada y nos devuelve los tokens son los tag de texto enviado, mediante la siguiente método:

```
ObjTag = Tokenizar()
Lista = []
Lista2 = []
for sentence in sent_tokenize(texto):
    tags=ObjTag.tagear(sentence)
```

Se define patrones para la extracción mediante NLTK(en la variable **chunker**), dicha extracion se la realiza mediante el método :

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

en el cual se clasifica y extrae las entidades mediante los tokens con sus tag en constrate a los patrones definidos en la variable **chuncker.**

Luego de todos estos procesos se regresa una lista (la variable ***Lista2***) que contiene todas las entidades extraídas con sus etiquetas y con el contexto de donde se las extrajo.

B. Idioma inglés

En este caso se llama al método:

```
def recuperarEntidadesEn(self,texto):
    ObjTag = Tokenizar()
    Lista = []
    Lista2= []
    for sentence in sent_tokenize(texto):
        tags=ObjTag.tagear(sentence)
        parsed = ne_chunk(tags)
        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
```

Esta función recibe el texto y lo envía a la case de Tokenizacion para ser procesada y nos devuelve los tokens son los tag de texto enviado, mediante la siguiente método:

```
    ObjTag = Tokenizar()
    Lista = []
    Lista2= []
    for sentence in sent_tokenize(texto):
        tags=ObjTag.tagear(sentence)
```

Y como el texto esta en ingles y el corpus de NLTK esta especificado para ese idioma, es mas sencillo la extracción, y se lo hace con el siguiente método:

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

se clasifica y extrae las entidades mediante los tokens con sus tag en constrate a los patrones definidos por NLTK en la variable **ne_chunck.**

Luego de todos estos procesos se regresa una lista (la variable ***Lista2***) que contiene todas las entidades extraídas con sus etiquetas y con el contexto de donde se las extrajo.

C. **Idioma no encontrado.**
Se presentará un mensaje en el cual se especifica que el idioma no se encuentra soportado por la clase.

2. **Entrada tipo Lista**
Si el tipo es un tipo list(lista) llama a este método:

```
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
```

Y como ya la entrada ya es una lista de tokens con sus respectivos tag en ingles, para corpus de NLTK le resulta mas sencillo para poder extraer las entidades; este método devuelve la lista de las entidades extraidas.

3. **Tipo no encontrado**

Se presentará un mensaje en el cual se especifica que el tipo de entrada no se encuentra soportado por la clase.

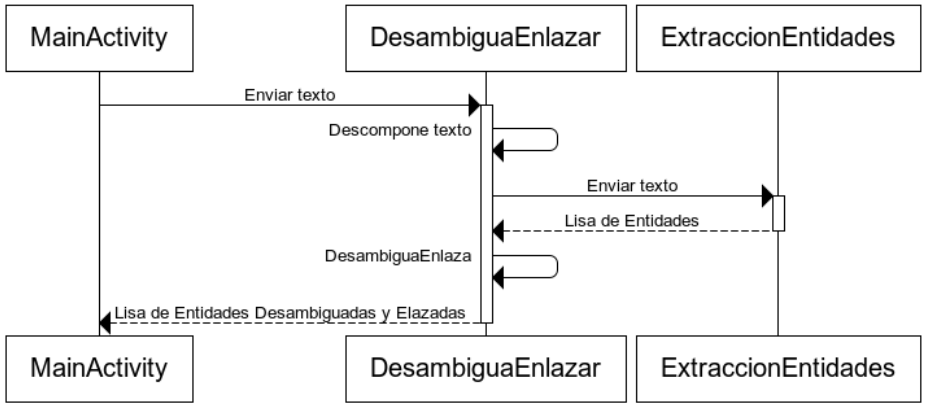
5.3.3.WS DESAMBIGUACIÓN Y ENLACE

El diseño de este prototipo se realiza la extracción de entidades. Para la elaboración de este prototipo se realizo lo siguiente:

- El texto introducido se lo envía al prototipo anterior(WS tokenización) obteniendo una lista con las entidades extraidas.
- Utilizando consultas Sparql se realiza consultas ala dbpedia para desambiguar y extraer Uris para enlazarlos.

- **ARQUITECTURA**

Diagrama de secuencia



- **IMPLEMENTACIÓN**

Parámetros.

Como parámetro de entrada se tiene :

- *Entrada,* esta variable puede ser de tipo string que almacena el texto o una lista con las entidades y su contexto a ser analizados y procesados .

