UNIVERSIDADE
DE VIGO

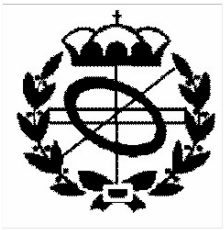
ESCOLA SUPERIOR DE ENXEÑERÍA INFORMÁTICA

Memoria e Manual de Usuario do Proxecto Fin de Carreira que presenta

Dna Sara Carrera Carrera

para a obtención do Título de **Enxeñeiro en Informática**

Adquisición semi-automática del conocimiento: una arquitectura preliminar



Xuño, 2007

Proxecto Fin de Carreira Nº: ENI 171

Director/a: Manuel Vilares Ferro
Área de coñecemento: Ciencias da Computación e Intelixencia Artificial
Departamento: Informática

"We must focus on seeding new trees, and hope that one day, when we will no longer be around, they will have mature, and bear new fruits..."

Gilles Khan.

AGRADECIMIENTOS

Ha llegado el momento de expresar mi gratitud a todas las personas que me han apoyado.

En primer lugar quiero dar las gracias a mi familia, especialmente a mis padres y a mi hermano. Siempre habéis estado cuando os he necesitado, me habéis proporcionado la mejor educación que cabe imaginar.

No quiero olvidarme de vosotras, Anita y Sabrina, por compartir conmigo tantos buenos momentos y por contar con vuestro apoyo cuando más lo he necesitado.

Existen unas personas que han sido las verdaderas protagonistas de mi vida a lo largo de estos años. Sois vosotras Isa, Mari, Rosa, Sandra y Tina. Gracias por compartir penas y alegrías.

A mis compañeros de fatigas, Miki, Santi, Iago, Sole, María. M., Alex y especialmente a ti María. B.

No me puedo olvidar de mis compañeros de laboratorio, Erica, Juan y Moli. Especialmente quiero dedicarle este proyecto a Mila, sin quien nada de esto hubiese sido posible. Gracias por la constante aportación de ideas y por el apoyo mostrado a lo largo de estos meses.

También dar las gracias a mi director Manuel Vilares por la oportunidad brindada al poder entrar en el laboratorio.

Finalmente estás tú, Celso. Tu apoyo incondicional es él que me da fuerzas cada día para seguir adelante. Estás ahí en los momentos flojos dándome tu apoyo y aportando una sonrisa siempre tan necesaria. En los bueno momentos sigues ahí, me acompañas. A ti quiero dedicar la culminación de mis años en la facultad.

A cada uno de vosotros gracias.

TABLA DE CONTENIDOS

1. IDENTIFICACIÓN DEL PROYECTO

titulo: Adquisición semi-automática del conocimiento: una arquitectura preliminar.

código: ENI - 171

alumna: Carrera Carrera, Sara

fecha: Junio 2007

director: Vilares Ferro, Manuel

área: Ciencias de la computación e inteligencia artificial

departamento: Informática

titulación: Ingeniería Informática

SITUACIÓN DEL PROYECTO

2. ORGANIZACIÓN DE LA DOCUMENTACIÓN

La documentación del proyecto se organiza en dos tomos, el Manual Técnico y la Memoria que incluye el Manual de Usuario.

En el Manual Técnico se abordan los siguientes temas:

- *Análisis del sistema*: Incluye la especificación de requisitos, el diagrama de clases del análisis junto a la descripción de las mismas, los detalles de los casos de uso así como los diagramas de secuencia del análisis.
- *Diseño del sistema*: Incluye los diagramas de clases de cada paquete así como la descripción de cada una de las clases, los patrones de diseño utilizados y el diseño de la base de datos.
- *Implementación del sistema*: En este apartado se incorporan todos aquellos detalles de implementación considerados relevantes.
- *Pruebas del sistema*: Finalmente se incorporan las pruebas llevadas a cabo para comprobar el correcto funcionamiento del sistema desarrollado.

En la Memoria (documento actual)se abordan los siguientes temas:

- *Introducción y situación del proyecto*: Se describe el problema abordado así como la solución desarrollada y los objetivos planteados.
- *Desarrollo del proyecto*: En esta sección se detalla la metodología de desarrollo utilizada así como aquellas tecnologías empleadas en la implementación.
- *Recursos de PLN*: En este apartado se hace una breve introducción al procesamiento del lenguaje natural (PLN) y se describen las características de las herramientas de PLN empleadas en el sistema.
- *Ontologías*: Incluye una descripción del término ontología y sus aspectos fundamentales.

- *Apéndice Técnico*: En el apéndice se describen dos tecnologías empleadas en el desarrollo que son Lucene y OWL.

En el Manual de Usuario se incorpora toda la información para la puesta en marcha de la aplicación, incorporando el manual de instalación, el Manual de Usuario para el manejo de la aplicación creada y el manual del usuario experto para poder llevar a cabo ciertas tareas que requieren unos conocimientos avanzados.

Además de la documentación aportada a través de los documentos descritos se incluye también un disco. Este disco tiene la documentación en formato digital así como el código fuente para instalar la aplicación. Se añaden también las herramientas necesarias para la puesta en marcha del sistema.

3. INTRODUCCIÓN

Actualmente el volumen de información textual disponible en formato digital hace necesario el desarrollo de herramientas que permitan la gestión de la misma. Cualquier aplicación que pretenda ser una solución a esta problemática debiera tener en cuenta algunos parámetros: la gestión automática de fuentes textuales independientemente del idioma, la adquisición de conocimiento y un acceso eficaz a los recursos lingüísticos.

Alrededor de la información disponible a través de documentos digitales se viene manifestando un interés creciente en fomentar el desarrollo de sistemas que procesen estos documentos automáticamente. De igual modo, hay que impulsar el perfeccionamiento de los sistemas de traducción. Ambos suponen un acercamiento al problema de la avalancha informativa y al problema derivado de la necesidad de comunicación translingüística.

Hoy en día, corrientes de investigación relacionadas con la Inteligencia Artificial (IA) y más concretamente con el procesamiento del lenguaje natural (PLN), centran su actividad en el concepto de adquisición automática de conocimiento y en su estructura de representación, las ontologías. Una ontología se define como una especificación explícita de un dominio que deseamos representar. Incluye un vocabulario de términos, el significado de los mismos y las relaciones existentes entre ellos. Es decir, define un vocabulario común, para aquellos usuarios finales (bien otros sistemas bien otras personas) que necesitan compartir información sobre dicho dominio.

El peso que a este contexto adquiere el acceso multilingüe a la información es creciente y constituye uno de los principales cuellos de botella de la sociedad de la información. Cada vez son más las aplicaciones que se centran en investigar, elaborar e integrar el uso de varias lenguas. Sería por lo tanto de interés el desarrollo de una estrategia que permita combinar fácilmente varios idiomas, transitando entre ellos de forma simple y eficiente. Es fácil ver entonces, el papel que puede jugar una ontología, permitiendo combinar distintas lenguas a través de la jerarquía de conceptos.

3.1.ORIGEN DEL PROYECTO

Este proyecto surge ante la necesidad del desarrollo de un marco que permita explorar los recursos textuales mediante la aplicación de técnicas y algoritmos de PLN. Concretamente la idea subyacente de este trabajo se centra en la adquisición de conocimiento que permite generar de manera semi-automática una ontología.

Los sistemas de gestión de información necesitan recursos ontológicos que permitan describir los términos y conceptos de un dominio. La lista de recursos con base terminológica es tan larga como la de los sistemas de tratamiento de información. Se pueden citar entre otros, los tesauros, los sistemas de indexación automática, los entornos de ayuda a la traducción o los de recuperación/extracción de información. A menudo abordados desde diferentes ópticas, es posible un acercamiento común a todas ellas. Para lograr este objetivo hay que facilitar la extracción del conocimiento contenido en los textos, es decir posibilitar la construcción semi-automática de una ontología a partir de las fuentes textuales. La tendencia sugiere además que para conseguir herramientas más eficientes, robustas y precisas para el dominio pertinente, el uso de recursos ontológicos parece indispensable [1]

Es, además, destacable que el desarrollo de una técnica de este tipo no es en si una herramienta final, sino que debería verse como un instrumento o medio para un determinado tipo de aplicación de mayor envergadura. Es el caso de un sistema de recuperación de información, un sistema de extracción de información o de un sistema de búsqueda-respuesta.

