UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ESCUELA DE CIENCIAS DE LA COMPUTACIÓN

IMPLEMENTACIÓN DE UN BUSCADOR SEMÁNTICO UTILIZANDO KIM, GATE Y FREELING PARA LA RECUPERACIÓN DE OBJETOS DE APRENDIZAJE (OA) DEL DSPACE-UTPL

Tesis previa a la obtención del título de Ingeniero en Sistemas Informáticos y Computación.

AUTOR*:*

Santiago Fernando Suárez Sánchez

DIRECTORA:

Ing. Janneth Alexandra Chicaiza Espinosa

Loja – Ecuador

2010

Ing. Janneth Alexandra Chicaiza Espinosa

DIRECTORA DE TESIS

CERTIFICA:

Haber dirigido y supervisado el desarrollo del presente proyecto de tesis previo a la obtención del título de **INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN**, y una vez que este cumple con todas las exigencias y los requisitos legales establecidos por la Universidad Técnica Particular de Loja, autoriza su presentación para los fines legales pertinentes.

Loja, octubre de 2010

Janneth Alexandra Chicaiza Espinosa

AUTORÍA

El presente proyecto de tesis con cada una de sus observaciones, análisis, evaluaciones, conclusiones y recomendaciones emitidas, es de absoluta responsabilidad del autor (s).

Además, es necesario indicar que la información de otros autores empleada en el presente trabajo está debidamente especificada en fuentes de referencia y apartados bibliográficos**.**

.....

F) Santiago Fernando Suárez Sánchez

CESIÓN DE DERECHOS

Yo, Santiago Fernando Suárez Sánchez 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) Santiago Fernando Suárez Sánchez

DEDICATORIA

A mis padres María y Segundo quienes han sido los pilares fundamentales en el desarrollo del presente trabajo, por estar siempre en los momentos más importantes de mi vida , sobre todo a mi madre por ser el ejemplo para salir adelante y enseñarme el camino de la vida.

A mis hermanos Erick, Gaby y Carlitos por estar siempre conmigo en todo momento y apoyarme en los buenos y malos momentos, muchas gracias por siempre estar ahí.

Santiago

AGRADECIMIENTOS

A Dios por llevarme siempre en sus bendiciones y regalarme cada día nuevas oportunidades para mejorar.

A Inés Jara por el apoyo brindado durante toda mi carrera, sobre todo por su motivación y consejos, de verdad muchas gracias.

A la Ing. Janneth Chicaiza por la dirección del presente trabajo en la parte final, sus directrices y correcciones sirvieron de mucho.

Santiago

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

El presente trabajo inicia con la revisión de Recuperación de Información, Procesamiento de Lenguaje Natural y Extracción de Información conceptos básicos de un buscador, de ellos se estudia principalmente su definición y modelos. También se presenta una clasificación de los buscadores, se revisa la Web Semántica con sus componentes y se hace el análisis de algunos buscadores semánticos que actualmente son reconocidos por la comunidad de Internet.

En el capítulo dos se realiza un análisis comparativo de herramientas especializadas en resolver las tareas básicas de un buscador, se dio preferencia a herramientas de carácter open source y compatibles con la tecnología java. Las temáticas tratadas en esta sección tienen la particularidad de contar con sus propios congresos y eventos científicos, esto dio la pauta de probar las herramientas con los datos de prueba de los mencionados eventos, y así someter a las herramientas a problemas reales.

Una vez determinadas las herramientas con las mejores prestaciones, estas pasaron a la siguiente fase que es la de desarrollo, donde se estableció los principales problemas, fuentes de información, stakeholder además de los principales requerimientos. Con lo anterior se procedió a determinar la arquitectura del sistema, las configuraciones y desarrollo de los componentes necesarios del sistema.

Desarrollado el sistema se procede a realizar un conjunto de pruebas con el fin de determinar el rendimiento, errores y posibles soluciones, en esta parte se definen métricas, datos de prueba, y escenario en que deben efectuarse.

Los Anexos contienen principalmente detalles de configuración de las diferentes herramientas analizadas, documentos utilizados para la captura, análisis de requerimientos y descripción de la arquitectura del sistema.

Capítulo 1 Estado del Arte

RESUMEN

La cantidad inmensa de información contenida en la Web a traído como consecuencia el uso de sofisticados paquetes de software llamados buscadores para ayudar a los internautas a filtrar los contenidos en la red; sin embargo, los resultados arrojados por estos programas son poco relevantes, además de ser numerosos lo que ha hecho que el usuario gaste tiempo y paciencia filtrando nuevamente la información arrojada por los buscadores. Para solventar éste y otros problemas aparecen producto del apogeo de la Web 3.0, los buscadores de tercera generación o semánticos. Este tipo de aplicaciones intentan solucionar el principal problema de sus predecesores (buscadores sintácticos), la escasa relevancia en los resultados. Por tal motivo este trabajo tiene como objetivo revisar los enfoques, estructuras y métodos utilizados por los diferentes componentes de un buscador semántico. Además se realiza un estudio de los problemas que poseen los buscadores actuales, que han dado lugar a nuevos mecanismos para buscar información; los problemas de los buscadores actuales tienen un trasfondo en la Web actual, por tal razón las técnicas de Recuperación de Información Aproximada son las que mejor se ajustan a la realidad de la información contenida en la Web y que existen maneras semánticas y no semánticas para dotar de inteligencia a un buscador.

INTRODUCCIÓN

La evolución de la Web actual a una Web con sentido (Web Semántica) ha originado que se busquen maneras más efectivas y precisas de buscar información, en una red de redes en donde el internauta ha pasado de ser un mero espectador a ser consumidor y productor de información lo que hace que el contenido en la Web crezca exponencialmente cada día. En 1990 se produce una carrera tecnológica por construir herramientas que ayuden a filtrar la información, en esos momentos el criterio con que se realizaban la discriminación de los documentos eran mediante una correspondencia de las palabras entre la búsqueda del usuario y los documentos en cuestión, esta solución se mantiene hasta la actualidad con algunas mejoras; pero debido a la diversidad de la información, spam y cantidad, la efectividad de estos buscadores ha bajado considerablemente, es por tal motivo que ahora se busca dotar a la Web y por ende a su contenido de significado que sea procesable tanto por máquinas como por humanos, lo que pretende la Web Semántica aún sigue siendo un concepto, hasta el día de hoy no existe una Web con sentido implantada totalmente, y todo indica que la Web actual como la Web semántica tendrán que coexistir durante mucho tiempo, ya que para cambiar toda la Web de hoy se requiere de un esfuerzo inimaginable.

Pese a esto existen trabajos en donde han logrado implementar buscadores altamente precisos, un ejemplo lo señala Amaral (2000) quien presenta un buscador semántico para el idioma portugués donde se hace uso de herramientas de Procesamiento de Lenguaje Natural y un corpus léxico multilingüe donde se evalúan las consultas del usuario, para la desambiguación de palabras polisémicas se hace uso de pivotes mostrados en la pantalla con los diferentes significados de la palabra donde el usuario escoge el sentido con el que desea realizar la consulta.

Otro trabajo importante sobre este tema se evidencia en Berrueta (2004), en este se describe el funcionamiento de un buscador semántico para recuperar información de la Administración Pública del Principado de Asturias, funciona en dos fases; por un lado una ontología basada en la naturaleza de la organización en la cual se describen entre otras cosas: vocabulario jurídico,

departamentos; y por otro un algoritmo activador que reconoce conceptos de la ontología presentes en la consulta del usuario.

En este capítulo está distribuido de la siguiente manera: En la sección 1 se presenta la estructura y enfoques de la Recuperación de Información, en la sección 1.2 se presenta la organización y componentes de un buscador sintáctico, además se hace una revisión de los principales problemas de los mismos. En la sección 1.3 se hace referencia a la Web Semántica, aquí se abarca como cada componente actúa dentro de la estructura organizativa, luego en la sección 1.4 se pasa a revisar los buscadores semánticos: su funcionamiento y principales implementaciones, para finalmente examinar el Procesamiento de Lenguaje Natural, origen y modelos en la sección 1.5.

1.1. Recuperación de Información

La Recuperación de Información (RI) es el proceso mediante el cual se evalúa un conjunto de documentos a fin de filtrar aquellos que satisfagan la consulta del usuario. Este proceso forma parte de muchas aplicaciones en la vida cotidiana, en el caso particular de los buscadores Web, su parte medular es la RI. En este punto se pasará a revisar la investigación de Llidó (2002), por la completitud de su trabajo en lo que se refiere a la RI, según este autor un sistema de RI consta de los siguientes componentes:

- Una colección de documentos: Un repositorio de donde se va a proceder a recuperar la información, cuyos documentos pueden ser estructurados y/o contener un descriptor de metadatos.
- Un lenguaje de consulta: Que son las palabras o términos de interés para el usuario, a partir de los cuales se procederá a filtrar los documentos.
- Una aplicación: Encargada de gestionar los diferentes procesos (organización, selección y presentación) de los documentos, posee ciertos componentes:
    - Un gestor del repositorio: Programa encargado de las funciones básicas del conjunto de documentos.
    - Proceso de emparejamiento: Un evaluador que será el encargado de determinar aquellos documentos que mejor asemejen a la consulta del usuario.
    - Proceso de relevancia: Aquel que determina el orden en que deben aparecer los documentos encontrados.

En el punto siguiente se hace una revisión de los diferentes modelos de RI, así como también su estructura y funcionamiento.

1.1.1. Modelos de recuperación

Durante mucho tiempo se han buscado nuevas formas para discriminar de manera efectiva los documentos relevantes con aquellos que no lo son. Dichas formas van desde las tradicionales como las booleanas, pasando por las estadísticas hasta llegar a enfoques basados en la inteligencia artificial, éstos se pueden clasificar en 2 grandes grupos:

- Sistemas de Recuperación de Búsqueda Exacta.
- Sistemas de Recuperación de Búsqueda Aproximada.

1.1.1.1. Sistemas de Recuperación de Información de Búsqueda Exacta

En este enfoque se realiza una comparación exacta entre la consulta y los documentos, los resultados son valores binarios 1 y 0. Cuando el resultado es 1 significa que el documento satisface la búsqueda y es 0 cuando el documento carece de información inherente a la búsqueda. Los documentos se listan en el mismo orden en que fueron encontrados. En este enfoque se pueden distinguir 2 modelos.

1.1.1.1.1. Por búsqueda de patrones

En este modelo la consulta es expresada como una sucesión de palabras, expresiones regulares o conjunto de cadenas con comodines. Luego el sistema busca directamente sobre la base de documentos aquel que contenga el patrón requerido por el usuario. Cabe destacar que en este modelo se utilizan técnicas de reconocimiento de patrones, no manejan índices por lo cual los tiempos de búsqueda en colecciones de documentos grandes son muy elevados, pero resultan ser efectivos en colecciones de documentos que sufren cambios frecuentes. Basado en lo expuesto anteriormente un ejemplo del funcionamiento por búsqueda de patrones se muestra en la Tabla 1.1, donde la columna "Patrón" y "Comodín" representan la consulta del usuario, la columna "Texto" contiene los documentos que han sido evaluados y el resultado es representado en la columna "Medida". Los documentos resultantes, es decir, los que satisfacen la consulta del usuario son presentados en el mismo orden en que fueron evaluados.

Tabla 1.1: Matriz patrón-texto Búsqueda Exacta

Patrón	Comodín	Texto	Medida
xyzsx	No "x,y"	Documento1	1
xyzsx	Si xyz	Documento2	0
xyzsx	Si xy + sx	Documento3	1
xyzsx	No s-x	Documento4	1
xyzsx		Documento5	0

1.1.1.1.2. Indexación Booleana:

Este modelo trata en indexar[1] los documentos a partir de valores booleanos. Este enfoque es uno de los más utilizados debido a su simplicidad, utiliza la lógica proposicional, algebra booleana y la teoría de conjuntos para expresar las consultas del usuario. Su funcionamiento básicamente consiste en crear una matriz de términos y documentos parecida a la que se muestra en la Tabla 2, en donde las columnas t_i contienen la lista de términos por los cuales se va a discriminar un documento y las filas d_k corresponde a cada documento, la intersección entre el termino t_i y el documento d_k se rellenan con unos cuando el termino t_i está presente en el documento d_k y cero cuando se produce lo contrario.

[1] Acción de registrar ordenadamente información para elaborar un índice

Tabla 1.2: Matriz de términos-documentos Llidó (2002)

	t1	t2	t3	⋯	t5	t6	t_i
d1	0	1	1	0	1	0	0
d2	1	1	1	1	1	1	0
d3	0	0	0	1	0	1	1
:	1	1	0	0	0	0	0
dk	0	1	0	1	0	1	1

Sin embargo, el método booleano posee varios problemas relacionados, el primero es que la mayoría de los usuarios no está familiarizado con la algebra booleana, lenguaje fundamental en que se realizan las consultas, este lenguaje puede tornarse poco entendible a medida que crezca la complejidad de las consultas. Otro problema relacionado es el rigor del algebra booleana, los resultados de toda consulta son binarios, es decir, pertenece o no pertenece, verdadero o falso, 1 o 0. No hay diversidad en la relevancia, un documento no puede ser medianamente relevante o algo relevante.

1.1.2. Sistemas de Recuperación de Información de Búsqueda Aproximada

Este enfoque apareció como una variación al modelo booleano donde la relevancia entre los documentos y la consulta de los usuarios ya no estaba limitada a un verdadero o falso, en cambio se optó por expresar la relevancia con cualquier valor real, teniendo como mínimo valor el número 0, el cual expresa la inexistencia de relevancia de un documento y como límite superior un valor extraído experimentalmente, dicho límite superior sirve como umbral en donde cualquier documento que alcance o sobrepase dicho umbral será considerado altamente relevante. Entre los diferentes modelos de recuperación pertenecientes a la Búsqueda Aproximada se destacan los siguientes:

- Búsqueda Probabilística.
- Redes de Inferencia Bayesiana.
- Modelo Vectorial.

1.1.2.1. Búsqueda Probabilística

En este modelo se utiliza la probabilidad como criterio de búsqueda. Los documentos como los términos de cada documento se disponen como se muestra en la Tabla 3, a diferencia de la matriz propuesta en la indexación booleana, la búsqueda probabilística asocia un peso w_{ij} que no es más que la probabilidad de que el termino t_i aparezca en un documento relevante d_j, lo cual se puede expresar matemáticamente de la siguiente forma: $P(R \mid t_i\ d_j)$ siendo R la relevancia; o puede ser todo lo contrario, puede expresar la poca relevancia del documento d_j con el término t_i que matemáticamente se expresa como: $P(-R \mid t_i\ d_j)$. En definitiva este modelo se encarga de determinar que tan probable es que el documento *x* con el término *y* sea relevante para el usuario.

Tabla 1.3: Matriz de Búsqueda Probabilística

	t1	t2	t3	⋯	t5	t6	t_i
d1	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}
d2	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}
d3	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}
:	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}
dj	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}	w_{ij}

1.1.2.2. Redes de Inferencia Bayesiana

En este modelo se utiliza una Red Bayesiana; un grafo acíclico dirigido en el cual cada nodo representa variables aleatorias y los arcos que unen los nodos entre sí representan relaciones de causa, por otra parte una probabilidad condicional representa la fuerza de inferencia entre las variables. Para un sistema de RI, una Red Bayesiana se estructura de la siguiente manera: Los nodos raíz representan los documentos sobre los cuales se va a realizar la búsqueda, los nodos hijos de cada nodo raíz (Documento d_j) representan los términos t_i presentes en cada documento. Los nodos terminales en cambio simbolizan la consulta del usuario, o en su defecto la raíz de los términos presentes en la consulta del usuario; lo expuesto anteriormente se puede observar en la Figura 1.1. Todos los arcos que unen los nodos padres (documentos) con los nodos hijos (términos de los documentos) representan una probabilidad condicional de que el término esté presente en el documento. Al final la relevancia es medida en base al apoyo evidencial[2] de las observaciones de cada documentos con la consulta.

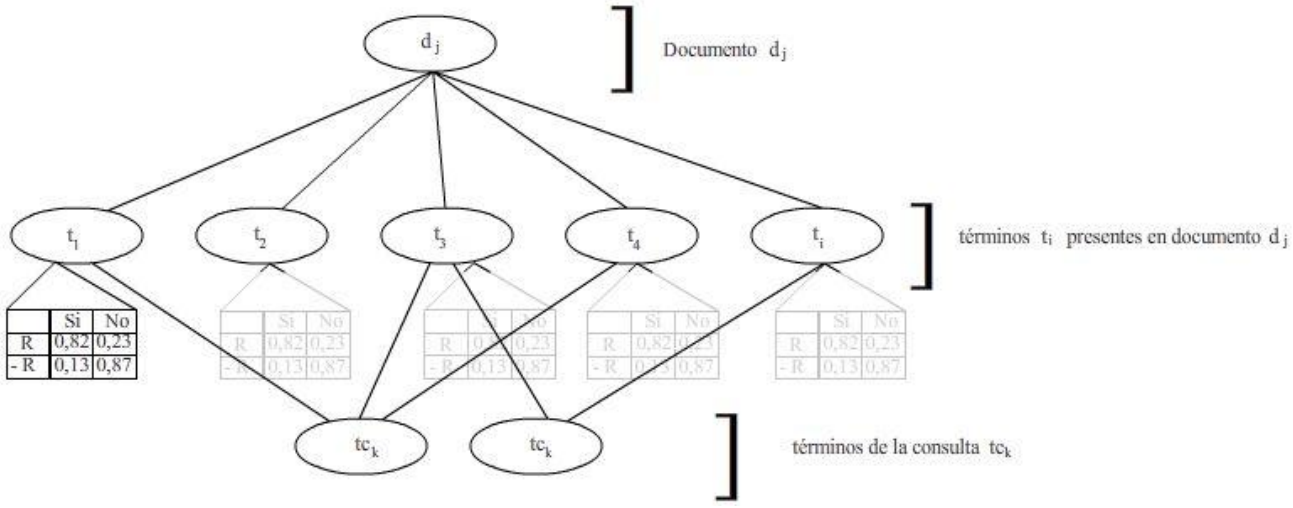


Figura 1.1: Red Bayesiana RI. Elaboración propia

1.1.2.3. Modelo Vectorial

Este modelo es el más utilizado en la actualidad por los Sistemas de RI, especialmente los basados en la Web, por lo cual se presenta un estudio más detallado de este modelo. El funcionamiento básico es tomado del modelo booleano, con marcadas mejoras. La

[2] Una probabilidad de implicación parcial entre enunciados. Carnap (1950)

representación lógica de los documentos en el modelo booleano consistía en un vector binario, en cambio en el modelo vectorial se trabaja con un vector de pesos w_{ij}, el cual indica el grado de relevancia del término t_i en el documento d_j, esta relevancia suele estar sujeta a la frecuencia con que el termino t_i aparece en el documento d_j. La naturaleza del peso w_{ij} puede ser de varios tipos: binario, entero o real. Estos vectores de peso se van creando por cada documento en cuestión, hasta formar una matriz como la que se ilustra en la Tabla 4 llamada matriz de frecuencias de términos.

Tabla 1.4: Matriz de frecuencia de términos Llidó (2002)

	t_1	t_2	t_3	⋯	t_i	⋯	t_m
d_1	w_{11}	w_{12}	w_{13}	⋯	w_{1i}	⋯	w_{1m}
d_2	w_{21}	w_{22}	w_{23}	⋯	w_{2i}	⋯	w_{2m}
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
d_i	w_{i1}	w_{i2}	$w_{i3}1$	⋯	w_{ij}	⋯	w_{im}
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
d_n	w_{n1}	w_{n3}	w_{n3}	⋯	w_{nj}	⋯	w_{nm}

En este modelo los términos constan de algunas propiedades, las cuales se detallan enseguida:

- tf_{ij}: Representa la frecuencia de aparición del término t_i en el documento d_j.
- df_j: Es el número de documentos en los que el término t_i aparece.
- $dv_j = Q - Q_j$: Es la capacidad de disminuir la semejanza entre dos documentos que comparten un mismo término t_i. El término Q es la densidad de semejanza sin considerar el termino t_i, en cambio Q_j es la densidad de semejanza pero con la consideración de t_i.

Considerando las propiedades de términos arriba mencionadas se procede a calcular el peso w_{ij} con las siguientes funciones:

- $w_{ij} = tf_{ij} * idf_j$, donde idf es la función inversa de df.
- $w_{ij} = tf_{ij} * dv_j$, tomando $Q = \frac{1}{N.(N-1)} \sum_i \sum_{j \neq i} sim\,(d_i\,,\,d_j)$.
- $w_{ij} = tf_{ij} * dv_j$, tomando $Q = \frac{1}{N} \sum_i sim\,(C, d_i)$, donde C representa el centroide[3] de colección de documentos.
- "idf_j = log ($\frac{d}{df_j}$) es la frecuencia inversa del documento, donde d es el número total de documentos, Martínez (2004).

Como paso final hay que calcular la semejanza entre un documento d_j y la consulta q_k, para lo cual se utilizan varias medidas de semejanza las cuales se detallan en la Tabla 5.

[3] La intersección de todos los hiperplanos que dividen a X en dos partes de igual n-volumen con respecto al hiperplano. Informalmente, es el promedio de todos los puntos de X.

Tabla 1.5: Distancias entre vectores de términos. Llidó (2002)

Medida de Similitud	**Modelo Booleano**	**Modelo Vectorial**
Producto Escalar	$\|\|X \cap Y\|\|$	$\sum_{j=1}^{m} X_j \,.\, Y_j$
Coeficiente de Dice	$\frac{2.\|\|X \cap Y\|\|}{\|\|X\|\| + \|\|Y\|\|}$	$\frac{2 \,.\, \sum_{j=1}^{m} X_j \,.\, Y_j}{\sum_{j=1}^{m} X_j^2 + \sum_{j=1}^{m} Y_j^2}$
Coseno	$\frac{\|\|X \cap Y\|\|}{\sqrt{\|\|X\|\|} \,.\, \sqrt{\|\|Y\|\|}}$	$\frac{\sum_{i=1}^{m} w_{j,i} * w_{k,i}}{\sqrt{\sum_{i=1}^{m} w_{j,i}^2} \,.\, \sum_{i=1}^{m} w_{k,i}^2}$
Coeficiente de Jaccard	$\frac{\|\|X \cap Y\|\|}{\|\|X\|\| + \|\|Y\|\| - \|\|X \cap Y\|\|}$	$\frac{\sum_{j=1}^{m} X_j \,.\, Y_j}{\sum_{j=1}^{m} X_j^2 + \sum_{j=1}^{m} Y_j^2 - \sum_{j=1}^{m} X_j \,.\, Y_j}$

De todas estas expresiones la medida de Similitud del Coseno es la más utilizada en los Sistemas de RI actuales.

$$sim\left(d_j, d_k\right) = \frac{\sum_{i=1}^{m} w_{j,i} * w_{k,i}}{\sqrt{\sum_{i=1}^{m} w_{j,i}^2} \,.\, \sum_{i=1}^{m} w_{k,i}^2} \quad (\mathbf{1})$$

Los sistemas de RI exacta tienen problemas fundamentales, la severidad con que realizan las búsquedas hace que en la práctica sean pocos productivos ya que un documento puede ser nada, poco, medio o altamente relevante; sin embargo, estos sistemas no dan cabida a ningún otro resultado que no sea verdadero o falso, relevante o no relevante. Los sistemas de RI de búsqueda aproximada, dan posibilidad de tener varios juicios sobre un documento, esta particularidad hace que se acercan más a la realidad de la RI en Internet. De lo revisado anteriormente se concluye que el modelo Vectorial es el que mejor se ajusta a las necesidades de RI de documentos, reúne las bondades del sistema booleano (RI exacta) y se vale de la diversidad de resultados de la estadística del modelo de búsqueda aproximada.

1.1.3. Recuperación de Información Semántica

1.1.3.1. Análisis de Semántica Latente (LSA)

El modelo LSA toma en cuenta el significado de los términos de la consulta así como también su relación con el documento, también es capaz de realizar inferencias, esto debido a que la combinación de determinadas palabras admiten que una tercera sea de un conjunto pequeño de posibilidades; así por ejemplo: *"El pescador usó su caña en el río y pescó..."*. La combinación de las palabras *"percador", "caña","río"* y *"pescó"* hacen que el universo de posibilidades que podría tener la siguiente palabra se reduzca al conjunto de peces, ya sea pescado, salmón, tilapia, etc. El significado de cada palabra es obtenido en base a un análisis estadístico de un repositorio de documentos o corpus. Es así que los documentos recuperados en este modelo son aquellos cuya semántica es semejante a la consulta del usuario. El LSA es una extensión del modelo vectorial es por esto que su funcionamiento es similar, como primer paso se construye

una matriz de frecuencia de términos igual a la que se muestra en la Tabla 4, en esta matriz que ahora en adelante la llamaremos A se muestra la asociación que existe entre los términos y los documentos. La matriz A es sometida a una descomposición singular de matrices (SVD) donde el resultado son tres matrices: U, W, V^t, cuya estructura se desglosa de la manera siguiente:

- U se reduce a s columnas.
- W se reduce a s columnas y a s filas.
- V^t se reduce a s filas.

Siendo *s* un valor arbitrario cuya única restricción es que sea menor al valor *r* resultante de la SVD. Entonces la primera matriz A se ha descompuesto en 3 submatrices, $A_s = U_s, W_s, V_s^t$, es decir que las matrices resultantes ya no contienen ruido (términos no coincidentes con la semántica de la consulta usuario), seguidamente se calcula la similitud de los documentos coincidentes con la consulta, para esto se utilizan las matrices sin ruido y procede a calcular los documentos relevantes con la siguiente fórmula, Maldonado (2002):

$$p^{\rightarrow} = q^{\rightarrow} U_s, W_s^{-1} \quad \textbf{(2)}$$

Donde $q^{\rightarrow}$ representa la consulta del usuario y $p^{\rightarrow}$ es el valor resultante, finalmente se procede a calcular la similitud de cada documento utilizando la medida de Similitud del Coseno **(1)** entre el vector resultante $p^{\rightarrow}$ y cada columna de la matriz V_s^t.

1.2. Buscadores de Información

La Internet inicialmente era de uso exclusivo para un grupo pequeño de personas (científicos e investigadores) así que es de suponer que la producción de información en esos momentos era relativamente mínima para la cantidad de información que existe en la actualidad, pese a esto la necesidad de encontrar algo en esa red de redes ya estaba presente. En 1990 se crea *Archie* una implementación de Unix, basada en el comando *grep* el cual listaba los contenidos presentes en Internet; sin embargo, esta forma sencilla de buscar no duró mucho, ya que en 1991 Tim Berners-Lee crea la World Wide Web (WWW) para compartir los trabajos de investigadores, pero esta nueva tecnología paso a tener otras connotaciones que ni el mismo creador las preveía. Con la WWW todos los usuarios están en condiciones de crear información, si actualmente los usuarios conectados a Internet bordean el billón[4] de usuarios, la tarea de listar todos los contenidos en la red se torna compleja, de ahí que el uso de buscadores se hace imprescindible para filtrar la inmensa cantidad de información que se encuentra en la Web, la misión principal de estos es recorrer la Web indexando contenido para luego ser recuperado por los usuarios, es necesario destacar que los buscadores no solo se limitan a rastrear páginas Web, si no también el contenido publicado dentro de ellas, entre dicho contenido podemos mencionar a archivos de texto de todo tipo y formatos, imágenes, video y audio.

Los buscadores o motores de búsqueda Camacho (2006) los define como:"Herramientas que permite a los usuarios encontrar información en internet sobre cualquier tema, partiendo de palabras claves." Los componentes de un buscador varían de 4 (Almeida (2006)) a 6 (Chau (2003)) dependiendo del autor, en esta investigación se trabajará con la arquitectura propuesta

[4] http://www.emarketer.com/Article.aspx?R=1006899

por de Chau por ser la más completa. Según este autor los componentes de un buscador básico consisten en:

- Spiders: También llamados robots, crawlers, worms o wanderers, que son los encargados de explorar la Web en busca de contenidos.
- Repositorio: Todo el contenido encontrado en la Web se guarda en un repositorio.
- Indexador: Encargado de almacenar ordenadamente el contenido encontrado.
- Indexador Invertido: Se realiza un índice invertido de los contenidos para mostrar más rápido los resultados.
- Motor de consulta: Es aquella rutina que acepta y ejecuta las consultas del usuario y que cuyos resultados los envía a la interfaz de usuario.
- Interfaz de Usuario de una Web: Permite al usuario ver los resultados arrojados por el motor de búsqueda.

A pesar que los buscadores tienen como fin recuperar la información, pueden ser de varios tipos, esto varía según ciertos criterios, los cuales según Vaquero (1997) pueden ser:

- ***Según la información que buscan***
    - *Generales:* Buscan información general en la red, por ejemplo: Infoseek, Google, Altavista.
    - *De Servicios:* Buscan información referente a un servicio en especial, ej: FTP, Telnet. Entre estos se destacan:
        - *Software:* Se encarga de recuperar software de tipo shareware o freeware para ser consumido por los usuarios, ej: Softonic
        - *Dirección:* Buscan información de personas o instituciones, son como una guía de teléfonos con la diferencia que buscan gente en Internet.
    - ***Temáticos:*** Recuperan información sobre un campo en particular, ej: *booksfactory un buscador de libros.*
- ***Según el acceso***
    - *Libres:* Son de libre acceso para cualquier usuario en la red.
    - *Privados:* Pensados y construidos para ser consumidos dentro de una intranet o grupo de personas cerrado.
    - *Limitados:* Motores de búsqueda de pago, que en su versión libre permiten en número limitado de resultados.
- ***Según la forma de adquirir el motor de búsqueda***
    - *Inadquiribles:* Son aquellos que no pueden ser personalizados, solo se los puede consumir.
    - *Shareware:* Se pueden adquirir de forma gratuita para uso personalizado. Ej: Lucene

- *Comerciales:* Se puede comprar un motor de búsqueda según nuestras necesidades.

En cambio Delgado (1998) presenta una clasificación más simple basada en la forma en que rastrean la Web:

- **Directorios:** Se manejan en base a una estructura jerárquica de contenidos, la forma en que las páginas son catalogadas en cada categoría y subcategoría es un proceso manual a cargo de un grupo de personas como lo hace Yahoo. Estos índices se componen de las páginas registradas estructuradas jerárquicamente. La forma en que se recupera las páginas puede ser por medio de 2 sistemas:
  - Navegando a través de la estructura temática.
  - Buscando mediante palabras clave.
- **Motores de Búsqueda:** En este modelo el rastreo de la Web se realiza de forma automática utilizando un programa (Spiders), el cual visita las páginas Web con cierta regularidad, luego crea una base de datos temporal donde va guardando la información que va encontrando en la Web, con un proceso propio del motor de búsqueda la información contenida en la base de datos temporal se va estructurando en la base de datos real de acuerdo a sus descriptores que en muchos casos son los términos que más se repiten en los documentos, para finalmente ser recuperados en base a palabras claves. Los motores de búsqueda constan de tres partes principalmente:
  - Robots encargados de recorrer la Web.
  - Base de datos estructurada por los robots.
  - Motor de búsqueda que ejecuta la consulta.
- **Meta buscadores ó Multibuscador:** Este modelo se encarga de trasladar la consulta del usuario a varios buscadores, después integran los resultados y los organizan de acuerdo a las referencias obtenidas de cada uno. En realidad este tipo de buscador no cuentan con una base de datos propia, simplemente son intermediarios entre la consulta del usuario y motores de búsquedas externos.

Una vez revisada la parte de clasificación y estructura de un buscador, se pasa a examinar algunos problemas que poseen los buscadores en la actualidad.

1.2.1. Problemas con los buscadores actuales

La mayoría de buscadores actuales por no decir todos trabajan mediante palabras claves para su búsqueda. Su funcionamiento de manera general se basa en el número de veces en la que la palabra es citada en todo el documento, la palabra que se repita el mayor número de veces terminará siendo el descriptor de ese documento. A este proceso también debe considerarse el concepto de Page Rank introducido por google[5] que no es más que el ranking de una página, este ranking está establecido por el número de enlaces que apunten a una página. Este proceso con el cual la mayoría de los buscadores funcionan posee marcadas deficiencias, de las cuales Abián (2009ª) describe las siguientes como principales:

[5] www.google.com

- **Escasa precisión en los resultados:** Aunque los resultados de las búsquedas arrojan millones de páginas, estas no contienen la información que realmente se desea.
- **Alta sensibilidad al vocabulario empleado en la búsqueda:** Algunas palabras tiene diferentes significados en cada región, como por ejemplo la palabra "piso", en España significa inmueble, mientras tanto en Latinoamérica hace referencia a la superficie en la que estamos parados, esta diferencia y otras más dificultan obtener los resultados esperados.

  Sin embargo, las deficiencias de los buscadores tienen un trasfondo en la Web actual, en sus inicios sus inventores no previeron las connotaciones que iba a tomar esta tecnología hasta el punto que en la actualidad son la estructura básica de todas las comunicaciones en el mundo. Según lo explica Abián (2005) los problemas cruciales de la Web actual son 2:

- **No incorpora mecanismos para el procesamiento automático de la información:** Y por ende la información no puede ser procesada por las máquinas, por ejemplo si buscamos las especies en peligro de extinción en el Ecuador y además queremos calcular que porcentaje representa estas especies a nivel mundial, salvo que algún autor lo haya calculado previamente el buscador es incapaz de calcular este dato con información relacionada.
- **No incluye mecanismo para la interoperabilidad de los sistemas de información:** Impide la facilidad de generar una comprensión común y compartida entre las personas, organizaciones y las propias maquinas.

Para que dicha interoperabilidad tenga efecto el mismo Abián (2005) expone los tres tipos de interoperabilidad que debe existir entre dos partes interesadas para la comprensión común, las cuales se detallan enseguida:

- **Interoperabilidad técnica:** No es más que la capacidad de intercambiar señales para la comunicación entre las partes interesadas, se exige que debe haber un medio físico o conexión física, como cables, fibra óptica, etc. Además de un conjunto de normas que establecen como van a ser procesados los diferentes tipos de mensajes, a esto se lo llama protocolos, la Web actual cumple a cabalidad con este tipo de interoperabilidad; sin embargo, esto no significa que ya puedan generar comprensión, es más tampoco asegura que existe una comunicación precisa entre dos entes, ya que no se abarca aquí el formato de intercambio de datos.
- **Interoperabilidad sintáctica:** Se refiere a establecer un formato común de intercambio de datos, con lo cual se busca facilitar a los sistemas de información leer los datos. Si no existiera ningún acuerdo previo sobre el formato difícilmente se podría dar una comunicación.
- **Interoperabilidad semántica:** Es la capacidad de intercambiar información referente a determinado dominio teniendo en cuenta el sentido exacto e inequívoco de los términos y expresiones usado dentro del dominio en cuestión. Si se toma en cuenta los bienes raíces españoles, la palabra "piso" hace referencia a un inmueble de alquiler y ya no la superficie en la que estamos parados.

La Web actual carece de interoperabilidad semántica completamente, no hace distinción entre dos conceptos totalmente distintos aun así el contexto en que aparecen lo sugiriera. Sobre la interoperabilidad sintáctica, se puede decir que se cumple a medias, por tal razón el mismo

creador de la Web actual Tim_Berners-Lee impulsa la Web Semántica, la cual se expone en el siguiente punto.

1.3. Web Semántica (WS)

Para solventar la poca relevancia y otros problemas relacionados a las búsquedas Tim_Berners-Lee creador de los protocolos HTTP[6] y HTML[7] y por ende creador de la Web tal cual la conocemos en estos días, propone instaurar una evolución de la Web actual, a una Web con sentido llamada la WS.

En la página del consorcio W3C impulsador del proyecto, oficina española (http://www.w3c.es/Divulgacion/Guiasbreves/WebSemantica) se presenta una definición de WS, la cual la describe como: "La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante."

A lo que apunta la WS es a mitigar los problemas actuales referentes a búsquedas y tratamiento de la información, para esto busca dotar a la Web actual de la interoperabilidad sintáctica y semántica y así mejorar el proceso de intercambio de información y añadir sentido a la misma.

Para tal efecto la WS establece metadatos y relaciones utilizando las ontologías de la inteligencia artificial, dichas ontologías forman su parte fundamental, ya que son las encargadas de dotar de significado a la información dentro de una página Web, sin embargo; las ontologías no son la única tecnología que aporta sus servicios dentro de la WS, como bien lo deja entre ver Castells (s/f) en la Figura 2, donde se muestra las 8 capas que forman la WS.

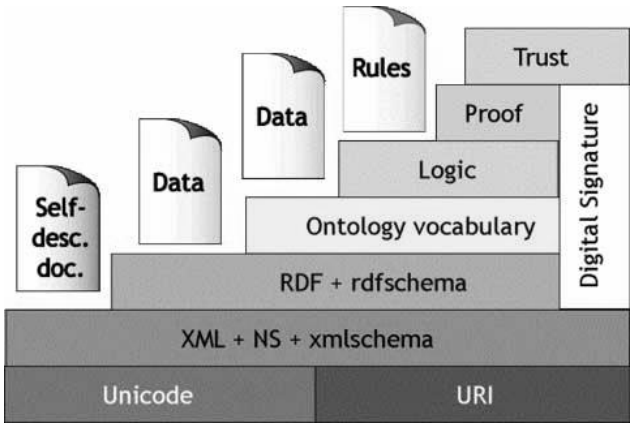


Figura 1.2: Estructura en capas de la WS. Tim Berners-Lee en Castells (s/f)

[6] http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html
[7] http://www.librosweb.es/xhtml/capitulo1/breve_historia_de_html.html

Cada una de estas capas tiene sus funciones específicas dentro de la WS, las cuales se explican a continuación, poniendo especial énfasis en las ontologías.

El **Unicode:** es un estándar que permite el tratamiento informático de cualquier texto escrito en cualquier lenguaje, consta en la primera capa de la estructura de la WS ya que es el encardado de proporcionar un lenguaje común para el tratamiento de cualquier recurso presente en la Web, para la localización de dichos recursos en la Web se utiliza **URI** (Uniform Resource Identifiers), un conjunto de caracteres que vienen hacer el identificador irrepetible de los recursos por medio del cual se puede localizar fácilmente la información, cabe destacar que los **URL** (U*niform Resource Locator*) son una subclasificación de las URI, con la diferencia de ser más específicas.

Siguiendo con la estructura semántica, en el nivel 2 se encuentra a **XML** (Extensible Markup Language); en el punto 3 Abián (2005) se refería que para la existencia de comprensión común, dos entes debía tener entre otras una interoperabilidad sintáctica; que no era más que un formato común para el intercambio de información, el encargado de esta misión es XML, el cual posibilita la comunicación entre agentes en la WS proporcionando una estructura común y de fácil lectura, entre los componentes principales de XML encontramos a los namespaces **NS** que son los encargados de dotar a los elementos un nombre único, así como también **xmlschema** que describe la estructura, restricciones de un XML en particular de manera muy precisa.

RDF (Resource Description Framework) como su nombre lo dice es un framework para el tratamiento y representación de metadatos, todo recurso en la Web posee metadatos como: titulo, autor, fecha, etc. Es muy útil ya que esta representación permite que tanto seres humanos como máquinas puedan interpretar estos datos.

Ontologías

La definición más utilizada es la de Gruber (1993) el cual la define como: "*a formal explicit specification of a shared conceptualization*", es decir una ontología es una estructura jerárquica de conceptos en cada uno de los cuales se pueden distinguir atributos, descripciones y relaciones con otros conceptos sobre un campo del saber humano. El objetivo de la WS con las ontologías es contar con una red de nodos cada uno de los cuales están debidamente tipificados y relacionados con otros nodos. Esto permite obtener una representación formal del conocimiento humano, el cual es legible para las máquinas, común para todos y reutilizable. Según el mismo Gruber (1993) las ontologías constan de 5 componentes:

- **Conceptos:** Son las ideas principales que se intenta formalizar, estas pueden ser distintas a saber: métodos, procesos, planes, objetos, etc.
- **Relaciones:** Representan las correspondencias de algún tipo (parte_de, inverso_a, etc.) entre dos conceptos en particular. Estas relaciones pueden ser taxonomías del mismo dominio.
- **Funciones:** Son relaciones que posee propiedades de cálculo y que involucran a varios elementos en la ontología. Por ejemplo: categorizar-clase, asignar-fecha, etc.
- **Instancias:** Son un caso particular de uno o varios conceptos. Por ejemplo una instancia del concepto animales mamíferos sería ballena.
- **Axiomas:** Son restricciones o propiedades descritas en las relaciones las cuales deber ser cumplidas por los elementos de la ontología, es así; Si $B \subset A$ y $C \subset B$ entonces $C \subset$

A. Estos axiomas permiten realizar operaciones que no necesariamente se encuentra en la ontología a este proceso se lo conoce como inferencia.

Según el propósito las ontologías se dividen en tres grandes grupos:

- **Ontologías de nivel superior:** Permiten modelar los niveles altos de una realidad, ofreciendo conceptos genéricos para la clasificación de términos. Ejemplos de ontologías de estas características son CyC, WordNet, SUMO, etc.
- **Ontologías generales:** Algunos conceptos como el tiempo, el espacio, eventos, etc., pueden reutilizarse a través de diferentes dominios.
- **Ontologías de dominio:** Ontologías de dominios específicos o para aplicaciones concretas modelan las particularidades de las realidades de acuerdo a los propósitos de explotación impuestos.

Aunque en la realidad y según la necesidad se pueden adaptar diferentes tipos de ontologías en un dominio en específico. También se ha desarrollado múltiples enfoques para el desarrollo de ontologías las cuales solo mencionaremos a continuación:

- Uschol and King´s.
- Methontology.
- OnTo-Knowledge.
- Ontology-101.

La capa Lógica (Logic): En esta capa se definen reglas y mecanismos para la producción de inferencias sobre los conceptos de la Web, cada inferencia está sujeta a **Pruebas (Proof)** las cuales validan si las inferencias son lógicas y van acorde al requerimiento propuesto. La capa de **Trust (confianza)** permite verificar que la información ha sido recabada de fuentes de confianza y creíbles para lo cual se utiliza las **Firmas Digitales** que no son más que mecanismos para comprobar si un recurso en la Web es verdaderamente quien dice ser.

Proof (Pruebas):

Se intercambiarán "pruebas" escritas en el lenguaje unificador de la WS. Este lenguaje posibilita las inferencias lógicas realizadas a través del uso de reglas de inferencia.

Trust (Confianza):

Hasta que no se haya comprobado de forma exhaustiva las fuentes de información, los agentes deberían ser muy escépticos acerca de lo que leen en la WS.

Digital Signature (Firma digital):

Utilizada por los ordenadores y agentes para verificar la autenticidad de cualquier emisor de información.

Una vez revisada la arquitectura de la WS y su funcionamiento ahora se pasa a revisar los buscadores semánticos.

1.4. Búscadores Semánticos

Utilizando a la WS como marco de trabajo los buscadores de cuarta generación o semánticos tienen como objetivo entender la búsqueda de los usuarios y ofrecer resultados concretos. Ya no se basan en palabras claves ya que esto se presta a la ambigüedad, si no que tratan de entender los conceptos en la consulta así como también su contexto. Las ontologías forman parte fundamental de los mecanismos de un buscador semántico ya que son las encargadas de definir formalmente los conceptos de un dominio en particular con la suficiente riqueza expresiva para que los usuarios puedan especificar sus consultas con mayor detalle, esto permitirá al buscador presentar los resultados más relevantes.

Técnicamente Abián (2009ª) define un buscador semántico como: "una aplicación que comprende las búsquedas de los usuarios y los textos de los documentos de la Web mediante el uso de algoritmos que simulan comprensión o entendimiento".

Dicha comprensión y entendimiento permitirá al usuario obtener información relacionada gracias al tratamiento automático de la información carente en la Web actual, en el ejemplo planteado en el punto 3 hablamos de calcular el porcentaje que representa las especies en peligro de extinción en el Ecuador a nivel mundial, con esta particularidad un buscador semántico está en la capacidad calcular dicho porcentaje sin la necesidad de buscar a algún autor que lo haya calculado previamente.

A la luz de las bondades de los buscadores semánticos aparecen nuevos problemas relacionados, por ejemplo las palabras polisémicas; que no es más que la particularidad de una palabra para tener diferentes significados. Para ilustrar este problema se hará uso de un ejemplo. Supongamos que se tiene una ontología que describe las señales de radio emitidas por medios informáticos y por insectos, el buscador que esté trabajando con esta ontología deberá contar con los mecanismos necesarios que le ayuden a distinguir cuando la palabra antena" hace referencia a un insecto o al aparato de captar señales electromagnéticas.

Sin duda los buscadores semánticos ofrecen muchas prestaciones; sin embargo, hasta ahora no existe la estructura ni la suficiente anotación semántica para tal efecto, algunos de los buscadores semánticos que existen en el mercado utilizan artificios para simular dichas anotaciones; por otro lado, algunos de los buscadores semánticos del mercado no son orientados a usuarios, existen una gran cantidad de buscadores que realizan consultas a una ontología, este hecho no se refiere precisamente a que puedan ser usados por cualquier usuario, más bien son considerados para usuarios con conocimientos avanzados de anotación semántica. Por este motivo que Hernández (2009) divide a los buscadores semánticos en 2 categorías, así:

- ***Buscadores Semánticos no orientados a usuarios:*** Estos hacen referencia a buscadores que realizan sus consultas en una ontología, ej: Un buscador de drivers para un computador. Esta clase de buscadores es recomendable para usuarios con conocimientos avanzados de notación semántica. Algunos nombres de buscadores ontológicos se muestran a continuación: NaturalFinder, Buscador Experimental BOPA, Lexxe, WebBrain, Blinky.
- ***Buscadores Semánticos orientados a usuarios:*** en cambio estos buscadores además de utilizar herramientas de la WS manejan Procesamiento de Lenguaje Natural con lo cual los usuarios plantean su consulta como si estuvieran planteándosela a otra persona, dentro de esta categoría destacan: Hakia, Wolfram Alpha, Bing.

En el siguiente apartado se presenta el funcionamiento de varios buscadores semánticos orientados al usuario, se empezará con el estudio de Hakia por ser el buscador más representativo y cuyos resultados son lo más relevantes.