Como parámetro de salida:

- *Lista entidades enlazadas,* es una lista desglosada, en la cual se encuentra las entidades extraídas, desambiguadas y enlazadas:

```
[u'Quito', [[1, u'http://dbpedia.org/resource/Quito', 'GEO']]]
[u'Ecuador', [[1, u'http://dbpedia.org/resource/Ecuador', 'GEO']]]
[u'Peru', [[1, u'http://dbpedia.org/resource/Peru', 'GEO']]]
[u'Argentina', [[1, u'http://dbpedia.org/resource/Argentina', 'GEO']]]
[u'Israel', [[1, u'http://dbpedia.org/resource/Israel', 'GEO']]]
[u'Korea', [[1, u'http://dbpedia.org/resource/Korea', 'ORG']]]
[u'China', [[1, u'http://dbpedia.org/resource/China', 'GEO']]]
[u'North Korea', [[1, u'http://dbpedia.org/resource/North_Korea', 'GEO']]]
[u'Loja', [[1, u'http://dbpedia.org/resource/Loja', 'GEO']]]
[u'Ambato', [[2, u'http://dbpedia.org/resource/Ambato,_Ecuador', 'GEO'], [2,
tp://dbpedia.org/resource/Ambato-Madagascar', 'Sin Tipo']]]
```

Funciones.

Esta clase implementa funciones para:

- Desambigua y Enlaza entidades a partir de:
  - Texto
  - Lista con entidades
- Lista de entidades desambiguadas con su enlace a la dbpedia.

Para acceder al servicio de esta clase se lo hace mediante la función:

```
def DesamEnlace(self,Entrada):
    DesamEnl=[]
    if type(Entrada) is str:
        gs= goslate.Goslate()
        idioma=gs.detect(Entrada)
        lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Alb
        idiomaCompleto=lenguajes[idioma]

        if idioma =='es' or idioma == 'en':
            Entrada=ObjExt.recuperarEntidadesEs(Entrada)
            DesamEnl=DesamEnlaceDescom(Entrada)
        else:
            mensaje='El idioma %s no es soportado'%idiomaCompleto
            #print mensaje
            mensaje=gs.translate(mensaje, idioma)
            #print mensaje
            DesamEnl=[]
    elif type(Entrada) is list:
        DesamEnl=DesamEnlaceDescom(Entrada)
    else:
        mensaje='Tipo de datos no soportados'
        #print mensaje
        DesamEnl=[]

    return DesamEnl
```

la misma recibe como parámetro el texto ingresado que se almacena en la variable `Entrada`.

Se procede a detectar el tipo de entrada mediante la función

```
    if type(Entrada) is str:
```

```
    elif type(Entrada) is list:
```

Pueden darce 3 casos de tipo de entrada:

1. **Entrada tipo String**

   Si el tipo es String se procede a detectar el idioma mediante:

   ```
   gs= goslate.Goslate()
   idioma=gs.detect(Entrada)
   lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Al
   idiomaCompleto=lenguajes[idioma]
   ```

   de esta manera se determina el lenguaje en que se encuentra escrito el texto ingresado para posteriormente llevar a cabo la tokenización y taggeo del mismo.

   Pueden darse 2 casos dentro de la desambiguación y enlace de entidades:

   A. **Idioma Español o Ingles.**

      Si el idioma del string de entrada esta en español o ingles llama al servicio web de extracción de entidades:

```
def ExtEntidades(self,Entrada):
    if type(Entrada) is str:
        gs= goslate.Goslate()
        idioma=gs.detect(Entrada)
        lenguajes={"auto":"Detect language","af":"Afrikaans","sq":"Al
        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=[]
    elif type(Entrada) is list:
        Lista=self.recuperarEntidadesToken(Entrada)
    else:
        mensaje='Tipo de datos no soportados'
        print mensaje
        Lista=[]
    return Lista
```

Este servicio web nos devolverá una lista desglosada mediante PLN, en la cual se encuentra las entidades extraídas con su contexto.

```
[[('Jap\xc3\xb3n', 'NNP')], 'El volc\xc3\xa1n Fuji, 13 de Enero del 2014
[[('Educaci\xc3\xb3n', 'NNP')], 'El volc\xc3\xa1n Fuji, 13 de Enero del 2
[[('Unesco', 'NNP')], 'El volc\xc3\xa1n Fuji, 13 de Enero del 2014 en  es
[[('Patrimonio', 'NNP'), ('Mundial', 'NN')], 'El Comit\xc3\xa9 del Patrim
[[('Phnom', 'NNP'), ('Penh', 'NNP')], 'El Comit\xc3\xa9 del Patrimonio Mu
[[('Fujiyama', 'NNP')], 'Los japoneses consideran sagrado el Fujiyama (mo
[[('Fuyi', 'NNP')], 'Los japoneses consideran sagrado el Fujiyama (monte
[[('Unesco', 'NNP')], 'Los expertos de la Unesco tambi\xc3\xa9n inscribie
[[('Honghe', 'NNP'), ('Hani', 'NNP')], 'Los expertos de la Unesco tambi\x
[[('China', 'NNP')], 'Los expertos de la Unesco tambi\xc3\xa9n inscribier
[[('Parque', 'NNP'), ('Nacional', 'NNP'), ('Sehlabathebe', 'NNP')], 'Los
[[('Lesoto', 'NNP')], 'Los expertos de la Unesco tambi\xc3\xa9n inscribie
[[('Hongye', 'NNP'), ('Hani', 'NNP'), ('cubren', 'NN'), ('16.603', 'CD')]
```

De este modo la entrada la convertimos en una lista.