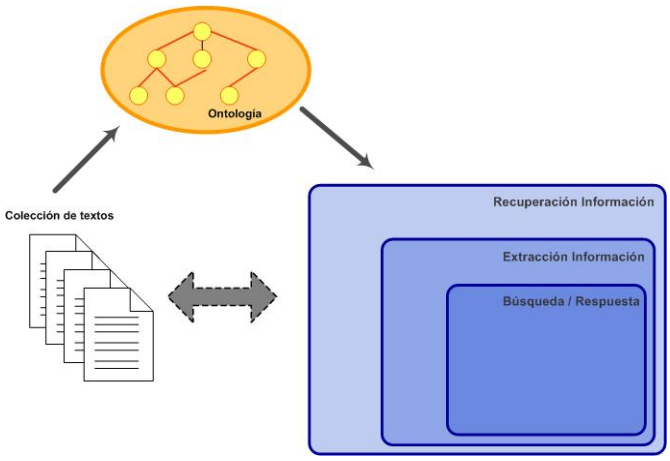


Figura 1 sistemas con ontología

En la Figura 1 se ilustra la situación de una ontología en los sistemas nombrados previamente. Brevemente se pasa a describir el cometido de cada uno para así comprender la relevancia de su integración.

Un sistema de **Recuperación de Información** tiene por objetivo identificar los documentos de una colección que sean relevantes a una necesidad de información del usuario. El resultado es un conjunto de documentos que se ordena en función de cómo refleja cada documento la necesidad planteada. Un ejemplo habitual de un sistema de recuperación de información es el de los motores de búsqueda en Internet tales como Google o Yahoo[1].

Un sistema de **Extracción de Información** va un paso más allá, buscando información más precisa en la colección de documentos. Tiene por objetivo detectar, extraer y presentar fragmentos muy concretos que respondan a la consulta del usuario. Un ejemplo sería un sistema orientado a la extracción de las matrículas los coches y sus titulares en documentos de una oficina de seguro.

Un sistema de **Búsqueda/Respuesta** tiene por objetivo permitir a un usuario satisfacer su necesidad concreta de información. Los datos que obtiene como respuesta son específicos a lo que solicitó. Es decir, el usuario mediante la formulación de preguntas muy específicas, que contengan información concreta, obtiene las respuestas adecuadas. Por ejemplo, el cliente podría formular la siguiente *petición "Donde está el Auditorio de Ourense ?"*; el sistema respondería: "*El Auditorio está en Rúa da Canle, 2*".

En los tres sistemas abordados y en su versión más básica, las consultas se formulan utilizando una serie de palabras clave que cada sistema se encargará de procesar. Es en este punto que entran en juego las ontologías. Su utilidad se basa en poder expandir la consulta realizada por el usuario. Navegando por la estructura jerárquica que define una ontología es posible recuperar los términos que tengan relación con las palabras clave de la petición. De esta manera los datos, que los distintos sistemas tienen que buscar en sus colecciones de documentos, son más concretos, precisos y completos.

[1] http://www.google.com y http://www.yahoo.com

3.2.OBJETIVOS DEL SISTEMA

El objetivo perseguido es el desarrollo de una arquitectura genérica que permita, a partir de un conjunto de fuentes textuales, la extracción de conocimiento. Éste se representará en forma de términos clave y en forma de ontología.

Para ello se parte de un conjunto de noticias recogidas de periódicos en sus secciones de economía. La principal característica es que dichas noticias, se encuentran en dos idiomas, francés y español. Tenemos así un conjunto de textos paralelos en ambas lenguas. Para la realización de un proyecto de este tipo es necesario considerar diferentes fases:

1. En un primer momento es necesaria la búsqueda y estudio de herramientas que se centren en la extracción de los términos relevantes del texto. Un término es un conjunto de palabras que tiene propiedades sintácticas y representa una realidad en un domino de conocimiento. La extracción terminológica deberá proporcionarse para los idiomas en juego.

2. En una segunda fase se deberá buscar y realizar un estudio sobre una aplicación que nos proporcione análisis sintáctico superficial robusto. El análisis sintáctico robusto consiste en poder recuperar la estructura total y/o parcial de una oración en todos los casos. El hecho de que sea superficial implica que se deba obtener información sintáctica de la oración en unidades de interés, de forma eficiente y fiable, sacrificando quizás la completitud y profundidad del análisis global.

3. Por último la tarea se centrará en la aplicación de un algoritmo para la extracción de una ontología. Para llevar a cabo esto, será necesario, la integración y adaptación de las herramientas citadas en los dos puntos anteriores.

El estudio realizado en cada uno de estos pasos conlleva analizar los algoritmos empleados, así como el tipo de información manejada. Los resultados han

demostrado la necesidad de un apoyo externo en forma de componente, como puede ser un etiquetador o un lematizador.

Una vez expuestas las etapas de desarrollo podemos citar los objetivos que se pretenden conseguir:

- La construcción de un prototipo para la generación semiautomática de los conceptos que se jerarquizan en una ontología. El hecho de que sea semiautomática implica la validación del usuario de los resultados obtenidos en cada paso para que éstos se ajusten a sus necesidades. Hay que ver este desarrollo como un proceso iterativo de forma que a cada paso la ontología se enriquece y la adquisición del conocimiento es cada vez más precisa.

- La generación y estructuración de los términos clave de los documentos tratados. Dichos términos serán presentados al usuario y utilizados en la construcción de la ontología.

- El objetivo que se pretende lograr con un enfoque multilingüe es poder transitar de un idioma a otro y obtener conocimiento en todas las lenguas implicadas. Podemos ilustrarlo con un ejemplo en el que esta herramienta se aplique al desarrollo de un sistema de recuperación de información, la idea es que la pregunta se puede formular en cualquiera de los idiomas considerados y a través de la ontología la respuesta generada sea además de precisa, devuelta también en las diferentes lenguas disponibles.

- Finalmente, la ontología creada podrá ser recuperada por el usuario. Además, como se ha comentado anteriormente, la utilidad fundamental de la ontología generada es su uso en una aplicación de mayor envergadura. Se incorpora así, un sistema de recuperación de información a través del cual el usuario introducirá consultas, ampliadas con los conceptos de la ontología, recuperando lo documentos relevantes a la petición.

La arquitectura a implementar debe ser abierta para facilitar la incorporación de nuevos recursos. Se pretende conseguir un sistema escalable que aporte las siguientes funcionalidades:

- Incorporación de un nuevo idioma. De esta manera se consigue un enfoque multilingüe permitiendo que el sistema lidie con los distintos recursos necesarios para la explotación de un idioma, éstos son los extractores y analizadores sintácticos.

- Incorporación de un nuevo tema. El tema concreto tratado en el desarrollo del proyecto es el económico. Sin embargo, pensando en un esquema abierto y escalable es justo pensar que los recursos de un idioma deben poder emplearse para cualquiera que sea el dominio encerrado en las fuentes textuales. Este hecho se ve, a menudo, sujeto a modificaciones dentro de los propios módulos empleados por las herramientas de explotación de los textos. Por ejemplo la adaptación de un lexicón al dominio tratado. Este es un objetivo al margen del presente proyecto, pues cada usuario, deberá adaptar sus herramientas al tema correspondiente.

- Contemplar la diversidad de formatos en los que pueden venir dados los textos.

- Posibilidad de realizar consultas y recuperación de los documentos relevantes a las mismas.

Expuestos los objetivos, así como las funcionalidades y pasos a seguir, en este punto es destacable que este sistema está enfocado hacia dos tipos de usuarios.

- Un primer usuario no necesita ninguna clase de conocimiento más que el necesario para manejar un programa que le ayude en un propósito específico, en este caso en la recuperación de conocimiento a partir de unos documentos.

- Otro tipo de usuario que pudiéramos considerar experto, será aquel que además de poseer el conocimiento para el manejo de los recursos disponibles, sea capaz de incorporar nuevas herramientas con lo que esto implica. Es decir, debe tener la experiencia suficiente para tratar los datos proporcionados por los recursos que incorpore y estructurarlos tal y como el sistema exige.

3.3.DESCRIPCIÓN DEL SISTEMA

En este apartado se expone la solución adoptada, así como las características de la arquitectura desarrollada.

Como se ha mencionado en el apartado anterior el desarrollo de una arquitectura abierta es el principal objetivo. Aunque los idiomas en los que se ha trabajado han sido el francés y el español y el tema de los textos manejado ha sido el económico, en todo momento el sistema se ha llevado a cabo considerando la posibilidad de integrar nuevas lenguas y temáticas.

A continuación se detallan las funcionalidades específicas de cada uno de los módulos de la arquitectura planteada, junto con una breve descripción del flujo de información entre las distintas partes. Se trata de mostrar un primer acercamiento al sistema, posteriormente en los capítulos dedicados al análisis, diseño e implementación, la descripción será más detallada.

La arquitectura planteada para el sistema de adquisición semi-automática del conocimiento se puede estructurar en cuatro capas.

La primera es una **capa de interfaz** con el usuario. Mediante la utilización de una sencilla interfaz gráfica el usuario puede interaccionar con la herramienta, solicitando todas aquellas operaciones que desee llevar a cabo.