1.4.1. HAKIA

Este buscador fundado en el 2004 es uno de los más representativos en el campo semántico, a pesar que se encuentra en fase beta, los resultados obtenidos son altamente relevantes Klein (2009). Entre las principales características se puede mencionar que toda la información en Hakia proviene de sitios creíbles como: Medical Library Association, Biblioteca Nacional de Medicina de los Estados Unidos y otros más. Otra característica importante es que no hace uso de artificios estadísticos para calcular la relevancia, según Riza C. Berkan, director ejecutivo de Hakia: "*El significado no emerge de la estadística, emerge del conocimiento Asociativo*"; es decir, que el poder de este buscador está en entender la consulta y recuperar aquella información que satisfaga dicha consulta. Sobre su funcionamiento Abián (2009b) hace un estudio pormenorizado sobre Hakia, señala que para el funcionamiento de este buscador se hace uso de 3 tecnologías cuya explicación se detalla enseguida:

Ontosem y Onto parser: Ontosem es el encargado de analizar los textos en lenguaje natural, para lo cual utiliza una ontología que es independiente del lenguaje y que abarca miles de conceptos debidamente relacionados. El diccionario que posee esta ontología posee más de 100.000 sentidos de las palabras, cada palabra se encuentra categorizada según los significados que puede tener. Además se contemplan nombres propios y conceptos de muchas áreas de conocimiento del saber humano.

La representación de cada texto a su respectiva representación semántica es llevado a cabo por OntoParser que es un conversor ontológico, cada vez que se introduce una consulta en lenguaje natural OntoSem se encarga de estructurar una representación donde se pueden distinguir conceptos, las diferentes relaciones entre ellos, además de los sentidos de cada palabra. Ontosem permite que Hakia pueda realizar inferencias e inclusive realizar búsquedas relacionadas con el tema en cuestión. Si la WS estuviera implementada Hakia dejaría de utilizar OntoParser para establecer el significado exacto de las palabras en una oración.

QDEX (Query Indexing techique) Para hacer una analogía con los buscadores actuales, QDEX vendría siendo el crawler con la diferencia de que por cada página que va revisando extrae las sentencias más importantes y genera preguntas a partir de esas sentencias, es así que si se pregunta a Hakia ¿Quién es Obama? QDEX ya ha generado y por ende se ha respondido esa pregunta previamente, y también todas aquellas preguntas relacionadas. QDEX utiliza a Ontosem para eliminar aquellas sentencias que carecen de sentido y hacer desambiguaciones de palabras. En la Figura 3 se muestra un ejemplo en el cual se quiere dar respuesta a la consulta ¿Dónde nació Max Planck?, como se muestra en la figura, QDEX ya ha generado previamente preguntas relacionadas y tiene enlazados los documentos que dan respuesta a dicho requerimiento.

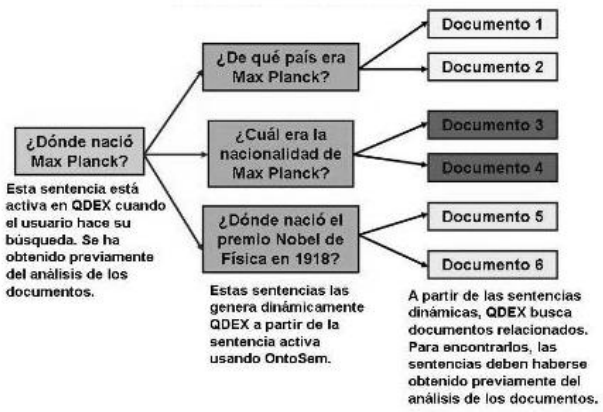


Figura 1.3: Ejemplo de funcionamiento de QDEX en las búsquedas de Hakia. Abián (2009ª)

SemanticRank: Cumple las mismas funciones que PageRank de Google; determinar la relevancia de un documento, pero lo hacen con mecanismos diferentes. El pageRank calcula la relevancia de los resultados a partir del número de enlaces que apuntan a la página en cuestión, en cambio SemanticRank determina la relevancia de los resultados en base a la concordancia de los conceptos relacionados con la búsqueda. SemanticRank recibe las preguntas relevantes de QDEX y determina la relevancia con la búsqueda en cuestión.

En la **Figura 1.4** se muestra gráficamente como estaría compuesto estructuralmente Hakia, esto en base a lo que se ha revisado anteriormente:

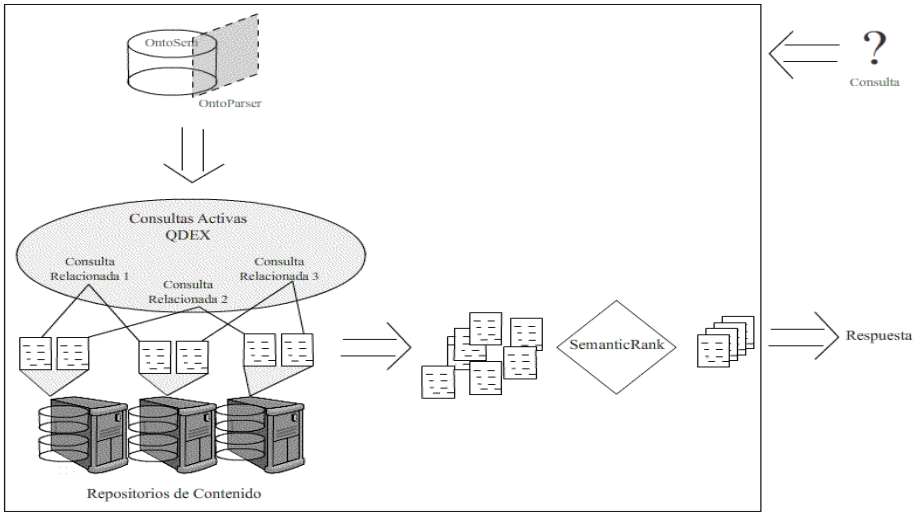


Figura 1.4: Estructura de Hakia

1.4.2. Wolfram Alpha

Este buscador ha originado una gran expectativa desde su lanzamiento en Mayo de 2009, esto se debe a su funcionamiento; es capaz de realizar cálculos matemáticos tal como lo haría cualquier persona, entre sus capacidades de cálculo constan la resolución de polinomios, integrales y demás operaciones. Esto ha llevado a la gran comunidad de Internet ha considerarlo como la revolución entre los buscadores semánticos y no semánticos. Su funcionamiento es muy particular, las inferencias son básicas y son realizadas por Cyc, un proyecto de inteligencia

artificial que intenta ensamblar una ontología compresiva y una base de conocimientos general, todo esto con el objetivo de realizar razonamientos de tipo humano. Otro factor importante es la RI, un grupo de expertos en Wolfram Alpha han procesado la información presente en la Web y la han almacenado en una base de datos desde la cual se extrae un informe con los datos más relevantes, evitando mostrar enlaces Web, este buscador elimina las páginas intermediarias y presenta el resultado directo en la interfaz del buscador. Si bien es cierto este buscador recupera la información desde una base de datos y realiza inferencias; sin embargo, es más un sistema de respuestas que un buscador semántico como tal, esto lo corrobora el propio Stephen Wolfram [8]creador del buscador, ya que considera que su aplicación es más un repositorio de conocimientos que un buscador.

1.4.3. Lucene – SIREN

Este es un motor de búsqueda completo de código de abierto escrito originalmente en Java, aunque en la actualidad ha extendido su desarrollo a otros lenguajes de programación. Entre sus funcionalidades podemos destacar:

- Proporciona la capacidad de buscar en muchos idiomas.
- Indexa cualquier archivo basado en texto, tal como HTML, o cualquier archivo que se pueda convertir en texto.
- Clasifica la búsqueda de manera que se muestren primero los mejores resultados.
- Realiza búsqueda booleana y por frase.
- Permite búsqueda por campo (por ejemplo, las búsquedas se pueden enviar por títulos, autores, contenidos, etc).
- Permite búsquedas en un gran rango de fechas para que los usuarios puedan tener acceso a información sujeta a plazos determinados.

Su funcionamiento según Akamai (s/f), consiste como primer paso la creación de un índice que no es más que toda la información de nuestro sitio compilada en una base de datos. Este índice debe contener un analizador o extractor por cada tipo de documento, Lucene emplea por defecto el HTML parser, el cual recibe como parámetros la URL de ubicación del archivo, extrae el texto y lo almacena en un objeto string, el cual si está indexado y almacenado significa que se puede buscar y mostrar el contenido posteriormente, luego de esto es convertido en token (unidad básica de indexación), en este proceso se quitan todas las palabras vacías (un, la, el, etc.), luego se saca la raíz de la palabra, proceso conocido como stemming, para finalmente transformar el texto en letras minúsculas, este proceso se repite hasta que el índice ha reducido su tamaño con los elementos principales.

Lucene es un buscador sintáctico por naturaleza, para proporcionarle funcionalidades semánticas se necesita de una extensión llamada SIREN (Semantic Information Retrieval Engine) cuyo funcionamiento se trata según la propia página del proyecto[9] de un conjunto de tripletas que poseen la descripción, representado en un grafo de estrella con un nodo común que especifica el enfoque de una palabra, como se muestra en la Figura 5:

[8] http://www.stephenwolfram.com/
[9] https://dev.deri.ie/confluence/display/SIREn/SIREn+Presentation

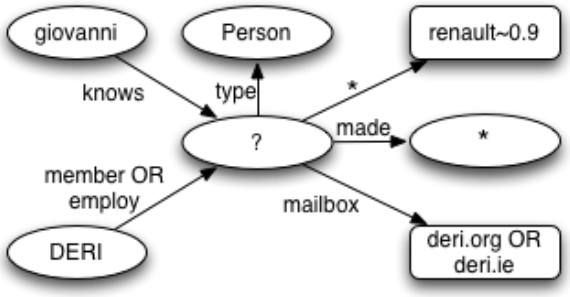


Figura 1.5: Estructura de estrella de una consulta- SIREN

Fuente: https://dev.deri.ie/confluence/display/SIREn/SIREn+Presentation

Una manera de representar datos semi-estructurados es utilizar un modelo de tabla tupla similar a una hoja de cálculo, como se ilustra a continuación. Los datos están organizados en filas (tuplas) y columnas, cada celda contiene algunos valores; además existe la posibilidad de contar de esta estructura de datos en un campo de documento Lucene. El modelo de tabla tupla da mucha más libertad para organizar los datos.

Tabla 1.6: Representación de datos Semi-estrcuturados

	P1	**O1**	**P2**	**O2**
T1	rdf:type	Person	rdfs:label	"A Person"
T2	foaf:knows	giovanni	foaf:name	"Giovanni Tummarello"
T3	foaf:knows	giovanni	foaf:homepage	http://g1o.net

1.5. Procesamiento de Lenguaje Natural (PLN)

Disciplina que inicia en 1940 y sus aplicaciones comenzaron a aparecer 2 años más tarde donde se tuvo particular interés en la traducción automática. A principios la traducción consistía simplemente en el reemplazo de palabras, pero esto degeneró en problemas de sintaxis, semántica, además de la falta de una estructura oracional de algunas lenguas. En 1970 la influencia de la inteligencia artificial centró el interés ya no en la traducción de texto si no que se optó por realizar una representación del significado, en esta época se construyo Eliza; el primer sistema de preguntas y respuestas basado en lenguaje natural. Parte de los '70s y comienzos de los años 80s aparecen gramáticas orientadas al tratamiento computacional de la información, en esta época tuvo gran crecimiento la programación lógica. En la actualidad se utilizan técnicas estadísticas y de representación de conocimiento basadas en la inteligencia artificial que buscan mejorar los resultados de consultas en texto.

Según Sosa (1997) el PLN se define como: "*el reconocimiento y utilización de la información expresada en lenguaje humano a través del uso de sistemas informáticos*.". Para esto PLN hace uso de varias disciplinas del saber humano entre ellas: filosofía, matemáticas y psicología. El tratamiento del lenguaje natural comprende según Sosa (1997) 4 fases:

- **Morfológico:** Etapa temprana que se encarga de verificar que las unidades que forman una palabra pertenezcan al léxico del que se está tratando.
- **Sintáctico:** Este etapa es la encargada de verificar si todas las combinaciones de los componentes de una oración son gramaticalmente correctas.
- **Semántico:** Es la fase encargada de la representación del significado.
- **Pragmático**: Encarga de añadir información en base al contexto en el que se está trabajando.

A continuación se revisan los modelos existentes para el PNL.

1.5.1. Modelos de Procesamiento de Lenguaje Natural

Según ha ido evolucionando el PLN han aparecido nuevas formas de tratar el lenguaje natural, Entre los diferentes modelos Contreras (2001) presenta una clasificación en cuatro grupos las cuales se pasan a revisar enseguida:

- Modelos Simbólicos.
- Modelos Empíricos o estadísticos.
- Modelos Conexionistas o Biológicos.
- Modelos de enfoques híbridos.

1.5.1.1. Modelos Simbólicos

Estos sistemas están basados en la rigurosidad y sistemática de la demostración de teoremas matemáticos y lógicos. Según este modelo un lenguaje natural, por ejemplo, el español puede representarse o escribirse utilizando un lenguaje artificial llamado formalismo gramatical, el cual es riguroso para facilitar las predicciones y evaluación de las frases.

Para facilitar el manejo del lenguaje natural se han creado diferentes tipos de gramáticas, sin embargo, las más conocidas son las gramáticas generativas de Chomsky, estas gramáticas constan de un conjunto de reglas que marcan la estructura interna de las oraciones. Existen varios tipos de gramáticas generativas, estás se dividen de acuerdo a las reglas que contienen en cuatro clases de gramáticas generativas, las cuales se presentan en la Tabla 7:

Tabla 1.7: Jerarquía de Chomsky. Moreno (1998)

Tipo	Gramáticas	Restricciones a la forma de las reglas	Lenguas	Autómatas
0	Irrestrictas	Ninguna: $\propto 1 \ldots \propto N \rightarrow \beta N$	Enumerables recursivamente	Máquinas de Turing
1	Dependientes del contexto	La parte derecha contiene como mínimo los símbolos de la parte izquierda: $\propto \rightarrow \beta / X\_Y$ o alternativamente: $X \propto Z \rightarrow \beta Z \ldots$	Dependientes del contexto	Autómatas linealmente finitos
2	Independientes del contexto	La parte izquierda solo puede tener un símbolo: $\propto \rightarrow \beta \ldots$	Independientes del contexto	Autómatas PDS (Push Down Store)
3	Regulares o de estados finitos	La regla solo puede tener estas dos formas: $A \rightarrow tB$ $A \rightarrow t$	Regulares	Autómatas finitos

Las reglas de las gramáticas pueden extenderse según la necesidad del lenguaje en cuestión, es más se pueden unir reglas de diferentes gramáticas.

1.5.1.2. Modelos Estadísticos

Este modelo nació de la necesidad de solventar el principal problema del PLN; la ambigüedad, estos sistemas tratan de establecer la interpretación o estructura más adecuada de una oración valiéndose de la probabilidad. Este modelo utiliza varias etapas las cuales se detallan enseguida, Moreno (1998):

- Recolección de datos: También llamados corpus lingüísticos, son una colección representativa de palabras.
- Anotaciones del corpus: Son descriptores del texto que pueden tener cualquier tipo de información, como por ejemplo concordancia, categorías, etc. Las anotaciones pueden ser automáticas o manuales, en las anotaciones manuales un grupo de personas expertas definen las ambigüedades las cuales servirán como conjunto de entrenamiento en anotadores automáticos. En cambio los métodos automáticos utilizan una gramática computacional para realizar anotaciones en el corpus.
- Cálculo de frecuencias: Es el proceso encargado de calcular cuantas veces una palabra, sintagma, concepto, morfema o fonema aparece en función a una categoría sintáctica.

Un ejemplo del funcionamiento básico de este modelo se puede observar en la Tabla 8, donde se trata de establecer utilizando la probabilidad, cuando la palabra *"sobre"* hace referencia a una preposición y cuando a un sustantivo concreto.

Tabla 1.8: Ejemplo de análisis de probabilidad. Contreras (2001)

Palabra	Ocurrencias	Probabilidad
Sobre, preposición	115 veces	115/10.255 = 0,0112
Sobre, sustantivo	13 veces	13/10.255 = 0,0012

El cálculo de la probabilidad es básico, se lo hace dividiendo el número de ocurrencias para el número total de palabras en el corpus.

1.5.1.3. Modelo Biológico

El nombre de este modelo se debe a que las técnicas que utilizan tratan de simular procesos de los organismos de la vida real, como lo son el cerebro (redes neuronales) o la evolución de las especies (algoritmos genéticos). Estos sistemas están estructurados de los siguientes módulos: reconocimiento léxico y morfológico, análisis sintáctico, interpretación semántica y pragmática.

1.5.1.3.1. Redes Neuronales (Conexionismo)

Como se dijo en la parte anterior el funcionamiento de esta técnica trata de simular el comportamiento del cerebro, aunque la gramática ni el léxico son explícitos en esta técnica; los fonemas, morfemas y oraciones si lo son. El reconocimiento de estructuras se basa en reconocimiento de patrones, en redes neuronales si una estructura activa los mismos nodos de un patrón se consideran similares. Aunque las redes neuronales utilizan la probabilidad en su funcionamiento al igual que el modelo estadístico del apartado anterior, el conexionismo tiene la capacidad de llevar el reconocimiento morfosintántico y semántico de forma paralela. Debido a que los modelos conexionistas están basados en patrones han sido de gran utilidad en algunos procesos del PLN entre los cuales se mencionan los siguientes:

- Reconocimiento del habla: En esta actividad se utilizan datos acústicos organizados por rasgos, fonemas, palabras, alófonos y silabas para el reconocimiento del habla.
- Desambiguación léxica: Simula el léxico mental del ser humano para resolver ambigüedades.
- Etiquetadores morfosintácticos: Son utilizados especialmente para reconocer signos de puntuación.

1.5.1.3.2. Algoritmos Genéticos

Las características principales de estas técnicas son la adaptación y el auto aprendizaje basado en información histórica. Para explicar el funcionamiento de esta técnica se expondrá un ejemplo de Moreno (1998). En este ejemplo se consideran a los fonemas del idioma español, los cuales son codificados en binario, por ejemplo /b/ es representado por el número binario 11001$$. En la Tabla 9 se muestra la tabla completa con algunos fonemas:

Tabla 1.9: Fonemas en español en formato binario. Moreno (1998)

	/p/	/b/	/m/	/t/	/d/	/n/	/a/	/i/	/u/
Consonante	1	1	1	1	1	1	0	0	0
Labial	1	1	1	0	0	0	#	#	#
Dental	0	0	0	1	1	1	#	#	#
Nasal	0	0	1	0	0	1	#	#	#
Sonora	0	1	1	0	1	1	1	1	1
Anterior	#	#	#	#	#	#	0	1	0
Central	#	#	#	#	#	#	1	0	0
Posterior	#	#	#	#	#	#	0	0	1

Los números binarios representan los cromosomas de una consonante, esta representación facilita los procesos de reproducción y mutación. La reproducción de un cromosoma se realiza tomando la parte izquierda del cromosoma de una consonante y uniéndola con la parte derecha de otra consonante dando lugar a un nuevo fonema. En cambio la mutación se realiza cambiando un dígito particular de un cromosoma.

1.5.1.4. Modelos de enfoques híbridos

En este enfoque se combinan algunos de los modelos anteriores y otros tomados de la Inteligencia artificial, todo esto con el fin de suplir deficiencias y potenciar ventajas; sin embargo, la aplicación de estos modelos conlleva mayor dificultad y tiempo en aplicarlos. Un ejemplo de la implementación de este modelo se encuentra en Rufiner (2004), en este trabajo se combina el cálculo paralelo de las redes neuronales, y el poder de selección de los árboles de decisión para crear un sistema de reconocimiento del habla.

1.6. Extracción de Información (EI)

La EI tiene como objetivo extraer aquellos hechos relevantes que se encuentran de forma explícita en los documentos, es un proceso posterior a la RI y utiliza intensamente PLN para su funcionamiento.

Hamish (2004), define a EI como el proceso de producir formatos fijos de datos no ambiguos a partir de texto libre (texto plano). La EI tiene marcadas ventajas con respecto a PLN ya que es capaz de trabajar con grandes colecciones documentos en los cuales se habla de diversos temas y poseen una longitud considerable, los sistemas de EI son capaces de encontrar y enlazar información relevante mientras ignoran aquella información que no lo es o que es extraña, Tellez (2005).

Con el objetivo de medir los avances de la EI ha se han promovido en Estados Unidos varias conferencias entre ellas Message Understanding Conference (MUC)[10], TIPSTER[11] (programa de investigación sobre recuperación y extracción de información del gobierno de EE.UU) y

[10] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html
[11] http://www-nlpir.nist.gov/related_projects/tipster/

Automatic Content Extraction (ACE)[12] cuya función principal es promover el desarrollo de nuevos y mejores métodos de EI, estas se celebran año tras año y se trata un tópico especifico, investigadores de todo el mundo ponen a prueba sus métodos sobre un corpus etiquetado. En las conferencias MUC, se han definido cinco tareas que se ejecutan en la EI; así:

- **Reconocimiento de Entidades con Nombre (NE):** Se trata de reconocer nombres (personas, organizaciones), lugares, fechas, tiempo, moneda etc.
- **Plantilla de Elementos (TE):** Tarea que añade información descriptiva a los resultados de NE, utiliza CO.
- **Tareas de Correferencia (CO):** Identifica expresiones de texto que hagan referencia a un mismo objeto, este proceso incluye las tareas de NE y TE.
- **Plantilla de Relación (TR):** Descubre las relaciones entre entidades, es una tarea muy importante dentro de EI, trabaja con número de partida de posibles relaciones entre los distintos elementos.
- **Plantilla de Escenario (ST):** Construye un escenario de eventos específicos donde los resultados de TE y TR coinciden y obtiene información relacionada a estos elementos.

No cabe duda que las conferencias MUC son ampliamente reconocidas dentro de EI, especialmente durante el lapso de 1990 y 1998 donde la competición entre diferentes grupos de investigación aportaron con metodologías, evaluación, diseños en definitiva experiencias que más tarde Hobbs (1993) las reunió y propuso una arquitectura general para EI, que a decir de su autor es "una cascada de módulos que en cada paso agregan estructura al documento, y algunas veces, filtran información relevante por medio de aplicar reglas" la cual consta de 8 procesos, secuenciales que se detallan en la siguiente figura:

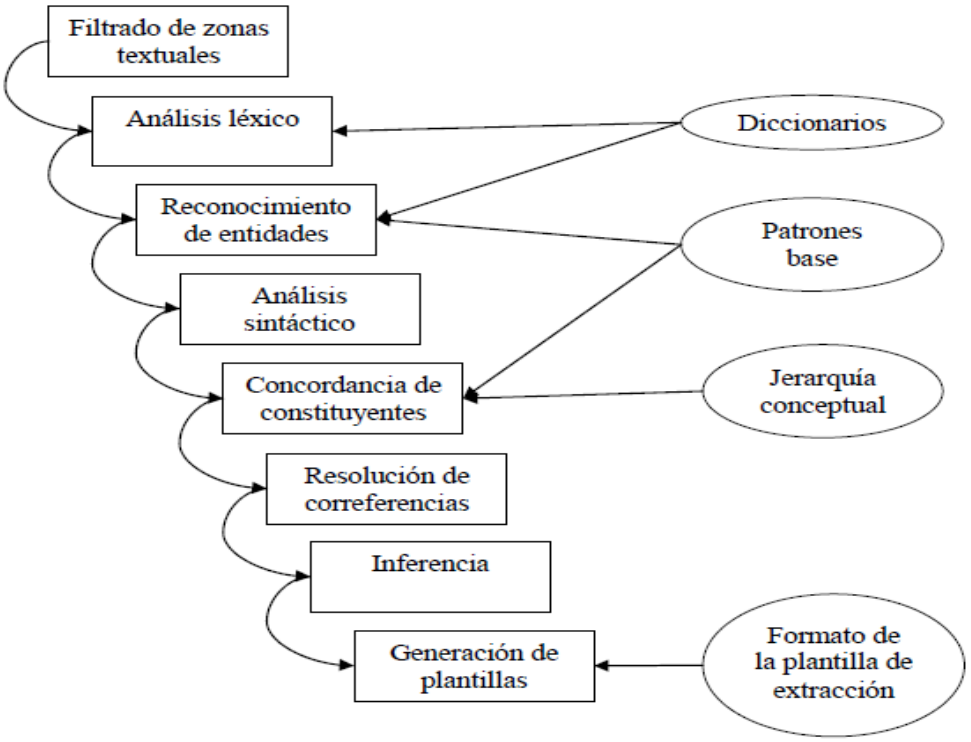


Figura 1.6: Arquitectura General de un sistema de EI. Hobbs (1993).

[12] http://www.itl.nist.gov/iad/mig//tests/ace/

Se han propuesto varias metodologías alternativas, sin embargo; estas siguen como línea base la arquitectura de Hobbs (1993), en ella se distinguen 4 módulos que varían del idioma en que se estén trabajando, Llidó (2003) y Tellez (2005) destacan las siguientes:

- **Segmentador:** Extrae las porciones del documento que se van a analizar como vocablos, oraciones; en algunos lenguajes como el español el proceso es trivial, en cambio, en otros lenguajes como el chino resulta ser bastante complejo lo que requiere procesos adicionales.
- **Procesamiento morfológico y léxico:** Agrega características morfológicas o léxicas a las porciones de texto analizadas, se utilizan diccionarios de dominio específico, lexicón o base de datos terminológicas; la principal características de estos recursos es que son extensos por consiguiente generan un problema, "la ambigüedad". Para tratar los distintos significados de cada plabla se utilizan alguno de los siguientes métodos:
    - **Part of Speech Tagging:** Etiqueta cada palabra del texto con la información de un lexicón. Para tratar palabras con significados poco comunes y eliminar ambigüedades se requiere contar con un lexicón muy completo y requiere grandes esfuerzos en textos largos.
    - **Word Sense Tagging:** Asigna el significado de las palabras en base a un diccionario, con las palabras poco comunes o ambiguas utiliza un listado con los significados posibles y utilizados con más frecuencia, no es un proceso tan perfecto por eso se lo utiliza como alternativa al Part of Speech Tagging.
- **Análisis sintáctico:** Utiliza PLN para descubrir entidades relevantes en un dominio específico; sin embargo, trabaja con ciertas restricciones para procesar las grandes cantidades de texto en tiempos manejables.
- **Análisis de dominio:** En esta etapa se distinguen tres operaciones:
    - **Resolución de anáforas, deixis, ambigüedades y correferencia:** Las entidades de un hecho relevante pueden ser citadas de varias formas a lo largo del texto, la mayoría de las veces se encuentran bastante separadas entre sí, la correferencia de nombres personas o empresas se resuelve mejor con un módulo de reconocimiento de nombres. Las referencias temporales relativas ("hoy", "hace un año") son otro problema de referencia, para solucionarlo se necesita establecer previamente fechas absolutas o determinar la fecha en que el texto fue publicado.
    - **Combinación de resultados parciales:** Se identifica el mismo suceso en varios documentos con el fin de obtener información más completa del evento aprovechando su redundancia.
    - **Construcción de salida:** Es la etapa más crítica de todo sistema de EI, extraen las frases que contienen la información relevante; existen otros sistemas más complejos que expresan la información extraída en un modo independiente al texto original.

1.6.1. Modelos básicos EI

Existen varios modelos de EI, Chakrabarti (2000) propone tres:

1.6.1.1. Modelos para texto

Estos modelos utilizan el espacio vectorial para representar el documento y PLN para su procesamiento, trabajan con dos enfoques. El modelo estadístico o binario, el más simple de todos; cada documento es un conjunto de términos que a su vez forman un subconjunto de términos posibles (subconjunto léxico) en este enfoque no se toma en cuenta la frecuencia de los términos, solo se representa su aparición con uno o cero.

1.6.1.2. Modelos para hipertexto

Trabajan en documentos donde se combinan el texto con enlaces de hipertexto, esta clase de documentos son más simples de manejar, cada link es un grafo directo (D, L) donde D son los nodos que representan documentos y páginas, y L son los enlaces.

1.6.1.3. Modelos para datos semi-estructurados

Las diferentes estructuras en la Web necesitan un manejo especial, por ejemplo el HTML, la oraciones no se encuentran completas ya que están divididas por alguna etiqueta, esto dificulta a las técnicas de PLN el tratamiento del texto y en consecuencia los proceso de EI, es por esta razón que se buscan mecanismos personalizados para el contenido estructurado (XML, XHTML, etc.).

1.6.2. Técnicas de EI

Estas técnicas ilustran el método empleado para descubrir información, todas tratan de alguna manera el texto para su mejor exploración, Cruz (2006) expone cuatro técnicas:

1.6.2.1. Técnicas Deterministas

Utilizan el modelo vectorial para representar los documentos, construyen una matriz *n* dimensional, donde *n* es el número de términos del documento, se toma en cuenta la frecuencia de los términos. Para evitar problemas de sobredimensionalidad de la matriz, se hace un pre-procesado de los documentos, se elimina las stop words o palabras vacías (él, la, los, ellos, (pronombres)) ya que estos sistemas son incapaces de diferenciar la importancia de las entidades, tratan de la misma forma la entidad *Rafael* como la palabra vacía *la*. Utilizan además la frecuencia inversa del documento (IDF) para asignar a cada palabra un peso dentro del documento.

1.6.2.2. Técnicas Probabilísticas

Estas técnicas trabajan bajo la suposición de que se tienen *m* clases de tópicos: $c_1, c_2, \dots\dots, c_3$ con algunos documentos semilla o de entrenamiento D_c, para calcular la probabilidad de que un documento pertenece a una determinada clase se utiliza:

$$\frac{|D_c|}{\sum_c |D_c|}$$

Dentro de esta técnica se encuentra la clasificación bayesiana, es una de las técnicas que más se han utilizado en aprendizaje automático, la simplicidad y efectividad con la que trabajan los hace altamente recomendables, funciona en base a la probabilidad de que una determinada palabra pertenezca a cierta clase (nombre, lugar, fecha); sin embargo, esta técnica asume que todas las clases evaluadas son independientes, si el vocabulario es mayor que el número de documentos, los documentos pequeños no son tomados en cuenta. Los modelos ocultos de Markov se utilizan para modelar procesos que varían durante el tiempo, se componen de nodos que representan estados, los cuales tiene asociados una distribución de probabilidad de emisión de símbolos y una probabilidad de transición entre estados. En la EI los modelos ocultos de Markov se entrenan para encontrar la matriz y el vector, de modo que pueda encontrar el camino más probable entre los estados, dado un símbolo de entrada.

1.6.2.3. Técnicas bio-inpiradas

Diseñan métodos heurísticos de los sistemas naturales o sociales, dentro de estos métodos se encuentran las Redes Neuronales, las cuales simulan el funcionamiento del cerebro humano, cada neurona artificial en un principio es entrenada con un conjunto patrones (textos) que arrojan un conjunto de salidas (entidades) ya conocidas para luego comparar el error. La arquitectura de una red puede ser lineal o una malla bidimensional donde se maneja como medida de asociación la vecindad. Por otra parte, los algoritmos evolutivos trabajan con la teoría de Darwin, para el problema de EI se lo maneja como un problema de optimización.

1.6.2.4. Otras Técnicas

Las técnicas revisadas anteriormente trabajan en el tratamiento del texto; sin embargo, existen otras aproximaciones como por ejemplo, utilizar un grafo de documentos, el cual permite realizar agrupaciones de documentos afines, extraer características de los grupos y sus relaciones. Otra técnica que se está difundiendo masivamente es el Análisis Conceptual de documentos o Análisis Formal de Conceptos (FCA) donde se modela matemáticamente los conceptos contenidos en textos y representan la relación de sus conceptos en forma de un grafo, generado a partir de atributos iniciales.

1.7. Aporte Personal

El PLN, RI y EI a pesar de que tratan problemas diferentes utilizan ciertos métodos comunes (tabla 1.10), por ejemplo el modelo determinista vectorial se utiliza en la mayoría de sistemas RI y EI, la principal razón de su uso se debe a que manipulan los términos en base a la frecuencia de aparición y relevancia, estos métodos son muy útiles cuando se manejan grandes cantidades de información. Los métodos probabilísticos son utilizados por los tres componentes (PLN, RI y EI). Al manejar grandes cantidades de información la probabilidad es la vía más aconsejable para conseguir resultados efectivos, las herramientas de PLN trabajan en su mayoría con corpus previamente etiquetados, para la clasificación de una nueva palabra los sistemas procesan la información almacenada estadísticamente y obtienen la probabilidad para cada caso posible, lógicamente la palabra es asignada a la clase con mayor probabilidad, de forma similar funcionan los sistemas de EI; sin embargo, todo no se puede dejar a la probabilidad ya que

existen casos particulares que no obedecen las normas generales, en estos casos se hace uso de restricciones o reglas conocidos como Modelos Simbólicos.

La RI puede plantearse como un problema de clasificación (relevante o no relevante), a partir de esto las técnicas de inteligencia artificial como las Redes Bayesianas, son utilizadas para mejorar los resultados del modelo vectorial y poder hacer predicciones. El PLN y EI son problemas en los que se tiene en cuenta el contexto, cierta palabra no tiene el mismo significado ni representan la misma entidad en diferentes escenarios, para lidiar con el problema del contexto estos sistemas hacen uso de técnicas bioinspiradas, las cuales les permite integrar cierta cantidad de información de partida y estas técnicas generan patrones, reconocen características, en definitiva aprenden y generan conocimiento iterativo, es así que cada vez que procesan una nueva palabra recurren los patrones generados y formulan una respuesta. Un resumen de lo expuesto anteriormente se encuentra a continuación:

Tabla 1.10: Cuadro comparativo entre RI, PLN y EI

Criterios	RI	PLN	EI
Métodos			
• ***Deterministas***			
○ *Modelo Vectorial*	x		x
• ***Empíricos o estadísticos***		x	
○ ***Probabilísticos***			
▪ *Redes Bayesianas*	x		x
• ***Simbólicos***		x	
• ***Modelos Biológicos***			
○ *Redes Neuronales*		x	x
○ *Algoritmos Genéticos*		x	x
Analizan	Documentos	Documentos + Datos	Datos

1.8. Discusión final

La RI consta de varios modelos entre los cuales se encuentra los Sistemas de RI de Búsqueda Exacta donde sólo pueden existir dos resultados, relevante o no relevante; pese a la precisión al menos teórica que puedan proporcionar estos sistemas, la rigurosidad de los juicios hacen que sean poco prácticos a la realidad de la información de la Web. Como solución a este problema aparecieron modelos más flexibles que hacen uso de la estadística para obtener los resultados más relevantes, estas técnicas son llamadas Sistemas de RI Aproximada, en donde el modelo vectorial es el más representativo, ya que reúne las bondades tanto del enfoque de búsqueda Exacta como el de búsqueda Aproximada. Por otra parte los motores de búsqueda son el campo de aplicación de los sistemas de RI, estos son ampliamente utilizados y mejorados cada día en el mercado, sin embargo a medida que aumenta el tamaño de la Web, los buscadores de información van perdiendo terreno en lo que se refiere a precisión; los resultados dejan de ser relevantes, los usuarios se ven en la necesidad de hacer artificios para encontrar la información que requieren, esto se debe en gran medida a la carencia de estructuras, por tal motivo se hace

necesario la evolución de la Web actual desordenada a una Web con sentido; la llamada WS, donde la disposición estructurada de la información permite recuperar información más útil y precisa.

La evolución de la Web a dado lugar al desarrollo de motores de búsqueda más especializados llamados buscadores semánticos, cuya meta principal es resolver de manera precisa, fácil y rápida los requerimientos del usuario, para esto utilizan métodos que les permiten entender los conceptos y el contexto de la consulta del usuario. Aunque la WS no se encuentra establecida totalmente y pasará algún tiempo para que lo este, algunos motores de búsqueda semánticos han echado mano a la creatividad para dotar de alguna manera de sentido a la Web actual y así poder prestar el servicio semántico que en términos generales todavía se encuentra en fase beta; ya que por un lado los usuarios dependen exclusivamente de las palabras claves y por otro aun no se tiene claro lo que realmente puede hacer un buscador semántico para el usuario (Mitos de los buscadores semánticos). Pese a la gran ayuda que se supone que representan los buscadores semánticos, estos seguirán siendo paquetes de software sofisticados a los cuales solo tendrán acceso unos pocos (personas especializadas) si no se cuenta con una interfaz que permita la comunicación en lenguaje natural entre el usuario y el buscador. El PLN disciplina encargada del reconocimiento y utilización de la información expresada en lenguaje natural haciendo uso de las máquinas parece ser la interfaz perfecta, para que la información expresada en lenguaje natural del usuario pase a lenguaje lógico de máquina y que a su vez los resultados expresados en lenguaje lógico sean representados en lenguaje natural entendible al ser humano. La EI es un proceso posterior a la RI, se encarga de extraer datos importantes de los documentos recuperados y utiliza activamente tareas de PLN para cumplir con su función. Las conferencias MUC son la principal actividad para el desarrollo de metodologías para la EI, a partir de estas se adquiere experiencia y nuevos enfoques para tratar este problema. Existen varios tipos de enfoques para el problema de EI, sobresalen principalmente las técnicas de aprendizaje automático (Inteligencia Artificial) las cuales se apoyan en la estadística y hechos históricos para predecir nuevos.

Capítulo 2 Análisis Comparativo de Herramientas Open Source sobre sistemas de RI, PLN, EI

2.1. Introducción

En este capítulo se presenta un análisis de las diferentes herramientas disponibles en la Web para resolver las tareas que existen dentro de un buscador semántico (RI, PLN, EI). Se han encontrado varias herramientas para resolver problemas particulares y otras de propósito general, de todo el conjunto de herramientas encontradas se ha preferido aquellas que trabajan con la tecnología java, debido a que el repositorio de objetos de aprendizaje, Dspace; de la Universidad13 trabaja con ella, también se ha visto la necesidad de trabajar con herramientas con suficiente documentación ya que es muy importante a la hora de personalizar las herramientas a intereses específicos.

Algunas de las herramientas que no se tomaron en consideración en el presente análisis constan: Para RI se han desechado SemacticVectors[14], SemanticSpace[15] por ser herramientas con poca documentación (solo javadocs) lo que hace complicado aprovechar las prestaciones de la herramienta. En PLN en un principio se consideró las herramientas de LingPipe[16], Mallet[17], Stanford NLP[18], sin embargo; se dejaron de lado por qué no procesan el lenguaje español o en su defecto no cuenta con modelos para hacerlo, otras herramientas como Natural Language Toolkit[19] trabajan con Phyton lo cual hace difícil la integración con java.

Los datos de prueba así como también las métricas utilizadas en la mayoría de componentes han sido tomados de eventos científicos internacionales, se toma en cuenta conferencias como: Senseval[20], CONLL[21] y MUC[22]. Cada una ofrece un conjunto de datos de entrenamiento y prueba los cuales han sido procesados y acoplados al escenario de cada componente.

Este análisis está enfocado a someter a las herramientas a problemas reales dentro de cada campo, por tal motivo se han preferido utilizar datos de prueba de congresos expertos en las problemáticas tratadas en este trabajo. Se busca también examinar las prestaciones de cada herramienta, así como también conocer su funcionamiento, detalles de implementación y uso.

[13] http://repositorio.utpl.edu.ec
[14] http://code.google.com/p/semanticvectors/
[15] http://code.google.com/p/airhead-research/wiki/DocumentRepresentations
[16] http://alias-i.com/lingpipe/web/models.html
[17] http://mallet.cs.umass.edu/index.php
[18] http://nlp.stanford.edu/software/index.shtml
[19] http://www.nltk.org/Home
[20] http://www.senseval.org/
[21] http://www.cnts.ua.ac.be/conll/
[22] http://www-nlpir.nist.gov/related_projects/muc/index.html

2.2. Marco Teórico

Las herramientas son analizadas en base a medidas específicas dentro de cada campo, es así que para RI se utiliza la precisión, que según Salton (1983) expresa la medida en que el sistema de RI muestra resultados pertinentes ante una consulta determinada, y la exhaustividad o recall que es la proporción de material relevante recuperado, del total de los documentos relevantes en un repositorio (keen (1971)); dichas métricas se calculan mediante las fórmulas siguientes:

$$Precisión = \frac{Documentos\ \ relvantes\ \ recuperados}{Documentos\ \ recuperados} \quad (1)$$

$$Exhaustividad = \frac{Documentos\ \ relevantes\ \ recuperados}{Documentos\ \ relevante\ s} \quad (2)$$

La herramienta analizada en este punto se describe a continuación.

Semantic Information Retrieval Engine-SIREN[23]

Es la extensión semántica del sistema de RI sintáctico Lucene, esta herramienta ha sido escogida por la facilidad que tiene para trabajar con cualquier tipo de contenido estructurado o semi-estructurado y puede ser combinada con los métodos de Lucene.

SIREN añade una etiqueta llamada "Tuples" a las que vienen por defecto en Lucene (title, date,etc.), esta etiqueta acepta información especial estructurada llamada N-Tuplas que es un formato basado en texto plano para codificar datos semi-estructurados como tripletas RDF u otros formatos, su sintaxis es derivada de las Tripletas RDF, cada N-Tupla es un súper conjunto de una N-Tripleta.

Posee algunas características como por ejemplo: Un eficiente manejo de grandes volúmenes de contenido semi-estructurado, permite realizar búsquedas semi-estructuradas o estructuradas de texto completo, el indexado de datos semi-estructurados es flexible, integración completa con Lucene o Solr, se puede utilizar como librería externa a cualquier proyecto java, integración de búsquedas estructuradas con consultas booleanas y contiene un ranking de búsqueda semi-estructurada.

Entre sus desventajas se tiene la incompatibilidad con algunas versiones de Lucene, solo se puede trabajar en contenido estructurado o semi-estructurado, la herramienta se encuentran en

[23] http://siren.sindice.com/

versiones iniciales y la documentación es básica, aunque posee código fuente de ejemplo. Las configuraciones de instalación y uso se encuentran descritas en el Anexo 1

Por otro lado el análisis de las herramientas de PLN se ha dividido en dos partes, en la primera; se analizan aquellas que realizan tareas de procesamiento morfológico y sintáctico del texto, en este análisis participan OpenNLP [24] y Freeling[25], se ha desechado otras herramientas por varias razones entre ellas: son incompatibles con la tecnología java, no realizan procesamiento para el lenguaje español o no poseen documentación. La segunda parte del análisis contempla a las herramientas OpenCalais[26], AlchemyApi [27]y Freeling las cuales realizan el análisis semántico y pragmático del texto. Cabe señalar que la mayoría utilizan aprendizaje automático, algunas de ellas manejan cierta parte de PLN y solo Freeling abarcan todos los procesos (morfológico, sintáctico, semántico y pragmático) de PLN.

Las métricas de evaluación utilizadas en los dos análisis de PLN son la proporción de aciertos y errores obtenidos en cada herramienta, la instalación y configuración de las herramientas de este apartado se encuentran en el Anexo2. Las herramientas analizadas se mencionan a continuación:

OpenNLP

Proviene de un conjunto de proyectos relacionados con este campo, posee varios procesos entre ellos: Detección de sentencias, tokenización, pos-tagging, segmentación y análisis de texto, detección de entidades nombradas. Utiliza máquinas de aprendizaje basadas en el paquete Maxent[28] para trabajar la correferencia en texto. Cuenta con modelos pre procesados para el tratamiento del texto, ofrece la posibilidad de crear un modelo propio a partir de un determinado contenido, utiliza el conjunto de etiquetas EAGLES [29]para el español y TreeBank [30] para el inglés y posee la ventaja de realizar el análisis en varios lenguajes: inglés, español, alemán y tailandés.

Como desventaja principal de la herramienta se tiene: Modelos limitados para el lenguaje español.

OpenCalais

Es una API que puede ser consumida como servicio Web, extensión de Firefox o modulo de Drupal, es desarrollada y mantenida por la empresa de noticias Thomson Reuters desde el 2008. Este servicio permite identificar entidades de cualquier entrada ya sea texto, HTML o XML y la respuesta puede ser en texto plano o en RDF enriquecido de metadatos. Este servicio es de uso libre para toda clase de fines comerciales o no comerciales, desde marzo de 2009

[24] http://opennlp.sourceforge.net/
[25] http://garraf.epsevg.upc.es/freeling/demo.php
[26] http://www.opencalais.com/
[27] http://www.alchemyapi.com/
[28] http://maxent.sourceforge.net/
[29] http://garraf.epsevg.upc.es/freeling/doc/userman/parole-es.html
[30] http://bulba.sdsu.edu/jeanette/thesis/PennTags.html

forma parte de Linked Data[31]. Está en la capacidad de reconocer a personas, hechos, regiones; en la figura 2.1 se muestra en detalle las entidades que pueden ser reconocidas por OpenCalais:

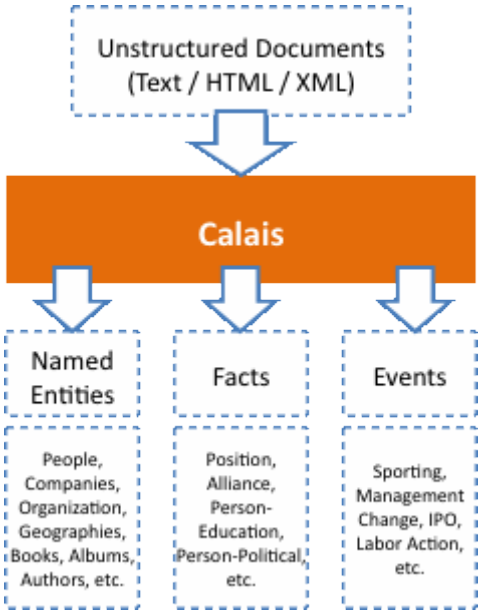


Figura 2.1: Entidades reconocidas por OpenCalais[32]

Su funcionamiento se basa en procesos de PLN, machine learning y otros métodos que al parecer son de clasificación ya que por cada entidad reconocida facilita su relevancia expresada en porcentaje. En la **Figura 2.2** se puede observar la relevancia calculada para la entidad "Wayne Rooney" que aparece en un texto.