B. Idioma no encontrado.

Se presentará un mensaje en el cual se especifica que el idioma no se encuentra soportado por la clase.

2. Entrada tipo Lista

Si el tipo es un tipo list(lista) llama a este método:

```
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]
        link=self.Linkear(entidades,contexto,idioma)
        DesamEnl.append(link)
    return DesamEnl
```

Este método separara la lista y envía las entidades separadas con el contexto a este otro método para ser procesadas:

```
def Linkear(self,entidad,contexto,idioma):
    linkear=[]
    entidad=entidad.decode('utf-8')
    try:
        results = self.ConsuDbpedia(entidad,'rdfs:label',idioma)
        #results = self.ConsuDbpedia(entidad,'rdfs:label','en')
        if results["results"]["bindings"]==[]:
            if idioma=='en':
                results = self.ConsuDbpedia(entidad,'rdfs:label','es')
            else:
                results = self.ConsuDbpedia(entidad,'rdfs:label','en')
    except Exception, e:
        print e
        return
    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 ==[]:
            UrlyTipo=[]
            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
                    try:
                        results4 = self.ConsuDbpedia2(link,'rdf:type')
                    except Exception, e:
                        #print e
                        continue

                    results4 = results4["results"]["bindings"]
                    TipoFinal=self.TipoEntidad(results4)
                    UrlyTipo.append([link,TipoFinal])
                linkear=self.SelecionaTipo(UrlyTipo,contexto)
            else:
                TipoFinal=self.TipoEntidad(results2)
                linkear=[[1,link,TipoFinal]]

        else:
            TipoFinal=self.TipoEntidad(results2)
            linkear=[[1,link,TipoFinal]]
        #break
    LisDesyEnlase=[entidad,linkear]
    return LisDesyEnlase
```

Esta función recibe la entidad con su contexto y el idioma en el que esta, una ves obtenido esto se envía la entidad para ser procesada en este otro método:

```
def ConsuDbpedia(self,entidades,predicado,idioma):
    sparql = SPARQLWrapper("http://dbpedia.org/sparql")
    sparql.setQuery("""
    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    SELECT *
    WHERE {
    ?label %s "%s"@%s
    FILTER regex(str(?label), "http://dbpedia.org/resource/", "i")
    }
    """%(predicado,entidades,idioma))
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()
```

Este método recibe la entidad a ser procesada, así como también su idioma y el valor de predicado para poder realizar una consulta sparql a la dbpedia, este método nos devolverá una lista con todos los resultados encontrados en la consulta. Dependiendo de lo q nos devuelva el método anterior se pueden dar dos casos:

A. Entidad Ambigua.

Si la entidad es ambigua, es decir la lista obtenida del método ConsuDbppedia devolvió un enlace a un recurso que contiene un predicado *wikiPageDisambiguates,* se procede a realizar otra consulta para extraer todos los recursos relacionados ha esa entidad:

```
def ConsuDbpedia3(self,link,predicado):
    sparql = SPARQLWrapper("http://dbpedia.org/sparql")
    sparql.setQuery("""
    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    PREFIX rdf: <http://www.w3.org/1999/02/20002-rdf-syntax-ns#>
    SELECT *
    WHERE {
    <%s> %s ?label
    FILTER regex(str(?label), "http://dbpedia.org/resource/", "i")
    }
    """%(link,predicado))
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()
```

Ese método contiene otra consulta sparql que nos devuelve la lista con todos los recursos posibles para la entidad extraída. Esta lista se lo envía a este método:

```
def SelecionaTipo(self,listaTipos,contexto):
    listafrec=[]
    for link in listaTipos:
        try:
            LabelsExtraidos=self.ConsuDbpediaExtraLabel(link[0],'rdfs:label')
        except Exception, e:
            #print e
            continue
        LabelsExtraidos=LabelsExtraidos["results"]["bindings"]
        labels=(' '.join(self.eliminasignos(c["label"]["value"]) for c in LabelsExtraidos)).lo
        labelsSinDuplicados=list(set(labels))
        count=0
        for label in labelsSinDuplicados:
            if label in self.eliminasignos(contexto.lower()).split(' '):
                count=count+1

        listafrec.append([count,link[0],link[1]])
    ListaOrdenado= sorted(listafrec,key=lambda x:x[0], reverse=True)
    return ListaOrdenado
```

A este método llegan como parametos de entrada la lista de los recursos de la entidad ambigua y el contexto de dicha entidad, para procede a determinar cual es el recurso valido o mas cercano, para esto nos servimos de otra consulta para extraer los *rdfs:label* de los recursos, para proceder ha hacer una comparación con el contexto de la entidad y los labels extraídos con la consulta ordenándolos en una lista de acuerdo a al que tenga mas concordancia en dicha comparación. El método devuelve la lista ordenada de los recursos.