La segunda es la **capa mediador**. Se encarga de revisar los datos procedentes del fichero de configuración para poder proceder al conjunto de tareas que llevan a cabo las capas inferiores. Es en si el centro del sistema, pues es donde el usuario define los temas que va a tratar a lo largo de sus acciones. Llegados a este punto es importante destacar que lo que en esta documentación llamamos **tema** es el contenido o materia tratada en las fuentes textuales disponibles, es decir, el corpus concreto tratado en un instante. En el caso concreto desarrollado para este sistema, el tema de los textos es el económico.

La tercera capa es la **capa de procesamiento**. Se caracteriza por contener todos los subsistemas que realizan las tareas concretas del proyecto. Estas tareas son el tratamiento de los documentos para la generación del corpus, la extracción de términos, el análisis sintáctico así como la generación de la ontología.

Por último se considera la **capa de recuperación**. En la misma se ubican los recursos necesarios para explotar los resultados de las operaciones de la capa anterior. Así mediante la consulta de las ontologías creadas y a través del índice de documentos, esta capa ofrece la posibilidad de recuperar aquellas fuentes textuales que responden a una petición de usuario.

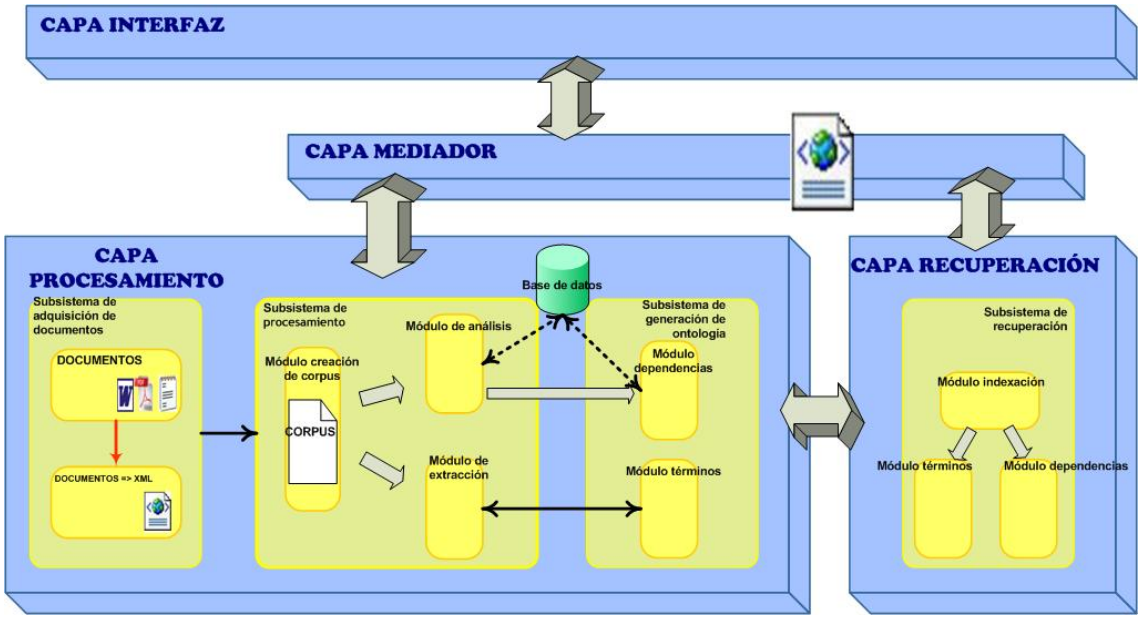


Figura 2 arquitectura del sistema

3.3.1.CAPA DE INTERFAZ DE USUARIO

En esta capa se maneja toda la lógica relacionada con el conjunto de vistas presentadas al usuario para la realización de las tareas que pretenda lleva a cabo. Según las opciones elegidas, el sistema mostrará una vista u otra, ofreciendo la posibilidad al usuario de introducir los datos que se le solicite para ejecutar una tarea.

3.3.2.CAPA MEDIADOR

Esta capa es la responsable de controlar la configuración del sistema. Es en si sencilla, sin embargo encierra el control de toda la aplicación. Se encarga de gestionar el fichero de configuración, que se puede considerar el corazón de la aplicación.

Hace transparente al usuario los recursos necesarios para la ejecución de una determinada tarea. A ella corresponde de cargar los recursos de forma dinámica y a medida que se vayan necesitando. De esta forma el usuario sólo tiene que preocuparse de las operaciones que desea llevar a cabo; como por ejemplo lanzar una extracción con un único o con varios extractores de entre los disponibles.

El sistema puede realizar su cometido mediante accesos al árbol de configuración que contiene información sobre todos los recursos a su alcance. En el apartado 10.2 del Manual Técnico se detalla la estructura del mismo.

Existe una serie de consideraciones que es importante conocer para comprender cómo el sistema interactúa con la capa mediadora y para entender las decisiones tomadas.

- En primer lugar hay que tener en cuenta que cada tema es independiente de las herramientas que se utilizan para explotar los textos.

  A modo de ejemplo se puede decir que en un mismo sistema podrían coexistir el tema económico y el tema jurídico; sin embargo las herramientas de explotación no variarían del uno al otro.

- En segundo lugar hay que tener en cuenta que si bien las herramientas no son dependientes de un tema concreto sí lo son del idioma sobre él que se apliquen.

  A modo de ejemplo se puede decir, que si en el sistema coexisten el tema de cocina con fuentes en español y en francés, y el tema de botánica con fuentes en francés e inglés; las herramientas de explotación para el francés serán aplicables en ambos casos.

Por lo tanto se puede concluir que dado un tema y sus fuentes textuales, los recursos necesarios deberán buscarse en base a los idiomas tratados en los documentos.

3.3.3.CAPA DE EJECUCIÓN

Si la capa mediadora era el corazón de la aplicación, ésta es el motor de la misma. Es la capa encargada de lanzar y ejecutar las funcionalidades fundamentales de la aplicación.

A través de la interfaz y con los recursos recuperados de la capa superior, ésta llama a sus distintos subsistemas para ejecutar las tareas que se le hayan encomendado.

Se divide en un conjunto de subsistemas:

(A) SUBSISTEMA DE ADQUISICIÓN DE DOCUMENTOS

En el subsistema de adquisición de documentos, se manejan los textos introducidas en el sistema. Al trabajar con un enfoque multilingüe los textos introducidos serán paralelos. Es decir, dados dos idiomas contra los que se va a aplicar una determinada tarea, las fuentes textuales de ambos han de ser traducciones literales la una de la otra. Para ilustrarlo se muestra a continuación un ejemplo:

Un euro à 1,50 dollar... Le scénario qui fait peur. L'ampleur du déficit américain pénalise le billet vert. l'Europe est la première victime de cette crise monétaire. Faut-il croire à un krach du billet vert à l'approche du cinquième anniversaire de l'euro qui caracole sur les plus hauts niveaux de son histoire encore récente ?	Un euro a 1,50 dólares... Un escenario que da miedo El gran déficit norteamericano penaliza el billete verde. Europa es la primera víctima de esta crisis monetaria. ¿Hay que pensar en una quiebra del billete verde cuando se acerca el quinto aniversario del euro que oscila en los más altos niveles de su todavía reciente historia?

A pesar de ser una arquitectura abierta, para el desarrollo de este proyecto se ha contado con un conjunto fijo de documentos sobre el tema económico. Sin embargo, está contemplado el poder añadir nuevas fuentes textuales, estando este hecho sujeto a ciertas condiciones.

Así, el usuario tiene que tener muy presente en qué tareas y en qué temas desea emplear los idiomas. Es decir, para un usuario concreto el francés y el español pueden ser las únicas lenguas con las que desee trabajar para el tema de economía. Sin embargo, otro usuario podría desear trabajar con otro tema, por ejemplo el jurídico, y contar con otros dos idiomas para ello, por ejemplo el inglés y el alemán. Será entonces el usuario el responsable de proporcionar los documentos paralelos en los idiomas en los que pretenda trabajar.

El subsistema de adquisición de documentos, se encarga entonces, de procesar los textos introducidos en el sistema. Sus dos funcionalidades principales:

1. Extracción de la información relevante de los textos.

2. Almacenamiento de esa información en un formato estándar como XML

Este hecho es fundamental, pues el subsistema de adquisición de documentos está orientado a manejar distintos tipos de formatos. Es decir, un documento podría ser proporcionado en cualquier tipo de fichero; como txt, pdf, doc, etc. Es el mismo subsistema que se encargará de recoger la información relevante y almacenarla en un formato estándar definido por un DTD (*Definición de tipo de documento*). Se consigue de este modo que el resto de módulos de la arquitectura sean independientes del formato original de los textos. Cabe destacar en este punto, que la arquitectura implementada queda abierta para la incorporación sencilla de nuevos formatos.