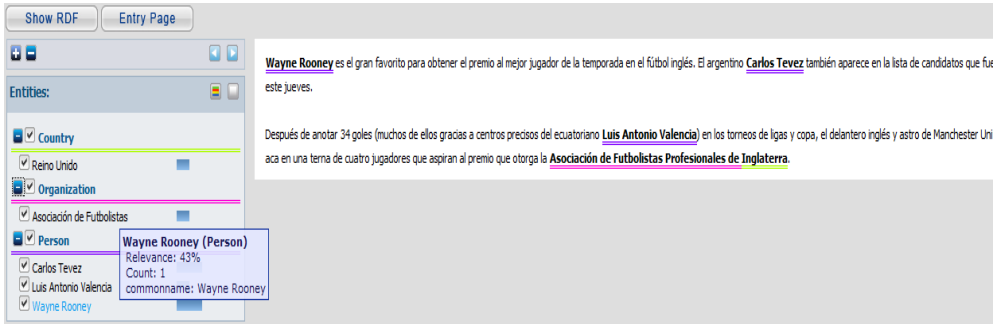


Figura 2.2: Relevancia para la entidad persona

Cuenta con las funcionalidades de reconocer varios idiomas (inglés, francés y español), capacidad de integrar la consulta con Linked Data, soporta datos en texto plano o estructurado,

[31] http://linkeddata.org/
[32] http://www.opencalais.com/about

requiere de pocas configuraciones, soporta varios lenguajes de programación entre ellos: Java, PHP, Action Script, .NET, Ruby y soporte para SOAP y REST.

Posee algunas características como suficiente documentación y soporte, rapidez en el procesamiento de consultas, personalizable en base a parámetros, simple de implementar y personalizar.

Pese a las ventajas anteriormente descritas las consultas del servicio Web son restringidas (50.000 por día), las entidades reconocibles para español y francés son limitadas.

AlchemyAPI

Es un servicio Web que permite extraer información semántica como lugares, compañías, temas, información acerca de personas. Utiliza un identificador automático de lenguaje que puede reconocer entre 97 lenguajes diferentes. Además contiene funcionalidades de extracción de entidades, etiquetado de conceptos y extracción de términos, clasificación de textos, categorización de tópicos. Es de tipo propietaria pero posee una parte de acceso libre que permite hasta 30.000 accesos a la API al día. En la siguiente figura se muestra todos los servicios que ofrece:

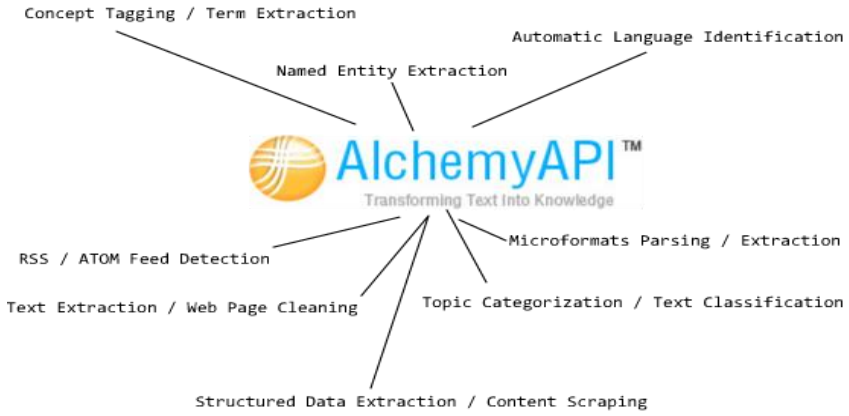


Figura 2.3: Componentes de AlchemyAPI. Elaboración propia

Como características, cuenta con soporte para datos en texto plano, estructurado, micro-formatos y páginas Web, posee algoritmos de desambiguación, servicio Web consumible con Rest, soporta varios lenguajes de programación: Java, Perl, Ruby, Python, PHP-5, C /C++, .Net.

Según la página de le herramienta las ventajas más relevantes son: Servicio de posicionamiento en buscadores, extracción de términos, reconocimiento de entidades, categorización de documentos, soporte para manejar RSS, resultados en texto plano o RDF, capacidad de integración con Wordpress.

FreeLing 2.1

Es una suite de tratamiento del lenguaje escrita en C, posee una API para trabajar con Java, Python y Perl; fue desarrollada por la Universidad Politécnica de Catalunya (UPC)[33] y el Centro de Tecnologías y Aplicaciones del Lenguaje y del Habla (TALP)[34] y se distribuye gratuitamente bajo licencia GPL.

Tiene soporte para varios lenguajes como son: español, catalán, gallego, italiano e inglés, los servicios que prestan son: lemantización[35], etiquetado, agrupamiento, el reconocimiento de entidades nombradas, reconocimiento de cantidades (moneda, relaciones, magnitudes físicas), fechas, hora y anotador del sentido.

Su arquitectura según Carreras y Pradó (2001) está basada en un modelo cliente servidor que consta de 2 capas: Una capa básica de servicios lingüístico y de análisis (morfológico, taggin, parsing, etc.) y otra capa de aplicación orientada a atender los requerimientos. En lo que se refiere a su funcionamiento esta herramienta trabaja con varias tecnologías, es así que para el etiquetado de la oración utiliza el algoritmo de los trigramas siguientes basado en Modelos Ocultos de Markov (HMM), Brants (2000). La clasificación de entidades nombradas utiliza el algoritmo de AdaBoost (Schapire y Singer (1999)). El análisis morfológico como sintáctico utiliza etiquetas EAGLES para representar la información morfológica de las palabras. La anotación del sentido de las palabras tanto del inglés como del español utiliza la versión reducida 1.6 de WordNet [36].

Esta herramienta tiene las ventajas de realizar el análisis completo del lenguaje (morfológico, sintáctico, semántico y pragmático), cuenta con suficiente documentación, cada módulo puede ser usado independientemente, resultados de fácil interpretación, altos niveles de precisión. Sin embargo cuenta con una versión reducida de Wordnet lo cual constituye un problema.

La evaluación de herramientas para la EI se toma las métricas de precisión y exhaustividad según las conferencias MUC, las cuales se calculan a partir de los siguientes parámetros, Díaz, J y Pérez, F (2009):

[33] http://www.upc.edu/
[34] http://www.talp.cat/talp/
[35] Proceso para reducir una palabra a su raíz.
[36] http://wordnet.princeton.edu/

Tabla 2.1: Parámetros utilizados para la evaluación de las herramientas de EI

Nombre	**Fórmula**	**Descripción**
Número correcto	COR	Ocasiones donde la clave y la respuesta coinciden
Número incorrecto	INC	Ocasiones donde la clave y la repuesta no coinciden
Numero perdido	MIS	Ocasiones donde existe una clave pero no una respuesta
Número falso	SPU	Ocasiones donde existe una respuesta pero no una clave
Número posible	POS = COR + INC + MIS	Número de registros en la clave
Número actual	ACT = COR + INC + SPU	Número de registros en la respuesta

En base a los parámetros anteriores la precisión y exhaustividad se expresa a partir de las siguientes fórmulas:

$$Precisión = \frac{COR}{ACT} \quad (3)$$

$$Exhaustividad = \frac{COR}{POS} \quad (4)$$

La herramienta analizada en esta parte se describe a continuación:

Apache UIMA (Unstructured Information Management Applications)[37]

Es un framework para el análisis de grandes volúmenes de información no estructurada con el fin de descubrir información relevante para el usuario. Fue creado como un proyecto de incubación por IBM en el 2006 y actualmente es un proyecto de alto nivel. Posee una arquitectura modular lo cual permite utilizar, escribir y reutilizar el código libremente. Cuenta con una serie de módulos que admiten realizar operaciones cotidianas de análisis del texto; cuenta con un analizador de entidades que reconocen personas, organizaciones y relaciones como: "trabaja para" o "ubicado en". La arquitectura modular permite integrar componentes personalizados y otro tipo de componentes. Soporte para varios lenguajes de programación (Java y C++) y trabaja con varios lenguajes (Español, Inglés, Francés, etc.). Su instalación puede ser como librería o complemento de Eclipse (entorno de desarrollo). Procesa cualquier tipo de documentos incluyendo archivos de audio y video. Puede ser integrado con otras herramientas, por ejemplo: OpenNlp, OmniFind, InfoSphere Warehouse. Permite el desarrollo de motores de análisis personalizados. La desventaja principal de esta herramienta es que cuenta con un número limitado de archivos descriptores para es español.

Las configuraciones de la herramienta se podrán revisar en el Anexo 3.

[37] http://uima.apache.org/

GATE (General Architecture for text Engineering)[38]

Es una herramienta que incluye un conjunto de métodos para EI y PLN en muchos lenguajes, es desarrollada y mantenida por la Universidad de Sheffield desde 1995. Esta escrita complemente en java y permite el análisis del texto utilizando un sistema llamado ANNIE (Nearly-New Information Extraction System), que es un compendio de algoritmos de estado finito, además incluye una herramienta llamada JAPE[39] la cual realiza la transducción de estados finitos en anotaciones basadas en expresiones regulares.

JAPE permite reconocer anotaciones utilizando gramáticas que están constituidas por pares ordenados de patrones y reglas. Por lo general, las expresiones regulares que contienen las anotaciones poseen cadenas de caracteres con una simple secuencia lineal de elementos, sin embargo el sistema JAPE aplica estructuras mucho más compleja de datos ya que no todas las cadenas de texto se pueden anotar usando u estado finito determinista (para mayor información del funcionamiento de JAPE véase (Dhava, Taha, & Phil, 2009)). Las configuraciones de GATE se encuentran en el Anexo 3.

2.3. Evaluación

En este apartado se evalúan las características más significativas de cada componente, también se describe el conjunto de datos de prueba utilizado y se presenta una discusión final de los resultados.

Análisis RI

En la Evaluación del sistema RI se utilizó como conjunto de pruebas los documentos contenidos en el repositorio de la Universidad, se trabaja específicamente con la información del índice-Lucene, la misma es transformada en datos tabulares[40], de esta información se formularon cinco consultas, así:

Tabla 2.2: Detalle de consultas aplicadas sobre los sistemas de RI

Consulta	Detalle
Consulta 1	Todos los ítems que son videoconferencias
Consulta 2	Diagnóstico y tratamiento de la hipatidosis
Consulta 3	Documentos que hablen de economía
Consulta 4	Documentos de ingenieros que hablen acerca del cálculo
Consulta 5	Videos acerca de sociología

Los resultados obtenidos por cada consulta expresados en términos de precisión y exhaustividad se presentan enseguida:

[38] http://gate.ac.uk/
[39] http://gate.ac.uk/sale/tao/splitch8.html#x12-2080008
[40] https://dev.deri.ie/confluence/display/SIREn/Indexing+and+Searching+Tabular+Data

Tabla 2.3: Resultados de la herramienta SIREN

Consultas	Precisión (%)	Exhaustividad (%)
Consulta 1	80	8,8
Consulta 2	50	50
Consulta 3	60	40
Consulta 4	42,9	50
Consulta 5	50	42,9

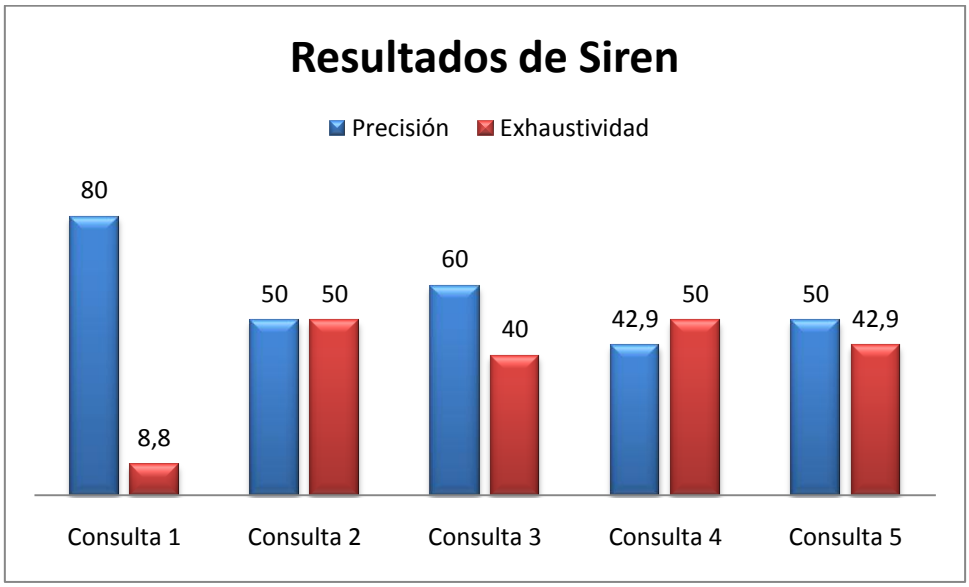


Figura 2.4: Análisis de SIREN (precisión y exhaustividad)

En la Figura 2.4 se evidencia que las variaciones más significativas entre precisión y exhaustividad son entre las consultas 1 y 3 donde la diferencia está por encima de los 10 puntos, se puede entender que el contraste tan marcado es por el hecho de que dichas consultas tienen un nivel de descripción muy general, en especial la consulta 1 que posee el índice de precisión más elevado de todas la consultas; la razón es que el repositorio posee una cantidad importante de OA con esta descripción.

En cambio para las consultas 2, 4 y 5 que son más descriptivas se observa que precisión no superan el 50%; sin embargo, se observa una relación directa con la exhaustividad. Los resultados de SIREN en si son buenos si consideramos el hecho que para el análisis se estructuraron datos generales basados en el índice de lucene, esto da la pauta que para datos con mayor nivel de descripción los resultados pueden mejorar significativamente.

Análisis PLN

Como se mencionó en el apartado anterior se análisis de PLN se dividió en dos partes, ya que no todas las herramientas escogidas realizaban un análisis completo de PLN. A continuación se presenta la primera parte del análisis.

Análisis morfológico y sintáctico

Este análisis utiliza los recursos de las conferencias Senseval, en el análisis morfológico se utiliza los recursos de la actividad "Spanish lexical sample"[41] los cuales están constituidos de extractos de noticias de la agencia EFE, en el análisis sintáctico se emplea el data set de la actividad de anotación semántica[42] de las mismas conferencias; la información sobre la estructura de estos recursos se encuentran en Márquez (2004) y Rigau (2000). Estos recursos están etiquetados morfológicamente y sintácticamente, de estos se han escogido 5 extractos de texto, los cuales se detallan en la Tabla 2.4.

Tabla 2.4: Extractos de texto utilizados para el análisis

Test	Área	Total Tokens	Total Etiquetas	Total Sentencias
text1	ECO:ECONOMIA,SECTORES,AGROALIMENTACION	130	130	3
text2	SOC:SOCIEDAD,SUCESOS	166	166	3
text3	CYT:CIENCIA-TECNOLOGIA,CIENCIA	163	165	3
text4	CYT:CIENCIA-TECNOLOGIA,AMBIENTE-NATURALEZA	143	143	3
text5	SOC:SOCIEDAD-SALUD,SALUD	128	128	3

En el análisis de estas herramientas se prueba básicamente la detección de oraciones o sentencias, segmentación de palabras (tokenización), el reconocimiento de componentes básicos o taggin (verbos, adjetivos, etc.) y el análisis sintáctico (parsing). En este punto cabe indicar que la herramienta OpenNlp no cuenta con modelos para el lenguaje español para realizar parsing por tal motivo en este criterio la herramienta no aplica para el análisis. Los resultados obtenidos se detallan a continuación:

Tabla 2.5: Resultados del Análisis Morfológico y Sintáctico

Criterios	OPenNlp		Freeling	
	Aciertos %	Errores %	Aciertos %	Errores %
Detección de Sentencias	86,68	19,98	93,34	6,66
Segmentación de Palabras (tokenización)	100	5	99,88	0,96
Reconocimiento de Componentes básicos (taggin)	79,9	18,44	88,32	11,68
Análisis sintáctico (parsing)	0	0	89,4	10,6

[41] http://www.senseval.org/senseval3/data.html
[42] http://www.lsi.upc.edu/~nlp/semeval/msacs_download.html#data

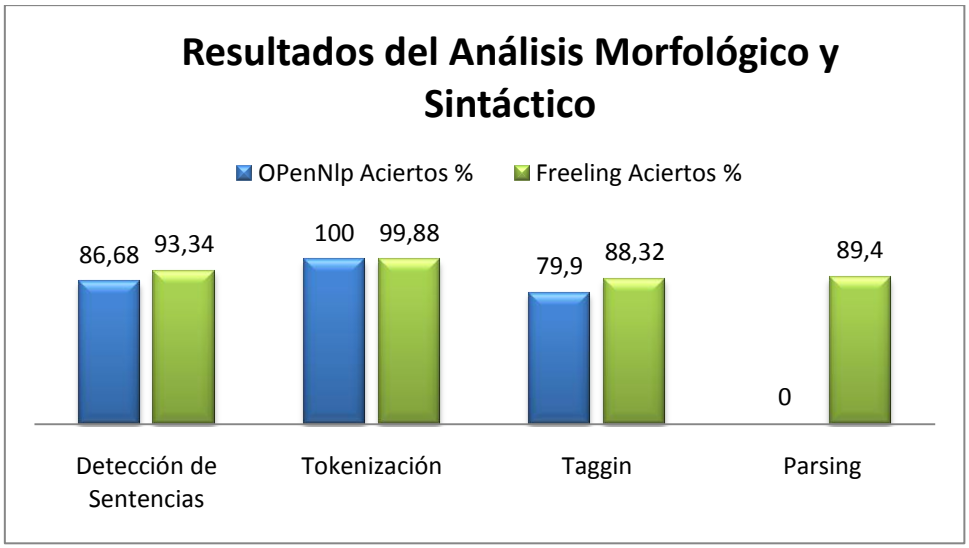


Figura 2.5: Resultados del análisis morfológico y sintáctico de Freeling y OpenNlp

Los resultados en los tres primeros criterios no existen grandes diferencias, en el porcentaje acumulado OpenNlp posee un 88,86 % de aciertos contra un 93,85% de Freeling, además se debe tomar en cuenta que este último realiza el proceso de parsing cuyos resultados son aceptables, en consecuencia se puede concluir que Freeling es la mejor opción en este proceso; sin embargo, esta herramienta participan en la segunda parte del análisis PLN por lo que dependiendo de los resultados obtenidos se podrá concluir sobre la pertinencia de utilizar Freeling en la implementación.

Análisis Semántico y Pragmático

En este análisis e escogieron aquellas herramientas que permiten asociar a cada palabra con un sentido acorde al contexto de la oración, en esta etapa del PLN se abordan algunos procesos como son: el reconocimiento de entidades nombradas (NER), desambiguación del sentido, descubrimiento de relaciones, los datos que se utilizan corresponden a las conferencias: CONLL 2002, específicamente de la actividad de reconocimiento de entidades nombradas[43] y a los recursos de la actividad "Spanish lexical simple" de las conferencias Senseval que ya se utilizo en el apartado anterior, los resultados se presenta a continuación:

Tabla 2.6: Criterios evaluados entre AlchemyApi, OpenCalais y Freeling

Criterios	OpenCalais (%)		AlchemyApi (%)		Freeling (%)	
	Aciertos	Errores	Aciertos	Errores	Aciertos	Errores
Reconocimiento de entidades nombradas	38,5	61,5	32,4	67,6	68,5	31,5
Desambiaguación del sentido	25	35	22	34	60	40
Descubrimiento de relaciones	10	34	43	45	40	40

[43] http://www.cnts.ua.ac.be/conll2002/ner/

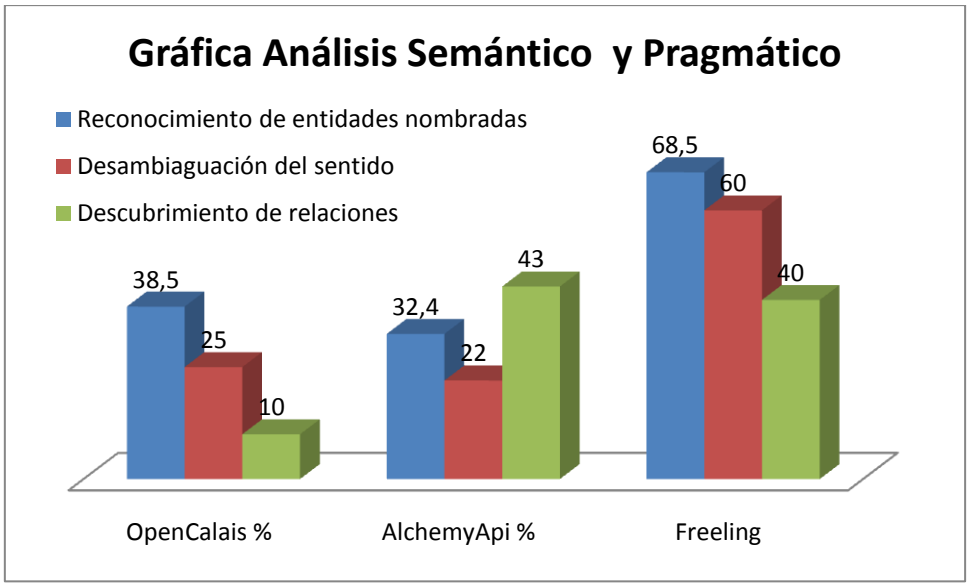


Figura 2.6: Análisis semántico entre AlchemyAPI , Freeling y OpenCalais

En los resultados se evidencia que Freeling es superior en las tareas de reconocimiento de entidades nombradas y desambiguación del sentido, en cambio en la tarea de descubrimiento de relaciones hay una supremacía de tres puntos de AlchemyAPI. Analizando los resultados acumulados entre las herramientas, Freeling es superior con 56 % comparado con el 24 y 32 % de OpenCalais y Alchemy API respectivamente, considerando los resultados de análisis anterior (morfológico- sintáctico) se concluye que la herramienta que presta mayores ventajas en lo que se refiere a PLN es Freeling, ya que es una herramienta completa y sus resultados han sido superiores en los dos análisis sobre las demás herramientas.

Análisis de herramientas de EI

Los datos de pruebas utilizados en este análisis corresponden a las conferencias MUC, evento especializado en EI, los recursos pueden ser obtenidos desde la sección de descargas[44] de las conferencias. Estos datos de prueba constan de un archivo que contiene reportes de noticias de la agencia EFE relacionados con actos terroristas, y un segundo archivo que posee una plantilla que detalla los hechos, lugares, personas, fechas y organizaciones que los sistemas de EI deben reconocer. Los resultados obtenidos se detallan a continuación:

[44] http://www-nlpir.nist.gov/related_projects/muc/muc_data/muc34.tar.gz

Tabla 2.7: Resultados obtenidos del proceso de EI

	Apache UIMA		GATE	
	Precisión	**Exhaustividad**	**Precisión**	**Exhaustividad**
texto1	60	50	70	60
texto2	60	50	53	40
texto3	40	40	60	80
texto4	70	70	74	80
texto5	90	90	70	70
	64	60	65,4	66

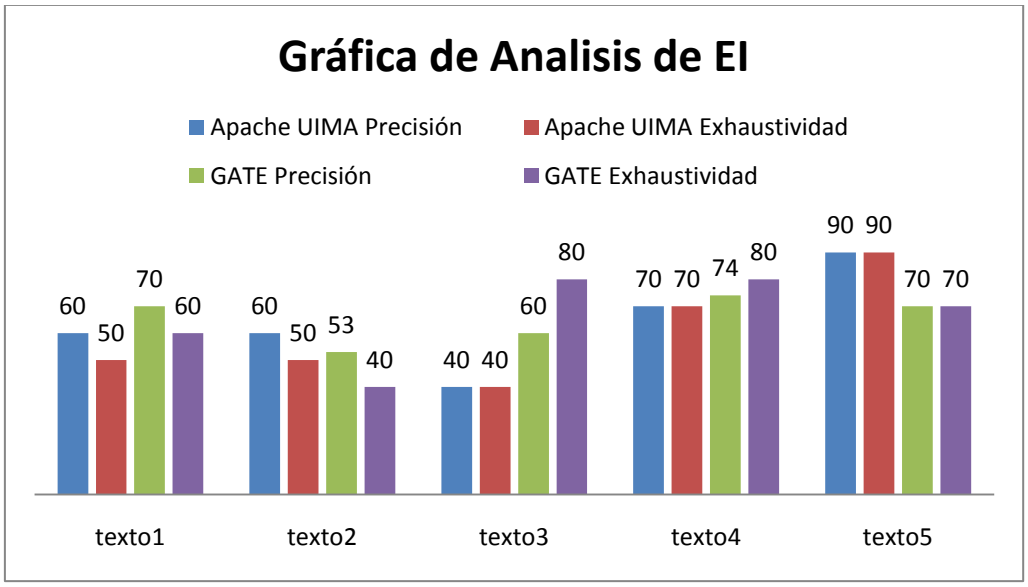


Figura 2.7: Gráfica de los resultados obtenidos de EI

Los resultados obtenidos al utilizar Apache UIMA son buenos, en la mayoría de los textos analizados la herramienta obtiene una precisión superior al 50 % con excepción del texto 3 donde los resultados son bajos. En cambio los resultados obtenidos con GATE son superiores en todos los textos analizados, los resultados más bajos corresponden al texto 2 con 53 y 40 % de precisión y exhaustividad respectivamente. Aunque Apache UIMA posee casi un porcentaje perfecto en el texto 5 (90 %); en términos generales, GATE posee resultados mucho más precisos así lo demuestran los porcentajes acumulados.

Tabla 2.8: Precisión y Exhaustividad Acumulada

Aplicación	**Precisión**	**Exhaustividad**
Apache UIMA	64	60
GATE	65,4	66

Adicionalmente se vio la necesidad de contar con un repositorio para administrar las entidades reconocidas en los documentos para realizar sobre ellos futuras búsquedas, se analizaron varias

herramientas que realizan este proceso entre ellas: WSMO Studio[45], Kim platform[46], IMS Tools[47] todos ellos son sistemas de Administración de Información y Conocimiento, (Knowledge and Information Management) a los cuales se los procedió a analizar y cuyos resultados se presentan en la siguiente tabla:

Tabla 2.9: Análisis de Resultados de Knowledge and Information Management

Aspectos	WSMO Studio	Kim-Platform	IMS TOOLS
Sistema Operativo soportado	todos	todos	todos
Tecnología que utiliza (java)	si	si	si
Requerimientos de Intalación y configuración	básicos	básicos	altos
Requerimiento de Hardware	básicos	básicos	altos
Manuales y Documentación	básica	completa	completa
Vetajas y Desventajas	Proyecto no muy estable	Integra GATE por defecto	Licencia Propietaria de IBM

De los resultados obtenidos se procedió a desechar a WSMO Studio ya que al parecer es una versión inicial de Kim-platform. Por otro lado IMS TOOLS es una herramienta madura y especializada en la administración información y conocimiento; sin embargo, debido a su licencia propietaria se decidió dejarla de lado, finalmente Kim-Platform es la herramienta que mejor se adapta al proyecto ya que entre sus prestaciones se encuentra que posee por defecto una ontología sobre la cual realizar búsquedas adicionalmente cuenta con la ventaja que integrar GATE por anotar para anotar semánticamente los documentos, posee una api para realizar búsquedas estructuradas y también da soporte para realizar búsquedas sintácticas.

2.4. Discusión Final

Después de haber realizado la evaluación de las herramientas para los diferentes componentes, se está en la capacidad de escoger las mejores para la fase de desarrollo. Los resultados demuestran que la herramienta para EI (GATE) funciona muy bien con la configuración por defecto, esto da la pauta que con ajustes adicionales los resultados obtenidos se pueden mejorar.

En el caso de PLN, Freeling es la mejor opción, los resultados demuestran que pueden ser iguales o superiores a los obtenidos por otras herramientas, además Freeling posee la ventaja de realizar el análisis completo del lenguaje.

La herramienta Kim-platform posee una suite completa de administración, búsqueda además posee una api completa para la RI ya sea estructurada como sintáctica por lo cual se dejará de lado a SIREN, se procederá a utilizar el módulo de RI de kim-platform que es mas completa.

En resumen las herramientas escogidas para la fase de desarrollo son: Para RI y la administración de entidades se utilizará la Api de Kim-platform, GATE realizará los procesos de EI y finalmente la interfaz entre el usuario y el buscador la proporcionará Freeling.

[45] http://www.wsmostudio.org/
[46] http://www.ontotext.com/kim
[47] http://www-01.ibm.com/support/docview.wss?uid=swg21261315

Capítulo 3 Diseño e Implementación del buscador semántico de Objetos de Aprendizaje

3.1. Introducción

En el presente capítulo se define el diseño del buscador semántico, se comienza describiendo los principales problemas que tiene el buscador actual, también se presenta una visión macro sobre las fuentes de información que posee el repositorio DSpace, se nombra los principales usuarios que interactúan con el sistema. Luego se presenta los requerimientos del sistema conjuntamente con los respectivos casos de uso.

En la sección 3.4 se presenta la arquitectura del Buspace, se especifica también los cambios realizados al modelo de datos del repositorio actual, en la sección 3.6 se describe brevemente la ontología utilizada, se especifican al arquitectura, instancias, principales conceptos que la conforman, así como también las el listado de las diferentes API's utilizadas en el desarrollo del buscador.

Las secciones 3.7 y 3.8 se definen los detalles de implementación de los módulos desarrollados, en el primer módulo de migración (sección 3.6) se especifica de manera puntual los metadatos tomados de cada OA y en la sección siguiente se muestra los componentes principales del búscador como tal.

3.2. Descripción del problema

Después de haber realizado el análisis correspondiente, (ver **Anexo 5** documento Visión) el cual consistió en determinar el funcionamiento del buscador, las principales stakeholders, determinar las fuentes de información y recabar los principales problemáticas se llegó a determinar que el repositorio de OA Dspace es alimentando principalmente por cuatro fuentes (ver **Figura 3.1**): La Unidad de Virtualización se encarga de dar mantenimiento al repositorio así como también subir las video-conferencias (OA) producidas por la Universidad. El departamento de Procesos Técnicos de Biblioteca es el encargado de publicar las tesis generadas por la Universidad como OA al repositorio. Los profesores y alumnos de la universidad también se encargan de subir su material y finalmente los usuarios externos que también tienen la capacidad de subir material al repositorio. El único requisito para poder realizar estas actividades es estar registrado en el sistema, bajo este escenario se ha podido identificar tres problemas principales:

- **Gestión Básica de los Metadatos:** Los metadatos de los OA pasan por controles de calidad generales y básicos y están encaminados a verificar la ortografía de los mismos, más no su pertinencia, lo que trae como consecuencia que los metadatos muchas veces puedan contener errores, no reflejan exactamente el contenido del recurso o estén vacios. Esta información es crucial al momento de filtrar la información.

- **Consultas no descriptivas:** La poca estructuración de contenido y la gestión básica de metadatos hace que sea imposible formular consultas con la suficiente especificación y el proceso de búsqueda sea ineficiente, empleé demasiado tiempo para encontrar resultados precisos y desperdicie recursos. La búsqueda avanzada actual solo filtra la información por tres campos a la vez y proponer consultas más descriptivas es realmente inadmisible.

- **Procesamiento básico de Consultas:** El procesamiento de las consultas es básico, solo se eliminan stop words, no se determina el sentido de las palabras ni el contexto en que son utilizadas, en consecuencia no se establece de forma efectiva el requerimiento del usuario.

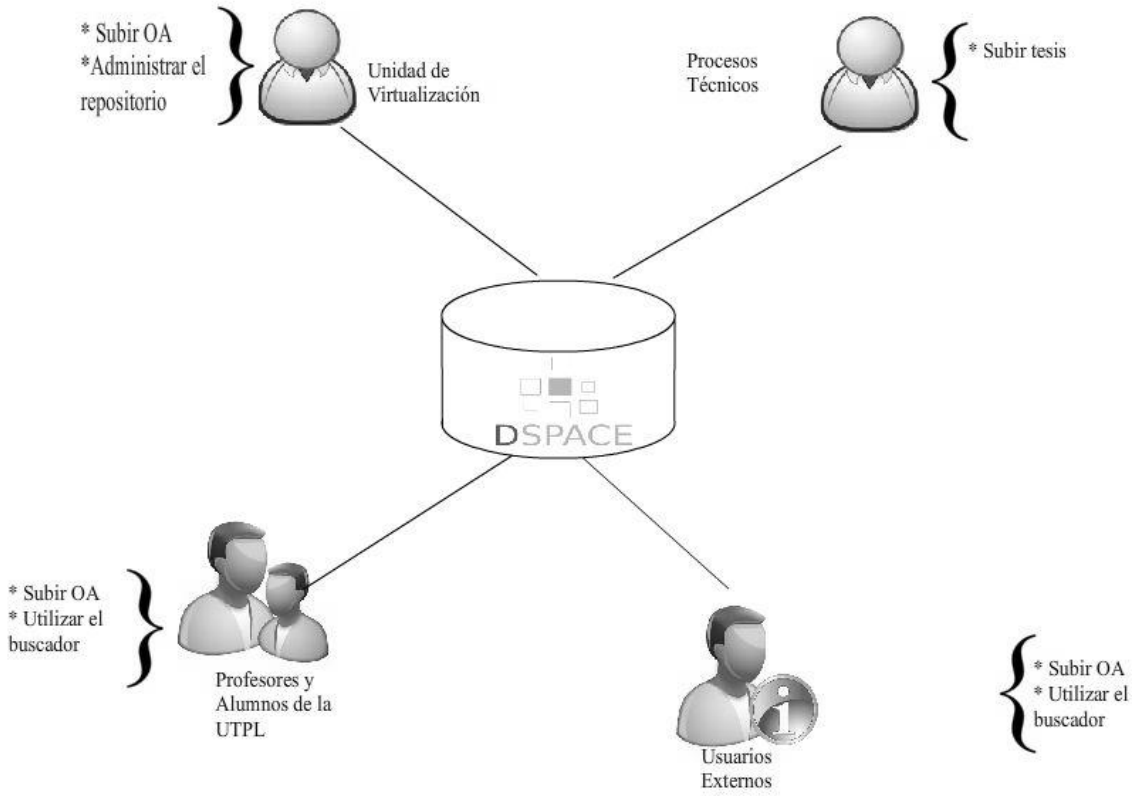


Figura 3.1: Fuentes de información de repositorio

Basados en el análisis anterior también se ha podido establecer los principales usuarios del sistema, los cuales se detallan enseguida:

- **Administrador:** Personal operativo de la Unidad de Virtualización, se encarga del mantenimiento y administración del repositorio y es usuario calificado en todas las actividades que posee el repositorio.
- **Usuario:** En este grupo entran profesores y alumnos y usuarios externos, tienen acceso a publicar OA, hacen uso del buscador, su nivel de expertise va de básico a experto del negocio.

3.3. Requerimientos

Los problemas encontrados en la sección anterior evidencian claramente que los metadatos y el procesamiento básico de las consultas resultan ser de poca ayuda ante el problema de encontrar resultados precisos, además se vio la necesidad de buscar alguna alternativa que permita estructurar el contenido con el fin de aumentar la riqueza en las consultas del usuario y así aumentar la precisión de los resultados considerando lo anterior se concluyó que el buscador semántico a construirse debía contar con las siguientes funcionalidades:

3.3.1. Administrador

Migración de Oas:

- Recuperar Contenido de OA
- Se extrae el contenido de los OA, esto dependerá del tipo de OA, es así que para los objetos de basados en texto (word, pdf, html, excel, presentaciones, etc.) se extrae todos sus caracteres. En cambio OA multimedia (audio, video, animaciones, gráficos, etc) se toma el metadato descripción de estos OA para su estructuración.
- Extraer entidades (Estructurar)
- El contenido recuperado de los OA pasa por un proceso de extracción de información donde se reconocen entidades relevantes de cada recurso como son: personas, organizaciones, fechas, posiciones laborales, regiones, magnitudes físicas y relaciones entre entidades.
- Obtener Metadatos de OA
- Junto a cada OA se describen ciertos metadatos en el repositorio como autor, fecha, tipo de OA, titulo y área académica de la Universidad a la que pertenece, estas también son considerados entidades.
- Guardar Información en KIM
- Todas las entidades reconocidas del contenido de los OA al igual que los metadatos asociados a cada recurso se almacenan en la herramienta KIM.

3.3.2. Usuarios (Profesor, estudiante, usuario externo)

Búsqueda Simple

La consulta ingresada al sistema pasa por los siguientes procesos:

- **PLN de la Consulta:** La consulta es analizada con el fin de determinar si su estructura es correcta, este procesamiento consta de los siguientes análisis:
  - **Análisis Ortográfico**
  - Analiza si la consulta no contiene caracteres especiales, caracteres no reconocibles por el buscador y errores ortográficos.

- **Análisis Morfológico**
- Determina la categoría a la que pertenece cada palabra en la consulta, así como también la estructura de la oración.

- **Análisis Sintáctico**
- Etiqueta cada uno de los componentes sintácticos que aparecen en la consulta y analiza la combinación de las palabras para formar construcciones gramaticalmente correctas.
-
- **Análisis Pragmático**
- Transforma la consulta en términos en que el sistema pueda entender (consulta SeRQL).

Búsqueda Avanzada

- **Búsqueda por entidades**

El usuario escoge las entidades por las que desea realizar la búsqueda e ingresa el valor en cada parámetro.

A continuación se presenta el esquema con los principales casos de uso de los usuarios definidos anteriormente:

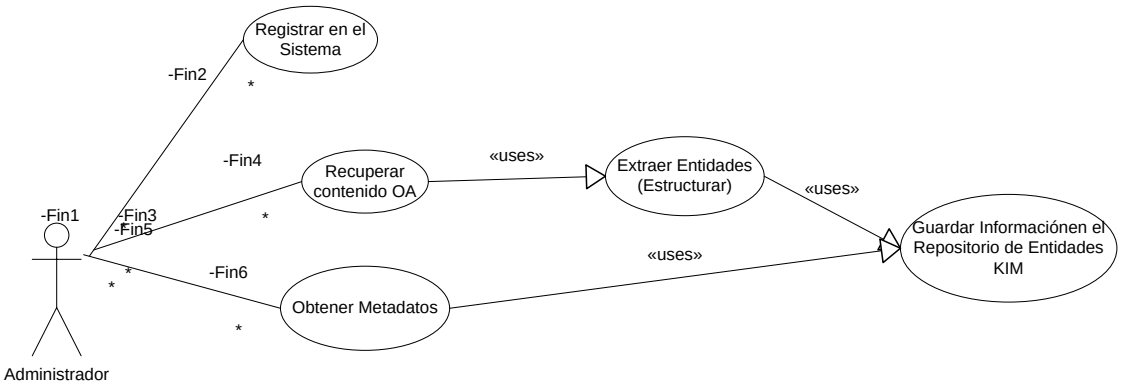


Figura 3.2: Casos de Uso para el Administrador

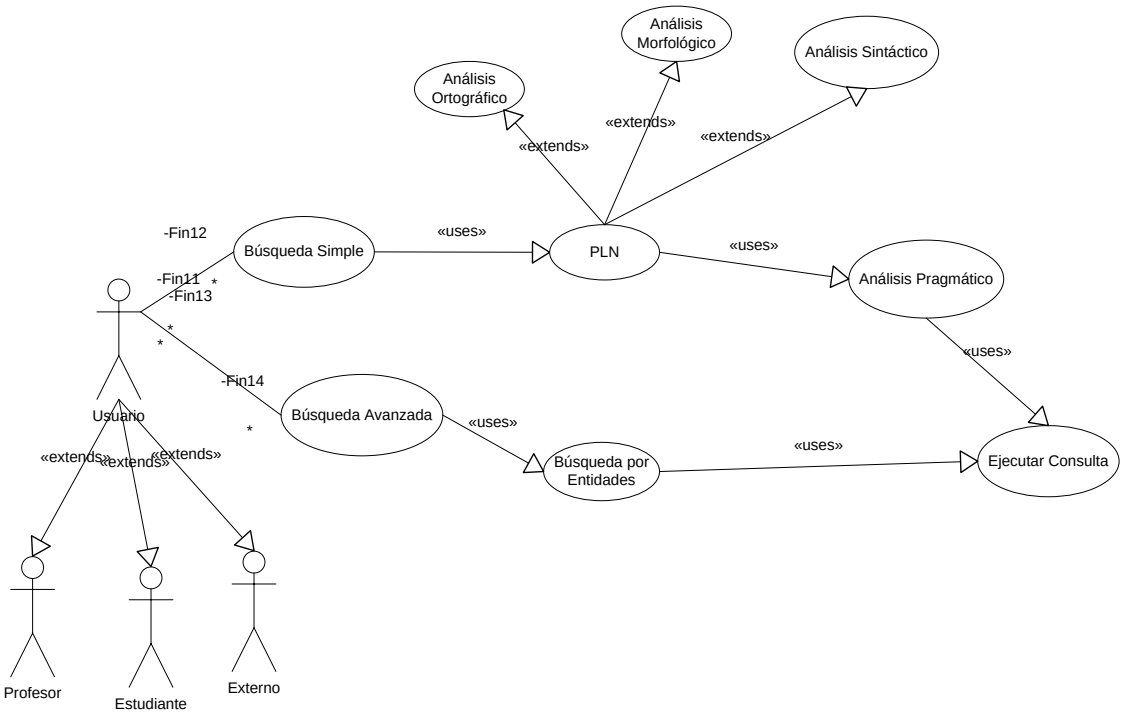


Figura 3.3: Casos de Uso para el Usuario

3.4. Vista de Implementación

El buscador Buspace trabaja conjuntamente con el repositorio de Dspace del cual se extrae el contenido y los metadatos. También utiliza la base de conocimiento de KIM para almacenar las entidades reconocidas en el proceso de migración y realizar las consultas semánticas del usuario, gráficamente la arquitectura del sistema se expone a continuación:

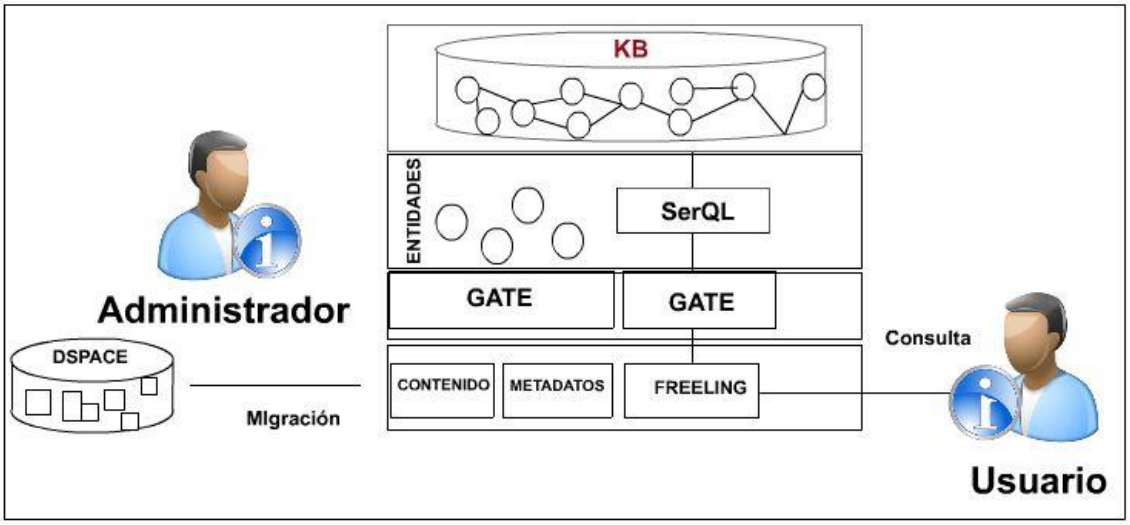


Figura 3.4: Arquitectura general del Buspace

3.5. Vista de Datos

Como paso previo al desarrollo de los componentes, se procedió a modificar el esquema por defecto de base de datos del repositorio Dspace[48], se hizo necesario agregar la tabla *"IMPORT"* para almacenar los OA que han sido migrados a la base de conocimiento de Buspace, también se creó una vista (*ITEMSBYTIPO_ARCHIVO*) la cual contiene información necesaria para el proceso de migración de OA y así evitar sobrecarga al servidor, el modelo de las tablas antes mencionadas se describe en la **Figura 3.5** conjuntamente con las tablas de donde se obtienen los datos.

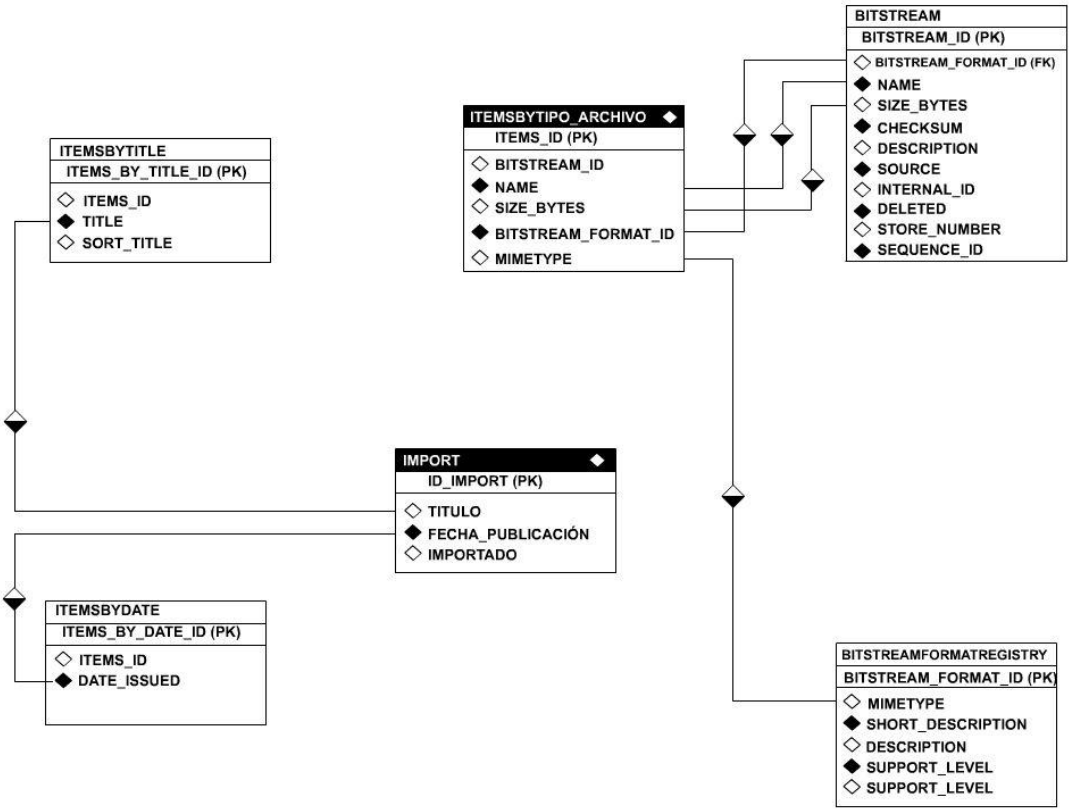


Figura 3.5: Modificaciones al esquema de Base de Datos de Dspace

3.6. Descripción de la ontología

Kim es un conjunto de ontologías algunas propias y otras de terceros como la ontología de propósito general PROTON49 desarrollada por el proyecto SEKT50 (detalles de instalación en), en ella se distinguen tres conceptos principales los cuales son: "Entity", "EntitySource", "LexicalResource" siendo "Entity" el concepto más importante y del cual se deprenden "Abstract", "Happening" y "Object concepts", la ontología como tal contiene 250 conceptos, 40

[48] http://www.dspacedev2.org/1_5_1Documentation/image/db-schema.gif
[49] http://proton.semanticweb.org/
[50] http://www.sekt-project.com/

propiedades y alrededor de 77500 Entidades por defecto instanciadas y más de 110000 alias. En la siguiente figura se presenta la estructura de la ontología:

Figura 3.6: Ontología Kim

La tabla a continuación muestra el número de conceptos que posee la ontología de Kim:

Tabla 3.1: Resumen de números de conceptos instanciados en Kim

Concepto	Número
Entidades	77561
Alias de Entidades	110308
Lugares	49348
Ciudades	4720
Compañías	7906
Compañías Publicas	515
Posiciones de Personas	55
Organizaciones	8365
Total	258778

Una vez descritos los componentes y la estructura de la ontología, se pasa a detallar las diferentes API's que brinda Kim para ser consumidas como servicios web o utilizando RMI. Los detalles de instalación y configuración de Kim se pueden revisar en el Anexo 5. Kim provee la infraestructura necesaria para el manejo de documentos y consultas sobre la base de conocimiento estas funcionalidades se basan en API's que son las siguientes:

SemanticAnnotationAPI service

Invoca el proceso de anotación semántica, es la rutina estándar de población de la ontología, en este servicio se distinguen dos métodos principales:

- **Execute:** Acepta como entrada un objeto de tipo de *KIMDocument* (URL) con el texto a estructurar y devuelve el documentos anotado semánticamente.
- **executeText:** Acepta entradas en texto plano y devuelve un arreglo de tipo *KIMAnnotation* con las anotaciones creadas por el procedimiento.