B. Entidad no Ambigua

Si la entidad no es ambigua nos devuelve la uri del recursos con el tipo de recurso

3. Tipo no encontrado

Se presentará un mensaje en el cual se especifica que el tipo de entrada no se encuentra soportado por la clase.

BIBLIOGRAFÍA

W3C. (n.d.). Web Services Architecture. Retrieved 19 de 01 de 2014 from W3C: http://www.w3.org/TR/ws-arch/

Python. (n.d.). About Python. Retrieved 19 de 01 de 2014 from Python: http://www.python.org/about/

González, M. (n.d.). ESTUDIO DE ARQUITECTURAS DE REDES ORIENTADAS A SERVICIO. Retrieved 19 de 01 de 2014 from UpCommons: http://upcommons.upc.edu/pfc/bitstream/2099.1/12312/1/ESTUDIO_DE_ARQUITECTURAS_DE_REDES_ORIENTADAS_A_SERVICIO.pdf

UPM. (n.d.). ¿Qué es OCW? Retrieved 20 de 01 de 2014 from OpenCourseWare de la Universidad Politécnica de Madrid: http://ocw.upm.es/bfque-es-ocw

Loper, E., & Bird, S. (n.d.). NLTK: The Natural Language Toolkit. Retrieved 21 de 01 de 2014 from Cornell University: http://arxiv.org/pdf/cs/0205028v1.pdf

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

La Web Semántica y las Tecnologías del Lenguaje Humano. (n.d.). Retrieved 23 de 01 de 2012 from e-Lis: http://eprints.rclis.org/15586/1/La%20Web%20Sem%C3%A1ntica%20y%20las%20Tecnolog%C3%ADas%20del%20Lenguaje%20Humano%20-%20Preprint.pdf

Unesco. (2011). A Basic Guide to Open Educational Resources. Retrieved 22 de 8 de 2013 from www.unesco.org/education

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

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

UNESCO. (n.d.). UNESCO. Retrieved 20 de 02 de 2014 from 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/

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

UTPL. (20 de 01 de 2014). OpenCourseWare UTPL. Retrieved 21 de 02 de 2014 from UTPL OCW: http://ocw.utpl.edu.ec/

PNL. (n.d.). Retrieved 23 de 02 de 2014 from El Procesamiento del Lenguaje Natural en la Recuperación de Información Textual y áreas afines: http://www.upf.edu/hipertextnet/numero-5/pln.html

Birrell, & Nelson. (n.d.). RCP. From Implementing Remote Procedure Calls: http://www.cs.princeton.edu/courses/archive/fall03/cs518/papers/rpc.pdf

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

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

Nltk. (n.d.). Retrieved 27 de 01 de 2014 from NLTK: http://nltk.org/

OCW Consortium. (2012). OpenCourseWare Consortium. Retrieved 7 de 02 de 2014 from http://www.ocwconsortium.org/

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

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

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

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

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

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

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

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

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

San Martin Oliva, C. R. (n.d.). portalhuarpe. Retrieved 25 de Julio de 2012 from http://www.portalhuarpe.com.ar/Seminario09/archivos/MetodologiaICONIX.pdf

Google Inc. (14 de Agosto de 2012). Android Developers. Retrieved 03 de Septiembre de 2012 from

http://developer.android.com/reference/org/apache/http/package-summary.html

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

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

OCWConsortium. (2012). OCWConsortium. Retrieved 03 de Octubre de 2012 from http://www.ocwconsortium.org/en/courses

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 .

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

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

W3C. (n.d.). RDF. Retrieved 25 de 04 de 14 from RDF: http://www.w3.org/RDF/

W3C. (n.d.). SPARQL Query Language for RDF. Retrieved 28 de 04 de 2014 from RDF sparql-query: http://www.w3.org/TR/rdf-sparql-query/

DBpedia. (n.d.). About. Retrieved 05 de 05 de 2014 from DBpedia: http://wiki.dbpedia.org/About

DBpedia. (n.d.). Datasets. Retrieved 05 de 05 de 2014 from DBpedia Datasets: http://wiki.dbpedia.org/Datasets