(B) SUBSISTEMA DE PROCESAMIENTO

El subsistema de procesamiento se encarga de aplicar los recursos lingüísticos a las fuentes textuales. Este subsistema se divide a su vez en tres módulos.

El **módulo de creación del corpus** se encarga de recuperar la información relevante de los documentos con las fuentes textuales proporcionadas por el usuario.

El corpus es otro elemento fundamental que se utilizará a lo largo de todo el sistema. En el mismo se encierra todo el conocimiento que se desea extraer y a partir del cual los módulos de extracción y de análisis sintáctico aplicarán sus algoritmos.

El **módulo de extracción de términos**, se encarga de la adquisición de la terminología relevante. En las secciones 13 y 14 se detallan cuales han sido los extractores empleados, así como las modificaciones que se han realizado para su adaptación a la herramienta construida.

Los resultados que proporciona la extracción terminológica son muy dispersos entre herramientas distintas. Así, algunos generan su salida a través de un fichero XML y otros a través de un simple fichero de texto. Este hecho debe ser tenido en cuenta por el usuario, pues será el responsable de analizar las salidas de los extractores que introduzca en el sistema para transformar dicha salida al formato esperado por la aplicación, concretamente los resultados de la extracción se almacenan en un fichero XML. El formato de dicho fichero se detalla en el apartado 10.2 del Manual Técnico.

El **módulo de análisis sintáctico** se encarga de analizar el corpus y generar las dependencias sintácticas que existen en el mismo. Estas dependencias serán almacenadas en una base de datos y aplicadas en el algoritmo de extracción de una ontología.

Al tener que aplicar los resultados del análisis sintáctico en un algoritmo concreto, es necesario que éstos estén adecuadamente organizados.

El análisis sintáctico es una tarea que depende exclusivamente del tema e idioma sobre el que se aplica. De este modo, el lanzamiento del análisis sintáctico vendrá definido por el tema tratado en los textos de entrada y por el idioma de los textos. La razón de esta solución se basa en que la ontología que se pretende generar en cada momento deberá guardar información en un único idioma y tratar de un único tema.

Dado el carácter genérico pretendido, estos dos últimos módulos tienen una interfaz externa que proporciona una entrada al mismo y abstrae de los detalles propios de cada herramienta. De este modo la incorporación bien de un nuevo extractor de términos bien de un nuevo analizador sintáctico se verá condicionada a

seguir una estructura predefinida. Esta estructura viene dada por una DTD o a través del diagrama de entidad/relación que define la base de datos.

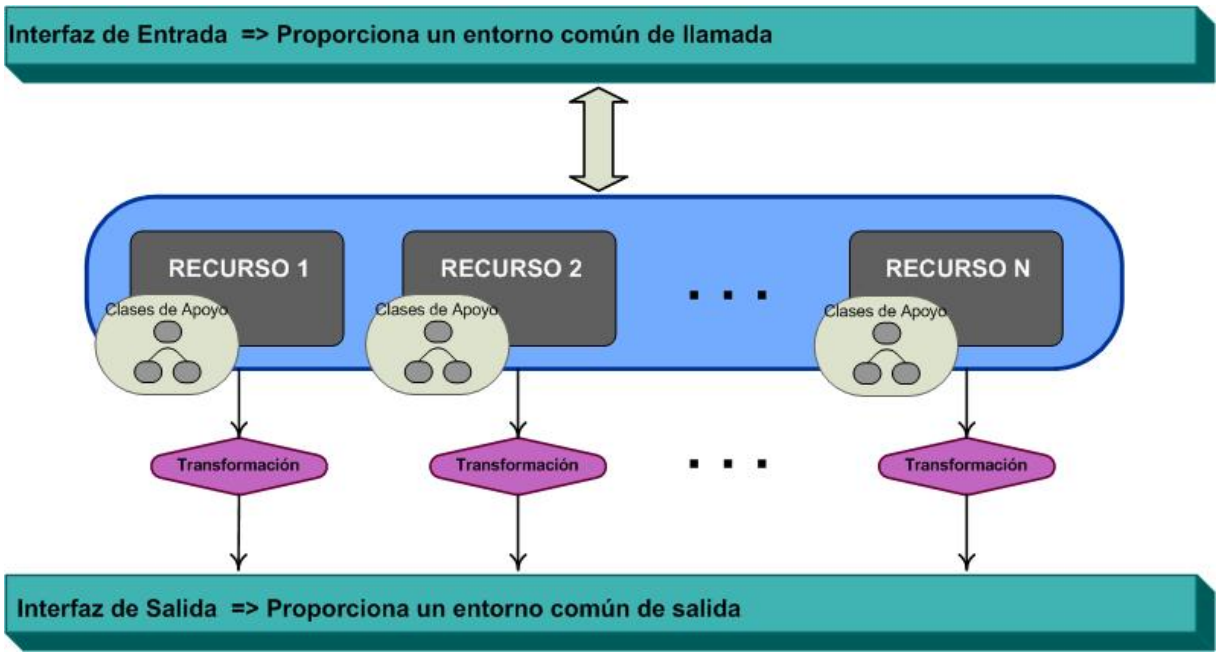


Figura 3 arquitectura de los recursos

(C) SUBSISTEMA DE GENERACIÓN DE ONTOLOGÍA

En este subsistema se lleva a cabo la creación de las ontologías disponibles en el sistema. Una ontología nos va permitir crear una estructura con los conceptos relevantes en el dominio tratado.

Se han seguido dos enfoques para la consecución de las mismas, reflejado en cada uno de los módulos del subsistema de generación de ontología.

En el **módulo de ontología por términos** se parte de los términos extraídos con los extractores y siguiendo una estrategia definida para este cometido se consigue crear una estructura jerárquica de los mismos. Esta técnica emplea conocimiento sobre los contextos de aparición de las palabras que conforman los términos. En este caso se generarán tantas ontologías como extracciones se hayan llevado a cabo. Para más detalle, consultar en el Manual Técnico la sección *12. implementación de la ontología por términos*.

En el **módulo de ontología por dependencias** se parte de las dependencias extraídas a partir del análisis sintáctico. Aplicando un novedoso algoritmo se consigue estructurar las dependencias generadas y se establecen un conjunto de relaciones

entre los elementos de las mismas. El proceso es iterativo de manera que en cada conjunto de iteraciones el usuario puede interactuar y modificar los datos obtenidos por el proceso de la forma que considere oportuna. Esta técnica se detalla en el Manual Técnico (apartado: *13.1. aplicación de un algoritmo de Error-Mining de extracción de conocimiento*)

En ambos casos las ontologías creadas se emplean para ampliar aquellas consultas introducidas por el usuario a la hora de recuperar documentos.

3.3.4.CAPA DE RECUPERACIÓN

Esta capa explota los resultados de la capa de procesamiento. Se encarga de indexar los textos introducidos en el sistema y recuperar los mismos en función de unos parámetros.

En un primer momento, esta capa debe crear el índice asociado a un conjunto de documentos, es decir, crea un índice relaciona con el tema e idioma actual. Este índice será automáticamente actualizado cada vez que se incorporen nuevos textos al sistema. Por lo tanto, cada vez que un usuario mete un nuevo texto en el sistema, éste deberá ser examinado e introducido en el índice.

El objetivo de introducir una tarea de indexación en el sistema se basa en poder posteriormente recuperar el conjunto de documentos relevantes a la solicitud de un usuario. Para ello, esta capa procesará tanto la consulta como las distintas ontologías creadas, ya que lo que se persigue es ampliar la petición del usuario con conceptos relacionados que se ubiquen en la ontología. De esta manera el sistema devolverá documentos más concretos y organizados según conceptos similares.

3.4.FUNCIONAMIENTO DEL SISTEMA

En este apartado, se presenta un ejemplo del funcionamiento del sistema. Se trata de mostrar el flujo de los datos a través de la arquitectura.

LANZAMIENTO DEL PROGRAMA

En un primer momento el usuario debe elegir la ubicación del fichero de configuración. El sistema analiza dicho fichero para encontrar posibles errores o subsanar ciertas inconsistencias, por ejemplo, crear un directorio si este no existe.

A partir de este momento varias tareas pueden ser lanzadas. Se presentan a continuación.

GENERACIÓN DEL CORPUS

El usuario indicará el directorio que contiene los documentos (también se pueden introducir de uno en uno). El sistema se comunicará a través de la capa mediador con el *subsistema de adquisición de documentos* para almacenar el nuevo documento en su correspondiente archivo XML. Guardará además una copia del archivo original.

Este subsistema se comunicará con el *módulo de creación del corpus* para actualizar el corpus con las nuevas fuentes textuales incorporadas. Se comunicará también con el *módulo de análisis sintáctico* para actualizar el mismo y con el *módulo de indexación* para añadir los nuevos documentos en el índice.