DocumentRepositoryAPI service

Se encarga del almacenamiento y recuperación de los documentos anotados en el repositorio, consta de los siguientes métodos:

- **addDocument:** Método para almacenar los documentos en el repositorio, los documentos deber ser anotados antes de ser almacenados.
- **getConstants:** Permite acceder a las contantes predefinidas de los Documentos almacenados.
- **getDocuments:** Permite recuperar el contenido estructurado de múltiples documentos almacenados repositorio, utiliza consultas estructuradas para filtrar los documentos.
- **getDocumentIds:** Retorna el identificador único con que ha sido almacenado un determinado documento, este identificador es utilizado para acceder al contenido del mismo.
- **getDocumentCount:** Retorna el número de documento recuperados por la consulta, acepta objetos de tipo query como entrada;
- **getDocumentFeatures:** Permite acceder a las características asociadas a cada documentos, como son titulo, fecha, autores, url, contenido, tipo, palabras claves. Acepta el identificador del documento y devuelve un arreglo de características.
- **loadDocument:** Permite abrir el contenido estructurado de un documento en particular denotado por su identificador.
- **getEntities:** Obtiene las anotaciones del documento dejando aparte el contenido, se pueden acceder a todas las anotaciones del documento o simplemente aquellas que satisfacen una consulta.
- **getEntitiesCount:** Devuelve el número total de entidades del documento o las entidades recuperadas para cierta consulta.

SemanticRepositoryAPI service

Se encarga de extraer información semántica como RDF, en si se encarga de ejecutar las consultas SeRQL del usuario.

- **evaluateSelectSeRQL:** Acepta como entrada un consulta en lenguaje SeRQL en texto plano (String) y devuelve los resultadso como una tabla utilizando el objeto *SemanticQueryResult* para mostrar los resultados.
- **evaluateSelectSERQLCount:** Obtiene el número de documentos que satisfacen la sonsulta SeRQL.
- **getInstanceCount:** Toma una clase URI como entrada y devuelve el número de instancias del repositorio que satisfacen el requerimiento.
- **importData:** Permite importar datos personalizados de los usuarios en formato RDF directamente al servidor KIM.
- **addStatement:** Añade una nueva declaración (statement) RDF al repositorio.
- **removeStatementsBatch:** Remueve sentencias RDF anadias por el usuario y las que viene por defecto con Kim, en esta operación hay que tener cuidado de no alterar el resto de la estructura de kim.

EntityAPI service

Provee una simple abstracción de SemanticRepositoryAPI, permite la carga de toda la información semántica disponible para una determinada entidad con una sola llamada.

- **getEntityDescription:** Permite obtener la descripción ontológica de una entidad, acepta la URI del la entidad como por ejemplo http://proton.semanticweb.org/2006/05/protonu#Company. Y retorna la descripción en un objeto EntityDescription como resultado.
- addEntityDescription : Permite añadir una nueva entidad al repositorio semántico.
- **serializeEntities:** Es la encargada de la transformación de una lista de descripciones de entidades a cadenas de texto (String)
- **DeserializeEntities:** Realiza el proceso contrario del método anterior transformando una cadena de texto a una definición estructurada de entidades.

QueryAPI service

A través de este servicio se puede realizar consultas complejas, estas pueden ser formales hacia la semántica de la ontología, QueryAPIservice combina la funcionalidad de SemanticRepositoryAPI y DocumentRepositoryAPI pudiéndose realizar consultas semánticas combinadas con consultas sintácticas y filtrando los documentos por entidades encontradas, esta proporciona los siguientes métodos:

- **getEntities:** Permite obtener las entidades asociadas al resultado de una consulta semántica, las restricciones de la consulta se describen a través del objeto SemanticQuery, el resultado se devuelve como un objeto SemanticQueryResult.
- **getDocuments:** Permite que en la recuperación de un consulta semántica de entidades se recuperen los documentos correspondientes que poseen las entidades especificadas. Para este método tanto en el conjunto de entidades de resultados y la parte del documento del objeto devuelto - están combinadas.

3.7. Migración

Luego de revisar los métodos para el desarrollo con Kim, se procedió a realizar la migración de los OA del Dspace a la base de conocimiento, cada documento en Kim es almacenado conjuntamente con unas características llamadas *features* que permiten identificar al documento, como requisito previo para almacenar el documento, éste ya debe estar anotado (pues una vez guardado un documento, su contenido no puede ser anotado ni modificado posteriormente).

Las *features* de los documentos Kim fueron tomadas de los metadatos de Dspace, en la **Tabla 3.2** se presenta el desglose completo de los datos involucrados. El metadato "text_value" es un caso particular ya que este puede utilizarse de dos maneras dependiendo el caso:

- Si el OA que se quiere migrar es multimedia ya sea video, animación, podcast, imagen, el metadato "text value" debe ser configurado como contenido del documento Kim, y se procederá a anotarlo semánticamente
- Si el documento es basado en texto, como archivos pdf, word, excel, texto plano y url's, se lee su contenido y se configura como tal en un documento Kim; la migración de este tipo de OA requiere más tiempo que los archivos multimedia.

Tabla 3.2: Descripción de Metadatos y Features

Dspace		KIM		Valor por defecto
Metadato	**Tabla que lo contiene**	**Feature**	**Tipo**	
title	itemsbytitle	<TITLE>	object	
date_issued	itemsbydate	<TIMESTAMP>	Long	La fecha del sistema
author	itemsbyauthor	<AUTHORS>	object	
handle	handle	<URL>	String	
name	community	<SUBJECT>	object	
name	collection	<SUBJECT>	object	
text_value (*)	metadatavalue	<KEYENTITIES>	object	
		<KEYPHRASES>	object	
		<SOURCE>	object	"Repositorio de Material Educativo -- Dspace UTPL"
		<LANGUAGE>	object	"Español"

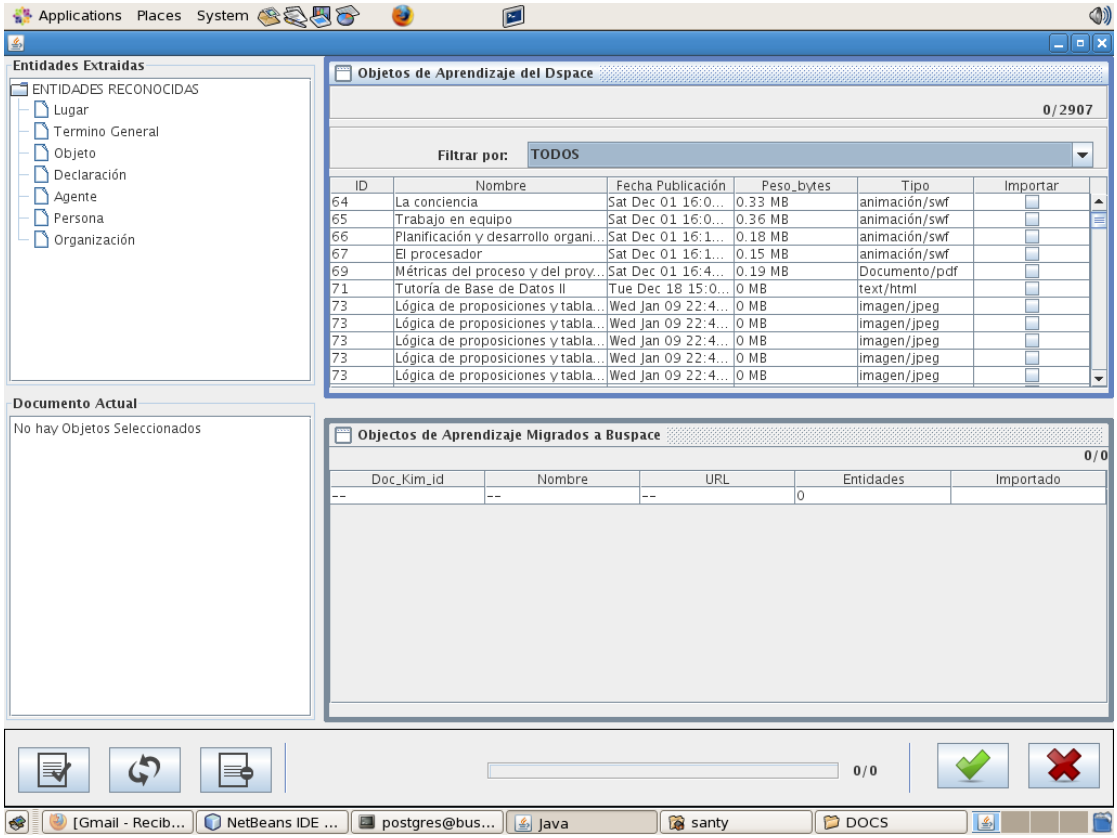


Figura 3.7: Módulo de Migración de OA

En la **Figura 3.8** se presenta un documento migrado a Buspace, el mismo que se anotado semánticamente, se puede observar que se han reconocido entidades como lugares (Catacocha) y términos generales.

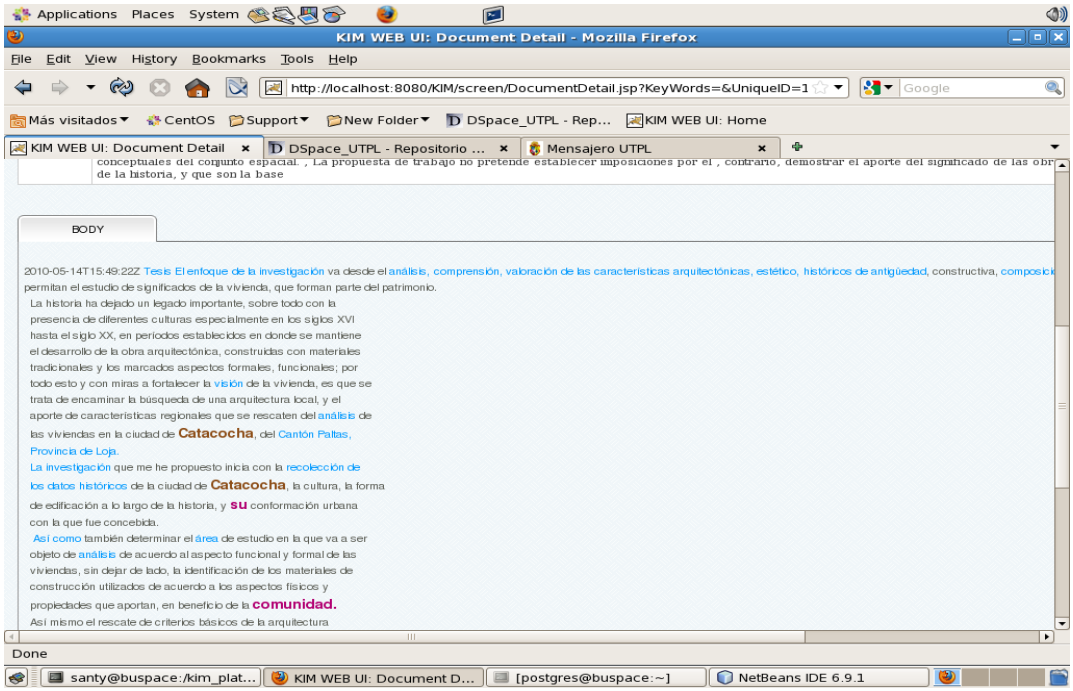


Figura 3.8: Documento Anotado en Kim

3.8. Buscador

Las consultas se realizan en su mayoría a través de conceptos, sin embargo también se combinan con restricciones tanto en las entidades como en las features de los documentos con el fin de encontrar los resultados más precisos. En la Figura 3.9 se muestra un ejemplo de la construcción de una consulta semántica la cual busca organizaciones ubicadas en Ecuador y que lleven por nombre "UTPL" a continuación se hace una restricción para que solo se recuperen aquellos documentos que por título contengan la palabra "Alteridad".

```
// ------------------------------------------------------------------------------
// SeRQL 2 Semantic Query
// ------------------------------------------------------------------------------
String serql2 = "";
    serql2 += "select distinct ORG,ORGMainLabel,LOC,LOCMainLabel ";
    serql2 += "FROM ";
    serql2 += "     {ORG} <" +  WKBConstants.KIMO_LOCATED_IN + "> {LOC}, ";
    serql2 += "     {ORG} <" + RDF_TYPE + "> { <" +  WKBConstants.CLASS_ORGANIZATION + "> }  , ";
    serql2 += "     {LOC} <" + RDF_TYPE + "> { <" +  WKBConstants.CLASS_LOCATION + "> } , ";
    serql2 += "     {ORG} <" + RDFS_LABEL + "> { ORGMainLabel } , ";
    serql2 += "     {LOC} <" + RDFS_LABEL + "> { LOCMainLabel } , ";
    serql2 += "     {LOC}  <" +  WKBConstants.KIMO_HAS_ALIAS + "> {} <" + RDFS_LABEL + ">  {LOCLa
    serql2 += "where  ORGLabel0 like \"*UTPL*\" ignore case AND LOCLabel0 like \"*Ecuador*\"";

    DocumentQuery query = new DocumentQuery();
   query.setKeywordRestriction("title:Alteridad");
    }
```

Figura 3.9: Combinación de consultas semánticas y restricciones de features

El proceso de consulta comienza procesando el requerimiento del usuario, los análisis que se realizan son ortográfico y sintáctico, para el análisis semántico y pragmático se utiliza el anotador semántico de kim para descubrir conceptos relevantes en la consulta y se procede a generar las respectivas consultas semánticas con los conceptos involucrados, como es lógico los resultados se presenta ordenados por relevancia.

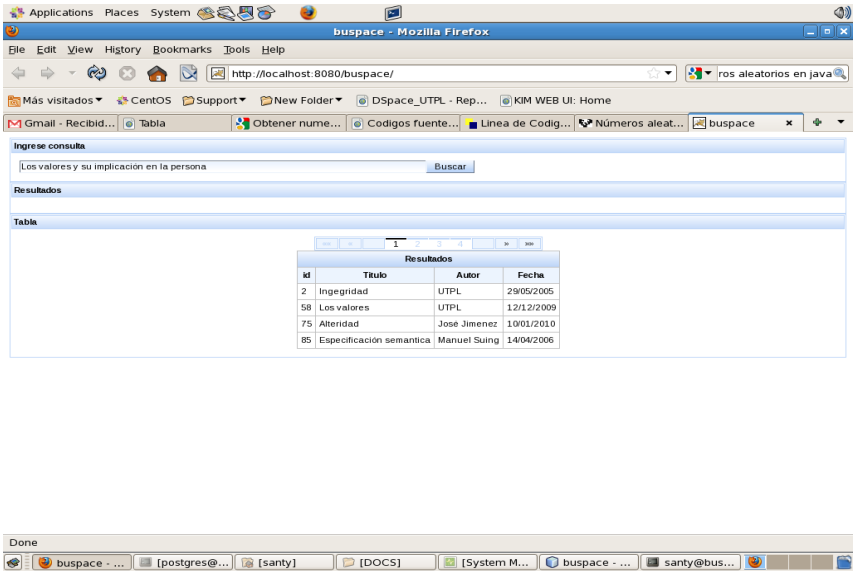


Figura 3.10: Buscador Buspace

Capítulo 4 Pruebas de Implementación

4.1. Introducción

En el presente capítulo se inicia definiendo, las métricas utilizadas para probar el buscador semántico Buspace, estas métricas corresponde a la Precisión y Exhaustividad (Recall) que son comúnmente utilizadas para evaluar buscadores en internet.

En la sección 4.3, se presenta el esquema bajo el cual se procedió a realizar las pruebas, en este escenario participa Buspace y el buscador sintáctico del Dspace, se habla brevemente sobre las muestras tomadas para la evaluación.

Finalmente, en la sección 4.4, se realiza un análisis de los resultados obtenidos y una interpretación de los datos.

4.1.1. Propósito

Se requiere verificar que el sistema funcione correctamente en escenarios reales de producción, arrojando datos precisos y en lo posible reaccionar y recuperarse lo mejor posible ante los errores.

El presente plan de pruebas se busca principalmente verificar la funcionalidad y usabilidad de Buspace, además de los siguientes objetivos específicos:

- Identificar errores del sistema y de sus diferentes componentes.
- Identificar posibles mejoras a futuro
- Constatar la integración satisfactoria de todas las herramientas seleccionadas en capítulos anteriores.

4.1.2. Alcance

Las pruebas intentan evaluar el componente de búsqueda del Buspace, el cual incluye las siguientes funcionalidades:

- Módulo de PLN
- Módulo de Reconocimiento de entidades
- Módulo de transformación de Consultas
- Módulo de RI

Las pruebas que se han realizado son de usabilidad y funcionalidad, además pruebas que considerarán aspectos de precisión, retentiva, relevancia, rapidez de los resultados arrojados por el buscador.

4.1.3. Audiencia

Las personas involucradas en el presente plan de pruebas se detallan a continuación:

- Desarrollador (1)
- Usuarios
    - Administrador (1)
    - Docentes (1)
    - Estudiantes (2)

TOTAL = 5

4.1.4. Referencias

La información útil para el presente documentos son los Casos de uso y funcionalidades de descritos en la Figura 3.2 y los requerimientos del sistema. Los casos de uso a probar son:

- Busqueda Simple
    - PLN (análisis Ortográfico, Morfológico, SIntáctico)
    - Análisis Pragmático
    - Ejecutar consulta

4.2. Identificación del sistema a probar

EL Buspace es un sistema que consta de dos funciones principales; migración que solo puede ser accedida por el administrador del sistema y el módulo de búsqueda la cual está disponible para cualquier usuario, precisamente este módulo es el objeto de estudio en el presente Plan de pruebas por cuanto interesa saber la efectividad del mismo comparado con el buscador por defecto del Dspace, este componente posee las siguientes funcionalidades:

- **Módulo de PLN:** El encargado de procesar la consulta del usuario, se encarga de verificar que la consulta se encuentre correctamente escrita ortográfica y sintácticamente, con el objetivo de que le modulo de reconocimiento de entidades no tenga problemas en reconocer los conceptos más relevantes de la consulta.
- **Módulo de Reconocimiento de entidades:** Encargado de reconocer las principales entidades presentes en las consultas del usuario, este módulo se encarga del procesamiento semántico y pragmático del requerimiento del usuario.
- **Módulo de transformación de Consultas:** Este módulo se encarga de transformar el requerimiento del usuario procesado por los módulos anteriormente descritos en consultas semánticas, para lo cual utiliza el Api de Kim para realizar consultas.
- **Modulo de RI:** Este módulo es el encarga de recuperar por orden de relevancia los OA que cumplen con el requerimiento del usuario, así como también establecer las principales características (features: Titulo, Autor, Fecha, etc.) y la URL del recurso.

4.3. Estrategias y Ejecución de Pruebas

4.3.1. Pruebas de Utilización de Buscador

4.3.1.1. Métricas de Evaluación

La cantidad inmensa de información en la red ha dado lugar a que los sistemas de RI cobren cada vez más importancia, al momento de evaluar esta clase de sistemas lo que principalmente se evalúa es: Qué tan pertinentes son los resultados? (Precisión) y Qué variedad de resultados despliega el sistema? (Exhaustividad o Recall). Aunque existen otras métricas de evaluación como son: la amigabilidad de las interfaces, formatos de presentación, conexiones con otros documentos y tiempo de respuesta, el presente trabajo se enfocará principalmente en aquellas métricas que permiten tener una noción de la calidad de los resultados desplegados, esta métricas como se ya se mencionó anteriormente se tratan de Precisión y Exhaustividad o Recall.

4.3.1.2. Precisión

Este concepto fue definido por primera vez por Keen (Keen, 1955) el cual lo define como un factor de pertinencia, otros autores se refieren a este concepto como "ratio de aceptación", entre todas las definiciones atribuidas a la precisión. (Salton, 1983) la define como "la proporción de material recuperado realmente relevante, del total de los documentos recuperados", esta definición es ampliamente utilizada dentro de la RI. (Frakes, 1992) Complementa que el valor de la relevancia está entre cero y uno, siendo la "recuperación perfecta" aquella que tenga un valor de uno, es decir todos los documentos relevantes han sido recuperados. La fórmula utilizada por Salton y empleada en el siguiente trabajo es la siguiente:

$$Precisión = \frac{Documentos\ relvantes\ recuperados}{Documentos\ recuperados}$$

Sin embargo, la medida de precisión está sujeta a dos problemas: el ruido documental o falsos positivos que no son más que aquellos documentos que se recuperan como relevantes cuando en realidad no lo son, y el silencio informativo que en cambio son documentos relevantes que no han podido ser recuperados por diferentes circunstancias.

4.3.1.3. Exhaustividad, Retentiva o Recall

La exhaustividad también es una de las métricas más utilizadas en RI, claro después de la Precisión, este concepto busca la proporción del material relevante recuperado del total de los documentos que son relevantes en la muestra analizada sin importar si éstos han sido recuperados por el sistema. Nuevamente se recurre a (Salton, 1983) para describir la forma de cálculo de este concepto:

$$Exhaustividad = \frac{Documentos\ relevantes\ recuperados}{Documentos\ relevantes}$$

Si el valor del cálculo de la retentiva es igual a uno, se tiene una exhaustividad máxima ya que se ha encontrado todo lo relevante que había en la muestra, por tanto el ruido y silencio informativo serán nulos y se ha procedido a obtener una "recuperación perfecta".

4.3.1.4. Esquema de pruebas

Las muestras utilizadas para realizar las pruebas son los OA migrados del DSpace a la base de conocimiento de KIM, los cuales son 211 y se encuentran anotados semánticamente, sobre esta muestra solo trabajará Buspace, en cambio para analizar el buscador sintáctico por defecto de DSpace se utilizará únicamente los OA de la comunidad que posee 2636 recursos, se ha escogido esta categoría debido a que la mayor cantidad de recursos publicados se encuentran bajo la misma, además los OA migrados al sistema Buspace han sido tomados de esta categoría.

Las temáticas de las dos muestras predominan los recursos que pertenecen al área administrativa con temas sobre negocio y empresas principalmente, luego se encuentran recursos pertenecientes al área técnica que hablan de Física e Informática y por último están los recursos de áreas como biológica y socio humanísticas en menor cantidad.

En la **Tabla 4.1** se presenta un resumen de la cantidad de OA que aún faltan por migrar agrupados por comunidades, en total son un conjunto de 2425 recursos. La migración de todos los recursos faltantes requiere de una revisión del Repositorio en miras de depurar aquellos OA obsoletos y eliminar enlaces rotos, todo esto con el fin de contar solo con recursos funcionales en sistema Buspace.

Tabla 4.1: Resumen por comunidades de OA que aún no han sido migrados a Buspace

Comunidad	Número OA
UTPL	2296
Artes	1
Arquitectura	39
Literaturas española y portuguesa	5
Comercio, comunicación y transporte	8
Computación, conocimiento y sistemas	7
Estadística general	8
Economía	5
Derecho	43
Administración pública y ciencia militar	4
Psicología	1
Ciencias de la vida, biología	1
Ingeniería Química	7
TOTAL	2425

Los objetivos que el presente esquema de pruebas persigue son los siguientes:

- Identificar errores del sistema y de sus diferentes componentes.
- Identificar posibles mejoras a futuro

- Constatar la integración satisfactoria de todas las herramientas seleccionadas en capítulos anteriores.

El alcance del presente esquema de pruebas solo comprende el módulo de búsqueda el cual integra las herramientas: Kim, GATE, Freeling.

Las pruebas realizadas se hicieron utilizando los siguientes recursos:

CPU	Intel(R) Pentium(R) 4 CPU 3.00GHz
RAM	512 MB
Sistema Operativo	Centos 5.3
Servidor	Tomcat 6.0.26
JVM	1.6
Disco	180 GB

El detalle de cada prueba se muestra a continuación:

4.3.1.5. Prueba 1

Esta prueba consiste en evaluar que tan preciso puede ser el buscador cuando se ingresa una consulta donde existen gran cantidad de recursos que contienen los términos empleados en la misma, en este caso los términos a los que se hace referencia son: *"Proyectos"* y *"Loja"* . Como se mencionó al inicio del presente Capítulo los recursos de las muestras con las que se trabaja tienen temáticas en su mayoría de empresas y negocios, además el término "Loja" se repite en gran cantidad de recursos.

Tabla 4.2: Resultados para Prueba 1

Consulta 1:	Proyectos desarrollados en la provincia de Loja		
		Buspace	**Buscador DSpace**
Doc. Relevantes en la muestra		**48**	**234**
Recuperados	**Relevantes encontrados**	49	156
	No relevantes	1	418
	Total	50	574
No recuperados		0	0
	Precisión	0,98	0,27
	Recall	1,02	0,67

Los resultados obtenidos muestran que Buspace tiene una precisión del 98 %, y una retentiva superior al 100 %, es decir recupera documentos relevantes; sin embargo, todavía existen problemas con un documento el cual se lo puede considerar como falso positivo (no relevante), esto puede ser a que los metadatos que describen estos OA no reflejan lo que realmente trata el recurso, es decir contienen errores. Además Buspace es más efectivo por cuanto reconoce a "Loja" como un lugar o zona geográfica y recupera aquellos documentos donde se mencionan a

esta entidad, sin embargo; el documento no relevante recuperado por Buspace se debe a que la entidad en cuestión es mencionada como parte de la región Sierra mas no por que se esté desarrollando un proyecto.

4.3.1.6. Prueba 2

En esta prueba se utilizó una consulta específica donde el número de documentos que la satisfacen corresponden a un número mínimo y poco representativo en las dos muestras, los OA relevantes en cada muestra son 1 y 46 respectivamente, además se establece una condición particular en esta consulta la cual es buscar la mención de la persona "Luis Miguel Romero" con el cargo de "rector de la UTPL" en el contenido del recurso.

Tabla 4.3: Resultados para la Prueba 2

Consulta 2:	Recursos en que se mencione a Luis Miguel Romero rector de la UTPL		
		Buspace	**Buscador DSpace**
Doc. Relevantes en la muestra		**1**	**46**
Recuperados	**Relevantes encontrados**	1	6
	No relevantes	0	1478
	Total	1	1479
No recuperados		0	0
	Precisión	1	0,041
	Recall	1	0,13

La poca precisión del buscador sintáctico del DSpace se debe a que recupera aquellos documentos que contiene las palabras "Luis" o "Miguel" o "Romero" y no lo trata como una sola entidad, en cambio Buspace considera a "Luis Miguel Romero" como una entidad de tipo "persona" y la condición de "rector de la UTPL" como una entidad "posición de trabajo" y "UTPL" como organización lo cual hace más especifico la búsqueda con Buspace.

4.3.1.7. Prueba 3

Consistió en buscar aquellos documentos pertenecientes a un autor en particular, además los documentos que satisfacen el requerimiento son poco representativos comparado con las muestras utilizadas lo que aumentan la complejidad de la misma.

Tabla 4.4: Resultados para la Prueba 3

Consulta 3:	Documentos escritos por Manuel Salazar		
		Buspace	**Buscador DSpace**
Doc. Relevantes en la muestra		**1**	**4**
Recuperados	**Relevantes encontrados**	1	4
	No relevantes	0	679
	Total	1	683
No recuperados		0	0
	Precisión	1	0,059
	Recall	1	1

Los resultados demuestran que el buscador sintáctico de DSpace recuperó todos los OA relevantes (retentiva), pero de igual forma recuperó una cantidad importante de documentos basura o con ruido, principalmente recupero aquellos documentos en los que aparecía las palabras "Manuel" o "Salazar" indistintamente en cambio Buspace reconoció las mismas palabras como una sola entidad de tipo persona y procedió a buscar aquellos documentos donde se menciona.

4.3.1.8. Análisis de Resultados

Los resultados demuestran que Buspace posee un índice mayor de OA's relevantes recuperados sobre los que recupera el buscador sintáctico de DSpace (ver Tabla 4.5). Buspace reconoció el 99 % de los documentos relevantes y el otro buscador solo el 37 %.

Tabla 4.5: Tabla acumulada de los resultados de Precisión

PRECISIÓN		
	Buspace	**Buscador DSpace**
Consulta 1	0,98	0,27
Consulta 2	1	0,041
Consulta 3	1	0,059
TOTAL	0,99	0,37

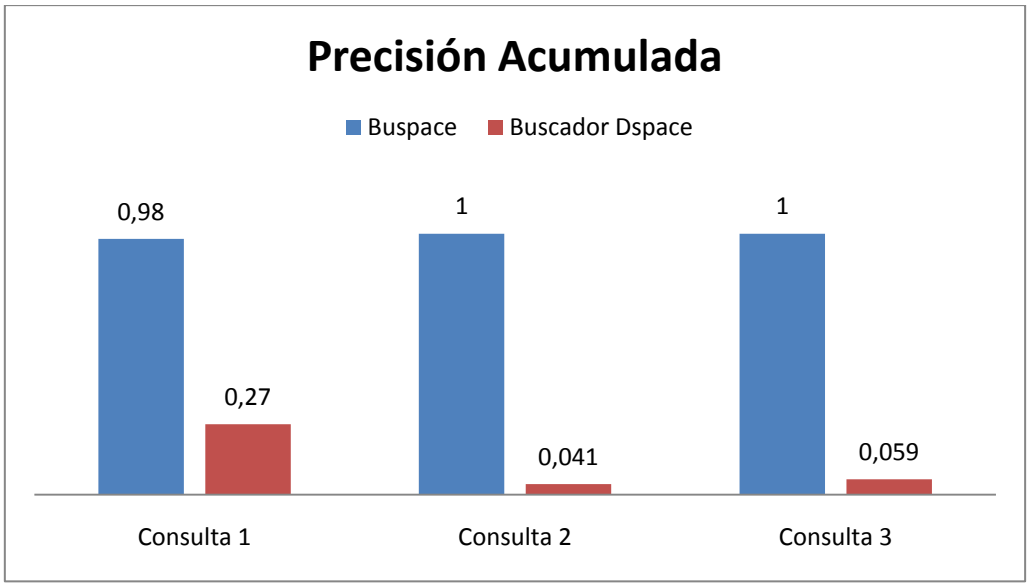


Figura 4.1: Precisión Acumulada

En la gráfica se puede observar que los resultados del buscador sintáctico no superan el 40 % en todas las mediciones hechas, por otra parte Buspace recupera el 99% de documentos relevantes, esto se debe a la búsqueda por entidades y no a la correspondencia de palabras.

En lo que tiene que ver a la retentiva (precisión - recall) la diferencia entre los dos buscadores analizados se reduce a un 8%, siendo el buscador del DSpace con mas documentos con ruido (no relevantes) recuperados.

Tabla 4.6: Tabla acumulada para la métrica de Retentiva

RECALL - RETENTIVA - EXHAUSTIVIDAD		
	Buspace	**Buscador DSpace**
Consulta 1	1,02	0,67
Consulta 2	1	0,13
Consulta 3	1	1
TOTAL	1,01	0,6

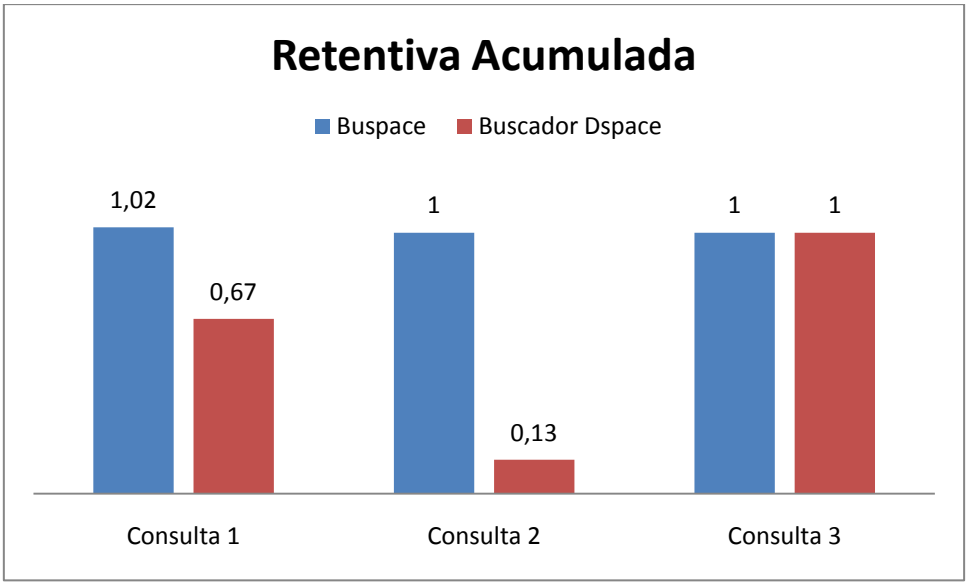


Figura 4.2: Retentiva Acumulada

4.4. Pruebas de Funcionalidad

El objetivo de las pruebas de funcionalidad es verificar la apropiada aceptación de los datos, el procesamiento de las consultas, el reconocimiento de entidades y la recuperación de los recursos que satisfacen la consulta. Este tipo de pruebas se realizan a través de la interfaz grafica de la aplicación. Los casos de pruebas se detallan a continuación:

4.4.1. PLN

Objetivo:				
Probar el procesamiento de la consulta del usuario antes de el proceso de búsqueda				
No.	**Caso de Prueba**	**Válido / Inválido**	**Resultado Esperado**	**Observaciones de la Prueba**
1.	El Usuario ingresa su requerimiento de búsqueda escrito correctamente y presiona el botón "*Buscar*"	**Válido**	**Verificar** que el sistema despliega la sección de resultados, con los OA encontrados	Ingresar consulta perfectamente escritas para constatar que el buscador está realizando efectivamente el análisis correcto del requerimiento del usuario.
2.	El Usuario ingresa su requerimiento de búsqueda con errores de escritura y presiona el botón "*Buscar*"	**Inválido**	**Verificar** que el sistema lanza una excepción/mensaje señalando la palabra que está mal escrita.	Ingresar consulta escrita incorrectamente para constatar que el buscador está realizando efectivamente el análisis correcto del requerimiento del usuario.
3.	El Usuario no ingresa consulta alguna y presiona el botón	**Inválido**	**Verificar** que el sistema despliegue un mensaje	Ninguna.

	"*Buscar*"		indicando que debe ingresar su requerimiento de consulta para proceder al procesarla.	
4.	El Usuario ingresa una consulta demasiado extensa (superior a 100 caracteres).	**Inválido**	**Verificar** que el sistema despliegue un mensaje indicando que el número de caracteres en el consulta supera el límite permitido.	Ninguna.

4.4.2. Reconocimiento de Entidades

Objetivo:				
Probar el módulo reconoce las entidades más relevantes de la consulta del usuario				
No.	**Caso de Prueba**	**Válido / Inválido**	**Resultado Esperado**	**Observaciones de la Prueba**
5.	El Usuario ingresa su requerimiento de búsqueda escrito correctamente y presiona el botón "*Buscar*"	**Válido**	**Verificar** que el sistema despliega las entidades reconocidas en el requerimiento del usuario	• Dependiendo de la consulta el sistema puede no reconocer ninguna entidad ya sea xq el sistema no las reconoce o simplemente no existen en la consulta del usuario. • También existe la posibilidad que algunas entidades sean etiquetadas erróneamente, por ejemplo etiquetar a una entidad como persona cuando realmente es una organización o cuando simplemente no significa nada.
6.	El Usuario ingresa su requerimiento de búsqueda con una entidad de tipo Persona que conste previamente en el sistema	**Válido**	**Verificar** que el sistema efectivamente reconoce dicha entidad como persona	El sistema puede etiquetar erróneamente la entidad ya que trabaja estadísticamente.

4.4.3. Transformación de Consultas

Objetivo:				
Probar que las consultas se están transformando efectivamente según las entidades reconocidas				
No.	**Caso de Prueba**	**Válido / Inválido**	**Resultado Esperado**	**Observaciones de la Prueba**
7.	El Usuario ingresa su requerimiento de búsqueda escrito correctamente y con una entidad de tipo organización que conste previamente en el sistema como tal y presiona el botón "*Buscar*"	**Válido**	**Verificar** que el sistema despliega la sección de resultados, con los OA encontrados en los cuales se encuentren al entidad de tipo organización	Ninguna.

4.4.4. Módulo de RI

Objetivo:				
Probar que el modulo de RI recupere información relacionada con la consulta planteada por el usuario				
No.	**Caso de Prueba**	**Válido / Inválido**	**Resultado Esperado**	**Observaciones de la Prueba**
8.	El Usuario ingresa su requerimiento de búsqueda escrito correctamente y con una entidades que se encuentren reconocidas en el sistema previamente y presiona el botón "*Buscar*"	**Válido**	**Verificar** que el sistema despliega la sección de resultados, con los OA encontrados en los cuales cumplan con la consulta del usuario	Ninguna.
9.	El Usuario ingresa su requerimiento de búsqueda escrito correctamente y con una entidades que no existen en la en el sistema previamente y presiona el botón "*Buscar*"	**Inválido**	**Verificar** que el sistema muestre 0 registros encontrados	Ninguna.

Con el desarrollo de las de funcionalidad se verifico que cada componente trabaje de manera correcta, a partir de las mismas se obtuvo los siguientes resultados:

Tabla 4.7: Resultados para las pruebas de Funcionalidad

Funcionalidad	# Errores	%
PLN	1	12,5
Reconocimiento de Entidades	3	37,5
Transformación de Consultas	2	25
Módulo de RI	2	25
TOTAL	8	100

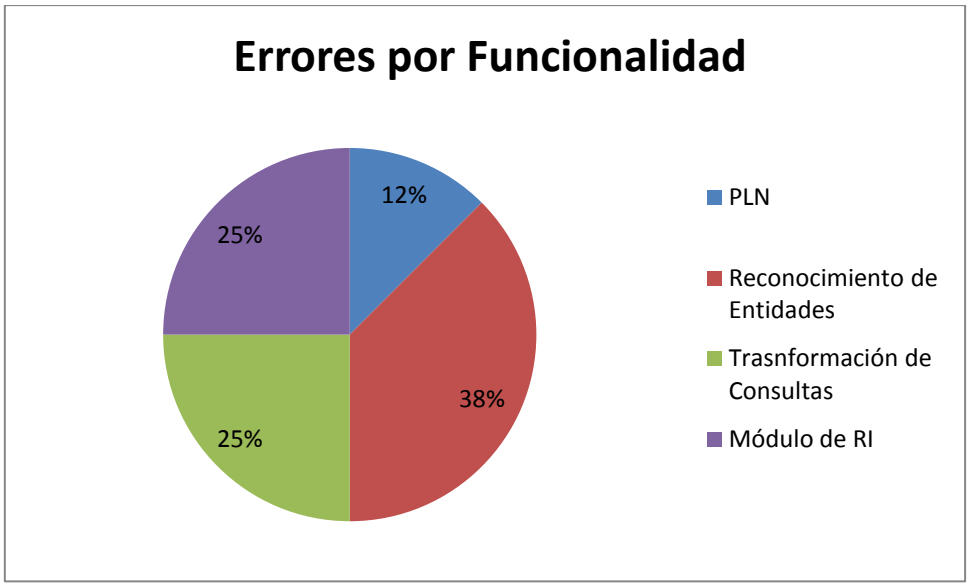


Figura 4.3: Grafica de Errores por Funcionalidad

Los resultados demuestran que la funcionalidad que posee el mejor desenvolvimiento es el módulo de PLN el cual es el encargado de verificar si la consulta ingresada se encuentra escrita correctamente, por otro lado la funcionalidad con mayores errores es la de reconocimiento de entidades la cual corresponde 38%, esto se debe a gran parte a la manera en que los usuarios ingresan la consulta, es decir el nivel de especificación de las mismas. Los resultados demuestran que la aplicación en forma general tiene un aceptable desenvolvimiento debido a que los errores no superan el 50 %.

4.5. Pruebas de Usabilidad

Estas pruebas aportan datos cuantitativos sobre como usuarios reales interaccionan con el software desarrollados, y poder evaluar algunos aspectos de accesibilidad y uso del programa. El listado de de personas encuestadas se presenta a continuación:

Nombre	Cargo
Rodrigo López	Administrador del DSpace
Yofre Tene	Docente
Byron Álvarez	Estudiante Mod. Abierta
Sergio Granda	Estudiante Mod. Presencial
Santiago Suárez	Desarrollador

La encuesta aplicada consiste en determinar el nivel de aceptación por parte de los usuarios, la cual contiene las siguientes preguntas:

Objetivo:				
Verificar que el sistema es usable				
No.	**Actividad**	**Pregunta**	**Resultado Esperado**	**Resultado Alto / Medio / Bajo**
10.	Poner a un usuario frente al sistema y pedirle que lo analice	**¿El sistema es intuitivo?**	Comprobar que el sistema por si solo le da una pauta al usuario de lo que tiene que hacer	SI / Algo / NO
11.	Pedirle al Usuario que interactué con el sistema	**¿El sistema es fácil de Usar?**	Verificar si el sistema es usable sin que exista alguna capacitación al usuario, es decir que pueda ser usado por cualquier persona	SI / Algo / NO.
12.		**¿El sistema desplegó los mensajes de error correspondientes?**		SI / NO
13.		**¿El sistema se cayó con algún dato erróneo del usuario?**		SI / NO
14.	Pedirle al usuario que realice consultas con algunas entidades previamente reconocidas en el sistema	**¿El sistema arrojo los resultados relevantes para su consulta?**	Verificar que el sistema es efectivo en los requerimientos del usuario	SI / Algo / NO
15.		**¿Qué le pareció el tiempo que empleo el sistema en devolver los resultados?**		Rápido / Normal / Lento
16.	Pedirle a los usuarios que interactúen con el sistema al mismo tiempo	**¿El sistema se cayó o volvió lento con varios usuarios a la vez?**	Verificar que el sistema soporta concurrencia de usuarios	SI / Algo / NO

Los resultados obtenidos demuestran que el sistema tiene un nivel de aceptación alto del 82 %, por lo tanto se puede concluir que el sistema es usable ya que posee un porcentaje mayor al 50 %.

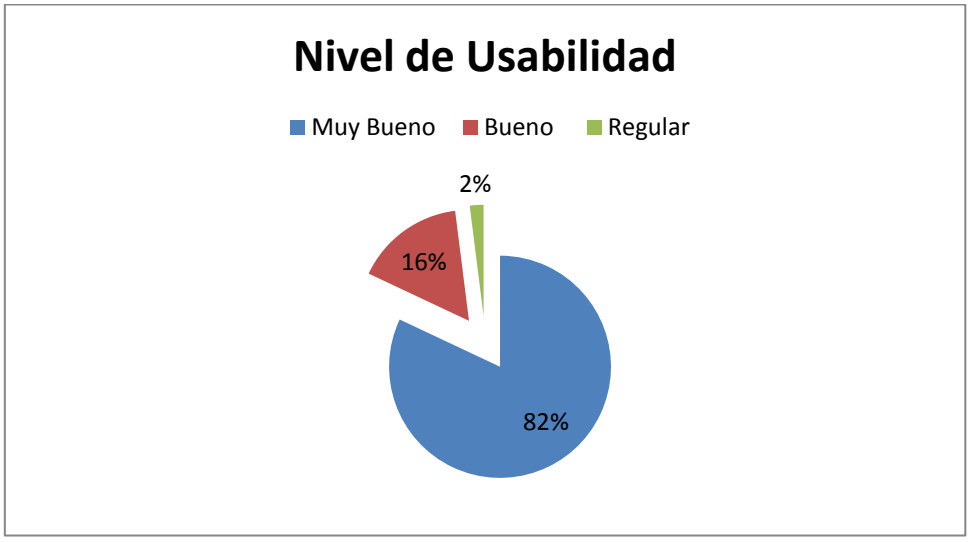


Figura 4.4: Nivel de Aceptación de la aplicación

4.6. Pruebas de Concurrencia

Estas pruebas están encaminadas a comprobar cómo el sistema reacciona ante varios clientes concurrentes, esta prueba se realiza en dos partes, en la primera se utiliza un conjunto de consultas las cuales son ejecutadas por 6 personas (estudiantes de GP y administrador del sistema) al mismo tiempo. La segunda parte se hace uso de la herramienta webwait[51] la cual permite crear un conjunto de conexiones a Buspace cada intervalo de tiempo para medir la concurrencia y disponibilidad del software construido. Los resultados que se muestran a continuación, demuestran que Buspace utiliza el 11.6 % de CPU, el 38,4% de Memoria y un tiempo promedio de 1.059 segundos en consultas concurrentes.