Cada vez que el usuario incorpore nuevos textos al sistema tendrá que indicar previamente sobre qué corpus está trabajando, es decir, elegirá el tema y el idioma. Estos datos se utilizarán en todos los subsiguientes procesos, estos son, como ya se ha mencionado, la actualización del corpus, el lanzamiento del análisis sintáctico y la actualización del índice de documentos para ese tema e idioma.

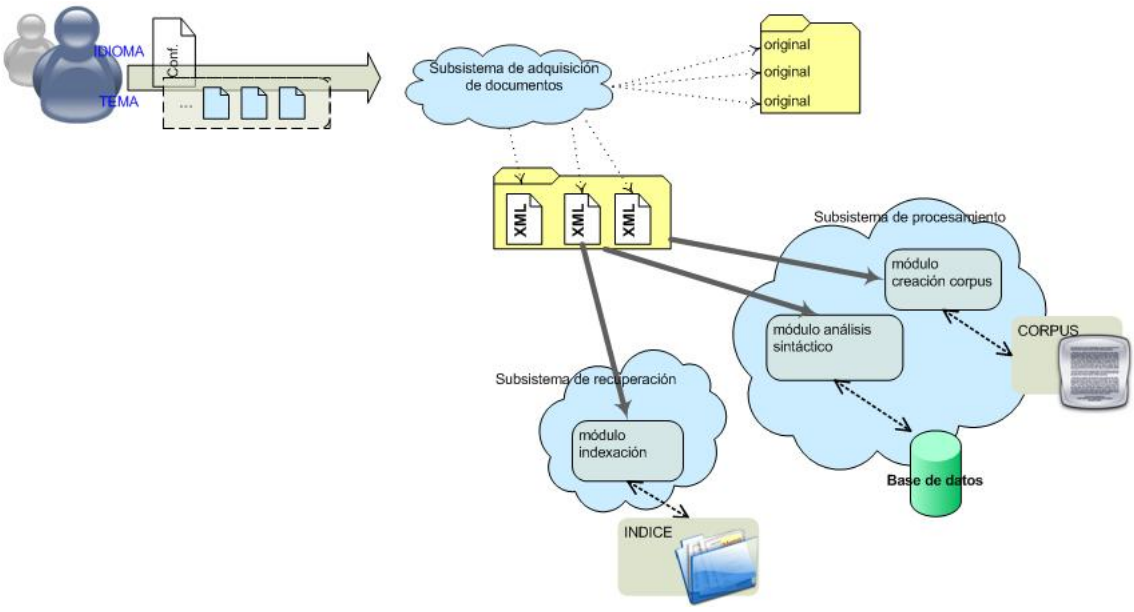


Figura 4 generación del corpus

CLASIFICACIÓN DEL CORPUS

La clasificación del corpus puede ser llevada a cabo utilizando dos enfoques, por un lado mediante la adquisición de terminología y por otro lado explotando el grafo de dependencias generado durante el análisis sintáctico.

Siguiendo el primer enfoque es necesario en un primer paso lanzar la extracción de términos. La capa mediador informa de cuales son los extractores disponibles para cada idioma. Una vez elegido un extractor, la capa mediador proporciona la información necesaria al módulo correspondiente del subsistema de procesamiento de la capa ejecución. De esta manera el módulo de extracción de términos con los recursos adecuados lanza la operación y genera unos resultados que son almacenados en un fichero con una estructura fija definida para tal caso.

Una vez se ha creado la terminología del dominio, se pueden clasificar los términos. Para ello se aplica en un primer momento una clasificación de los mismos, en base a sus contextos, y a continuación se estructuran los resultados en una ontología. El subsistema encargado de este cometido es el de ontología, concretamente su módulo *ontología por términos*.

Siguiendo el segundo enfoque se parte de los resultados obtenidos del análisis sintáctico. Estudiando las dependencias extraídas, el módulo de *ontología por dependencias* genera un conjunto de conceptos que pueden estructurarse en una

ontología. Para la realización de esta tarea el usuario habrá proporcionado el tema e idioma sobre el cual se está trabajando. Este proceso tiene una base iterativa, como se ha comentado anteriormente. El usuario lanzará tantas veces el proceso como lo desee. Entre cada ejecución podrá acceder a los ficheros oportunos (descritos en el apartado 13.2 del Manual Técnico) y realizar las modificaciones que considere oportunas, para ajustar los datos obtenidos a sus necesidades.

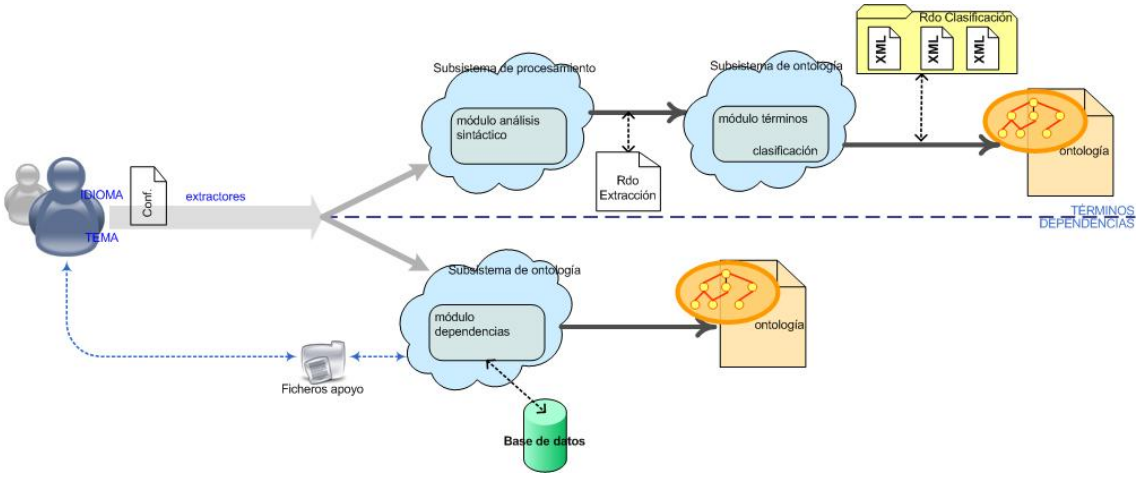


Figura 5 clasificación del corpus

BÚSQUEDA EN LOS DOCUMENTOS

Con esta actividad se realiza la tarea de recuperar los documentos relevantes a una consulta de usuario. A través de la interfaz, el usuario introduce una petición de recuperación de documentos.

La capa de recuperación examina la consulta. Dependiendo del tipo de búsqueda que se hay solicitado (bien a través de los términos, bien a través de las dependencias) la consulta será procesada de una forma u otra:

- Si la búsqueda es a través de los términos el sistema proporcionará términos relevantes a la consulta mediante la exploración de la correspondiente ontología.

- Si la búsqueda es a través de las dependencias, en un primer momento se recuperarán todos los idiomas relacionados al tema actual. Se efectúa un proceso de traducción y es posible, entonces, recuperar los términos relevantes a la consulta en los idiomas disponibles ya través de las respectivas ontologías.

La petición junto con los términos relacionados es manipulada para obtener los documentos relevantes en cada caso.

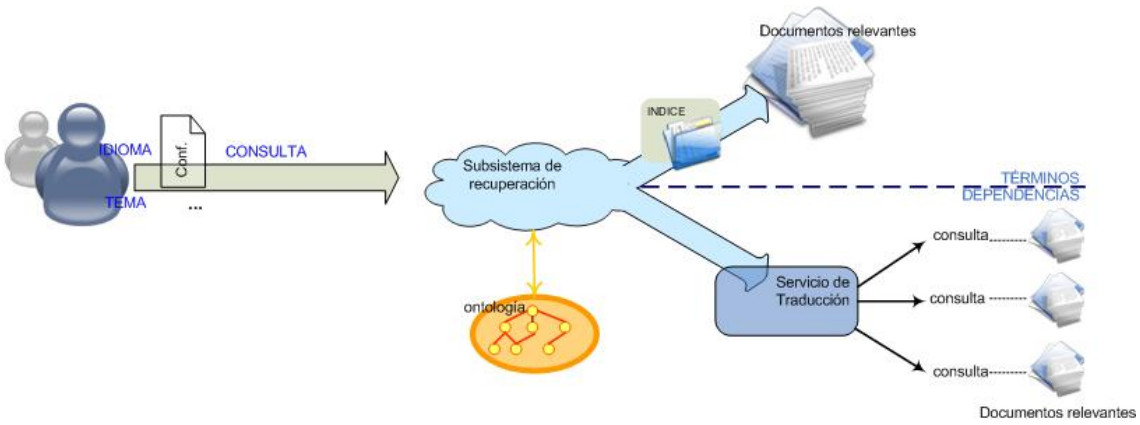


Figura 6 búsqueda de documentos relevantes

MODIFICACIÓN DE LA CONFIGURACIÓN.

Por último, la siguiente funcionalidad se basa en ofrecer la posibilidad de manejar los datos de configuración del sistema. Para ello, el usuario podrá añadir un nuevo tema así como borrarlo. Estas mismas tareas pueden ser aplicadas a los idiomas, extractores y analizadores sintácticos.

El hecho de cambiar la configuración con respecto a los extractores y analizadores sintácticos lleva implícito la incorporación del recurso que maneje los mismos. Es decir, cada vez que se añada un nuevo extractor o un nuevo analizador, deberá indicarse el recurso que maneja el mismo, a nivel de implementación esto debe ser visto como la clase que vaya efectuar la tarea. El sistema facilita la incorporación del recurso, realizando la carga automática del mismo en el momento que se solicite.

DESARROLLO DEL PROYECTO

4. METODOLOGÍA UTILIZADA

En este apartado se detallan diversos aspectos relacionados con el desarrollo del proyecto, como son la metodología empleada para el análisis y el diseño así como la solución adoptada para la implementación del modelo de proceso seguido

4.1 .LENGUAJE DE MODELADO

El análisis y diseño del sistema se desarrolló utilizando el Lenguaje Unificado de Modelado (UML: Unified Markup Language). Este lenguaje es el sucesor de diversos métodos de análisis y diseño orientados a objetos que surgió a finales de la década de los 80 y principios de los 90. Unifica en su seno los métodos de Rumbaugh, Jacobson y Booch. Hoy en día UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de desarrollo de código. A pesar de ser UML un lenguaje, posee más características visuales que programáticas.

Cuanto más complejo es el sistema que se desea crear, más beneficios presenta el uso de UML. Esto se debe a dos puntos básicos: (1) mediante una visión global resulta más fácil detectar las dependencias y dificultades implícitas del sistema; (2) los cambios en una etapa inicial resultan más fáciles de llevar a cabo que en una etapa final del sistema.

A continuación algunas características de UML:

- El sistema de software es diseñado y documentado antes de que sea codificado.
- Los lógicos 'agujeros' en la etapa de diseño podrán ser detectados con anterioridad. El software se comportará de la forma esperada y surgirán menos imprevistos.
- El diseño total del sistema dicta el modo en que se desarrollará el software. Las decisiones finales se harán antes de que se encuentre código mal escrito. Con esto se ahorra tiempo en el desarrollo.

- Cualquier modificación en el sistema será más fácil de llevar a cabo sobre la documentación UML.

En el proceso de desarrollo del proyecto se han utilizado los siguientes diagramas:

Diagrama de casos de uso: modela la funcionalidad del sistema agrupándola en descripciones de acciones ejecutadas para obtener un resultado.

Diagrama de clases: muestra las clases componentes del sistema y como se relacionan entre si. Una clase es una descripción descripciones de objetos que comparten características comunes.

Diagrama de secuencia: muestra la interacción entre los objetos y los mensajes que intercambian entre si, junto con el orden temporal de los mismos.

Diagrama de paquetes: muestra la organización de los subsistemas agrupando elementos de modelado. Cada paquete corresponde a un submodelo(subsistema) del modelo(sistema).

Además de estos diagramas, se ha utilizado también un diagrama de entidad/relación que a pesar de no estar incluido en la notación UML, se hace necesario para mostrar la notación y estructura empleada en el almacenamiento del análisis sintáctico.

4.1.1.MODELADO DEL SISTEMA

A continuación se detallan los diagramas realizado en cada una de las etapas de desarrollo del modelado del sistema.

ANÁLISIS

En el capítulo de análisis del Manual Técnico se describe el análisis del sistema. Esta fase comienza con la especificación de requisitos, que aunque no está definida en UML, es una parte fundamental del análisis para determinar las funciones del sistema así como sus restricciones.

Esta fase continúa con la elaboración de los diagramas de casos de uso y la descripción de cada uno de sus escenarios. Estos diagramas permiten comprender el

comportamiento del sistema desde el punto de vista de los usuarios. Posteriormente se procede a la elaboración de los diagramas de secuencia (uno por cada caso de uso) con los que se observará la interacción entre los objetos. Esta fase se completa con el diagrama de clases inicial.

DISEÑO

En el capítulo de diseño del Manual Técnico se detalla el diseño del sistema. En una primera etapa se desarrolla un diagrama de clases correspondiente a cada paquete. A diferencia del diagrama de clases del análisis, éstos aportan un mayor nivel de detalle al modelado del sistema. A continuación se prosigue con los diagramas de secuencia correspondientes a cada uno de los paquetes, y que aportan una mayor profundidad, aproximando al lector al sistema real implementado. Para completar esta fase, se aplican patrones de diseño que permiten optimizar y facilitar el desarrollo del sistema.

IMPLEMENTACIÓN

En el capítulo de implementación del Manual Técnico se describirán varios aspectos y tareas llevadas a cabo para la completa y correcta puesta en marcha del sistema. Se aclararán varios aspectos relacionados con el fichero de configuración. Se detallarán las DTD's que dan formato a los distintos ficheros de almacenamiento XML utilizados. También en este apartado se reflejará el diagrama de entidad/relación realizado para almacenar los resultados del análisis sintáctico.

4.2.CICLO DE VIDA

El ciclo de vida o modelo de proceso describe las fases principales de desarrollo de software y las tareas realizadas en cada una de las fases. Ayuda a administrar el progreso del desarrollo, y provee un espacio de trabajo para la definición de un proceso detallado de desarrollo de software[2]. Se pueden citar sus objetivos como:

- Definición de las actividades que se deben llevara cabo en un proyecto de desarrollo de software.
- Proporcionar unos puntos de control y de revisión administrativos, de tal forma que el sistema pueda ser evaluado.

Los pasos que proponen los modelos de ciclos de vida abarcan métodos, herramientas y procedimientos. La elección de uno de ellos depende del tipo de proyecto que se está efectuando. En el caso concreto que atañe este sistema se ha optado por el modelo de proceso de desarrollo conocido con el nombre de RUP.

RUP (Rational Unified Process), es uno de los modelos más generales de los que se utilizan actualmente. Es un proceso de ingeniería plantado por Kruchten en 1996, cuyo objetivo es producir software de alta calidad. Trata de cumplir con los requerimientos de los usuarios dentro de una planificación y presupuestos establecidos. RUP está dirigido por casos de uso, centrado en la arquitectura, iterativa (mini-proyectos) e incremental (versiones).

Toma en cuanta las mejores prácticas en el modelo de desarrollo de software en particular las siguientes:

- *Desarrollo de software en forma iterativa* (repite una acción). Un proceso iterativo permite una comprensión creciente de los requerimientos a la vez que crece el sistema. Aborda en primer lugar las tareas con más riesgos. Con esto se logra reducir los riesgos del proyecto y tener un subsistema ejecutable rápidamente.

[2] Cabe señalar que se han empleado ficheros XML para el almacenamiento de los documentos, la extracción de los términos y la clasificación de los términos, así como para estructurar el fichero de configuración.

- *Manejo de requerimientos*. RUP, describe como organizar los requerimientos y documentarlos.

- *Utiliza una arquitectura basada en componentes*. El proceso se basa en diseñar tempranamente una arquitectura base ejecutable. Esta arquitectura debe ser flexible, fácil de modificar, comprensible y debe promover la reutilización de componentes.

- *Modela el software visualmente* (modela mediante UML). Los bloques de construcción permiten ocultar detalles y analizar la consistencia desde dos puntos de vista: consistencia entre componentes y consistencia entre el diseño y la implementación.

- *Verifica la calidad del software*. No sólo la funcionalidad es esencial, también el rendimiento y la confiabilidad son dos requisitos básicos. RUP ayuda a planificar, diseñar, evaluar y ejecutar pruebas que verifiquen estas cualidades. Asegurar la calidad debe ser parte del propio proceso de desarrollo.

- *Controla los cambios*. RUP indica como controlar, rastrear y monitorear los cambios dentro del proceso iterativo de desarrollo.

4.2.1.CICLOS Y FASES

RUP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada uno. Cada ciclo se divide en cuatro fases:

Figura 7 ciclo de vida rup

Inicio: El objetivo en esta etapa es determinar la visión del proyecto. Se define la iteración de las distintas partes a un alto nivel de abstracción. En esta etapa se identificaran todos los casos de uso.

Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima. Para ello hay que analizar el dominio del problema y eliminar los elementos de mayor riesgo.

Construcción: El objetivo es obtener la capacidad operacional inicial. Es decir, se trata de desarrollar los componentes restantes e incorporarlos al producto.