Tabla 4.8: Resultados de pruebas de concurrencia con usuarios reales

Operación	%CPU	%MEM	Tiempo (s)
Operación 1	8	34,1	1,16
Operación 2	3	45	0,4
Operación 3	0	34	1,58
Operación 4	4	56	0,14
Operación 5	14	34	1,17
Operación 6	28	34	1,17
Operación 7	7	34	1,18
Operación 8	15	34	1,43
Operación 9	3	34	1,13
Operación 10	34	45	1,23
Promedio	11,6	38,41	1,059

Las operaciones que mayores recursos consumieron son la operación 4 y operación 10, en la primera se hace mayor uso de la memoria, por lo que se concluye que en esta operación hubieron mas accesos al disco lo cual implica mayor consumo de tiempo, esto se debe en que la operación 4 se ejecutaron las consultas que implicaban mayor número de OA recuperados. En tanto la operación 10 tiene el mayor índice de carga en procesamiento (CPU), memoria y tiempo, esta condición es conocida como etapa de apogeo[52] donde existen gran cantidad de procesos concurrentes, el consumo de procesador y memoria (acceso al disco) dan la pauta de que al menos 4 de las 6 operaciones se ejecutaron en ese momento.

Mientras tanto las pruebas con la herramienta anteriormente descrita demostraron que un escenario de 100 llamadas concurrentes cada 5 segundos se obtiene un promedio de 0.21 segundos de transacción atendida

4.7. Discusión Final:

Los resultados evidencian que la estructuración de contenido con GATE, la ontología de KIM y Freeling han ayudado de manera notable en la precisión de los resultados, con lo que tiene que ver la retentiva si bien sus resultados no son malos, en términos de falsos positivos; es decir,

[51] http://webwait.com/
[52] Apogeo: Intervalo de tiempo donde ocurre la mayo carga de procesos en un computador.

catalogar de relevantes a aquellos documentos cuando en realidad no lo son siempre será un problema de cara al usuario.

Las pruebas realizadas en este capítulo son las básicas de todo buscador, sin embargo existen otra pruebas que podría ser aplicadas, sobre todo aquellas que ayuden a mejorar la experiencia del usuario como son: amigabilidad de las interfaces, formatos de presentación, tiempos de respuesta y otras que van más orientadas a los administradores, para mejorar el servicio de búsqueda por ejemplo la medida F[53].

[53] La Medida-F, F-measure, combina la preción y la cobertura (exhaustividad) según un parámetro $\alpha \in (0,+\infty)$

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- El problema de la polisemia en las consultas ingresadas puede ocasionar que algunos documentos recuperados no sean relevantes a los intereses del usuario o en su defecto producirse el olvido de documentos relevantes, es decir un silencio informativo. En este caso se hace necesario la intervención del usuario para establecer el sentido estricto del término o consulta en cuestión.
- Resultado del análisis del Capítulo I se pudo determinar que existen técnicas semánticas y no semánticas para proveer de inteligencia a un buscador, lo semántico no necesariamente se relaciona con ontologías.
- Se encontraron algunas herramientas maduras para PLN desarrolladas en lenguajes como Perl, Python y C, inclusive la herramienta escogida para este propósito Freeling está escrita enteramente en C; sin embargo, posee una API para java. Estos lenguajes ofrecen facilidades para el desarrollo de esta clase de aplicaciones.
- De las herramientas revisadas en el Capítulo II, las de tipo open source ofrecen mejores prestaciones que las tipo privativo, esto se debe principalmente al desarrollo colaborativo que poseen; sin embargo, la documentación puede llegar a ser básica o descuidada.
- La integración de Kim y el anotador semántico basado en GATE fue fundamental en el rendimiento de Buspace en la fase de pruebas, que con un 99 y 101 % respectivamente en precisión y retentiva fueron superiores al buscador sintáctico de DSpace (37 % precisión, 60 % retentiva).
- Implementar herramientas opensource de diferente naturaleza ocasiona problemas de integración del sistema final y toma mayor tiempo de desarrollo depurar los problemas de integración.
- Una de las ventajas de anotar semánticamente la consulta del usuario como parte de PLN, es que se pueden descubrir relaciones adicionales a las que Freeling puede detectar y así aumentar la especificidad de la consulta para obtener resultados más precisos.
- Producto del desarrollo del presente trabajo se ha podido determinar que existen un conjunto de trabajo de investigación derivados los cuales se describen a continuación:
    - Complementar el análisis semántico del PLN con la ayuda de la Base de Datos Léxica EuroWordnet, las definiciones cortas y las relaciones semánticas entre sinónimos de una palabra pueden ayudar a resolver el problema de polisemia, así como también aumentar la precisión en obtener el significado de un término en la consulta.

- Un trabajo aparte de investigación correspondería al desarrollo modelos de EI especializados para la notación semántica de documentos en determinados campos del saber humano.
- Así mismo el desarrollo de ontologías específicas en diferentes temáticas, todo esto con el fin de aumentar la riqueza expresiva de las anotaciones semánticas encontradas en los documentos.
- El módulo de EI trabaja estadísticamente para reconocer entidades relevantes y por lo cual está sujeto a errores, por lo tanto se hace necesario que los propios usuarios que suben sus recursos al repositorio DSpace de la Universidad tenga la facultad de administrar las entidades reconocidas en los documentos, para ir corrigiendo errores y aumentar las entidades presentes en los documentos.

Recomendaciones

- Informar e incentivar a los usuarios del repositorio educativo de la universidad como el contenido estructurado mejora significativamente el proceso de búsqueda de OA.
- Desarrollar estrategias y componentes para que los OA puedan ser anotados semánticamente por la comunidad de usuarios del repositorio educativo de la universidad.
- Extender la base de conocimiento con ontologías propias de la universidad o con otras de propósito general como por ejemplo DBPedia.
- En el presente trabajo se realiza un PLN medio para el tratamiento de consulta, en trabajos futuros debe considerarse el desarrollo más completo de este componente y su integración con Word net para el español.
- Al momento de trabajar con herramientas open source percatarse que tenga suficiente documentación (documentación básica no sirve de mucho), manuales de personalización descriptivos, lista de correo - FAQ, y de ser posible considerar si cuenta con wiki.
- Realizar una revisión a los metadatos de los OA en el DSpace ya que algunos están vacios o la descripción no corresponde al OA en cuestión, además de realizar un control de los recursos que se encuentran duplicados y perdidos.
- En los posible no trabajar con demasiadas herramientas ya que pueden presentarse problemas al momento de integrarlas, además pueden consumir muchos recursos (tiempo y hardware).

GLOSARIO DE TÉRMINOS

- **API:** Aplication Programming Interface. Descripción de alto nivel de una librería o utilidad que permite al desarrollador usarla sin preocuparse por la detalles ni estructurar interna de la utilidad.
- **Bitstream:** Conjunto de datos digitales, comúnmente codificados, que son transmitidos entre sistemas (audio, video, imágenes, datos, señales, etc.).
- **OA:** Objeto de Aprendizaje, cualquier recurso (pdf, página web, video, podcast, Word, Excel) que contengan un conocimiento intrínseco.
- **Handle:** Identificador único de un OA dentro del repositorio que es propio o administrado por un proceso.
- **Metadatos:** Se tratan de datos sobre los propios datos, estos están asociados a cada OA y permiten describir el contenido de los mismos.
- **RI:** La recuperación de información es el conjunto de tareas mediante las cuales el usuario localiza y accede a los recursos de información que son pertinentes para la resolución del problema planteado.
- **PLN:** El PLN se concibe como el reconocimiento y utilización de la información expresada en lenguaje humano a través del uso de sistemas informáticos.
- **EI:** Extracción de hechos relevantes de un texto, se apoya en el PLN para realizar su tarea.
- **URI:** (Uniform Resource Identifier) es una cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Siempre es el mismo con respecto a un recurso, incluso si éste ha cambiado de localización (servidor).
- **URL:** (Uniform Resource Locutor), localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.
- **Repositorio:** Depósito donde se almacena y mantiene información digital.
- **Ontología:** Esquema conceptual de uno o varios dominios del saber humano, con el objetivo de facilitar el intercambio y comunicación entre diferentes sistemas.
- **Documento Kim:** Recurso o OA anotado semánticamente y almacenado en la base de conocimiento.

Anexo 1: Instalación de herramientas de RI

1. SIREN – LUCENE

1.1. Introducción

Es la extensión semántica del conocido motor de búsqueda sintáctica Lucene[54]. Esta extensión ha sido desarrollada por Renaud Delbru del departamento de Digital Enterprice Research Institue (DERI) de la Universidad Nacional de Irlanda, está escrito enteramente en Java y es compatible para trabajar con los métodos nativos de Lucene. SIREN añade una nueva etiqueta llamada *"Tuples"* a las que vienen por defecto en Lucene (Title, date, etc.), esta etiqueta acepta información especial estructurada llamada N-Tuples que es un formato basado en texto plano para codificar datos semi-estructurados como tripletas RDF u otros formatos. En si SIREN adopta la estructura de árbol de un XML y ordena cada entidad en un árbol donde cada nodo representa una entidad o una descripción.

1.2. Hardware y Software utilizado

Las pruebas se realizaron utilizando el siguiente hardware y software:

- Código fuente de Siren-0.2[55] o el archivo jar.
- Lucene 3.0.1: Que es compatible con la versión de SIREN[56].
- Entorno de desarrollo Netbenas 6.8
- Máquina virtual JDK 1.6
- Herramienta Apache Maven[57], necesaria en caso de que se quiera construir los archivos jar desde los fuentes.

1.3. Instalación de la herramienta

A continuación se presenta como generar los archivos jar desde los archivos fuentes de la herramienta.

[54] http://lucene.apache.org/
[55] http://siren.sindice.com/download.html
[56] Versiones anteriores de Lucene (2.4.0 y 2.3.1) presentan problemas al integrar con SIREN, se lo puede obtener desde: http://lucene.apache.org/java/docs/index.html
[57] http://maven.apache.org/

1. Desde una ventana de comandos o terminal en Linux, se sitúa bajo el directorio en el que se encuentran los fuentes, dentro está un archivo XML llamado POM.xml que describe los targets de construcción, para ejecutar estos targets se digita el comando “mvn package”, el cual generará el archivo jar de la herramienta bajo el directorio target

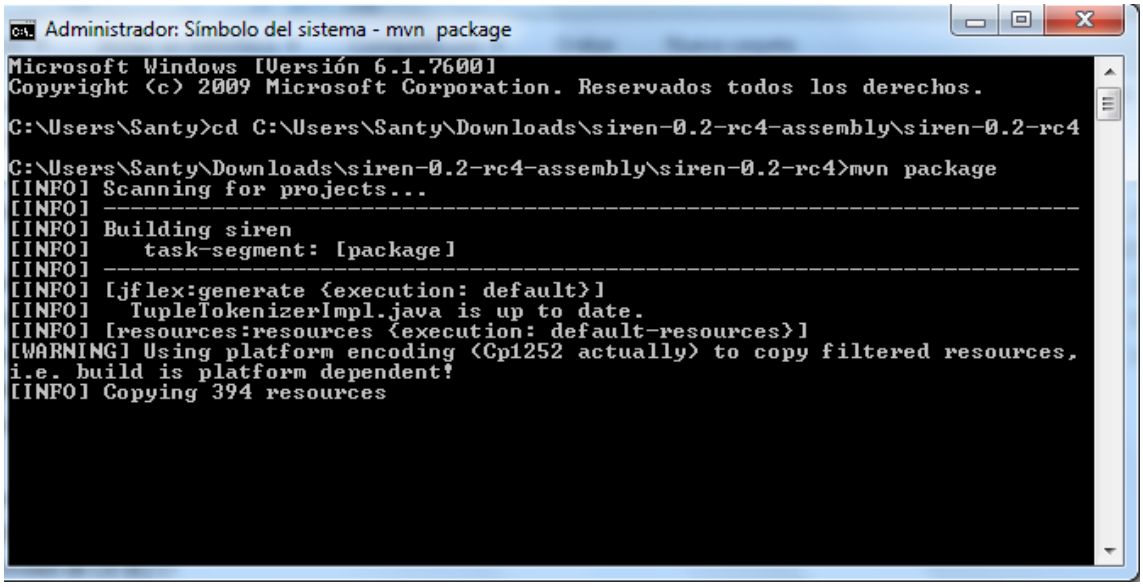


Figura A1.1: Construir los archivos jar con Apache ANT

2. Una vez generados los archivos jar, se agregan como librería junto con las librerías de Lucene en el entorno de desarrollo NetBeans.

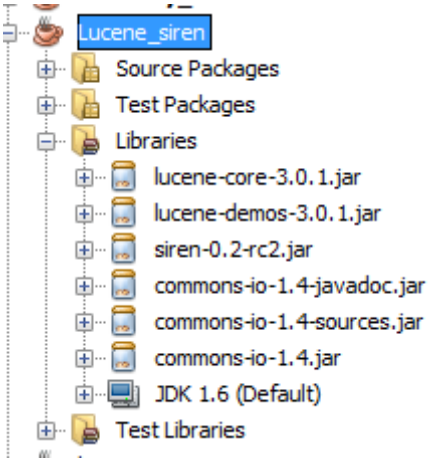


Figura A1.2: Agregar librerías al proyecto java en NetBeans

1.4. Descripción de las clases

Después del análisis realizado con SIREN, se destacan las siguientes funcionalidades:

- **Clase Tuple Analyzer:** Es la extensión de la clase Standar Analyzer de Lucene, esta clase posee métodos que permiten manejar de mejor manera el contenido estructurado que contiene la etiqueta N-Tuples , los métodos destacables dentro de esta clase son:
- **URITrailingSlashFilter:** Permite remover barras innecesarias de un URI, por ejemplo:

  http://xmlns.com/foaf/0.1/ → http://xmlns.com/foaf/0.1

- **URINomralisationFilter:** Este método permite descomponer un URI en tokens fácilmente manejables;

	posición	token
http://xmlns.com/foaf/0.1/name"	0	"http"
	1	"Xmlns.com"
	2	"foaf"
	3	"0.1"
	4	"name"
	5	"http://xmlns.com/foaf/0.1/name"

Tabla A1.1: Descomposición de un URL con el método URINomralisationFilter

- **Integración con Lucene:** también se pueden combinar con los métodos de análisis primitivos de Lucene como son:
  - **LowerCaseFilter:** transforma todo el contenido de las etiquetas en minúsculas
  - **StopFilter:** Elimina todas las palabras vacías (él, la, los, etc.).
  - **LengthFilter:** Remueve las palabras demasiado cortas (2 caracteres) o muy largas (128 caracteres).

- **Consultas:** Al momento de realizar consultas, estas pueden ser semi-estructuradas, combinarse con operadores booleanos o combinarse con métodos primitivos de Lucene, las clases principales para realizar las consultas son:

- **SirenTermQuery:** Cumple similares funciones que la clase nativa de Lucene TermQuery, que permite realizar consultas que coincidan con determinado término.
- **SirenPharesQuery:** Es similar al método nativo de Lucene PhraseQuery, el cual busca correspondencias exactas con una frase determinada.
- **SirenCellQuery:** Tanto las consultas de tipo SirenTermQuery como SirenPharesQuery pueden ser combinadas mediante esta clase. Como se dijo anteriormente SIREN representa el contenido estructurado en forma de tabla, cada vez que se crea un objeto de tipo SirenCellQuery se crea una celda dentro de esta tabla. A continuación se presenta de manera gráfica su funcionamiento:

```
SirenCellQuery cq1 = new SirenCellQuery();
```

Dentro de este objeto se puede estructurar cualquier tipo de consulta, como por ejemplo consultas booleanas de Lucene:

```
cq1.add(new SirenTermQuery(new
Term(leer.document(i).getField("tipo").stringValue(), "tesis")),
SirenCellClause.Occur.MUST);
```

En la porción de código anterior se ha estructurado dentro del objeto SirenCellQuery cq1 un objeto de tipo SirenTermQuery que a su vez contiene un objeto nativo de Lucene de tipo Term.

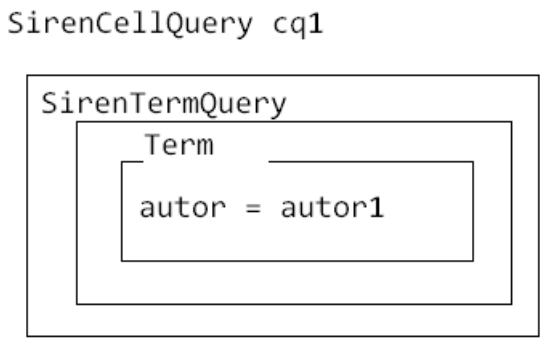


Figura A1.3: Representación de una consulta con la clase SirenTermQuery

1.4.6 SirenTupleQuery: Esta clase permite combinar varios objetos de tipo SirenCellQuery, en una tupla:

```
SirenTupleQuery tq = new SirenTupleQuery();
tq.add(cq1, SirenTupleClause.Occur.MUST);
tq.add(cq2, SirenTupleClause.Occur.MUST)
```

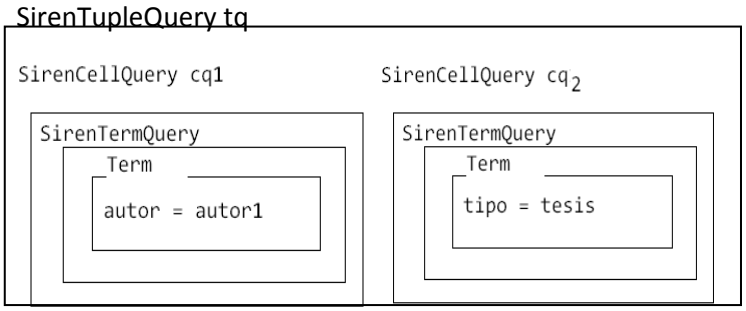


Figura A1.4: Combinar varios consultas en un objeto SirenTupleQuery

1.4.6.1 SirenCellClause.Occur: Con esta clase se establece el tipo de coincidencia para una consulta. Esta puede ser de varios tipos:

Método	Tipo	Detalle	EQUIVALENTE
SirenCellClause.Occur.	MUST	Cuando se requiere que la correspondencia sea exacta	AND
	MUST_NOT	Aplicable cuando se	NOT

		quiere dejar de lado un término o palabra	
	SHOULD	Cuando la correspondencia puede ser aproximada	OR

Tabla A1.3: Opciones del método SirenCellClause.Occur.

1.4.6.2 BooleanQuery: Tanto a las consultas de tipo SIREN como a las consultas primitivas de Lucene se pueden combinar en una consulta booleana.

```
/COMBINADOS LA BUSQUEDA ESTRCTURADA DE SIREN CON LA BUSQUEDA BOOLEANA DE LUCENE
        BooleanQuery q = new BooleanQuery();//// Unimos la consulta tq AND tq1:
          //Documentos de tipo tesis cuyo autor es autor1
          q.add(tq,
org.apache.lucene.search.BooleanClause.Occur.SHOULD);
           // OR que hablen del modelo vectorial matriz de términos documentos y Sistemas de Recuperación de Información de Búsqueda Aproximada
          q.add(query,
org.apache.lucene.search.BooleanClause.Occur.MUST);
```

1.5 Código

```
public class test_siren {
  private static final int   COLUMN_TUPLE_TYPE  = 0;
  private static final int   COLUMN_PERSON_NAME = 1;
  private static final int   COLUMN_MOVIE_TITLE = 1;
  private static final int   COLUMN_BIRTH_DATE  = 2;
  private static final int   COLUMN_BIRTH_PLACE = 3;
  private static final int   COLUMN_ANY        = -1;
```

```
 public RAMDirectory       dir;
// public File carpeta = new File("./index_lucene");
 //public File carpeta = new File("./index2");
 public Directory dir_index;
 public IndexWriter        writer;
 public IndexReader        reader;
 public static final String DEFAULT_FIELD     = "content";
  @SuppressWarnings("static-access")
 public test_siren() throws CorruptIndexException, LockObtainFailedException, IOException {
  dir = new RAMDirectory();
    writer = new IndexWriter(dir, new TupleAnalyzer(), MaxFieldLength.UNLIMITED);
 // writer = new IndexWriter(dir_index, new TupleAnalyzer(), MaxFieldLength.UNLIMITED);
  reader = IndexReader.open(dir, true);
   //dir_index = FSDirectory.open(carpeta);
  //reader = IndexReader.open(dir_index, true);
}
 public void addDocument(final File input)  throws IOException {
  final Document doc = new Document();
  final BufferedReader br = new BufferedReader(new InputStreamReader(
    new FileInputStream(input), "ISO-8859-1"));
  //final BufferedReader br = new BufferedReader(new InputStreamReader(
    //new FileInputStream(input), "ISO-8859-1"));
  final StringBuilder sbTupleContent = new StringBuilder();
  String line;
  while ((line = br.readLine()) != null) {
    if (line.matches("^\\w+:.*")) {
      // title and url fields.
      final String[] parts = line.split(":", 2);
      doc.add(new Field(parts[0], parts[1], Store.YES,
        Index.NOT_ANALYZED_NO_NORMS));
```

```
      if ("title".equals(parts[0])) {
        //System.out.println("adding: " + parts[1] + " from " + input.getName());
      }
    }
    else {
      // tuple fields
      if (sbTupleContent.length() > 0) {
        sbTupleContent.append("\n");
      }
      sbTupleContent.append(line);
    }
  }
  br.close();
  if (sbTupleContent.length() > 0) {
    doc.add(new Field(DEFAULT_FIELD, sbTupleContent.toString(), Store.NO,
      Field.Index.ANALYZED_NO_NORMS));
  }
  writer.addDocument(doc);
  writer.commit();
}

public ScoreDoc[] search(final Query q)
throws IOException {
  final IndexSearcher searcher = new IndexSearcher(dir);
  return searcher.search(q, null, 10).scoreDocs;
}

public Document getDocument(final int docId)
throws CorruptIndexException, IOException {
  reader = reader.reopen();
```

```
    return reader.document(docId);
  }

  public void close()
  throws CorruptIndexException, IOException {
    writer.close();
    reader.close();
//    dir.close();
    dir.close();
  }

  /**
   * create a query to find films in which stallone and fuller acted.
   */
  public Query Consulta1() {//Recupera todas las videoconferencias

     final SirenTupleQuery tq1 = new SirenTupleQuery();
    tq1.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
      SirenTupleClause.Occur.MUST);
    tq1.add(this.termInCell("videoconferencias", COLUMN_PERSON_NAME),
      SirenTupleClause.Occur.MUST);

    final SirenTupleQuery tq2 = new SirenTupleQuery();
    tq2.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
      SirenTupleClause.Occur.MUST);
    tq2.add(this.termInCell("ingeniería civil", COLUMN_PERSON_NAME),
      SirenTupleClause.Occur.SHOULD);

    // Combine two tuple queries with a Lucene boolean query
    final BooleanQuery q = new BooleanQuery();
```

```
 q.add(tq1, Occur.MUST);
 q.add(tq2, Occur.MUST);

 return q;

}//end consulta1

 public Query Consulta2() {

   //Diagnóstico y tramiento de la hipatidosis
    // Create a tuple query that combines the two cell queries
 final SirenTupleQuery tq1 = new SirenTupleQuery();
 tq1.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
   SirenTupleClause.Occur.MUST);
 tq1.add(this.termInCell("diagnóstico", COLUMN_PERSON_NAME),
   SirenTupleClause.Occur.SHOULD);

 final SirenTupleQuery tq2 = new SirenTupleQuery();
 tq2.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
   SirenTupleClause.Occur.MUST);
 tq2.add(this.termInCell("contagio", COLUMN_PERSON_NAME),
   SirenTupleClause.Occur.SHOULD);

 final SirenTupleQuery tq3 = new SirenTupleQuery();
 tq3.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
   SirenTupleClause.Occur.MUST);
 tq3.add(this.termInCell("hipatidosis", COLUMN_PERSON_NAME),
   SirenTupleClause.Occur.MUST);

 // Combine two tuple queries with a Lucene boolean query
```

```
 final BooleanQuery q = new BooleanQuery();
 q.add(tq1, Occur.MUST);
 q.add(tq2, Occur.MUST);
 q.add(tq3, Occur.MUST);

 return q;

}//end consulta2

 public Query Consulta3() {
       final SirenTupleQuery tq1 = new SirenTupleQuery();
       tq1.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
       SirenTupleClause.Occur.MUST);
       tq1.add(this.termInCell("ing.", COLUMN_PERSON_NAME),
       SirenTupleClause.Occur.SHOULD);

       final SirenTupleQuery tq2 = new SirenTupleQuery();
       tq2.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
       SirenTupleClause.Occur.MUST);
       tq2.add(this.termInCell("economía", COLUMN_PERSON_NAME),
       SirenTupleClause.Occur.SHOULD);

    final BooleanQuery q = new BooleanQuery();
    q.add(tq1, Occur.MUST);
    q.add(tq2, Occur.MUST);
    //q.add(tq3, Occur.MUST);
 return q;
 }
 public Query Consulta4() {
       final SirenTupleQuery tq1 = new SirenTupleQuery();
```

```
        tq1.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
        SirenTupleClause.Occur.MUST);
        tq1.add(this.termInCell("cálculo", COLUMN_PERSON_NAME),
        SirenTupleClause.Occur.MUST);

        final SirenTupleQuery tq2 = new SirenTupleQuery();
        tq2.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
        SirenTupleClause.Occur.MUST);
        tq2.add(this.termInCell("economía", COLUMN_PERSON_NAME),
        SirenTupleClause.Occur.SHOULD);

    final BooleanQuery q = new BooleanQuery();
    q.add(tq1, Occur.MUST);
   // q.add(tq2, Occur.MUST);
    //q.add(tq3, Occur.MUST);

return q;
}

 public Query Consulta5() {
        final SirenTupleQuery tq1 = new SirenTupleQuery();
        tq1.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
        SirenTupleClause.Occur.MUST);
        tq1.add(this.termInCell("video", COLUMN_MOVIE_TITLE),
        SirenTupleClause.Occur.MUST);

        final SirenTupleQuery tq2 = new SirenTupleQuery();
        tq2.add(this.termInCell("keyword", COLUMN_TUPLE_TYPE),
        SirenTupleClause.Occur.MUST);
        tq2.add(this.termInCell("sociología", COLUMN_PERSON_NAME),
```

```
        SirenTupleClause.Occur.SHOULD);

     final BooleanQuery q = new BooleanQuery();
     q.add(tq1, Occur.MUST);
    q.add(tq2, Occur.MUST);
     //q.add(tq3, Occur.MUST);
  return q;
  }

 private SirenCellQuery termInCell(final String term, final int cell) {
  final SirenCellQuery scq = new SirenCellQuery();
  scq.add(new SirenTermQuery(new Term(DEFAULT_FIELD, term)),
   SirenCellClause.Occur.MUST);
  if (cell != COLUMN_ANY) {
   scq.setConstraint(cell);
  }
  return scq;
 }

 public static void main(final String[] args) throws CorruptIndexException,
LockObtainFailedException, IOException {
  final File folder = new File("./repo2/");

  final File[] files = folder.listFiles(new FilenameFilter()
    {public boolean accept(final File dir, final String name) {
    return name.endsWith(".txt");
   }
  });

  final test_siren indexer = new test_siren();
```

```
    for (final File file : files) {
      indexer.addDocument(file);
    }
    System.out.println("adicionados " + files.length + " archivos");

    ScoreDoc[] results = indexer.search(indexer.Consulta1());
    System.out.println("Consulta 1: " + results.length);
      results = indexer.search(indexer.Consulta2());
    System.out.println("Consulta 2: " + results.length);

      results = indexer.search(indexer.Consulta3());
    System.out.println("Consulta 3: " + results.length);

    results = indexer.search(indexer.Consulta4());
    System.out.println("Consulta 4: " + results.length);

    results = indexer.search(indexer.Consulta5());
    System.out.println("Consulta 5: " + results.length);

    indexer.close();
  }
}
```

Anexo 2: Instalación de herramientas para PLN

1 OpenNlp

1.1. Introducción

OpenNlp es una librería escrita en java, posee un conjunto de métodos para el PLN, trabaja con un conjunto de datos llamados modelos, los que poseen reglas y características para cada método de la herramienta; para el idioma español existe cuatro modelos que vienen por defecto en el paquete de instalación que corresponden a los procesos de: detección de sentencias, tokenización, etiquetado (taggin); los demás métodos (parsing, correferencia, reconocimiento de entidades nombradas) tienen que ser entrenados con un conjunto de datos etiquetados previamente, existen métodos especiales de entrenamiento que generan el modelo, para luego ser utilizados por los métodos descritos anteriormente.

1.2. Prerrequisitos

Para utilizar OpenNlp se debe contar con las sigueintes herramientas:

- Entorno de desarrollo, por ejemplo: NetBeans
- Máquina virtual JDK 1.6 o superior
- Para la construcción del archivo jar de OpenNlp se debe tener instalado la herramienta Apache ANT: http://ant.apache.org/
- Descargar los fuentes de OpenNlp : http://opennlp.sourceforge.net/
- Obtener los modelos de funcionamiento de los métodos desde: http://opennlp.sourceforge.net/models.html

1.3. Instalación de la herramienta

Una vez descargados los fuentes de OpenNlp, en una ventana de comandos (terminal en Linux) y se ubica en el directorio donde se encuentra los fuentes y se ejecutan los comandos de la herramienta ANT para generar el jar:

- ./build.sh →Linux genera el jar
- ./build.sh javadoc →Linux genera la documentación de la herramienta
- Ant package →Windows genera el jar
- Ant javadoc →Windows genera la documentación de la herramienta

El jar de la herramienta se guarda bajo el directorio *"Target"*, el cual puede ser añadido como librería en el entorno de Desarrollo (NetBeans) para comenzar a trabajar con ella.

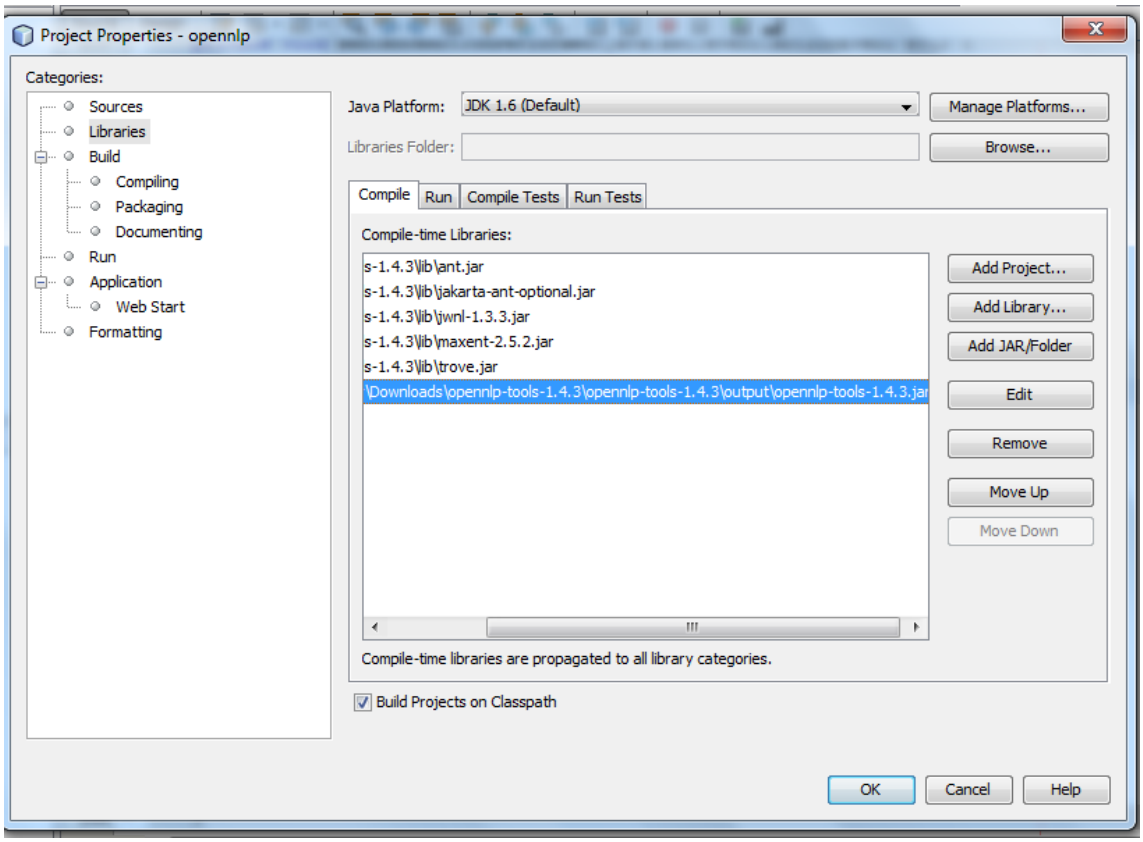


Figura A2.1: Agregar librerías a un proyecto java en NetBeans

Nota: Mayor en información en: http://opennlp.sourceforge.net/README.html

1.4. Descripción de las Clases

A continuación se describen las diferentes clases y métodos de esta herramienta:

1.4.1. Clase: *SentenceDetector:* Esta clase es la encarga de dividir el texto en oraciones, recibe como parámetro de entrada la ruta donde se encuentra el modelo y el nombre del archivo, en este caso "SpanishSent.bin.gz".

1.4.1.1. Método sentDetect: Obtiene el número de oraciones presentes en el texto, recibe como parámetro inicial un String con el contenido y devuelve un array de String con las oraciones separadas.

```
String[] sent = std.sentDetect(texto);
```

1.4.1.2. **Método getSentenceProbabilities:** Devuelve por cada oración la probabilidad con la que ha sido reconocida una oración en un array de tipo double.

```
double[] cos = std.getSentenceProbabilities();
```

1.4.1.3. Método sentPosDetect: Cuenta el número de caracteres total de una oración.

int[] no = std.sentPosDetect(texto)

#	probabilidad	sentencia
1	100 %	Cuando falta poco para la medianoche, las emisoras de radio colocadas a todo volumen, dan la cuenta atrás y ya a las ...
2	100 %	Los más religiosos, tras el abrazo a los seres queridos, reciben el año atendiendo una misa en la Iglesia; pero hay otro...

Figura A2.2: Resultados de la clase SentenceDetector

1.4.2. Clase Tokenizer: Esta clase tiene a cargo todas las operaciones de división de palabras o tokens, utiliza el modelo "SpanishTok.bin.gz".

Tokenizer tok = new Tokenizer(ruta + "SpanishTok.bin.gz");

1.4.2.1. Método tokenizer: Recibe como entrada el texto y devuelve un array de strings con las palabras divididas.

String [] toke = tok.tokenize(texto)

1.4.2.2. Método getTokenProbabilities: Obtiene por cada token o palabra la probabilidad con la que ha sido reconocido como token.

double[] valores = tok.getTokenProbabilities()

#	token	Probabilidad
1	Cuando	100 %
2	falta	100 %
3	poco	100 %
4	para	100 %
5	la	100 %
6	medianoche	100 %
7	,	100 %
8	las	100 %
9	emisoras	100 %
10	de	100 %
11	radio	100 %
12	colocadas	100 %
13	a	100 %
14	todo	100 %
15	volumen	100 %
16	,	100 %
17	dan	100 %
18	la	100 %
19	cuenta	100 %
20	atrás	100 %

Figura A2.3: Resultados de la clase Tokenizer

1.4.3. Clase PosTagger: Se encarga de las operaciones de etiquetado de los tokens, utiliza el conjunto de etiquetas EAGLES para el lenguaje español, trabaja con los modelos: "SpanishPOS.bin.gz" y "SpanishTokChunk.bin.gz" de los modelos el más eficiente es el primero.

1.4.3.1. Método Tag: Recibe como entrada el array de tokens obtenido por la clase Tokenizer y devuelve otro array con las etiquetas que corresponden a cada token.

String[] tage=tagr.tag(toke);

#	token	tag
1	Cuando	CS
2	falta	VIP
3	poco	RG
4	para	SPS
5	la	DA
6	medianoche	NC
7	,	FC
8	las	DA
9	emisoras	NC
10	de	SPS
11	radio	NC
12	colocadas	AQ
13	a	SPS
14	todo	DI
15	volumen	NC
16	,	FC
17	dan	VIP
18	la	DA
19	cuenta	NC
20	atrás	RG

Figura A2.4: Resultados de la clase PostTAgger

1.5. Código

```
package opennlp;

import java.io.IOException;
import opennlp.tools.lang.spanish.PosTagger;
import opennlp.tools.lang.spanish.SentenceDetector;
import opennlp.tools.lang.spanish.Tokenizer;
import opennlp.tools.parser.Parse;

/**
 *
 * @author Santy
 */
public class Main {
     //public static Object[] tokenes=null;
  //               File                directory=                new
File("C:\\Users\\Santy\\Documents\\DSPACE_ESTRCUTURA\\");
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        // TODO code application logic here
        String texto = "En este análisis se toma en cuenta las
herramientas OpenNlp y Freeling las cuales han sido instaladas y
```

configuradas, la información relacionada a estos procesos se encuentran en los Anexos 2 y 3 respectivamente. En el análisis de estas herramientas se prueba básicamente la segmentación de palabras, oraciones, el reconocimiento de componentes básicos (verbos, adjetivos, etc.), funciones gramaticales, para la asignación de valores se consideran los expuestos en la tabla 9.";

```
        String ruta = "C:\\Users\\Santy\\Documents\\NetBeansProjects\\opennlp\\models\\";

        //Sentence Detector: solo particiona las oraciones segun lo spuntos que va enocntrando
        SentenceDetector std = new SentenceDetector(ruta+"SpanishSent.bin.gz");
        String[] sent= std.sentDetect(texto);
        double cos[]= std.getSentenceProbabilities();
           int no []= std.sentPosDetect(texto);
//Imprime el número de carácteres del texto
        System.out.println(" Resultado del Sentence Detector:");

        presentar(sent);
        presentar2(sent, cos);
        for(int g=0;g < no.length;g++){
                  System.out.println("no "+no[g]+"lengt "+texto.length());
        }

        //**TOKENIZADOR
           Tokenizer tok = new Tokenizer(ruta+"SpanishTok.bin.gz");
           tok.setAlphaNumericOptimization(true);
           String[] toke = tok.tokenize(texto);


           //for(int i =1;i<=tokes.length;i++){System.out.println(tokes[i]);}
           double[]valores=tok.getTokenProbabilities();
```

```
            System.out.println("    Resultado de Tokenizer");
            presentar2(toke,valores);
                //**Postagger
        PosTagger tagr = new PosTagger(ruta+"SpanishPOS.bin.gz");

                String[] tage=tagr.tag(toke);
        Parse par =null;
        String auxi;
            System.out.println("  Resultado de Postagger");
            for(int i=0;i<tage.length;i++){
              System.out.println(toke[i]+"  ---> "+tage[i]);
              auxi = tage[i];
              System.out.println(Parse.parseParse(auxi));
            }
    }//end main

    public static void presentar(String[]dato){
        //System.out.println("\n");

            for(int i=0;i<dato.length;i++){
           System.out.println(dato[i]);
          }
      System.out.println("*****************************");
       //System.out.println("\n");
    }//end presentar

     public static void presentar2(String[]dato,double[] valores){
        for(int i=0;i<dato.length;i++){
              System.out.println(dato[i]+"  ---> "+valores[i]);
              //tokenes[i]=dato[i];
            }
        System.out.println("*****************************");
    }//end presentar2

}//end class
```

2. OpenCalais

2.1. Introducción

Esta herramienta se consume como servicio Web, se debe contar con una ApiKey la cual se obtiene previo registro en la página principal. Es un servicio que puede ser personalizado en base a un XML el cual describe los parámetros de entrada del Servicio Web. Puede ser consumido en SOAP, REST y HTTP Traffic Compression.

2.2. Prerrequisitos

- Entorno de desarrollo, por ejemplo: NetBeans con soporte para aplicaiones Web.
- Máquina virtual JDK 1.6 o superior
- Una Api key puede ser obtenida desde: http://www.opencalais.com/key
- Dirección del descriptor de servicios (WSDL): http://api.opencalais.com/enlighten/?wsdl
- Catálogo de parámetros de entrada: http://www.opencalais.com/documentation/calais-web-service-api/forming-api-calls/input-parameters

2.3. Instalación

2.3.1. En el entorno de desarrollo NetBeans y se crea un nuevo proyecto de tipo : “Java Aplication”

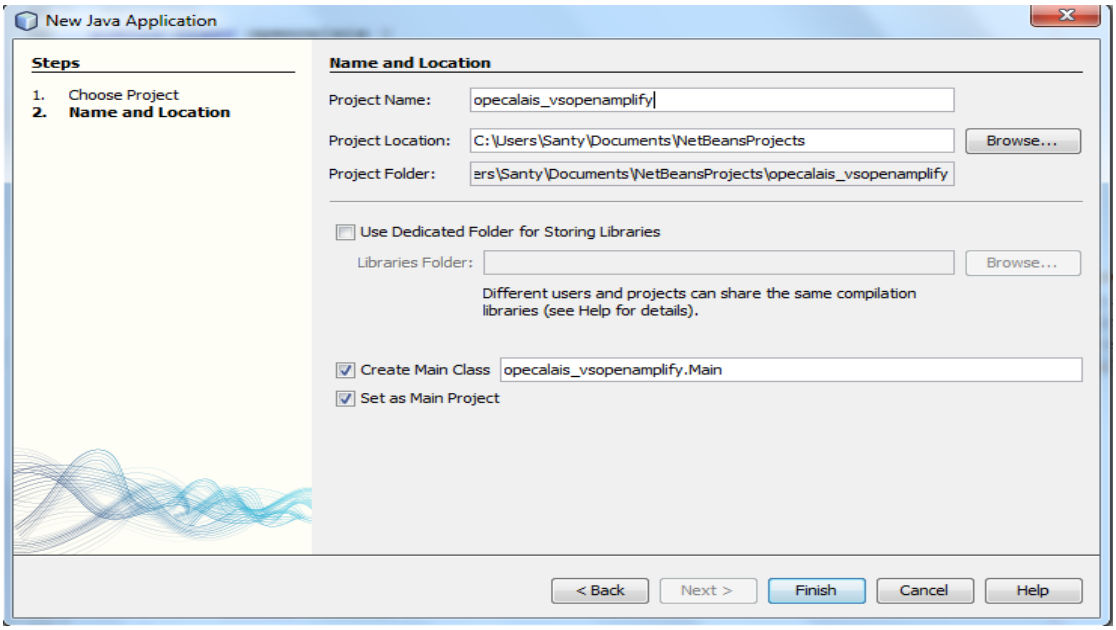


Figura A2.5: Crear un nuevo proyecto en NetBeans

2.3.2.Se da clic derecho en el proyecto y seleccionamos “New” y luego “Web Service Client”, en la ventana que aparece se selecciona la opción “WSDL URL” y se ingresa la dirección del descriptor de servicios: http://api.opencalais.com/enlighten/?wsdl

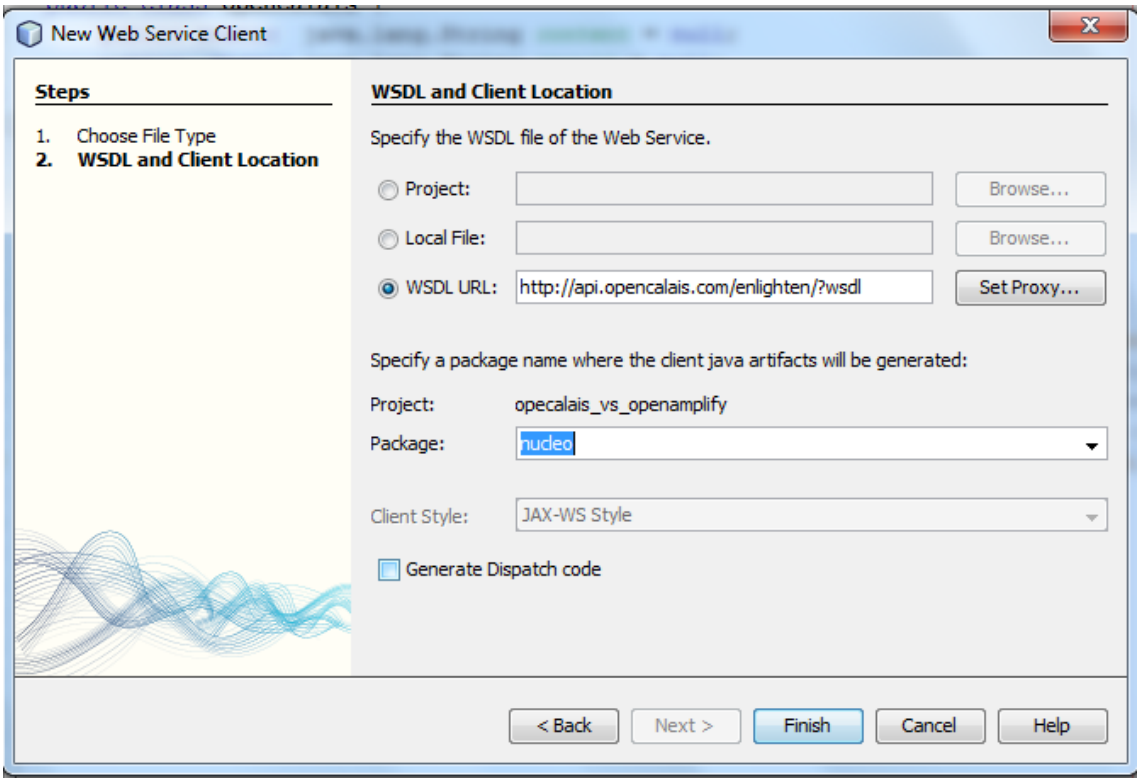


Figura A2.6: Configurar el WSDL del servicio Web

2.3.3.En la carpeta de paquetes fuentes se crea las clases que van a contener el código.

2.4. Descripción

2.4.1.Para utilizar OpenCalais primero se debe hacer una llamada al Web Service

```
com.clearforest.Calais service = new com.clearforest.Calais();
```

2.4.2.Se debe especificar el puerto que se va a utilizar, por ejemplo SOAP:

```
com.clearforest.CalaisSoap port = service.getCalaisSoap();
```

2.4.3.Luego se debe definir tres argumentos para llamar al Servicio Web, estos son:

2.4.3.1. Un String que contiene el id de licencia o Apikey: ”8g24ever3bk9xxxxxxxxxxxxx”

2.4.3.2. Un parámetro que contenga las configuraciones de entrada de Servicio Web: "<c:params xmlns:c=\http://s.opencalais.com/1/pred/\……."

2.4.3.3. Y una variable que contiene el texto a analizar: "Las reservas de oro y divisas de Rusia subieron 800….."

2.4.4. Luego se hace la llamada al servicio Web, se envía como parámetros las variables definidas anteriormente y este devuelve un String con los resultados.