Transmisión: El objetivo es llegar a obtener una versión del producto para su prueba con los usuarios. Una vez instalado surgirán nuevo elementos que implicaran nuevos ciclos. Esta etapa conlleva la realización de pruebas, entrenar a los usuarios y distribuir el producto.

Las iteraciones se realizaron por medio de una cascada como se muestra en la Figura 8. Se pasa por los flujos fundamentales (requisitos, análisis, diseño, implementación y pruebas). Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

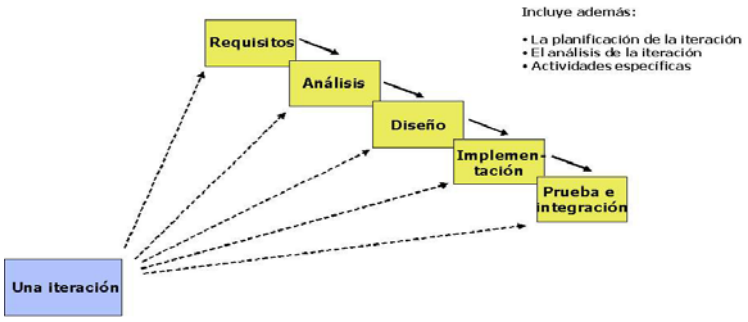


Figura 8 iteraciones en el ciclo de vida

5. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

A continuación se hace una breve descripción de las tecnologías y herramientas que se han empleado para el proyecto destacando las principales características de las mismas.

5.1. TECNOLOGÍAS EMPLEADAS

5.1.1. JAVA

Java es un lenguaje de programación con el que podemos realizar cualquier tipo de programa. En la actualidad muy extendido y popular, tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems (www.sun.com) con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras. Es un lenguaje orientado a objetos desarrollado al inicio de la década de los 90. Java inspira gran parte de su sintaxis en C y C++, pero tiene un modelo de objetos mucho más simple y elimina características de bajo nivel como los punteros. Sus principales características son:

Lenguaje simple.

Java posee una curva de aprendizaje muy rápida. Todos aquellos desarrolladores familiarizados con C++ encontrarán que Java es más sencillo, ya que se han eliminado ciertas características, como los punteros.

Orientado a objetos.

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como sus métodos (o funciones).

Interpretado y compilado a la vez.

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina llamado *bytecodes*, semejantes a las instrucciones de ensamblador. Por otra parte es interpretado ya que los *bytecodes* se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real.

Robusto.

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores, la aritmética de punteros, ya que se ha prescindido por completo de los mismos. La recolección de basura (*garbage collector*) elimina la necesidad de la liberación explícita de memoria.

Seguro.

El hecho de que Java sea interpretado también permite hacerlo seguro. Como la ejecución de los programas Java está controlada por el intérprete de Java, éste puede contener al programa y evitar que provoque efectos no deseados en el sistema.

Independiente de la arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos, desde Unix a Windows, pasando por Mac. La independencia de la plataforma es una de las razones por las que Java es interesante ya que muchas personas deben poder acceder desde distintos ordenadores. Pero no se queda ahí, Java está siendo desarrollado incluso para distintos tipos de dispositivos además del ordenador de sobremesa, como móviles y agendas.

Portable

Al ser los programas Java interpretados en lugar de compilados, resulta más sencillo ejecutarlos en una variedad de entornos.

Dinámico

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas.

Variedad en gestores de bases de datos

Java proporciona diversos controladores para el manejo de distintos gestores de bases de datos.

5.1.2.JDBC (JAVA DATABASE CONNECTIVITY)

JDBC, es un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java. Funciona de manera independiente al sistema operativo donde se ejecuta el código o de la base de datos a la cual se acceda. Emplea el lenguaje SQL para el acceso a los datos.

En JDBC, las tareas más sencillas sobre la base de datos, como son las peticiones, creaciones y actualizaciones, pueden realizarse empleando métodos simples y directos. Para tareas más complejas, como resultados múltiples o procesos de almacenamiento con parámetros de entrada y salida, JDBC tiene sentencias aparte.

El API en sí consiste en un conjunto de clases e interfaces que permiten que cualquier programa Java acceda a sistemas de bases de datos de forma homogénea. Es decir, con este API no es necesario escribir un programa para acceder a Oracle y otro para acceder a MySQL; se puede crear un solo programa que sea capaz de enviar sentencias SQL a la base de datos apropiada. Lo único imprescindible es que el programa Java tenga acceso a un controlador JDBC adecuado, esto es, al controlador específico del gestor de base de datos empleado. Es el controlador el que implementa la funcionalidad de todas las clases y proporciona la comunicación entre el API JDBC y la base de datos real.

5.1.3.XML (EXTENSIBLE MARKUP LANGUAGE)

XML es un lenguaje para estructurar la información en un documento o en general, en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años por ser un estándar abierto y libre, creado por el consorcio *World Wide Web*, W3C (los creadores de la *www*) en colaboración con representantes de las principales compañías productoras de software.

Se presentan a continuación algunas características de XML que se han tenido en cuenta a la hora de elegirlo como formato predeterminado en la elaboración del proyecto:

- Es un estándar basado en un conjunto de reglas para la definición de etiquetas semánticas que organizan los documentos en diferentes secciones.
- Es una arquitectura más abierta y extensible. Los identificadores pueden crearse de manera sencilla y ser adaptados.
- Mayor consistencia, homogeneidad y amplitud de los identificadores descriptivos de los documentos.
- La codificación del contenido en XML consigue que la estructura de la información resulte más accesible. Además, la independencia entre el contenido de los datos y la presentación de los mismos hace de XML un formato adecuado para el desarrollo de un sistema como el que aquí se presenta.
- Es un formato ideal para guardar datos de configuración de las aplicaciones.
- Existen abundantes herramientas para trabajar con información representada con el formato XML. Estas herramientas facilitan el acceso, tratamiento y transformación de los datos. Dos de estas herramientas se tratan a continuación, los APIs DOM y SAX.

El tratamiento de un documento XML pasa por la utilización de un analizador. Los analizadores disponibles para XML ofrecen los servicios de serializar y deserializar documentos con este formato.

5.1.4.SAX (SIMPLE API FOR XML)

Es una librería que permite analizar y recuperar los datos de un archivo XML.

Esta librería funciona por eventos y métodos asociados. A medida que el analizador SAX analiza el documento va encontrando los distintos componentes del mismo, es decir, elementos, atributos y valores. Estos componentes son lo que se conoce como *eventos*, la detección de uno de ellos desencadena una acción asociada, conocida con el nombre de *método*. A medida que se recorre el archivo, SAX detecta la presencia de eventos invocando a los métodos que el programador ha desarrollado.

El API consta de una serie de clases con sus correspondientes métodos, que permiten trabajar con un documento XML desde un programa escrito en Java, pudiendo acceder a los datos, comprobar si está bien formado, si es válido, etc. Su principal característica es que el fichero es leído secuencialmente de principio a fin, sin tener que cargar todo el documento en memoria.

SAX se ha empleado para recorrer el archivo devuelto por el extractor de términos Acabit (Sección 13). Se ha optado por le mismo debido al tamaño del fichero XML devuelto de la extracción que resultaba demasiado extenso para su carga en memoria.

5.1.5.JDOM (JAVA DOCUMENT OBJECT MODEL)

JDOM es un API para leer, manipular y crear documentos XML de forma cómoda desde un programa Java. Al ser un API específica para Java es mucho más intuitiva y sencilla de utilizar que la anterior.

JDOM utiliza DOM para manipular los elementos del fichero XML. DOM es la especificación del W3C que permite manipular y recorrer un documento XML como si se tratase de un árbol de nodos. DOM proporciona una representación en memoria del documento bajo la forma de un árbol de objetos, el recorrido y manipulación del documento se realiza de forma mucho más intuitiva.

JDOM ha sido específicamente creado para desarrollar aplicaciones que engloben a XML y DOM. Permite por lo tanto construir documentos, navegar por su estructura y modificar el contenido del mismo.

JDOM se ha empleado en el fichero de configuración, debido a su escaso tamaño los recursos de memoria no eran excesivos y debido a la modificación que dicho fichero podría sufrir en tiempo de ejecución. También se ha utilizado en la creación del fichero resultado de la extracción y en el fichero de clasificación de los términos.

5.1.6.SWING

Swing es una biblioteca gráfica para Java que forma parte de las *Java Foundation Classes* (JFC). Incluye *widgets* para la interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas.

Swing existe desde la JDK 1.1 (como un agregado). Antes de la existencia de Swing, las interfaces gráficas con el usuario se realizaban a través de AWT (*Abstract Window Toolkit*), de quien Swing hereda todo el manejo de eventos. Usualmente, para toda componente AWT existe una componente Swing que la reemplaza.