String result = port.enlighten(licenseID, content, paramsXML);

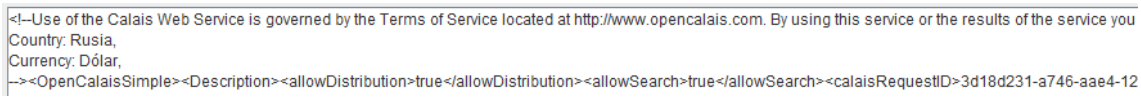


Figura A2.7: Resultados de OpenCalais

2.5. Código

```
public class opencalais {

   public static  java.lang.String content = null;

   public  static java.lang.String result = null;

  //public static void main(String[] args) {

   public String consulta(String cont){

   // TODO code application logic here

        content = cont;

       try { // Call Web Service Operation

          com.clearforest.Calais service = new com.clearforest.Calais();

          com.clearforest.CalaisSoap port = service.getCalaisSoap();

          // TODO initialize WS operation arguments here

          java.lang.String licenseID = "8g24ever3xxxxxxxxxxxxx";

           //content = "Las reservas de oro y divisas de Rusia subieron 800 millones de dólares y el 26 de mayo equivalían a 19.100 millones de dólares, informó hoy un comunicado del Banco Central.";

          java.lang.String paramsXML ="<c:params xmlns:c=\"http://s.opencalais.com/1/pred/\" xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"><c:processingDirectives c:contentType=\"text/txt\" c:enableMetadataType=\"GenericRelations,SocialTags\" c:outputFormat=\"Text/Simple\" c:docRDFaccesible=\"true\"
```

```
></c:processingDirectives><c:userDirectives c:allowDistribution=\"true\"
c:allowSearch=\"true\" c:externalID=\"utpl001\"
c:submitter=\"ABC\"></c:userDirectives><c:externalMetadata></c:externalMetadata></c:para
ms>";

          // TODO process result here

           result = port.enlighten(licenseID, content, paramsXML);

          //result = new String ("Hola");


          System.out.println(result);

      } catch (Exception ex) {

          // TODO handle custom exceptions here

                ex.printStackTrace();

      }


        return result;

   }


}
```

3. Freeling

3.1. Introducción

Freeling más que ser una herramienta para el tratamiento de lenguaje natural es una suite de componentes, está escrita completamente en C pero posee API's para diferentes lenguajes de programación, entre ellos java. Los propios autores de la suite sostienen que dan soporte exclusivamente para entornos Unix-Linux, sin embargo; algunos usuarios han provisto tutoriales de cómo instalar esta herramienta en ambientes Windows y Mac aunque no son efectivos en todos los casos. Para realizar las pruebas se utilizó la herramienta en Ubuntu 9.04, la instalación puede hacerse desde varios recursos: paquetes deb, compilando los fuentes y desde repositorios SVN, la instalación de paquetes deb es la más sencilla y permite utilizar la herramienta desde línea de comandos en una terminal.

3.2. Prerrequisitos

- Obtener el paquete deb desde la página: http://devel.cpl.upc.edu/freeling/downloads
- Tener instalado C para Ubuntu, puede ejecutar en una terminal: sudo apt-get install build-essential.
- Instala las librerías libdb4.6++ libpcre3 libboost-filesystem1.34.1, para esto se ejejuta en una terminal el comando: sudo apt-get install libdb4.6++ libpcre3 libboost-filesystem1.34.1

3.3 Instalación

En el directorio donde se encuentra el paquete deb de Freeling y se ejecuta el comando:

```
sudo dpkg -i FreeLing-2.1.deb
```

Terminada la instalación se cambia al directorio de instalación el cual es: /usr/share/Freeling/config/ y se procede a editar el archivo que contiene las configuraciones para el lenguaje español: es.cfg, es este archivo se cambian las siguientes líneas:

NEClasificattion= yes

NeRecongnition=basic

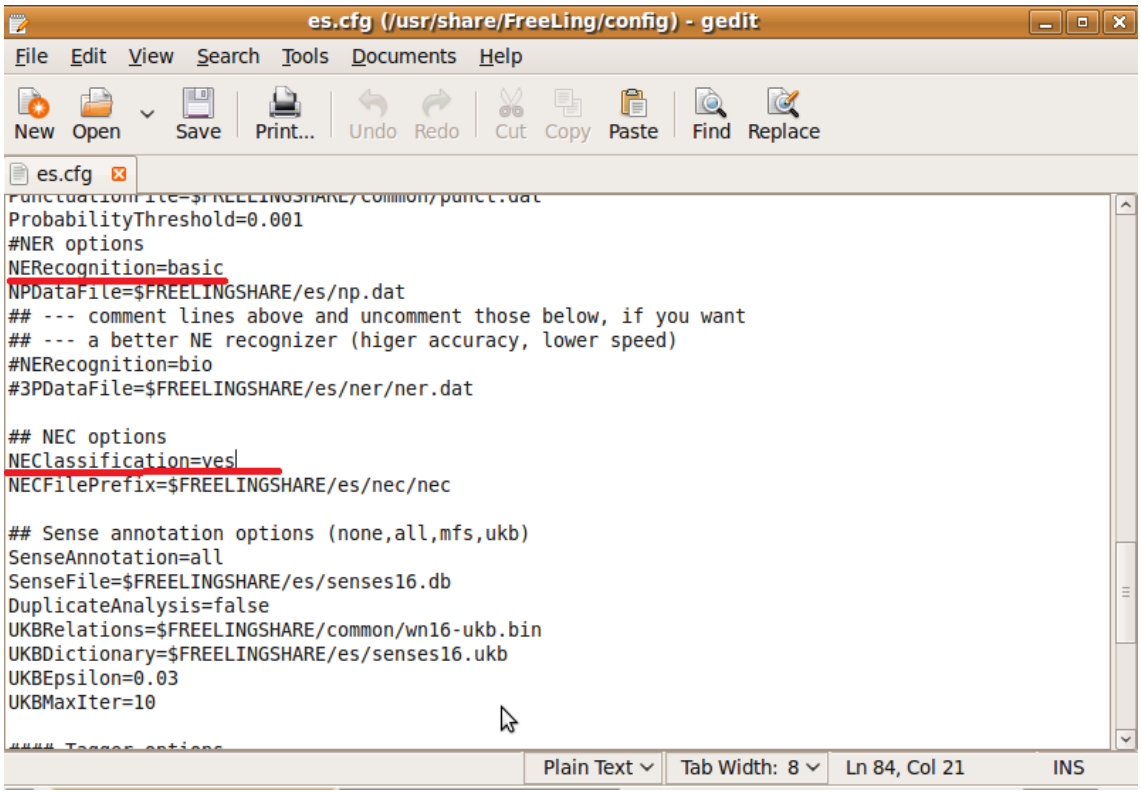


Figura A2.8: Modificaciones al archivo es.cfg

Se guarda los cambios en el archivo y ya se puede utilizar la herramienta.

3.4 Descripción

Con este tipo de instalación se utiliza Freeling desde línea de comando, las opciones y ejemplo de cómo utilizar se encuentran en la página: http://garraf.epsevg.upc.es/freeling/doc/userman/html/node59.html, con el comando "analyze" se tiene acceso a todas las funcionalidades.

3.4.1 Tokenización: Para la tokenización se utiliza la opción "token" recibe como entrada un texto plano y devuelve un archivo con un token en cada línea, las oraciones se dividen con un espacio en blanco.

Analyze –f es.cfg –outf token <entrada.txt >salida.txt

3.4.2 **Análisis Morfológico:** Con la opción morfo se divide el texto en oraciones y palabras (tokens), y hace un análisis morfológico.

Analyze –f es.cfg –outf morfo <entrada.txt >salida.txt

3.4.3 Etiquetado: La opción tagged permite el etiquetado de cada token, devuleve un archivo con tres datos: lemma, etiqueta y probabilidad.

Analyze –f es.cfg –outf tagged <entrada.txt >salida.txt

```
La el DA0FS0 0.972146 -
Federación_Nacional_de_Cafeteros_de_Colombia
federación_nacional_de_cafeteros_de_colombia NP00000 1 -
explicó explicar VMIS3S0 1 00635987:0.000000/00645379:0.000000
```

```
que que CS 0.4375 -
el el DA0MS0 1 -
nuevo nuevo AQ0MS0 1
01581807:0.000000/00131251:0.000000/01627275:0.000000/01582587:0
.000000/00026167:0.000000/00778835:0.000000/02461222:0.000000/01
241234:0.000000/01583317:0.000000
valor valor NCMS000 1
```

3.4.4 Análisis Sintáctico: Para este análisis se utiliza la opción parsed, la cual se encarga de construir un árbol donde cada nodo padre contiene la información sintáctica de la palabra que se encuentra como nodo hoja.

Analyze –f es.cfg –outf parsed <entrada.txt >salida.txt

```
S_[
  sn_[
    espec-fp_[
      +j-fp_[
        +(Las el DA0FP0 -)
      ]
    ]
    +grup-nom-fp_[
      +n-fp_[
        +(reservas reserva NCFP000
09625793:0.000000/06347607:0.000000/06208663:0.000000/06332265:0
.000000/00790310:0.000000/09626584:0.000000/07620124:0.000000/00
524765:0.000000/03671263:0.000000/09597262:0.000000)
      ]
    ]
  ]
  sp-de_[
    +(de de SPS00 -)
    sn_[
      +grup-nom-ms_[
        +n-ms_[
          +(oro oro NCMS000 09629533:0.000000/10487505:0.000000)
        ]
      ]
    ]
  ]
  coord_[
    +(y y CC -)
  ]
```

3.4.5 Análisis Semántico: La opción sense es la encargada de encontrar el significado y desambiguar el sentido de las palabras.

Analyze –f es.cfg –outf sense <entrada.txt >salida.txt

```
Según según SPS00 0.986667 -
los el DA0MP0 0.97623 -
ecologistas ecologista NCCP000 0.491028 -
, , Fc 1 -
la el DA0FS0 0.972146 -
```

carga carga NCFS000 0.935897
09580808:0.000000/00627257:0.000000/02934345:0.000000/02388459:0
.000000/04491717:0.000000/09918904:0.000000/07792707:0.000000/00
458391:0.000000/02934556:0.000000
está estar VAIP3S0 0.996032 01811792:0.000000/01867419:0.000000
destinada destinar VMP00SF 1
00490924:0.000000/01524852:0.000000/00479979:0.000000/00479444:0
.000000
fundamentalmente fundamentalmente RG 1 -a a SPS00 0.99585 - la
el DA0FS0 0.972146 - alimentación alimentación NCFS000 1

4 AlchemyAPI

4.1 Intorduccion

Es un servicio Web de pago, pero puede consumirse una parte de él en forma libre, su uso puede ser directo desde la página Web ya que posee algunos demos, o bien descargando un código de ejemplo que puede ser ejecutado con la herramienta apache ANT

4.2 Prerrequisitos

- Maquina virtual JDK 1.5 o superior
- Entorno de desarrollo, por ejemplo NetBeans
- Descargar el código de prueba de AlchemyApI desde: http://www.alchemyapi.com/tools/
- Una Apikey, puede ser obtenida desde: http://www.alchemyapi.com/api/register.html

4.3 Instalación de la herramienta

En el entorno de desarrollo NetBeans se crea un nuevo archivo, sobre la naturaleza del proyecto se escoge “Java Project with Esisting Sources”

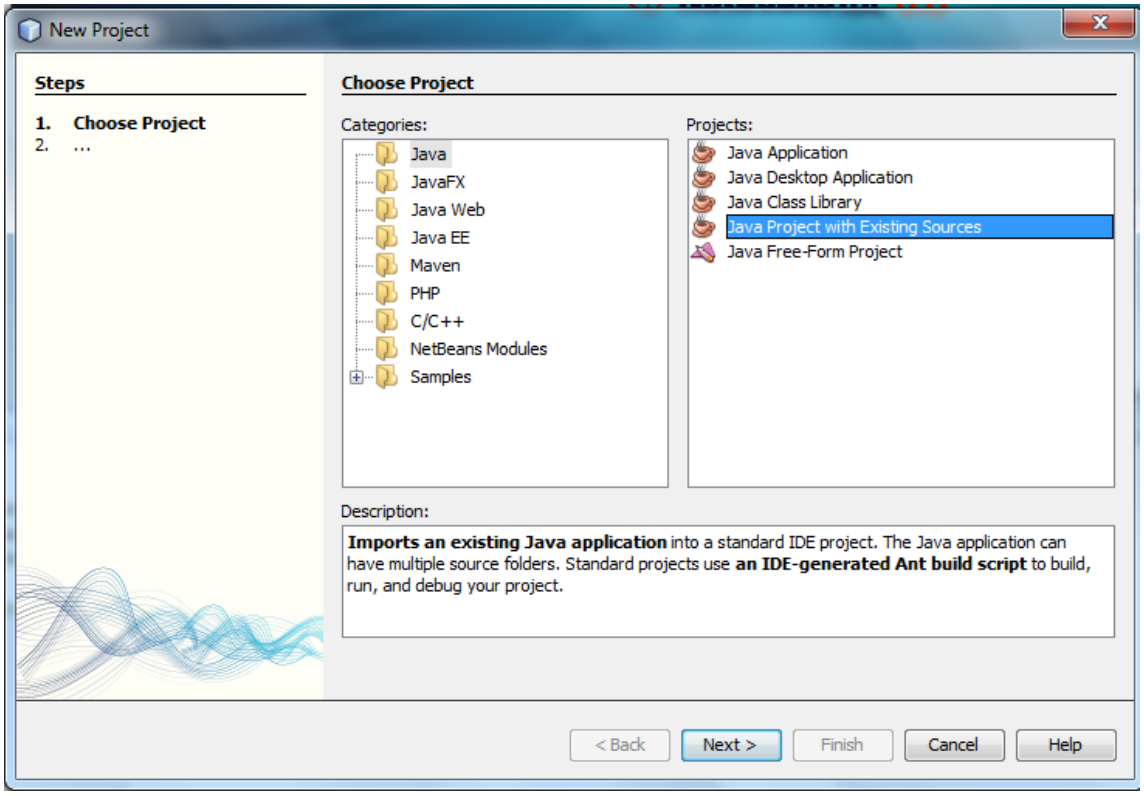


Figura A2.9: Crear un nuevo proyecto java con código fuente existente en NetBeans

En la ventana siguiente se debe ingresar el nombre del proyecto:

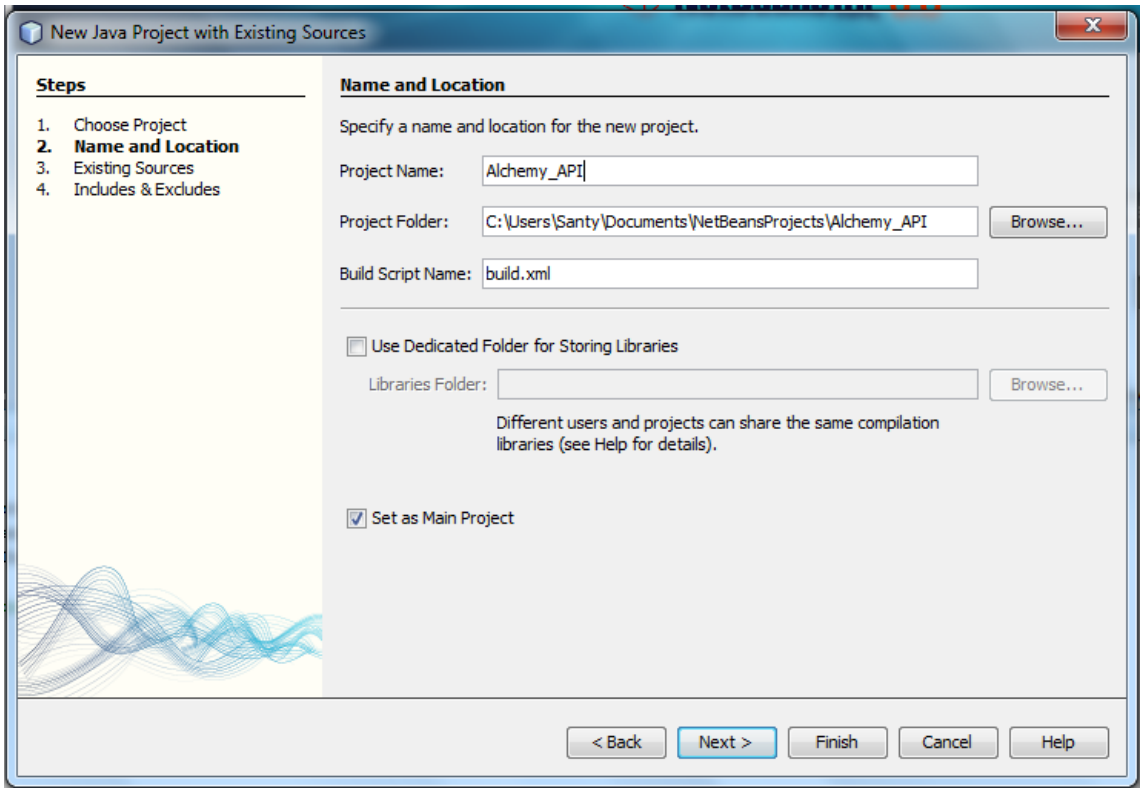


Figura A2.10: Configurar un nombre para el proyecto

A continuación se especifica la ruta donde se encuentra los archivos fuentes de AlchemiAPI y luego se da clic en “Finish”, así:

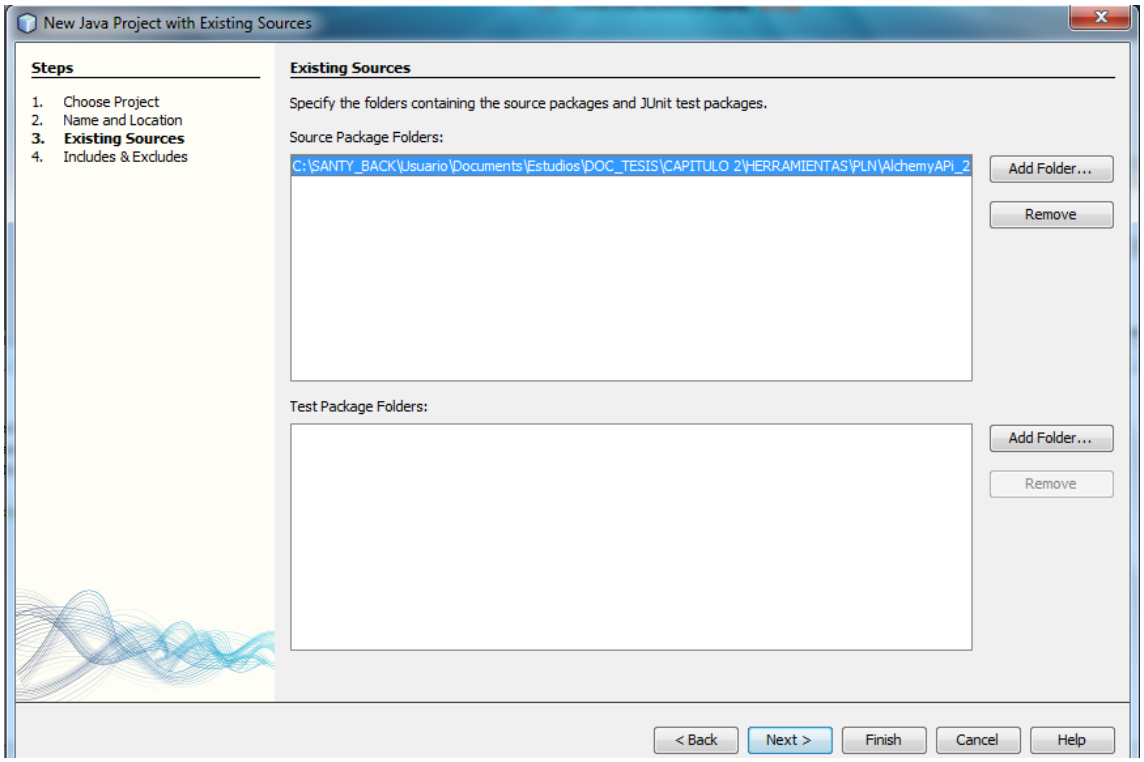


Figura A2.11: Ingresar ruta del código fuente en NetBeans

Con estos pasos se ha creado un nuevo proyecto java, se ha especificado un nombre y se ha importado el código fuente para su posterior uso.

4.4 Descripción

Para probar el reconocimiento de entidades con la herramienta se debe acceder al archivo java "EntityTest" contenido dentro del paquete "src.com.orchestr8.test", en este archivo se realizan las siguientes operaciones:

4.4.1 **Clase *AlchemyAPI***, esta clase recibe como parámetro la ruta completa de un archivo txt, que contiene la Apikey:

```
AlchemyAPI alchemyObj = AlchemyAPI.GetInstanceFromFile("api_key.txt");
```

4.4.1.1 Método *URLGetRankedNamedEntities:* Este método extrae todas entidades nombradas de una página web, recibe como parámetro un String con la URL de la página que se quiere analizar y devuelve los resultados en una variable de tipo Document, que luego puede ser leída mediante el método *"getStringFromDocument"*:

```
Document doc =
alchemyObj.URLGetRankedNamedEntities("http://www.techcrunch.com/");
System.out.println(getStringFromDocument(doc));
```

4.4.1.2* Método *TextGetRankedNamedEntities: Obtiene las entidades nombradas a partir de un texto contenido en un String, igual que el método anterior los resultados se guardan en una variable de tipo Document:

```
doc = alchemyObj.TextGetRankedNamedEntities(
"Hello there, my name is Bob Jones.  I live in the United States of America.
" +
"Where do you live, Fred?");
System.out.println(getStringFromDocument(doc));
```

4.4.1.3* Método *HTMLGetRankedNamedEntities: Extrae el conjunto de entidades ordenadas por ranking de un documento HTML, es similar al método "URLGetRankedNamedEntities" con la diferencia que trabaja con las páginas web guardadas en el equipo.

```
String htmlDoc = getFileContents("data/example.html");

// Extract a ranked list of named entities from a HTML document.
doc = alchemyObj.HTMLGetRankedNamedEntities(htmlDoc,
"http://www.test.com/");
System.out.println(getStringFromDocument(doc));
```

```
run:
<?xml version="1.0" encoding="UTF-8" standalone="no"?><results>
    <status>OK</status>
    <usage>By accessing AlchemyAPI or using information generated by AlchemyAPI,
    <url/>
    <language>spanish</language>
    <entities>
        <entity>
            <type>Country</type>
            <relevance>0.920333</relevance>
            <count>1</count>
            <text>Rusia</text>
        </entity>
    </entities>
</results>
BUILD SUCCESSFUL (total time: 4 seconds)
```

Figura 2.12: Resultados de la herramienta AlchemyAPI

4.5 Código

```
import org.xml.sax.SAXException;

import org.w3c.dom.Document;

import java.io.*;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.xpath.XPathExpressionException;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;


class EntityTest {

    public static void main(String[] args)

        throws IOException, SAXException,

               ParserConfigurationException, XPathExpressionException

    {

        // Create an AlchemyAPI object.
```

```
        AlchemyAPI alchemyObj =
AlchemyAPI.GetInstanceFromFile("api_key.txt");


        // Extract a ranked list of named entities for a web URL.
        Document doc =
alchemyObj.URLGetRankedNamedEntities("http://www.techcrunch.com/");
        System.out.println(getStringFromDocument(doc));


        // Extract a ranked list of named entities from a text string.
        doc = alchemyObj.TextGetRankedNamedEntities(
            "Hello there, my name is Bob Jones.  I live in the United
States of America.  " +
            "Where do you live, Fred?");
        System.out.println(getStringFromDocument(doc));


        // Load a HTML document to analyze.
        String htmlDoc = getFileContents("data/example.html");


        // Extract a ranked list of named entities from a HTML
document.
        doc = alchemyObj.HTMLGetRankedNamedEntities(htmlDoc,
"http://www.test.com/");
        System.out.println(getStringFromDocument(doc));
    }


    // utility function
    private static String getFileContents(String filename)
        throws IOException, FileNotFoundException
    {
        File file = new File(filename);
```

```
        StringBuilder contents = new StringBuilder();

        BufferedReader input = new BufferedReader(new
FileReader(file));

        try {
            String line = null;

            while ((line = input.readLine()) != null) {
                contents.append(line);
                contents.append(System.getProperty("line.separator"));
            }
        } finally {
            input.close();
        }

        return contents.toString();
    }

    // utility method
    private static String getStringFromDocument(Document doc) {
        try {
            DOMSource domSource = new DOMSource(doc);
            StringWriter writer = new StringWriter();
            StreamResult result = new StreamResult(writer);

            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer transformer = tf.newTransformer();
            transformer.transform(domSource, result);
```

```
            return writer.toString();
        } catch (TransformerException ex) {
            ex.printStackTrace();
            return null;
        }
    }
}
```

Anexo 3: Instalación de herramientas para EI

1. Apache UIMA

1.1. Introducción

Es un framework de arquitectura modular para el análisis de texto no estructurado, posee analizadores de texto y anotadores que pueden ser integrados con componentes personalizados, se lo utiliza como librería o directamente como herramienta. La instalación más sencilla es complemento del entorno de desarrollo Eclipse, desde el propio entorno se puede desarrollar nuevos componentes o ejecutar los ejemplos que vienen por defecto en la herramienta.

1.2. Prerrequisitos

Para la instalación de Apache UIMA como complemento de eclipse se necesita:

- Entorno de desarrollo eclipse: http://www.eclipse.org/
- Máquina virtual JDK 1.5 o superior
- Eclipse Modeling Framework , puede ser obtenido desde la página: http://www.eclipse.org/modeling/emf/
- Fuentes apache uima, pueden ser obtenidos desde: http://uima.apache.org/downloads.cgi

1.3. Instalación de la herramienta

*1.3.1.*Se accede al entorno de desarrollo, y se sitúa sobre el menú "*Help"* y luego se escoge la opción *"Install new Software"*

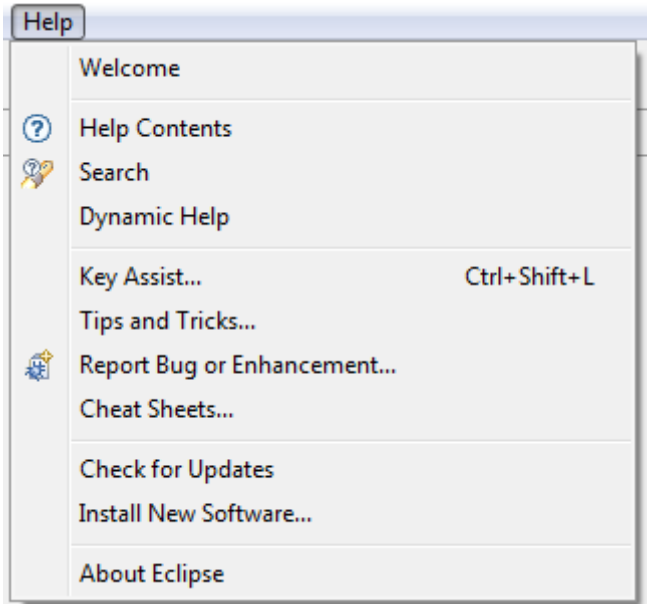


Figura A3.1: Instalar nuevos complementos en Eclipse

1.3.2.En la ventana que aparece se presiona sobre el botón "*Add*" y se introduce la página donde se encuentra los fuentes del complemento apache uima, el cual se encuentra en: http://www.apache.org/dist/incubator/uima/eclipse-update-site.

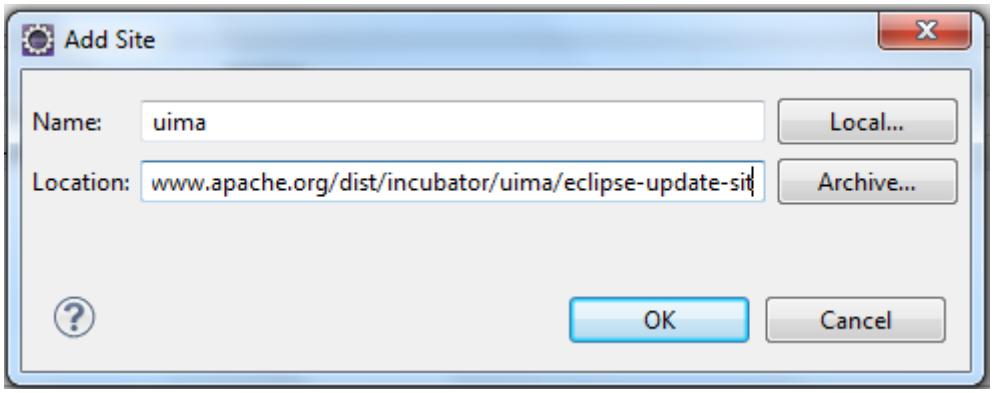


Figura A3.2: Agregar una nueva fuente de descarga de software

1.3.3. Luego se escoge la versión del complemento que se desea instalar y se presiona el botón *"Next"*:

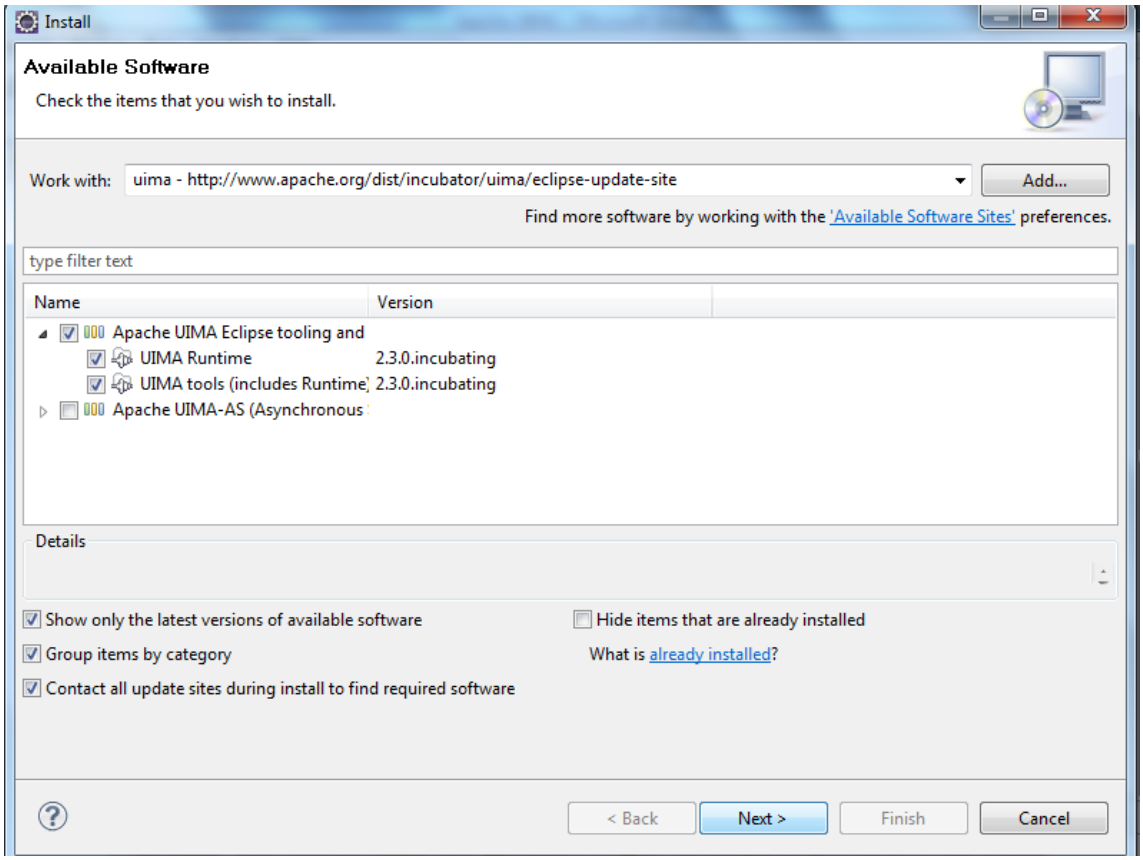


Figura A3.3: Seleccionar los complementos a instalar

1.3.4. Terminada la instalación se escoge la opción *"File"* y luego *"Restart"*, para que los nuevos complementos empiecen a funcionar en eclipse.

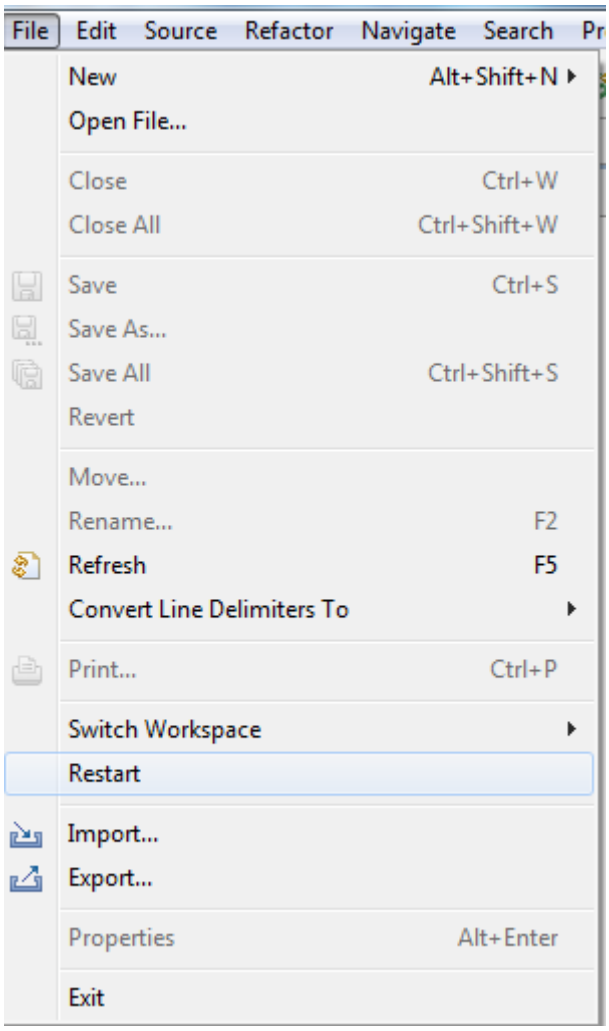


Figura A3.4: Reiniciar el entorno de desarrollo Eclipse

1.3.5. Se desempaqueta los fuentes de Apache UIMA en cualquier lugar del computador y se configura una variable de tipo class_path con el nombre UIMA_HOME dentro del entorno de desarrollo eclipse, esta variable contiene la ruta donde se encuentra los fuentes, en eclipse se selecciona el menú “Window” → “Preferences” → “Java” → “Build” “Path” → “Classpath Variables” se seleciona la opción “New” y se ingresa el nombre de la variable y la ruta de Apache UIMA:

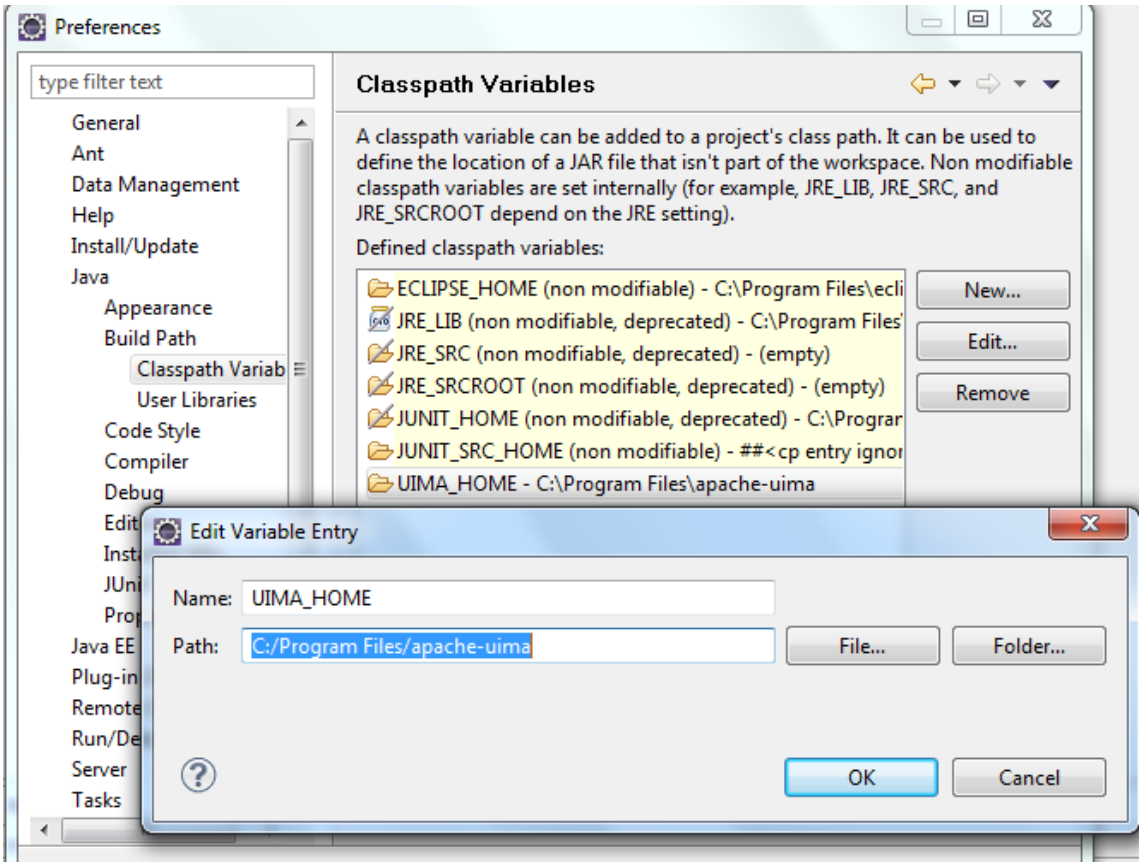


Figura A3.5: Creando variable de entorno UIMA_HOME

1.3.6. Para comenzar a utilizar la herramienta se puede cargar los ejemplos que vienen por defecto, para esto se selecciona el menú “File” luego “Import” en la ventana emergente se escoge la opción “General” y luego “Existing Project into Workspace” se da clic en “NEXT”.

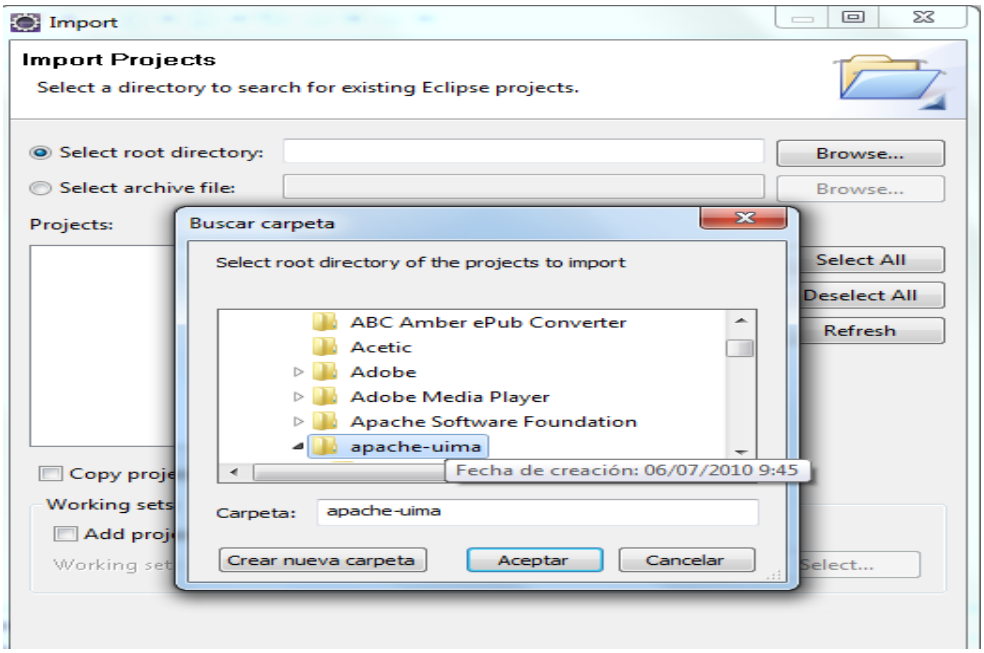


Figura A3.6: Configurando la ruta de los fuentes de Apache UIMA

En la siguiente ventana se ingresa la ruta donde se encuentra los fuentes y se da clic en "Finish", esta operación creará un nuevo proyecto con el nombre *uimaj-examples*

1.4. Descripción

1.4.1. Seleccionar el proyecto recientemente creado, se da clic derecho y se escoge la opción *"RUN"* lo cual desplegará una nueva ventana con las opciones de análisis que ofrece el framework:

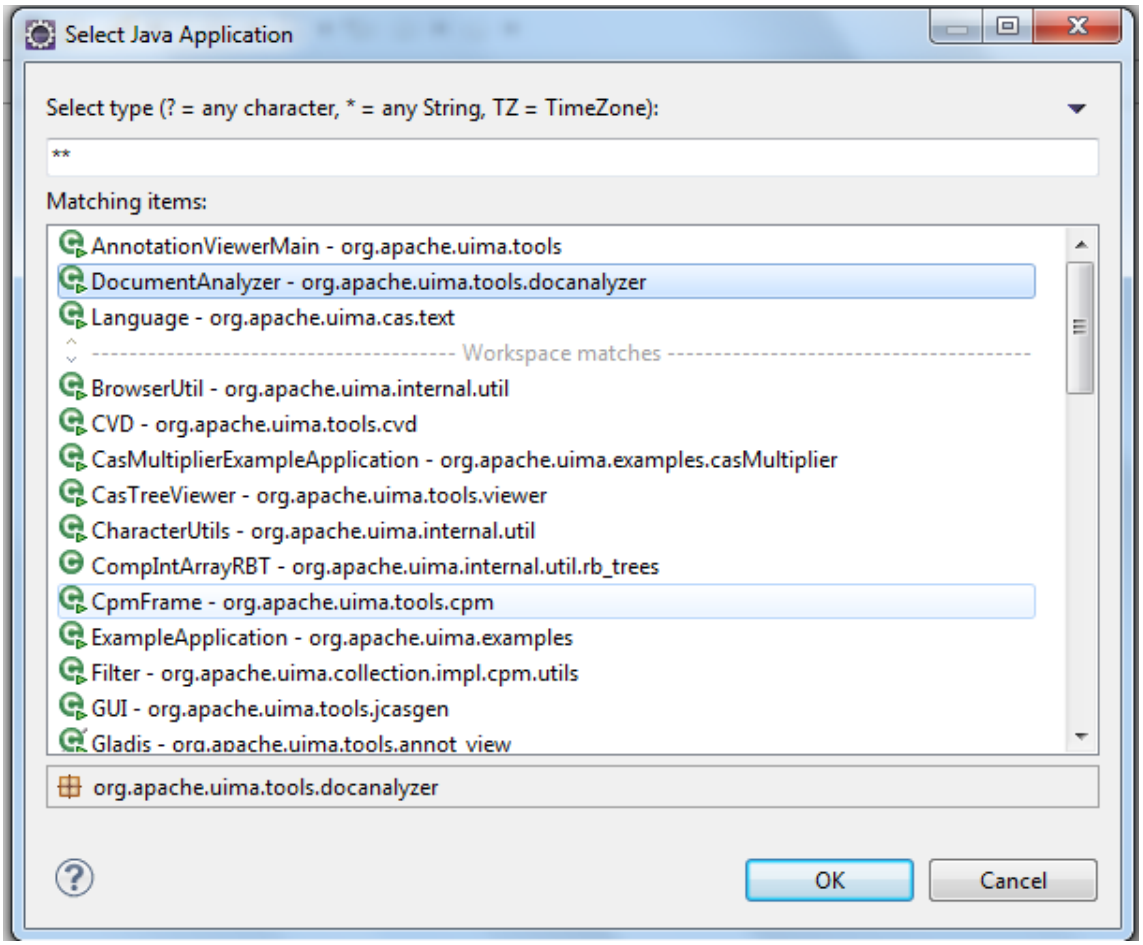


Figura A3.7: Ventana principal de ejemplos de Apache UIMA

1.4.2. Para el análisis de documentos se escoge la opción "DocumentAnalyzer" :

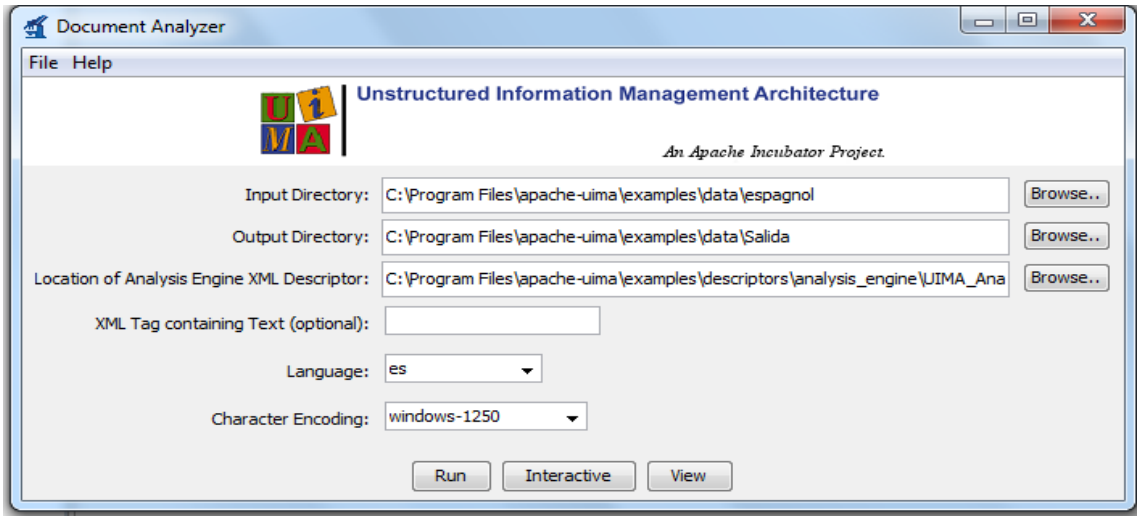


Figura A3.8: Ventana de configuración del Analizador de Documentos

1.4.3. En esta nueva ventana hay que ingresar los parámetros de configuración siguientes:

- **Input Directory:** Directorio con los documentos para el análisis
- **Output Directory:** Directorio donde se va a presentar los resultados
- **Location of Analysis XMl Descriptor:** El descriptor que se utilizará para el análisis; se puede escoger de varios que vienen por defecto.
- **Language:** EL lenguaje en que se va a realizar el análisis
- **Character Encoding:** El conjunto de caracteres que se va a utilizar, para el reconocimiento de tildes y demás caracteres del español se escoge el conjunto de caracteres :"Window 1250"

1.4.4. La opción "Interactive" permite realizar el análisis de varios documentos a la vez, con las configuraciones anteriormente mencionadas:

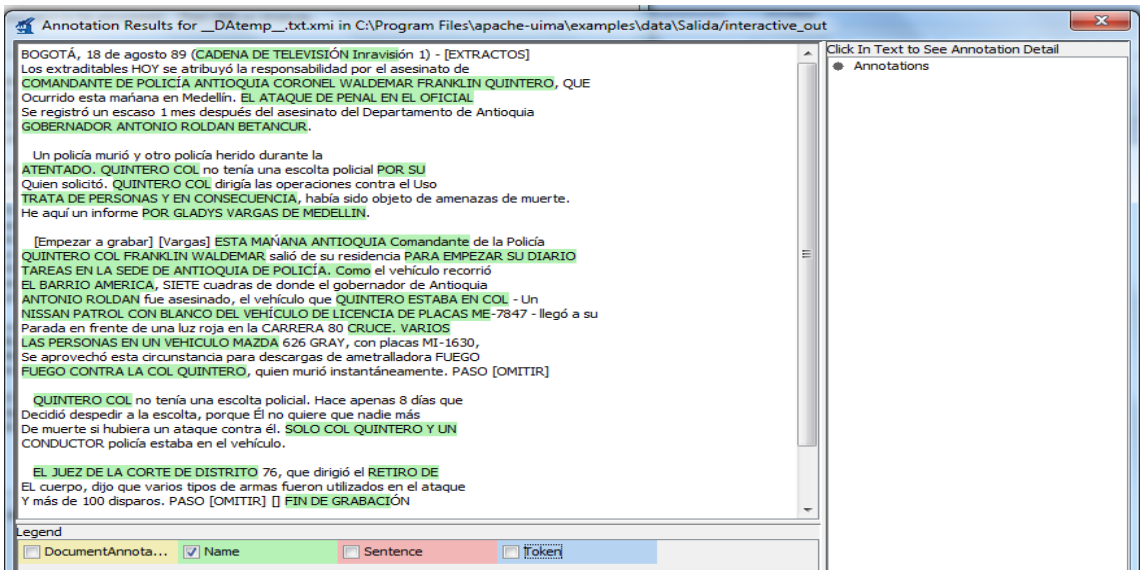


Figura A3.9: Resultados del Analizador de Documentos

2. GATE

2.1. Introducción

Es una herramienta que incluye un conjunto de métodos para EI y PLN en muchos lenguajes, es desarrollada y mantenida por la Universidad de Sheffield desde 1995. Esta escrita complemente en java y permite el análisis del texto utilizando un sistema llamado ANNIE (Nearly-New Information Extraction System), que es un compendio de algoritmos de estado finito, además incluye una herramienta llamada JAPE[58] la cual realiza la transducción de estados finitos en anotaciones basadas en expresiones regulares.

2.2. Pre-requisitos

Los requisitos para utilizar GATE se muestran a continuación:

- Máquina virtual JDK 1.5 o superior
- Archivos bin de GATE, que se obtiene desde: http://gate.ac.uk/download/

2.3. Instalación

GATE puede ser obtenido de diversas formas, desde fuentes java hasta aplicaciones listar para usar en Windows en este anexo se detalla el uso de la herramienta con la segunda opción:

2.3.1. Verificar que se cuenta con java instalado en la máquina

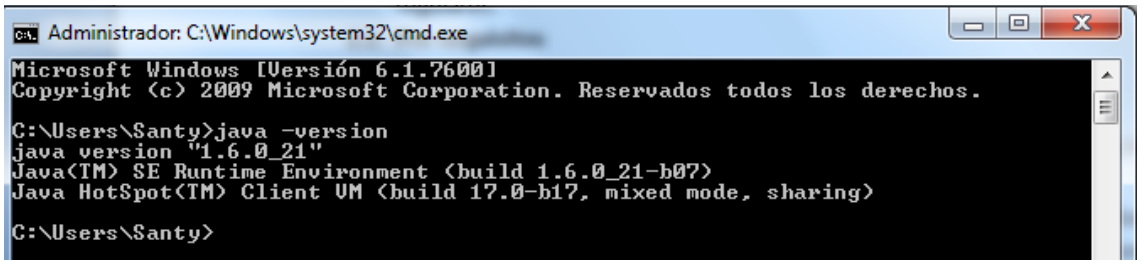


Figura A3.10: Verificar la versión de java instalada

2.3.2. Se crea la variable de JAVA_HOME en caso de no estar configurada previamente, para esto se usa las variables de entorno del sistema.

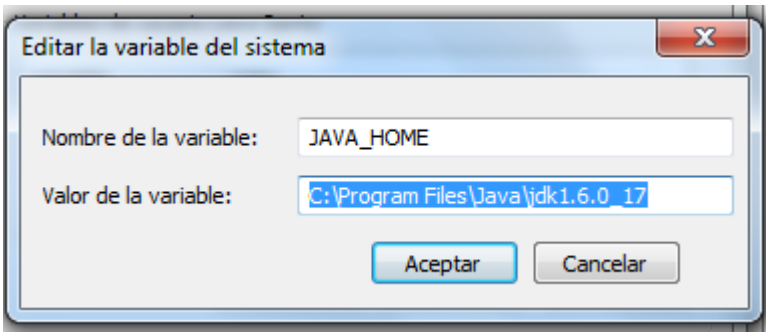


Figura A3.11: Configurar variables de entorno

2.3.3. Ejecutar la aplicación GATE y aparecerá la siguiente interfaz

[58] http://gate.ac.uk/sale/tao/splitch8.html#x12-2080008

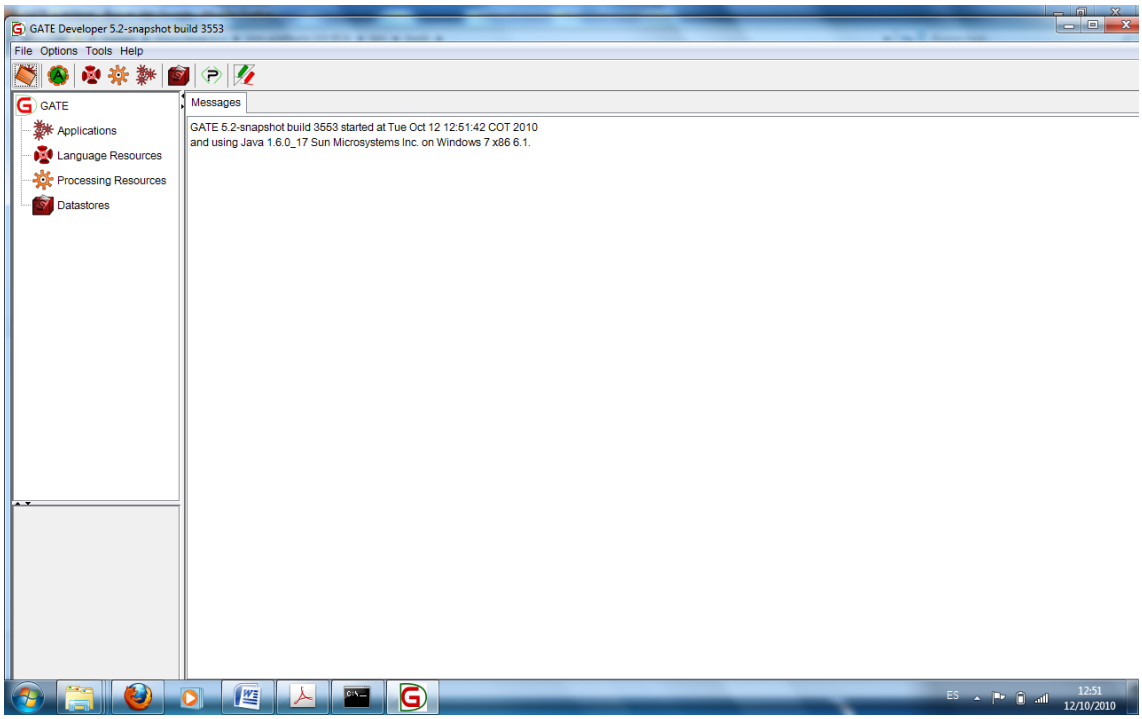


Figura A3.12: Interfaz por defecto de GATE

2.4. Descripción

2.4.1. Seleccionar la opción "*Language Resource*" y dar clic derecho y seleccionar New y luego *Document*

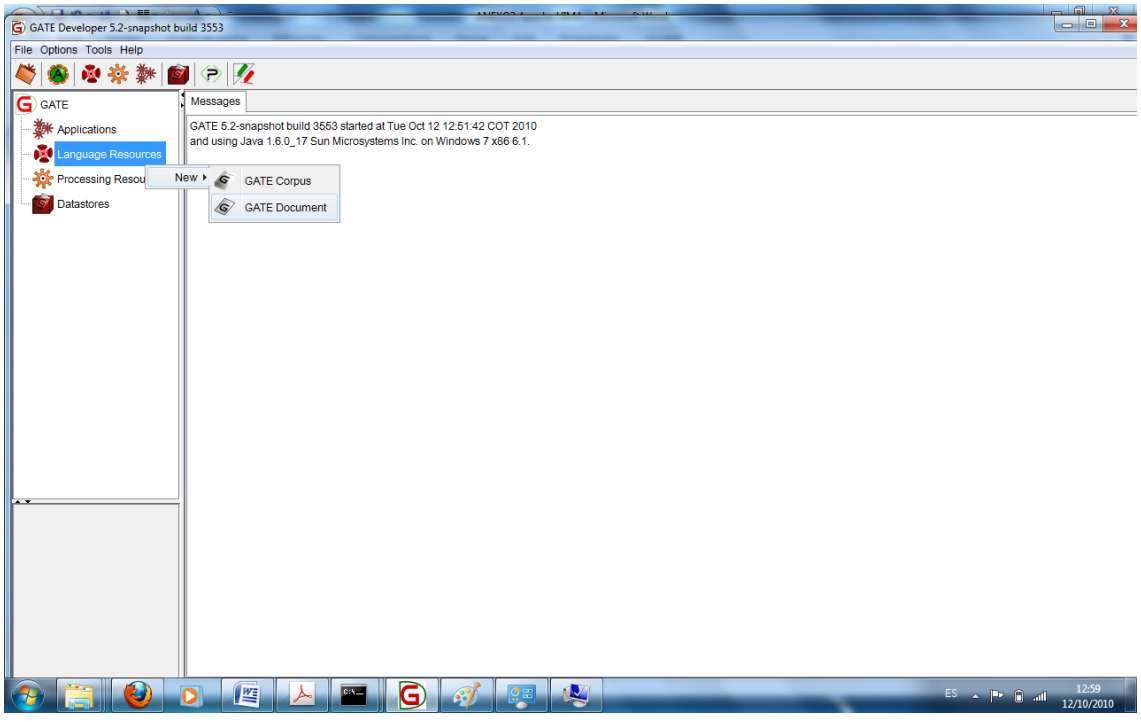


Figura A3.13: Crear un nuevo documento GATE

2.4.2.En la ventana emergente se ingresa los parámetros necesarios como son Nombre del documentos y contenido:

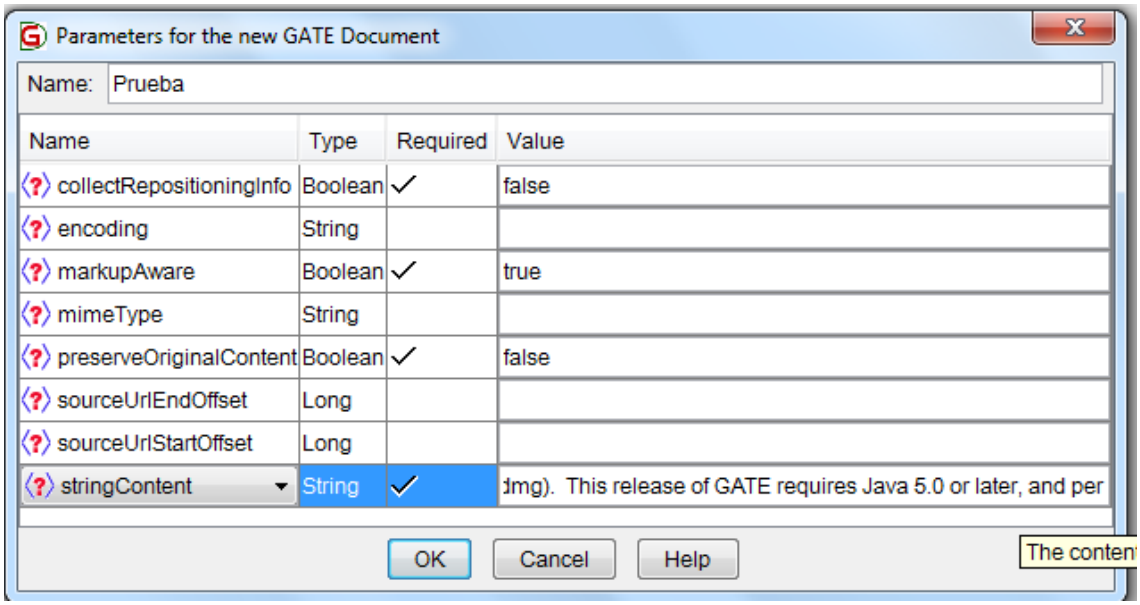


Figura A3.14: Configurar opciones del documento

2.4.3.Luego se selecciona la opción "Anotation set" para ir navegando sobe las anotaciones obtenidas

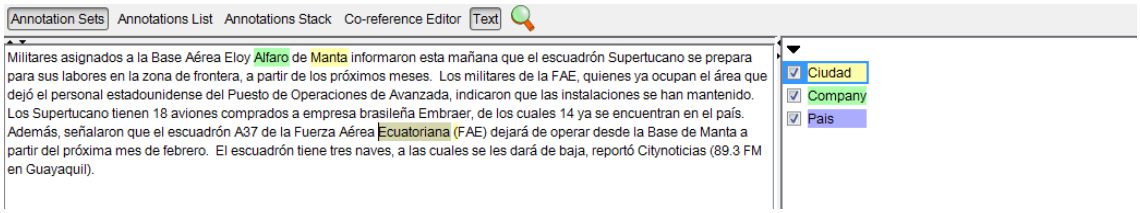


Figura A3.15: Anotaciones obtenidas

Anexo 4: Instalación de Kim

4 Kim-Plataform

4.1 Introduccción

Está formado por un conjunto de ontologías propias e independientes del lenguaje (español e inglés) que son una representación del Conocimiento del Mundo Real (Common World Knowledge), a través de su anotador semántico basado en GATE instancia cada una de las clases de estas ontologías con las entidades reconocidas en los documentos (ver Figura 3.6).

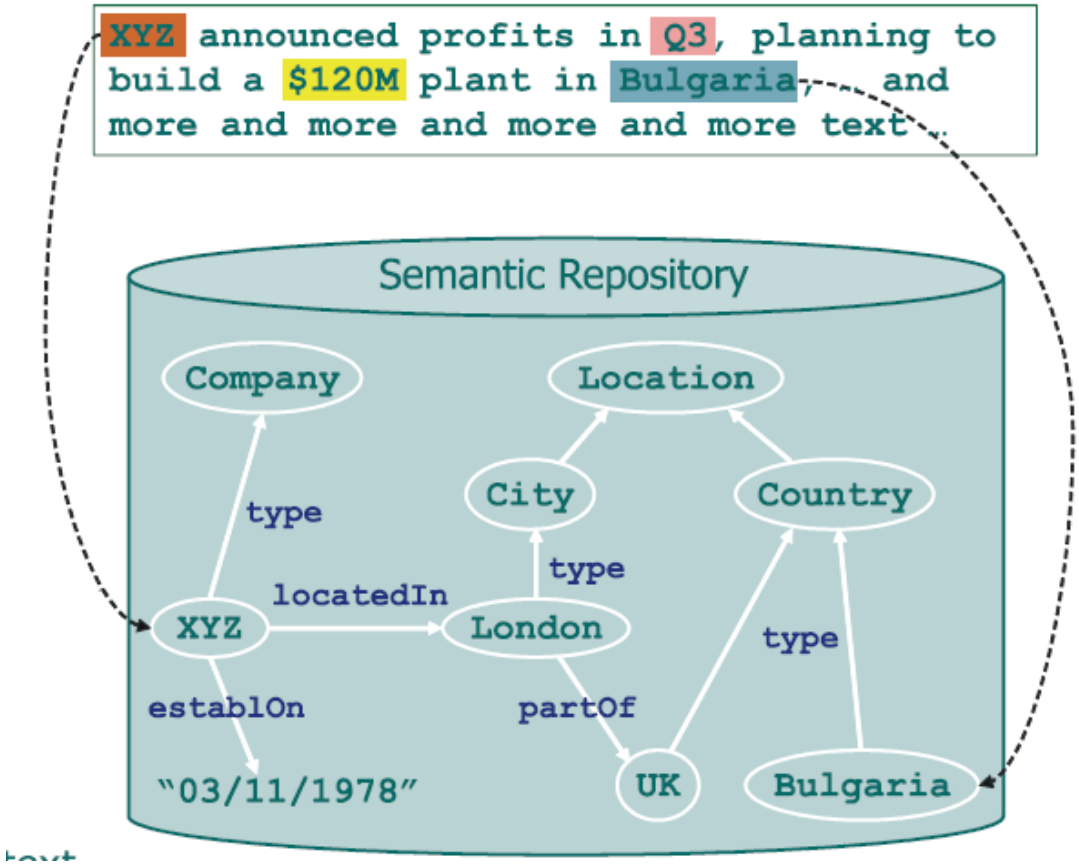


Figura A4.1: Funcionamiento de la anotación semántica y instanciación de las ontologías[59]

Además integra varias funcionalidades entre las cuales se cuenta Minería de texto, anotación semántica, co-ocurrencia, detección de relaciones y una suite completa para la RI, la cual permite combinar consultas sintácticas como semánticas, asimismo permite realizar restricciones de tipo coincidencia

[59] http://www.ontotext.com/kim/KIMPlatform.pdf

(igual, contiene, comienza por, etc.) o intervalos de tiempo. Estas funcionalidades son proporcionadas gracias a las tecnologías que integra KIM entre ellas se cuenta al buscador Sintáctico Lucene el cual permite hacer consultas por la coincidencia de palabras claves, también ya se ha mencionado que utiliza GATE para anotación semántica, incluye el repositorio semántico OWLIM[60] para la administración de RDF's, además trabaja conjuntamente con una implementación de Sesame[61] para el almacenamiento y consulta de tripletas RDF, una visión gráfica de la arquitectura de Kim se presenta a continuación:

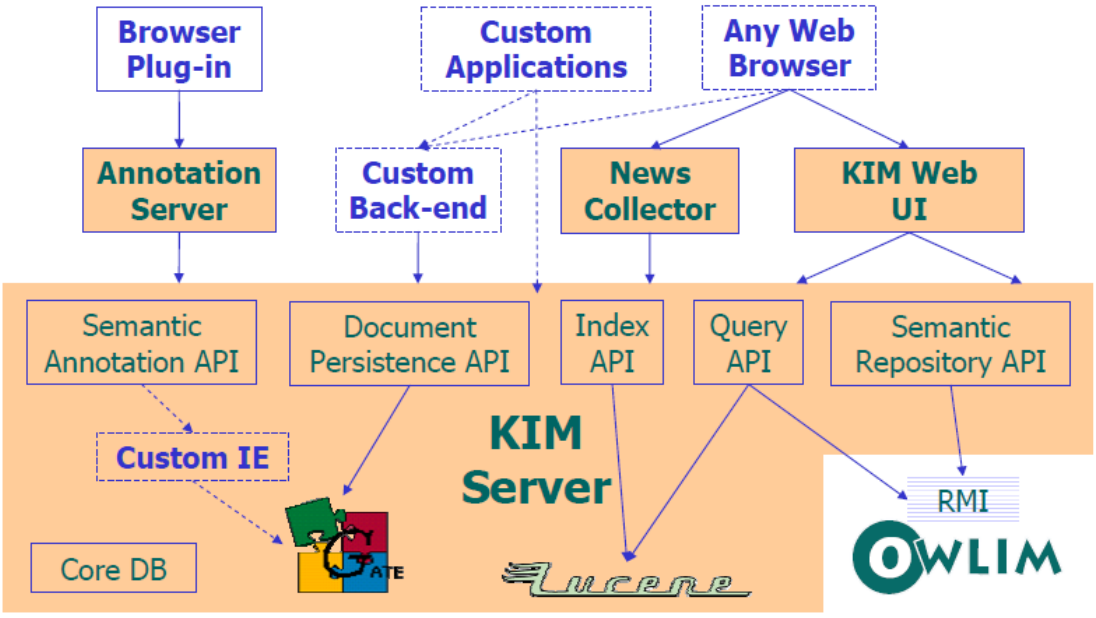


Figura A4.2: Arquitectura Kim

4.2 Pre-requisitos

Esta herramienta por ser contener un conjunto de herramientas tiene requerimientos tanto de hardware y software, lo cuales son los siguientes:

- Procesador Intel Pentium IV 1GHz como mínimo o procesador compatible (recomendado 2,5GHz)
- 1.5 GB de RAM como mínimo (recomendado 2 GB)
- 500 MB de espacio libre en disco
- Máquina virtual JDK 1.5 o superior
- Contenedor de Servlets (Tomcat 5 – 6, o similares)

[60] http://ontotext.com/owlim/
[61] http://www.openrdf.org/doc/sesame2/2.3.2/users/userguide.html#chapter-sesame2-whats-new

- Fuentes de Kim-platform, los mismo que son obtenidos desde: http://www.ontotext.com/kim/
- Fuentes de Apache Axis, los mismos están disponibles en: http://ws.apache.org/axis/

4.3 Instalación de la herramienta

4.3.1 Extraer las fuentes de Kim en cualquier ubicación del computador, preferible en el directorio root.

4.3.2 Editar el archivo */etc/profile* y configurar la variable de entorno *JAVA_HOME* con la ubicación donde se encuentra instalado java

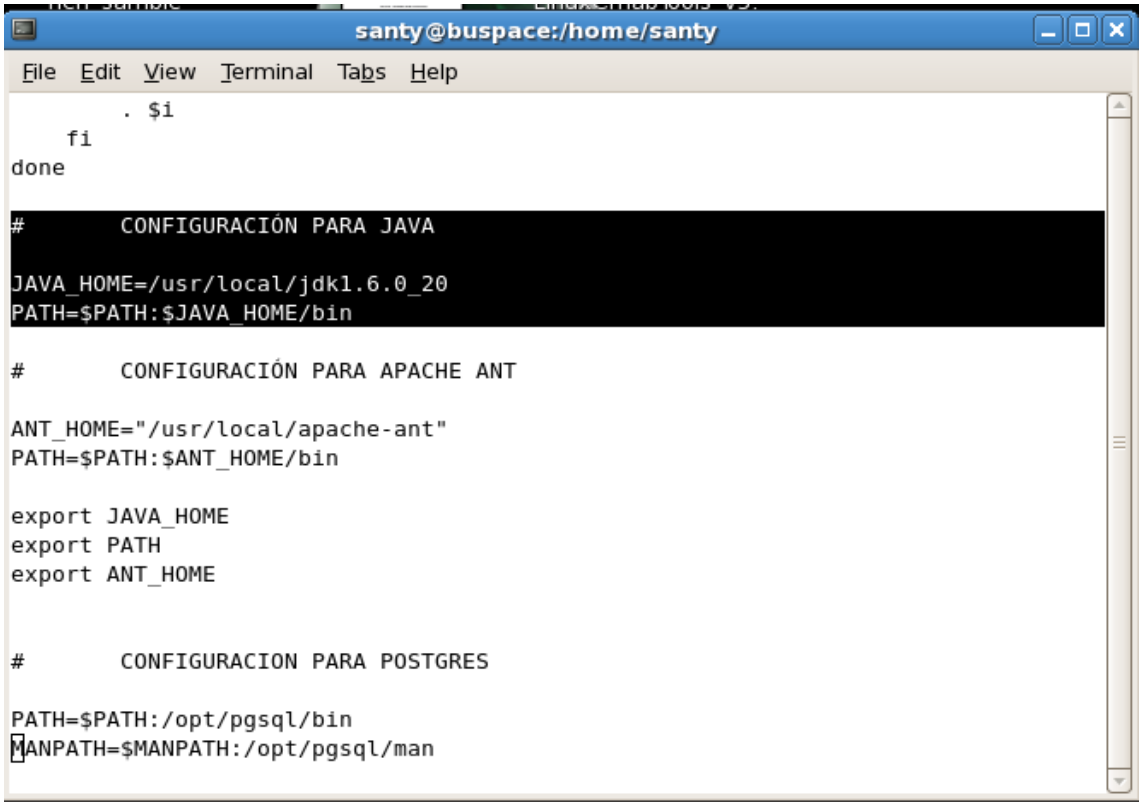


Figura A4.3: Configuración de la Variable de Entorno JAVA_HOME

4.3.3 Ingresar al directorio donde se encuentran las fuentes de Kim (de aquí en adelante <KIM_HOME>) y ubicarse bajo el directorio *"/bin/config "* y editar en el archivo "config" la línea "export JAVA_HOME =" con la ubicación del directorio de instalación de java:

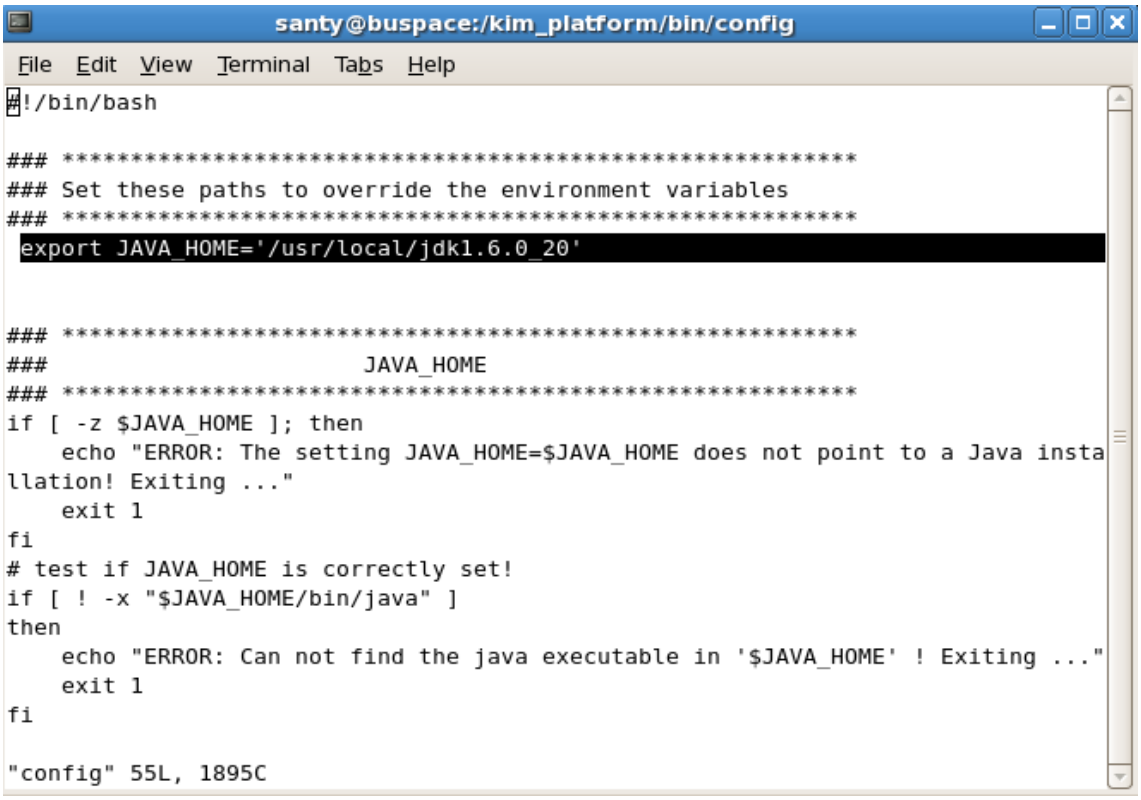


Figura A4.4: Configuración de JAVA_HOME en los archivos de configuración de Kim

4.3.4 Cambiar los permisos de ejecución a todos los scripts que se encuentran bajo el directorio *"bin"* , para lo cual se ejecuta el siguiente comando como súper usuario:

```
chmod -R +x KIM/bin/*.sh
```

4.3.5 Extraer las fuentes de Apache Axis en cualquier directorio del computador.

4.3.6 Modificar el archivo /etc/profile e ingresar las variables de entorno *"AXIS_HOME"* con la ubicación de los fuentes de Axis, *"AXIS_LIB"* con la ubicación del directorio de librerías de Axis que es la carpeta llamada *"lib"* y una variable llamada *"AXISCLASSPATH"* en donde se listan todas las librerías que se encuentran en el directorio *"lib"*:

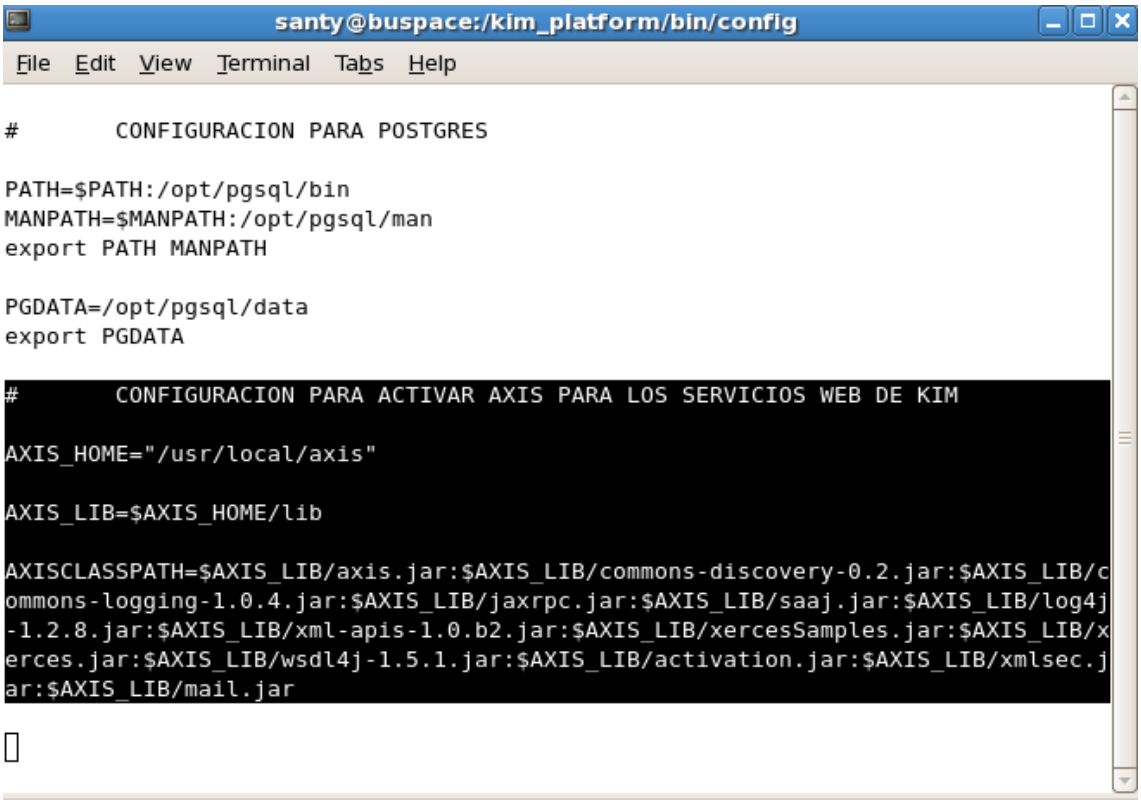


Figura A4.5: Configuración de las variables de entorno de AXIS

4.3.7 Ir al directorio "<KIM_HOME/KIM Clients/>" y copiar todos los archivos .war al directorio *"webapps"* que se encuentra bajo el directorio de instalación de Tomcat.

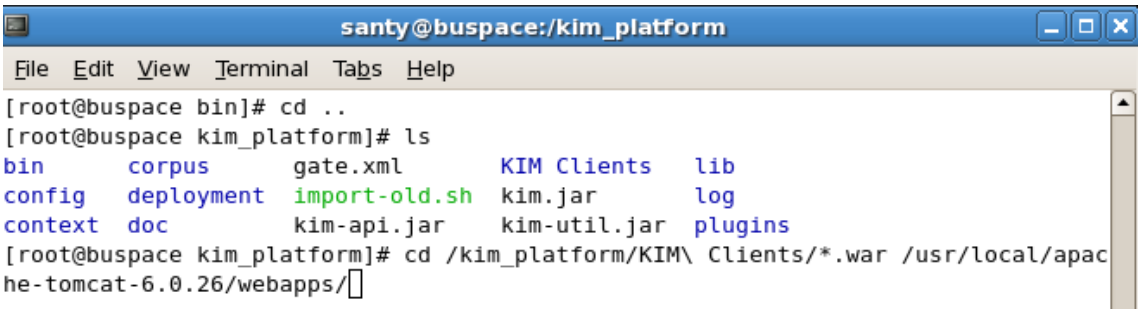


Figura A4.6: Transferir los clientes Web de Kim

4.3.8 Ubicarse sobre el directorio "$KIM_HOME/bin/" y ejecutar el script de inicio de kim:

./kim start

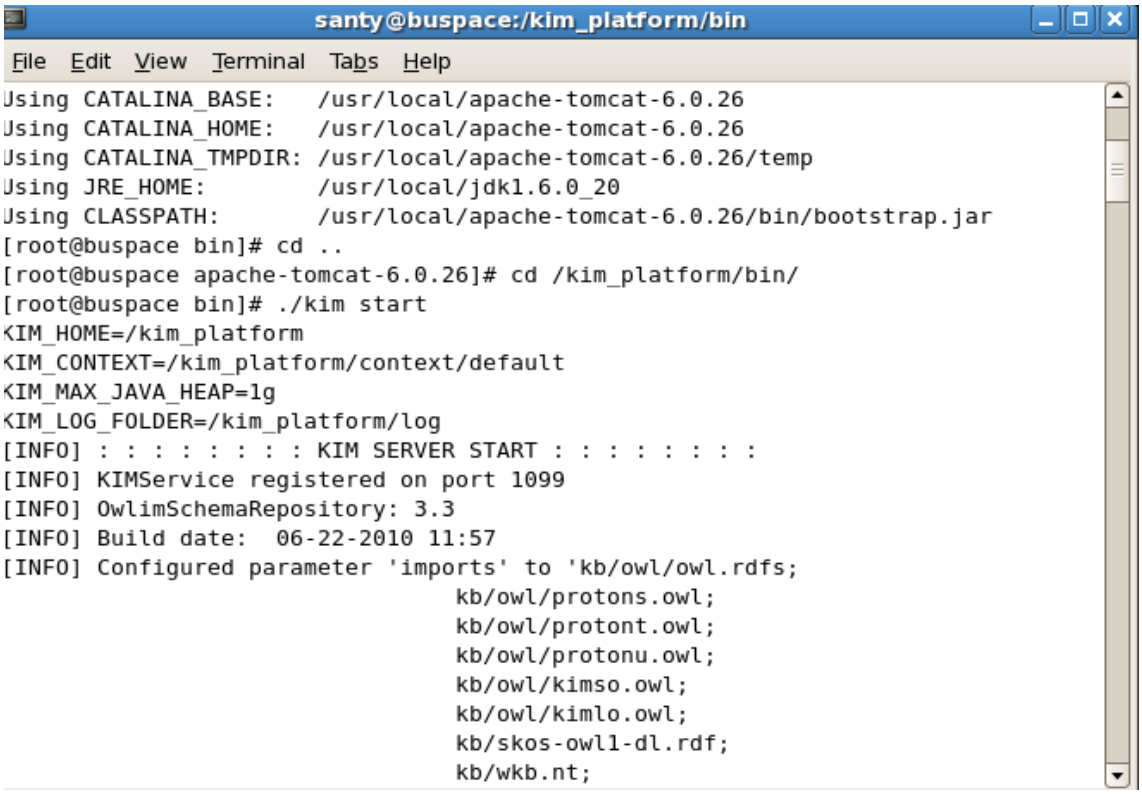


Figura A4.7: Inicio del servicio de Kim

4.3.9 Iniciar el servidor Tomcat y digitar el navegador: *http://localhost:8080/KIM*, en otra ventana del navegador digitamos *http://localhost:8080/kim-ws-api* si las páginas se despliegan sin ningún error, es porque se ha instalado satisfactoriamente Kim.

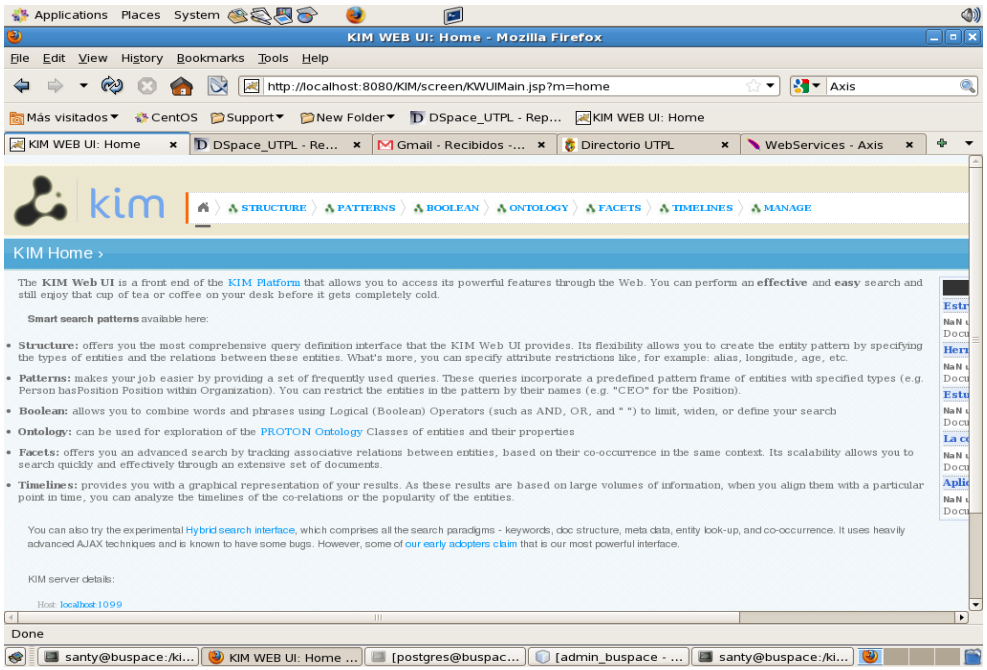


Figura A4.8: Pantalla de inicio de Kim

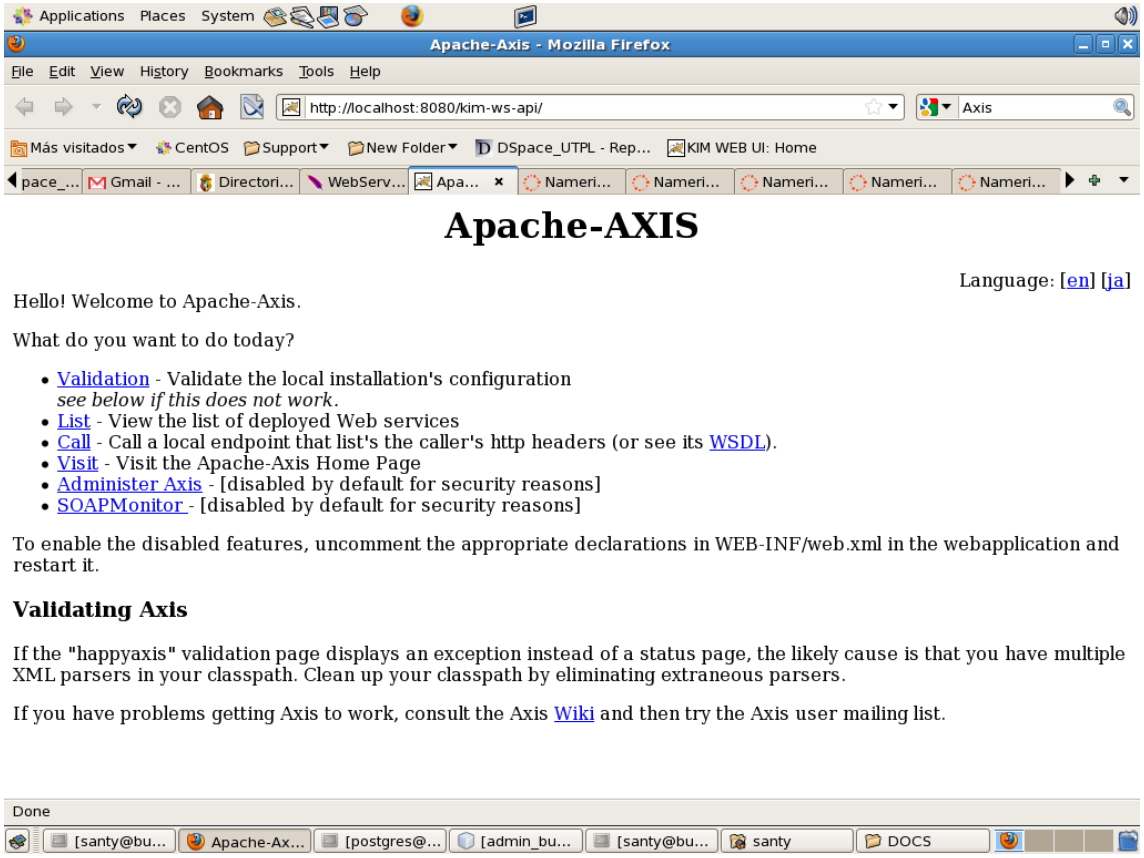


Figura A4.9: Pantalla de inicio servicios web de Kim

4.3.10 Abrir otra ventada de comando y ubicarse sobre el directorio "<KIM_HOME>/bin" y ejecutamos el script para parar el servicio de Kim.

4.3.11 Ahora se va realizar algunas configuraciones al servicio Kim: En primer lugar editamos el archivo "*config*" ubicado en el directorio: "<KIM_HOME>/bin/config" y se modifica la memoria que utilizará Kim para realizar las anotaciones.

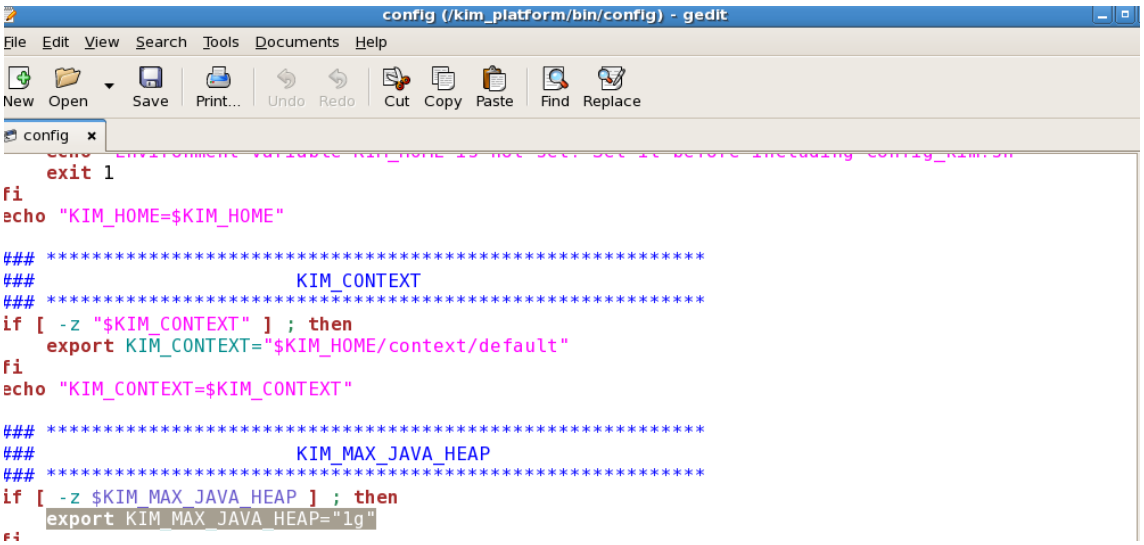


Figura A4.10: Configuraciones de memoria para Kim

4.3.12 Luego se configura las opciones del repositorio, el archivo de configuración se encuentra en: "<KIM_HOME>/config/document.repository.properties" y se modifican las siguientes líneas:

```
com.ontotext.kim.KIMConstants.DOCUMENT_REPOSITORY_TYPE = lucene
com.ontotext.kim.KIMConstants.CORE_INDEX_ADDON = rdfranked
com.ontotext.kim.KIMConstants.LUCENE_MAX_SEQUENTIAL_OPERATIONS = 50
com.ontotext.kim.KIMConstants.LUCENE_MAX_BOOLEAN_CLAUSES = 1000000
com.ontotext.kim.KIMConstants.LUCENE_MAX_RESULT_COUNT = 100
com.ontotext.kim.FileStore.Serialization=gate-xml
```

4.3.13 Finalmente se configura el puerto y dirección IP que va a recibir las conexiones RMI, esto se logra modificando el archivo <KIM_HOME>/config/install.properties y se modifican los siguientes variables:

```
com.ontotext.kim.KIMConstants.RMI_HOST= localhost  o alguna IP
com.ontotext.kim.KIMConstants.RMI_PORT=1099 u otro puerto
```

Anexo 5: Visión del Sistema (DEV-VIS)

BUSPACE

[Buscador Semántico para la recuperación de Objetos de Aprendizaje del Dpspce]

Posicionamiento del Producto

Definición del Problema

El problema de	*El repositorio de la universidad actualmente contiene alrededor de 3637 OA, los cuales están distribuidos en las diferentes áreas de estudio, cuenta con un buscador cuyo funcionamiento se basa en la correspondencia de palabras claves lo que dificulta el proceso de obtener información relevante, también posee una opción de búsqueda avanzada; esta permite realizar consultas más específicas, sin embargo esta elección solo admite filtrar la información por tres campos a la vez.* *Adicionalmente el sistema es incapaz de realizar consultas complejas ya que en el sistema los OA's no poseen ninguna estructuración., además los metadatos ingresados contienen información general sobre su contenido.*
Afecta a	• *Profesores y alumnos de la universidad* • *Administradores de repositorio* • *Usuarios externos interesados en OA del repositorio*
Cuyo impacto es	• *Pérdida de tiempo y recursos para obtener datos pertinentes.* • *Búsqueda avanzada es compleja para usuarios con conocimientos básicos* • *Los metadatos por defecto del repositorio no describen el contenido de los OA.* • *Errores en los resultados por ausencia de procesamiento de las consultas semánticamente*
Una solución exitosa es	• *Una buena solución debe permitir acceso a la información de forma rápida y precisa. Ésta debe pasar por un proceso de estructuración previo para facilitar el manejo de los datos. La solución a construirse debe coexistir y*

	complementarse con el proceso de subir OA que se lleva actualmente

Posicionamiento del Producto

Para	• *Profesores de la UTPL.* • *Estudiantes de la UTPL.* • *Usuarios externos.*
Quién(es)	• *Requieren un sistema de búsqueda para obtener la información del repositorio.* • *Suben Objetos de aprendizaje al repositorio*
El (Buspace)	*Buscadores Semánticos.*
Que	• *Permite extraer información relevante (OA).* • *Procesar las consultas en lenguaje natural.* • *Permitir a los usuarios acceder a los OA de formas más eficiente y rápida.*
A Diferencia	• *El sistema actual solo se basa en búsquedas de palabras claves.* • *Los metadatos son extraídos manualmente y reflejan el contenido del OA de forma general.* • *Los archivos no se actualizan y algunos están repetidos.* • *En las consultas no se tiene encuentra el sentido de las palabras*
Esta Aplicación	• *Proporciona consultas en lenguaje natural.* • *Ofrece la capacidad de buscar en base a conceptos.* • *Permite extraer información relevante del contenido del OA automáticamente.* • *Permite estructurar la información de los OA*

Descripción breve de stakeholders.

Nombre	***Descripción***	***Responsabilidades***
Administrador del repositorio	*Se encargan de gestionar el contenido del repositorio.*	• Subir nuevos OA al repositorio. • Revisar los metadatos de los OA. • Categorizar el contenido.
Usuario	*Persona (Profesores, Alumnos y Usuarios externos) encargada de subir nuevos OA en el repositorio o en su defecto consulta los OA del repositorio para dictar clases.*	• Registrar datos personales en el sistema. • Filtrar información
Estudiante	*Profesionales en formación que publican sus trabajos en el repositorio, además de consultar información en el repositorio*	• Registrar datos personales en el sistema. • Filtrar información
Usuarios externos	*Usuarios ocasionales que acceden al repositorio con fines investigativos o publican su propio contenido.*	• Utilizar el buscador • Registrar datos personales en el sistema.

Descripción breve de Usuarios

Nombre	***Descripción***	***Afectado al que representa***
Administrador del repositorio	*Se encargan de gestionar el contenido del repositorio.*	Administrador del repositorio.
Profesor	*Persona encargada de subir nuevos OA en el repositorio o en su defecto consulta los OA del repositorio para dictar clases.*	Profesor
Estudiante	*Profesionales en formación que publican sus trabajos en el repositorio, además de consultar información en el repositorio*	Estudiante
Usuarios externos	*Usuarios ocasionales que acceden al repositorio con fines investigativos o publican su propio contenido.*	*Usuarios externos*

Entorno de usuarios

Actualmente subir OA al repositorio lo puede realizar cualquier persona que haya registrado sus datos en el sistema previamente, también se sube contenido desde dos departamentos de la Universidad. La unidad virtualización se encarga de administrar y dar soporte al sistema, además de subir video conferencias. EL departamento de Procesos metodológicos sube tesis generadas en la universidad. En los dos departamentos estas actividades son llevadas por estudiantes de Gestión Productiva, cada proceso de subir un nuevo OA tiene una duración de tres a cinco minutos y se siguen los pasos de la plataforma. En el departamento de procesos metodológicos se exportan las tesis a formato pdf y se integran metadatos descriptivos sobre el documento los cuales son replicados en el repositorio al momento de subirlos, la figura siguiente representa las fuentes de información del Repositorio:

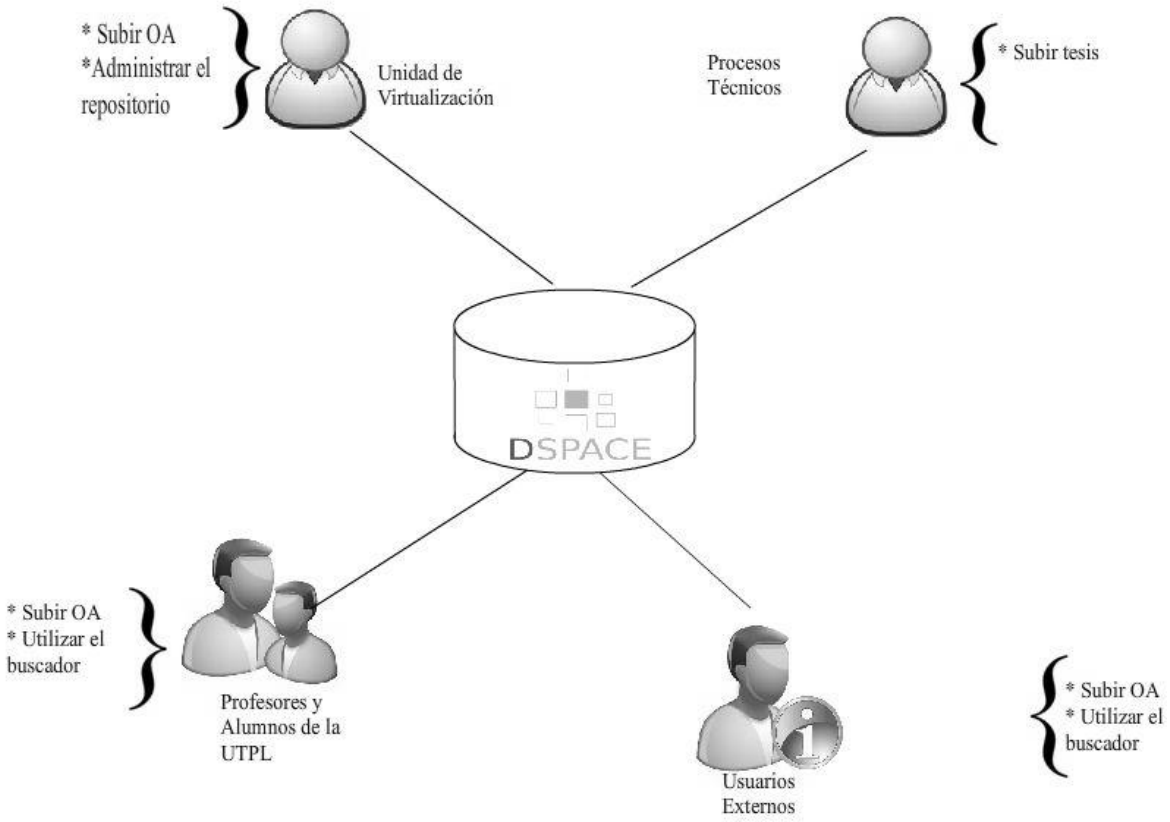
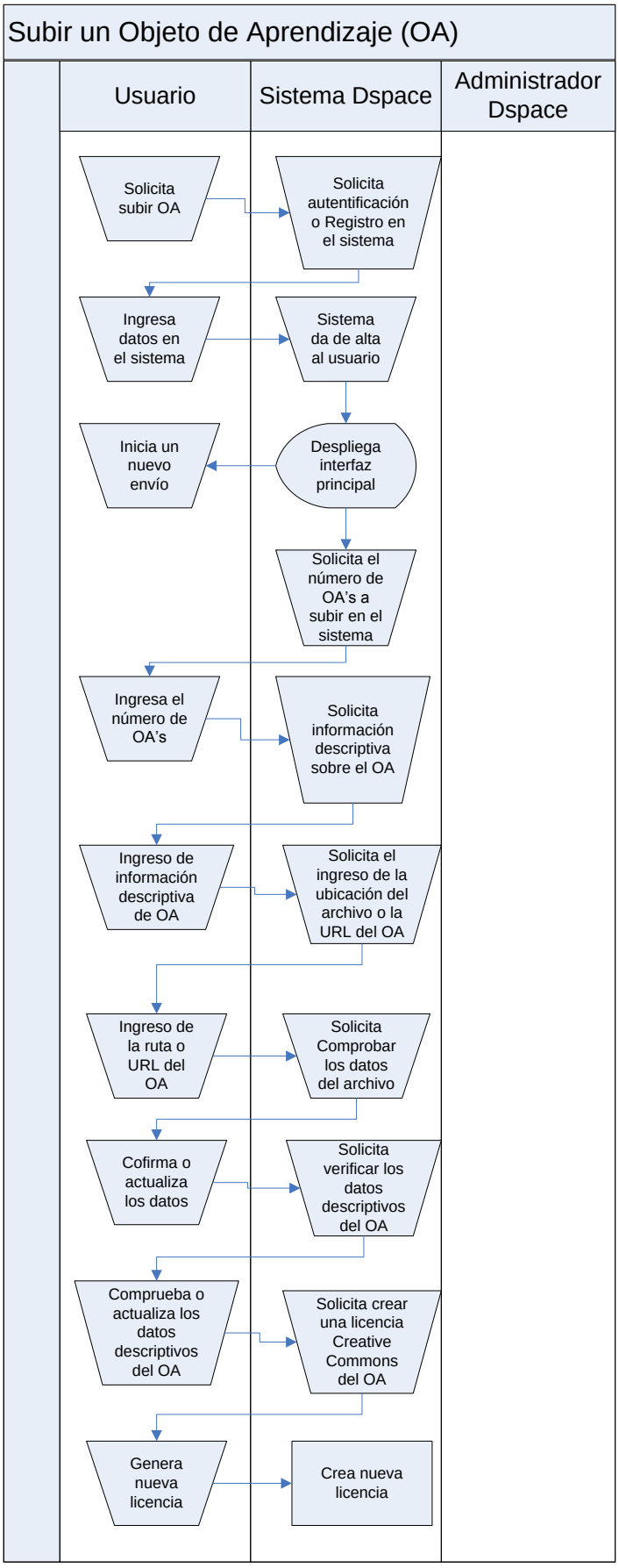


Figura A5.1: Fuentes de información de repositorio

Proceso actual de subir OA al repositorio:


Subir un Objeto de Aprendizaje (OA)
Usuario
Sistema Dspace
Administrador Dspace
Solicita subir OA
Solicita autentificación o Registro en el sistema
Ingresa datos en el sistema
Sistema da de alta al usuario
Inicia un nuevo envío
Despliega interfaz principal
Solicita el número de OA's a subir en el sistema
Ingresa el número de OA's
Solicita información descriptiva sobre el OA
Ingreso de información descriptiva de OA
Solicita el ingreso de la ubicación del archivo o la URL del OA
Ingreso de la ruta o URL del OA
Solicita Comprobar los datos del archivo
Cofirma o actualiza los datos
Solicita verificar los datos descriptivos del OA
Comprueba o actualiza los datos descriptivos del OA
Solicita crear una licencia Creative Commons del OA
Genera nueva licencia
Crea nueva licencia

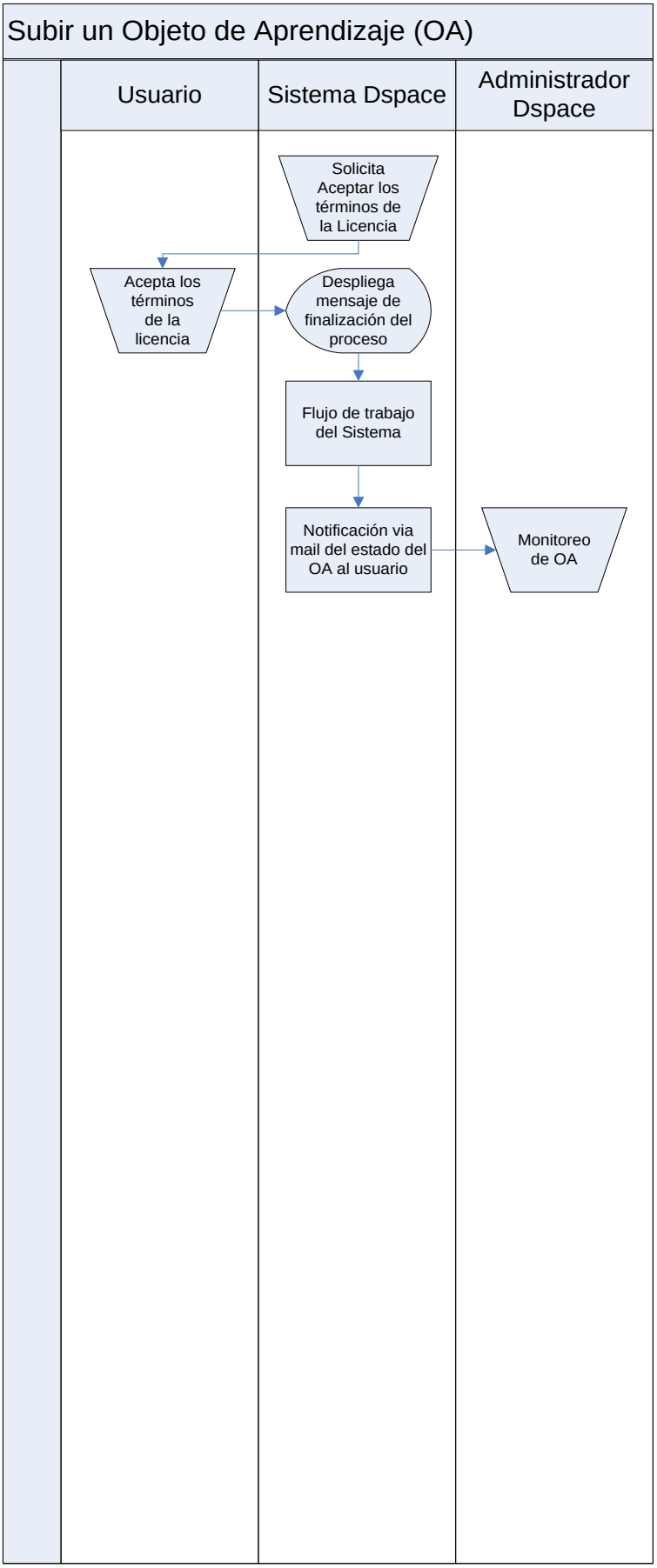


Figura A5. 0.1: Proceso de subir un OA

Perfil de stakeholders

Administrador del repositorio

Representante	Personal operativo de la Unidad de Virtualización.
Description	Personas encargadas de la administración del repositorio y gestión de los OA.
Type	Experto
Responsibilidades	Tienen a cargo tareas como, subir, actualizar, eliminar OA
Criterio de exito	Sistema que se adapta a las necesidades de la organización, es escalable y funcional
Involucramiento	Participación directa en el desarrollo del sistema, en diseño, implementación y pruebas.
Comentarios	

Profesor

Representante	Docentes Investigadores.
Description	Son los que generan y suben nuevos OAs
Type	Experto del negocio
Responsibilidades	.Registrarse en el sistema, generan OA’s
Criterio de exito	El sistema permite recuperar sus contenidos de manera fácil y precisa
Involucramiento	Participa en la especificación de requerimientos del buscador
Comentarios	

Estudiante

Representante	Profesionales en Formación.
Description	Se encarga de subir nuevos OA’s
Type	Usuario casual
Responsibilidades	.Registrarse en el sistema, subir nuevos OA’s, utilizar el buscador
Criterio de exito	El sistema permite recuperar sus contenidos de manera fácil y precisa
Involucramiento	Participa en pruebas del producto
Comentarios	

Perfil de usuarios

Profesor- Estudiante

Representante	Personas que utilizan el buscador
Description	Los usuarios del buscador filtran la información a través de los componentes del sistema
Type	Usuario casual
Responsibilidades	Utiliza el sistema, sube nuevos OA
Criterio de exito	El sistema provee un proceso de búsqueda simple, resultados precisos y rápidos.
Involucramiento	Se vincula en la especificación de requerimiento y realización de las pruebas.
Comentarios	

Usuarios Externos

Representante	Usuario Ocasionales
Description	Usuario externos a la universidad que consultan los OA's del repositorio y suben nuevo contenido
Type	Usuario casual
Responsibilidades	Filtrar información, registrarse en el sistema, subir nuevos OA's
Criterio de exito	El sistema permite recuperar sus contenidos de manera fácil
Involucramiento	Ninguno
Comentarios	

Necesidades de los Afectados/Usuarios

Necesidades comunes de todos los afectados

Necesidad	***Prioridad***	***Solución Actual***	***Soluciones Propuestas***	***Preocupación***
Proceso de Búsqueda Ágil	*Alta*	*Recolección de metadatos a mano*	*Estructuración de Contenido*	*El proceso se ralentice*
Precisión en los resultados	*Alta*	*Metadatos + Búsqueda Booleana*	*Estructuración de contenido + PLN + EI*	*Demasiado consumo de recursos*

Proceso de extraer información relevante de los OA	*Alta*	*Se maneja metadatos de información general extraídos a mano*	*Componente automático de Extracción de información*	

Necesidades de <Administrador del repositorio>

Necesidad	***Prioridad***	***Solución Actual***	***Soluciones Propuestas***	***Preocupación***
El proceso de gestión de OA se siga manteniendo como ha venido llevándose hasta el momento	*Alta*	*Participación de estudiantes de Gestión Productiva*	*El componente de estructuración a construirse debe coexistir con el sistema anterior*	
Los OA que se encuentran en el repositorio sean sometidos a proceso de extracción de información	*Alta*	*Solo se maneja metadatos extraídos a mano*	*Se construya un componente de estructuración masivo de OA*	*El nuevo proceso consuma demasiado tiempo*
El nuevo sistema no debe duplicar los OA	*Alta*		*Se recuperan los OA del repositorio actual*	

Necesidades de <Profesor Estudiante – Usuarios Externos>

Necesidad	***Prioridad***	***Solución Actual***	***Soluciones Propuestas***	***Preocupación***
Integrar el modulo de búsqueda avanzada	*Alta*	*Módulo de búsqueda avanzada basado en búsquedas booleanas y metadatos*	*Componente de búsqueda avanzada que permite filtrar la información por metadatos y entidades relevantes (Lugar, fecha, etc.).*	
Los resultados presenten el porcentaje de relevancia	*Media*	*No aplica*	*El sistema despliega los resultados con sus respectivos porcentajes de relevancia.*	

Resumen del Producto

Perspectiva del Producto

El buscador incluye un modulo de extracción de información, con una interfaz al repositorio actual, los datos capturados se guardan en una estructura predefinida de un repositorio de entidades.
Las consultas son ingresadas en Lenguaje de alto nivel y pasan por procesamiento de lenguaje natural y reconocimiento de entidades nombradas, además de mecanismos de desambiguación del sentido.

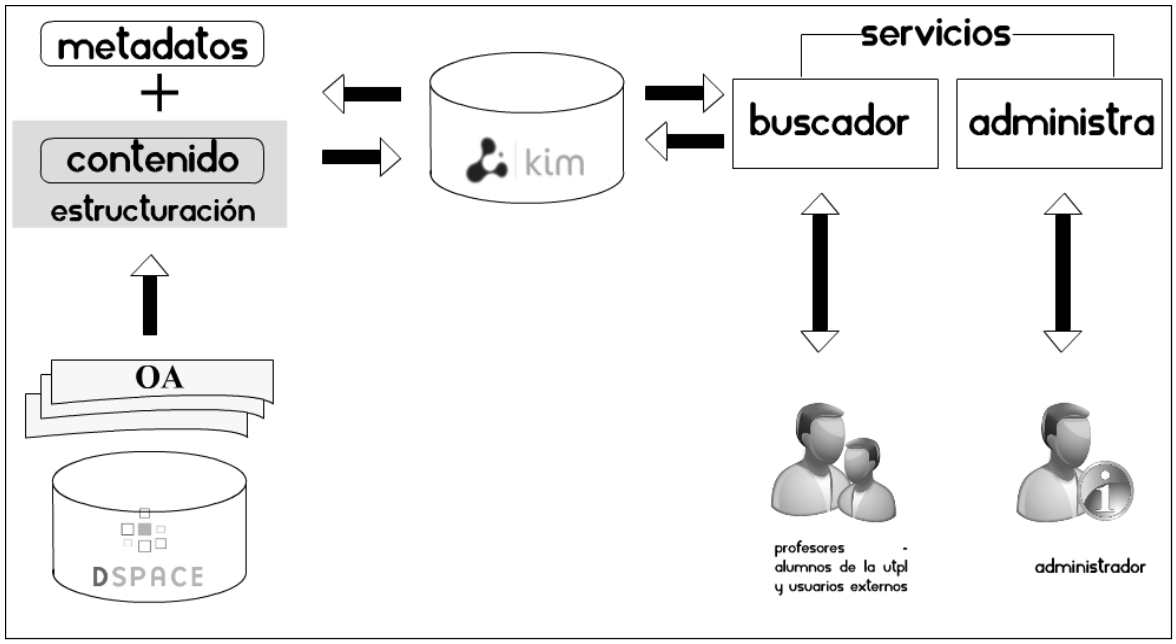


Figura A5 0.2: Arquitectura general del buscador

Resumen de Capacidades

Beneficios para el cliente	Características que lo soportan
Procesa consultas de alto nivel	• Posee mecanismos de PLN para el mejor entendimiento de la consulta • Cuenta con procesos de reconocimiento de entidades importantes • Módulo de desambiguación del sentido.
Resultados ágiles y precisos	• Contenido estructurado para describir de mejor manera contenido del OA
Permite una descripción más amplia de las consultas	• Contenido estructurado para describir de mejor manera contenido del OA • Cuenta con procesos de reconocimiento de entidades importantes

Supuestos y Dependencias

Las forma en que los OA's son subidos al repositorio se seguirá manteniendo durante el desarrollo de proyecto.

Características del Producto

Característica	*Descripción*
• Posee mecanismos de PLN para el mejor entendimiento de la consulta	La consulta es procesada morfológica, sintáctica y semánticamente para
• Contenido estructurado para describir de mejor manera contenido del OA	Permite al usuario describir con mayor riqueza sus consultas y al sistema buscar con mayor precisión.
• Módulo de desambiguación del sentido.	Permite establecer el sentido de las palabras utilizadas en la consulta para aumentar la precisión.
• Cuenta con procesos de reconocimiento de entidades importantes	Reconoce lugares, fechas, organizaciones para facilitar el proceso de búsqueda

Restricciones

Glosario

DRS	Documento de Requerimientos de Software
ANSI	American National Standards Institute
OA	Objeto de Aprendizaje
PLN	Procesamiento de Lenguaje Natural
RI	Recuperación de Información
EI	Extracción de información

Anexo 6: Especificación de Requerimientos de Software (ERS)

BUSPACE

Búscador semántico para la recuperación de Objetos de Aprendizaje (OA) del Dspace

Introducción

Descripción

El buscador semántico para el repositorio de objetos de aprendizaje del Dspace permitirá a estudiantes y profesores de la UTPL, así como también a usuarios externos llevar de forma más eficiente el proceso de búsqueda y filtro de información, empleando el procesamiento de lenguaje natural, reconocimiento de entidades y la desambiguación del sentido de las palabras con el fin de establecer de forma precisa el requerimiento del usuario.

También se contempla trabajar con el contenido de los OA (específicamente los basados en texto) para extraer información relevante, guardarlos en una base de entidades y facilitar la descripción de consultas a mayor detalle para aumentar la precisión de los resultados.

Problemas Conocidos

En el análisis realizado, se ha logrado establecer que la poca precisión en los resultados del buscador actual son a causa de los siguientes problemas principales:

Gestión Básica de los Metadatos: Los metadatos de los OA pasan por controles de calidad generales y básico encaminados para verificar la ortografía de los mismos, mas no su pertinencia, en este punto vale mencionar que el repositorio es alimentado colaborativamente por cualquier persona que se ha registrado en el sistema lo que trae como consecuencia que los metadatos muchas veces puedan contener errores, no reflejan exactamente el contenido del recurso o estén vacios. Esta información es crucial al momento de filtrar la información.

Consultas no descriptivas: La poca estructuración de contenido y la gestión básica de metadatos hace que sea imposible formular consultas con la suficiente especificación y el proceso de búsqueda sea ineficiente, empleé demasiado tiempo para encontrar resultados precisos y desperdicie recursos. La búsqueda avanzada actual solo filtra la información por tres campos a la vez y proponer consultas mas descriptivas es realmente complejo.

Procesamiento básico de Consultas: El procesamiento de las consultas es básico, solo se eliminan stop words, no se determina el sentido de las palabras ni el contexto en que

son utilizadas, en consecuencia no se establece de forma efectiva el requerimiento del usuario.

Referencias

[1] Definición de Requerimientos de Software ANSI/IEEE std. 830-1984
[2] Pressman, R. Ingeniería del Software - Un enfoque práctico. MCgrawHill

Descripción General

El buscador semántico para la recuperación de OA del Dspace tiene como función principal llevar de una forma eficiente el proceso de consulta del contenido del repositorio; para su implementación y correcto funcionamiento, dependerá de las funcionalidades, configuraciones y más procesos de integración con la herramienta KIM, la cual servirá como repositorio de conocimiento para el buscador.

Las funcionalidades del buscador semántico se encuentran organizadas en dos grandes grupos: Estructuración de contenido, y el proceso de consulta cada uno de los cuales emplean subprocesos.

Estructuración de Contenido: Permitirá recolectar datos de personas, organizaciones, fechas y más hechos descriptivos del contenido de los OA, estos son guardados en una base de conocimiento la cual servirá posteriormente para el proceso de búsqueda.

Procesamiento de Consulta: Las consultas ingresadas por el usuario pasan por mecanismos de procesamiento de lenguaje natural para establecer de manera precisa el requerimiento del usuario.

Búsqueda avanzada: Facilitará mecanismos para el filtrado avanzado de la información, el usuario tendrá la capacidad de proponer consultas estructuradas, además de filtrar la información por entidades y conceptos.

Perspectiva del Software

Este sistema trabaja con los OA publicados en el Dspace, su implementación y correcto funcionamiento, dependerá de las configuraciones y procesos de la herramienta KIM, la cual es una plataforma para la anotación semántica, almacenamiento de entidades y recuperación de información.

Características del Producto

El buscador semántico Buspace dispondrá de las siguientes funcionalidades:

1. **Administrador**

   Migración de OA

   - **Recuperar Contenido de OA**

     Se extrae el contenido de los OA, esto dependerá del tipo de OA, es así que para los objetos de basados en texto (word, pdf, html, excel, presentaciones, etc.) se extrae todos sus caracteres. En cambio OA multimedia (audio, video, animaciones, gráficos, etc) se toma la descripción de estos OA para su estructuración.

   - **Extraer entidades (Estructurar)**

     El contenido recuperado de los OA pasa por un proceso de extracción de información donde se reconocen entidades relevantes de cada recurso como son: personas, organizaciones, fechas, posiciones laborales, regiones, magnitudes físicas y relaciones entre entidades.

   - **Obtener Metadatos de OA**

     Junto a cada OA se describen ciertos metadatos en el repositorio como autor, fecha, tipo de OA, titulo y área académica de la Universidad a la que pertenece, estas también son considerados entidades.

   - **Guardar Información en KIM**

     Todas las entidades reconocidas del contenido de los OA al igual que los metadatos asociados a cada recurso se almacenan en la herramienta KIM.

2. **Usuarios (Profesor, estudiante, usuario externo)**

   Búsqueda Simple

   La consulta ingresada al sistema pasa por el modulo de PLN que contempla los siguientes procesos:

   - **Análisis Ortográfico**

     Analiza si la consulta no contiene caracteres especiales, caracteres no reconocibles por el buscador y errores ortográficos.

   - **Análisis Morfológico**

Determina la categoría a la que pertenece cada palabra en la consulta, así como también la estructura de la oración.

- **Análisis Sintáctico**

  Etiqueta cada uno de los componentes sintácticos que aparecen en la consulta y analiza la combinación de las palabras para formar construcciones gramaticalmente correctas.

- **Análisis Pragmático**

  Transforma la consulta en términos en que el sistema pueda entender (consulta SeRQL).

Búsqueda Avanzada

- **Búsqueda por entidades**

  El usuario escoge las entidades por las que desea realizar la búsqueda e ingresa el valor en cada parámetro.

USABILIDAD

<u>Guías para los procesos: búsqueda, y estructuración.</u>

- **Guiar a los usuarios en los procesos de búsqueda y estructuración.**

  Se pretende realizar documentos de información y ayuda para los usuarios en general, los cuales ayudarán a los usuarios a entender lo que son entidades y como utilizar la búsqueda avanzada para obtener mejores resultados.

- **Generación de informes.**

  Permitirá realizar estadísticas sobre el uso del buscador, OA estructurados y recuperados.

Características del Usuario

Administrador: Educación superior, conocimiento, experiencia y mantenimiento del repositorio, además realiza la gestión de los OA.

Estudiantes y Profesores.- Profesionales en formación con conocimientos básicos en los procesos de subir OA y búsqueda de información.

Usuario Externos.- Niveles de educación varios, interesados en investigar los recursos del repositorio y publicar su contenido.

Limitaciones Generales

La funcionalidad de gestión de entidades es de uso exclusivo del Administrador del repositorio.

Asunciones y Dependencias

Asunciones:

La publicación de los OA seguirá manteniéndose como hasta el momento, ya que la base de conocimiento recupera información del repositorio.

Requerimientos Funcionales.

REQ001 Recuperar Contenido de OA.

Entrada

Este requerimiento recibe como entrada la URL o la ubicación de uno o varios OA así como también el tipo (word, pdf, audio, video, etc.).

Proceso

El sistema clasifica según el tipo de OA, en recursos basados en texto captura todos los caracteres del mismo, para otro tipo de recursos como por ejemplo video, consulta los metadatos descripción y resumen para tratarlos como contenido del OA.

Salida

La salida de este proceso independiente al tipo de OA será un archivo temporal de texto plano con el contenido del recurso y la URL del mismo, todo esto con el fin de evitar sobrecarga en el procesamiento del servidor.

REQ002 Extraer entidades (Estructurar).

Entrada

Los datos utilizados en este requerimiento son lo obtenidos en el paso anterior (archivos de texto plano).

Proceso

Se extrae el contenido de cada texto plano y se procede a identificar entidades importantes dentro del contenido, las cuales pueden ser: Objeto, Lugar, Declaración, Organización, Compañía, Número, Abstracción de Empresas,

Industria, Tema, Término General, Titulo Laboral, Abstracción Sociales entre otras.

Salida

Extrae las cadenas de texto consideradas como entidades den cada OA.

REQ003 Obtener metadatos de los OA del Dspace.

Entrada

Cada OA en el repositorio Dspace tiene asociado un compendio de información sobre el recurso los cuales son llamados metadatos los cuales son buscados con el identificador del OA.

Proceso

Los metadatos que son recuperados dentro de este requerimiento son: autor, título, área académica, tipo idioma, palabras claves, resumen y descripción.

Salida

Se visualizan en el sistema todos los metadatos capturados.

REQ004 Guardar información en el repositorio de entidades de KIM

Entrada

Los datos iniciales son el conjunto de entidades reconocidas en el proceso de extracción de entidades (estructuración) y los metadatos capturados en el requerimiento anterior.

Proceso

Se procede a instanciar cada una de las entidades reconocidas en el contenido del OA y llenar con los datos obtenidos del reconocimiento de entidades, de igual manera se procede con los metadatos capturados.

Salida

Se realiza la población de la base de conocimiento con la información obtenido en requerimientos anteriores.

REQ005 Gestionar entidades.

Entrada
Tiene como entrada el listado de todas las entidades asociadas a cada OA.

Proceso

EL administrador tiene la capacidad de eliminar, actualizar o editar los valores de estas entidades siempre que lo considere necesario.

Salida
Refinamiento del repositorio entidades, con el propósito que solo las entidades estrictamente necesarias participen en el proceso de recuperación información.

REQ006 Eliminar entidades de OA dados de baja en el Dspace.

Entrada
El sistema tiene la capacidad de eliminar los OA que sean obsoletos a partir de la URL del mismo.

Proceso

En el repositorio Dspace se pueden eliminar OA, este requerimiento permite buscar y eliminar todas las entidades asociadas al recurso.

Salida

La salida es un repositorio si duplicación de información (entidades) innecesarias en el sistema.

REQ007 Análisis Ortográfico.

Entrada
Recibe la consulta del usuario en lenguaje natural.

Proceso
El sistema verifica que la consulta del usuario no tenga caracteres extraños, y que las palabras se encuentran correctamente escritas.

Salida
La consulta depurada sin faltas de ortografía.

REQ008 Análisis Morfológico.

Entrada
Recibe la consulta depurada del requisito anterior.

Proceso

Procesa la consulta para asignar la categoría gramatical a la que corresponde cada palabra en la consulta.

Salida
Consulta etiquetada morfológicamente.

REQ009 Análisis Sintáctico.

Entrada
Utiliza la consulta etiquetada morfológicamente en el paso anterior.

Proceso
Procesa la consulta para determinar la estructura de la oración y genera por cada palabra la respectiva etiqueta.

Salida
Consulta revisada sintácticamente.

REQ0010 Análisis Semántico.

Entrada
Recibe la consulta etiquetada estructuralmente del paso anterior.

Proceso
Este requerimiento se encarga de establecer el sentido correspondiente a cada palabra en el contexto en que son utilizadas dentro de la oración. Este proceso también contempla en reconocimiento de entidades como fechas, intervalos de tiempo y magnitudes dentro de la consulta.

Salida

La consulta ha sido etiquetada semánticamente, además de contener cierta información relevante para el proceso de búsqueda (entidades).

REQ0011 Análisis Pragmático.

Entrada
Recibe los datos de salida del requerimiento anterior.

Proceso
Se encarga de transformar la consulta del usuario en términos en que el sistema puede entender. El repositorio en el que se realizan las consultas está basado en una ontología donde se instancian las entidades reconocidas en el Requerimiento 1 (REQ002), a esta ontología se accede a través del idioma SerQL.

Salida

Consulta en términos de SerQL.

ReQ0012 Ejecutar consulta

Entrada
Hace uso de la consulta en términos de Serql.

Proceso
Realiza la búsqueda en el repositorio de entidades de los OA que cumplan con las condiciones de la consulta de entrada. En este paso el sistema puede realizar algunas inferencias.

Salida

EL listado de OA ordenados por relevancia que cumplen con los parámetros de entrada.

REQ0013 Búsqueda por entidades.

Entrada
Recibe como entrada los parámetros seleccionados para realizar la búsqueda.

Proceso
El usuario procede a ingresar los datos en las entidades escogidas previamente, estas ayudan a crear consultas más específicas y obtener resultados más precisos. Esta funcionalidad se utiliza como un método avanzado de filtrar información en el sistema.

Salida
Conjunto de OA que cumplen con el requerimiento del usuario.

REQ0014 Establecer el sentido de las palabras.

Entrada
En este requerimiento de contempla el uso de la consulta en lenguaje natural.

Proceso
El sistema genera los sentidos que las palabras pueden tomar en el contexto en que son utilizadas en la oración y el usuario se encarga de elegir el sentido más acorde con su requerimiento.

Salida

Conjunto de OA que cumplen con el requerimiento del usuario.

REQ0015 Extraer Entidades al publicar un nuevo OA.

Entrada

Un usuario autenticado en el Dspace, que se encuentre subiendo un nuevo OA.

Proceso
El usuario tiene la capacidad que al momento de publicar el nuevo OA, estructurar el contenido del mismo y dependiendo del tipo de recurso podrá estructurar el contenido o crear las entidades necesarias para la descripción del OA.

Salida
Entidades instanciadas en el repositorio KIM.

REQ0016 Guiar a los usuarios en los procesos de búsqueda y estructuración.

Entrada
Documentación de los procesos de búsqueda y estructuración.

Proceso
Se publicará la información necesaria dentro de los módulos correspondientes para guiar al usuario en los procesos del sistema.

Salida
En este requerimiento se tienen tutoriales, y tips para los usuarios.

REQ0017 Generación de informes.

Entrada
Los datos de entrada de este requerimiento son toda la información contenida en los procesos anteriores de estructuración, migración, ejecución de consulta, administración de entidades.

Proceso
Se visualizará las estadísticas realizadas a partir de los datos que provee el sistema.

Salida
Informe que el usuario o administrador puede consultar.

Requerimientos de Rendimiento

El sistema de búsqueda debido a su naturaleza debe responder de inmediato (5-8 segundos) a las interacciones de los usuarios. En cuanto a los procesos web de administración del repositorio como la migración, el sistema responderá en un tiempo prudente, la razón es que estos procesos manejan gran cantidad de información.

Limitaciones de Diseño.

Los recursos en cuanto al proceso de reconocimiento de entidades masiva de los OA son amplios, esto sobrecarga al servidor por lo que se utilizará archivos de texto plano para bajar el procesamiento además se planea que las tareas de extracción de contenido, metadatos de los OA, el reconocimiento de entidades se realiza de manera secuencial par alogerar la varga en el servidor.

Otros Requerimientos.

Otros módulos:

Se requiere que a través del repositorio Dspace de la UTPL se conceda implementen los requerimientos REQ006 y REQ0015.

Glosario

DRS	Documento de Requerimientos de Software
ANSI	American National Standards Institute
IEEE	The Institute of Electrical and Electronics Engineers, Inc.
OA	Objeto de Aprendizaje

BIBLIOGRAFÍA

- Abián, M (2005). El Futuro de la Web. Extraído el 22 Agosto de 2009 desde http://www.javahispano.org/tutorials.item.action?id=55
- Abián, M (2009ª). "Buscadores Semánticos: HAKIA por dentro y por fuera". Extraído el 1 Noviembre de 2009 desde http://www.wshoy.sidar.org/index.php?2009/07/01/46-buscadores-semanticos-comprender-para-encontrar-parte-1
- Abián, M (2009b). Buscadores Semánticos: Comprender para encontrar parte 1. Extraído el 20 de Octubre de 2009 desde http://www.wshoy.sidar.org/index.php?2009/07/01/46-buscadores-semanticos-comprender-para-encontrar-parte-2
- Akamai (s/f). Como se hace la búsqueda. Extraído el 31 de Octubre de 2009 desde http://spanish.akamai.com/enes/html/technology/how_search_works.html
- Almeida, Y. Puig A. (2006). Nutchifiler: perfiles de usuario para un motor de búsqueda. Extraído el 22 de Diciembre de 2009 desde http://www.informaticahabana.com/evento_virtual/files/SWL37.pdf
- Amaral C. (2000). " Design and Implementation of a Semantic Search Engine for Portuguese". Extraído el 20 de Septiembre de 2009 desde http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.4090&rep=rep1&type=pdf
- Berrueta D. (2004). "Buscador Semántico para la Documentación de la Administración Pública del Principado de Asturias". Extraído el 24 de Octubre de 2009 desde http://idi.fundacionctic.org/esperteyu/docs/articulo-blogak-2006.pdf
- Camacho, G. (2006). Motores de búsquedas. Trabajo presentado en la Pontifica Universidad Católica de Puerto Rico. Agosto. Puerto Rico. Extraído el 31 de Octubre de 2009 desde http://www.slideshare.net/gerinaldocamacho/motores-de-bsquedas-332516
- Carnap, R. (1950): Logical Foundations of Probability (p.p), Chicago, University of Chicago Press. (Second edition, 1962). http://evans-experientialism.freeWebspace.com/carnap.htm
- Castells, P (s/f). La Web Semántica. Extraído el 14 de Octubre de 2009 desde http://arantxa.ii.uam.es/~castells/publications/castells-uclm03.pdf
- Chau M, Huang Z y Chen H. (2003) "Teaching Key Topics in Computer Science and Information Systems through a Web Search Engine Project". The University of Hong Kong, ACM Journal of Educational Resources in Computing, Vol. 3, No. 3, September 2003. Extraído el 20 de Diciembre de 2009 desde: http://www.personal.psu.edu/faculty/h/u/huz2/Zan/papers/teaching.jeric.pdf
- Contreras, H. (2001). Procesamiento del Lenguaje Natural basado en una "gramática de estilos" para el idioma español. Universidad de los Andes. Facultad de Ingeniería. Extraído el 20 de Diciembre de 2009 desde: http://www.saber.ula.ve/bitstream/123456789/13157/1/hc_propuestatesis.pdf
- Gruber,T (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition. Extraído el 14 de Octubre de 2009 desde http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.7493
- Hernández, M. Villa, G (2009). "Tendencias en los buscadores: Web semántica". NoticIEEro. Extraído el 22 de Diciembre de 2009 desde: http://www.noticieeero.org/?p=1111
- Klein, P (2009). "Hakia, el buscador semántico. Buscar más allá de Google".Marketing Online. Extraido el 10 de Febrero de 2010 desde http://www.pauklein.com/hakia-el-buscador-semantico/.

- Llidó, María D. (2002). Extracción y Recuperación de Información Temporal. Tesis Doctoral. Universidad Jaume I. Extraído el 12 de Diciembre de 2009 desde http://www.tesisenxarxa.net/TESIS_UJI/AVAILABLE/TDX-0630104-124212//llido.pdf
- Maldonado M. (2002). Hermes: Servidor y biblioteca de modelos de recuperación de información**.** Tesis. Universidad de las Américas Puebla. Extraído el 19 de Enero de 2010 desde: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/maldonado_n_mf/capitulo_2.html.
- Martínez, Francisco J. (2004). Recuperación de Información: Modelos, Sistemas y Evaluación. (pp. 16). Murcia. España. El Kiosko JMC. Extraído el 4 de Enero de 2009 desde: http://digitum.um.es/xmlui/bitstream/10201/4316/1/libro-ri.PDF?sequence=1
- Moreno A. (1998). "Lingüística Computacional. Introducción a los modelos simbólicos, estadísticos y biológicos". Madrid. Editorial Sintesis. Extraído el 4 de Enero de 2009 desde: http://avaxhome.ws/ebooks/encyclopedia_dictionary/lasijdhfljashdlfkjhasldkjfhasldfh.html
- Rufiner (2004). Sistema de Reconocimiento automático del habla. Universidad Nacional entre Ríos. Argentina Extraído el 6 de Enero de 2010 desde: http://redalyc.uaemex.mx/redalyc/pdf/145/14502806.pdf
- Sosa, E (1997). Procesamiento del lenguaje natural: revisión del estado actual, bases teóricas y aplicaciones (Parte I) Extraído el 20 de Octubre de 2009 desde http://www.elprofesionaldelainformacion.com/contenidos/1997/enero/procesamiento_del_lenguaje_natural_revisin_del_estado_actual_bases_tericas_y_aplicaciones_parte_i.html
- Vaquero, R. (1997). Recuperación de la información en Internet: motores y otros agentes de búsqueda. Extraído el 17 de Diciembre de 2009 desde: http://ibersid.eu/ojs/index.php/scire/article/view/1078/1060
- Hamish, C. (2004). "Information Extraction, Automatic". Department of Computer Science, University of Sheffield. Extraído el 9 de Junio de 2010 desde: http://gate.ac.uk/sale/ell2/ie/main.pdf.
- Tellez, A. (2005): "Extracción de Información con Algoritmos de Clasificación". Instituto Nacional de Astrofísica, Óptica y Electrónica. Puebla-Mexico. Extraído el 9 de Junio de 2010 desde: http://ccc.inaoep.mx/~mmontesg/tesis estudiantes/TesisMaestria-AlbertoTellez.pdf.
- Hobss, J. (1993). The generic information extraction system. In Proceedings of the 5th Message Understanding Conference, Morgan Kaufmann, pp. 87-92, 1993.
- Chakrabarti, S. (2000). "Data mining for hypertext: A tutorial survey". Indian Institute of Technology Bombay. Extraido el 14 de Junio de 2010 desde: http://www.sigkdd.org/explorations/issue1-2/chakrabarti.pdf
- Cruz, J. y González, F. (2006). "Herramienta para la extracción de conocimiento en correos electrónicos personales". Universidad Nacional de Colombia. Bogotá. Extraído el 14 de Junio de 2010 desde: http://disi.unal.edu.co/~ypinzon/2013326-206/docs/Articulo1Cruz.pdf
- Carreras, X. y Pradó, L (2001). "A Flexible Distributed Architecture for Natural Language Analyzers". Departamento de Lenguaje y Sistemas Informáticos de la Universidad de Catalunya. Barcelona, España. Extraído el 21 de Abril de 2010 desde: http://www.lsi.upc.edu/~nlp/papers/carreras02b.pdf
- Brants, T (2000). "TnT -- A Statistical Part-of-Speech Tagger". Saarland University Computational Linguistics. Saarbriicken, Alemania. Extraído el 21 de Abril de 2010 desde: http://acl.ldc.upenn.edu/A/A00/A00-1031.pdf

- Schapire, R. y Singer, Y. (1999).” Improved Boosting Algorithms Using Confidence-rated Predictions”. AT&T Labs. Extraído el 21 de Abril de 2010 desde: http://www.cs.iastate.edu/~honavar/singer99improved.pdf
- Márquez, L y otros (2004). “*Senseval 3: The Spanish Lexical Sample Task*”. Universidad Politécnica de Cataluña y Universidad de Barcelona. Extraído el 23 de Junio de 2010 desde: http://www.cse.unt.edu/~rada/senseval/senseval3/proceedings/pdf/marquez2.pdf
- Rigau, G y otros (2000).”Framework and Results for the Spanish Senseval”. Universidad Politécnica de Cataluña, Universidad de Barcelona, Universidad Autónoma de Barcelona y Universidad Nacional de Educación a Distancia, España. Extraído el 23 de Junio de 2010 desde: http://nlp.uned.es/docs/german-rigau-mariona-taul.pdf
- Salton, y M. J. (G 1983) . Introduction to Modern Information Retrieval. New York: McGraw Hill. 1983
- Keen, E.M. (1971) Evaluation parameters. The SMART retrieval system Experimentes in automatic document processing. New Jersey:Prentice-Hall, 1971 p 74-111
- Díaz, J y Pérez, F (2009). Extracción y Recuperación de Información. Según patrones: léxicos, sintácticos, semánticos y de discurso. Madrid, DC: Curso de Ingeniería Informática de la Universidad Carlos III de Madrid. Extraído el 7 de julio de 2010 desde: http://galeon.com/recuperacionpatrones/evaluacion.html
- Dhava, l. T., Taha, O., & Phil, L. (2009). *GATE JAPE Grammar Tutorial.* Nottingham: GATE.