Son muchas las ventajas que ofrece el uso de Swing. Por ejemplo, la navegación con el teclado es automática, cualquier aplicación Swing se puede utilizar sin ratón. Todo ello, sin tener que escribir ni una línea de código adicional.

5.1.7.LUCENE

Lucene es una herramienta que permite tanto la indexación cómo la búsqueda de documentos. Creada bajo una metodología orientada a objetos e implementada completamente en Java, no se trata de una aplicación que pueda ser descargada, instalada y ejecutada sino de una API a través de la cual se pueden añadir capacidades de indexación y búsqueda a cualquier sistema.

5.1.8.OWL

OWL es un lenguaje desarrollado por el *W3C Ontology Working Group (WebOnt)* para publicar y compartir ontologías en la Web. El OWL se deriva de DAML+OIL y, al igual que este lenguaje, es una extensión de RDFS.

OWL ofrece la posibilidad de escribir ontología. Se diferencia principalmente de RDF y RDFS en que éstos son lenguajes para definir grafos, mientras que OWL es un lenguaje que nace específicamente para la creación de ontologías.

5.1.9.JENA

Para desarrollar aplicaciones basadas en OWL, se precisan librerías para leer y procesar las ontologías definidas en este lenguaje. Existen una gran multitud de analizadores y herramientas que se han desarrollado para tal efecto. Sin embargo, el analizador de OWL más popular es Jena. Permite leer, recorrer y modificar grafos OWL desde un programa Java.

Jena es una API para el lenguaje de programación Java, desarrollada por Brian McBride de Hewlett-Packard, utilizada para la creación y manipulación de grafos RDF. El API Jena posee objetos, clases para representar los modelos, recursos, propiedades y literales.

En Jena, un grafo es denominado modelo y es representado mediante la interface Model.

Para dar soporte a la manipulación de ontologías, la segunda versión de Jena posee un paquete específico, llamado “API Jena 2 Ontology”. En este paquete existen clases para la manipulación de ontologías en RDFS, DAML+OIL y OWL. Para dar soporte a estos lenguajes de ontologías la API posee las siguiente clases: OntClass y ObjectProperty.

5.1.10.SQL (STRUCTURED QUERY LANGUAGE)

La comunicación con la base de datos se ha realizado mediante sentencias SQL. Éste es un lenguaje estándar que se ha visto consolidado por el Instituto Americano de Normas (ANSI) y por la Organización de Estándares Internacional (ISO-SQL 99).

Es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos

Es un lenguaje de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro. SQL proporciona una rica funcionalidad más allá de la simple consulta (o recuperación) de datos.

5.1.11.PERL

Es un lenguaje creado por Larry Wall con el objetivo principal de simplificar las tareas de administración de un sistema UNIX. Hoy en día se ha convertido en un lenguaje de propósito general.

PERL es un lenguaje que hereda ciertas estructuras de los intérpretes de comandos de UNIX y de otras utilidades estándar, como `awk` y `sed`. Está diseñado para realizar las mismas funciones que dichas utilidades, pero de manera conjunta y de forma más comprensible y fácil de depurar. Es un lenguaje interpretado, aunque el intérprete PERL compila los programas, antes de ejecutarlos, al lenguaje máquina nativo del ordenador y sistema operativo en él que se ejecuta.

Aunque desarrollado originalmente en un entorno UNIX, actualmente hay versiones para casi todos los sistemas operativos: Windows XP, MacOS. Los programas PERL son compatibles entre las diversas plataformas, de forma que es un verdadero lenguaje multiplataforma.

Programar en PERL no es vistoso, pero sí funcional. La facilidad de manipulación de archivos y textos que otorga, permite que sea ampliamente empleado para actividades que incluyen el desarrollo rápido de prototipos, sistemas de utilerías, herramientas de software, actividades de administración de sistemas, acceso a bases de datos, programación gráfica y redes.

5.1.12.TWIG

Es un módulo PERL que proporciona un modo de procesar los documentos XML. Ofrece una interfaz de árbol al documento. Permite la salida de partes que hayan sido completamente procesadas. El uso de CPU y memoria es mínimo al construir el árbol tan sólo para las partes de los documentos que necesitan procesamiento actual. La idea es, entonces, de procesar tan sólo las ramas (*twigs*) del árbol, a las que aplica un modelo de datos basado en DOM, pero descartando el resto. A partir de aquí es posible procesar esas ramas; suprimir o añadir elementos o modificar su contenido.

Esta librería se ha utilizado para el proporcionar la entrada al extractor Acabit (ver la sección: *13 acabit* de este manual).

5.2.HERRAMIENTAS EMPLEADAS

En este apartado se hace una breve descripción de las herramientas utilizadas para llevar a cabo las diferentes tareas de este proyecto. Para ello, se indica cual ha sido el cometido de cada una de ellas durante la realización del proyecto junto con algunas de sus principales características.

5.2.1.EL IDE ECLIPSE

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

El IDE de Eclipse[3] es un entorno de desarrollo libre para crear aplicaciones clientes de cualquier tipo. El proyecto Eclipse es un proyecto de desarrollo de software de código abierto dedicado a proporcionar una plataforma industrial robusta, con amplias características y con calidad comercial para el desarrollo de herramientas altamente integradas.

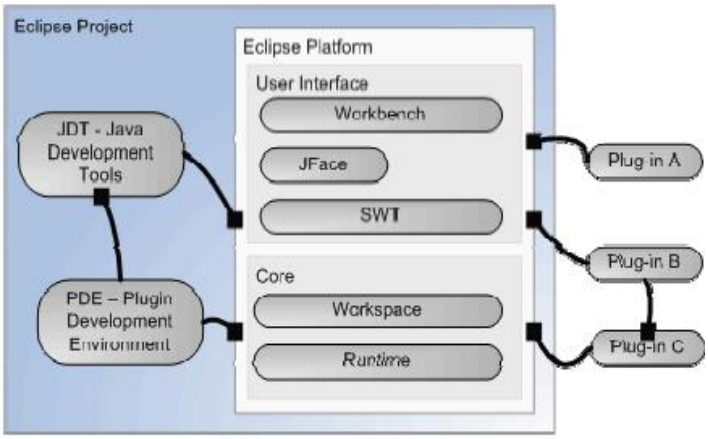


Figura 9 IDE Eclipse

[3] En la comunidad informática es común llamar a este IDE 'Eclipse'. Sin embargo como se describe en la presente documentación, Eclipse es un proyecto de desarrollo de software que engloba varios subproyecto, uno de los cuales es el IDE. En lo sucesivo de esta documentación cuando se haga referencia al IDE nos referiremos a él directamente con el término Eclipse.

Eclipse está formado por su propio núcleo y por multitud de *plugins* que yacen sobre éste. Este tipo de arquitectura basada en *plugins* es la que le proporciona una flexibilidad y escalabilidad muy elevadas. Todo componente con funcionalidad es tratado como un *plugin* en Eclipse. El SDK de Eclipse incluye la plataforma básica y dos herramientas principales que resultan de utilidad para el desarrollo de *plugins*. Las herramientas de desarrollo Java (JDT) implementan un entorno de desarrollo Java de función completa.

En la Figura 9 anterior puede observarse la arquitectura de Eclipse. La plataforma básica la forman el núcleo, el entorno de trabajo (*Workspace*) y el área de desarrollo (*Workbench*). Puede encontrarse toda la información referente a Eclipse en la página web *www.eclipse.org.*

Eclipse se ha utilizado a lo largo de todo el desarrollo para la generación del código fuente.

5.2.2.NETBEANS

NetBeans es una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un Entorno integrado de desarrollo (IDE).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con los APIs de NetBeans y un archivo especial (*manifest file*) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios. Se ha utilizado para generar la interfaz gráfica de la aplicación.

5.2.3.MYSQL

MySQL es un sistema de gestión de bases de datos relacional, bajo la licencia GNU/GPL. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

MySQL surgió cómo una necesidad de un grupo de personas sobre un gestor de bases de datos rápido, por lo que sus desarrolladores fueron implementando únicamente lo que precisaban, intentando hacerlo funcionar de forma óptima. Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Sin lugar a duda, lo mejor de MySQL es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
- Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.

Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación. Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

MySQL se ha utilizado como gestor de base de datos a lo largo del proyecto.

5.2.4.JDK 1.5

Puesto que el desarrollo del sistema se ha llevado a cabo con el lenguaje de programación Java, se hace necesario el uso de un compilador, así como de un entorno de ejecución.

Java Development Kit se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones en Java,

proporcionado por Sun Microsystem. Incluye las librerías básicas de Java, el JRE (*Java Runtime Environment*), un compilador y otras funcionalidades definidas por Sun. Proporciona por lo tanto el entorno de ejecución necesario para ejecutar programas hechos en Java.

5.2.5.VISUAL PARADIGM

Visual Paradigm es una herramienta CASE que utiliza UML como lenguaje de modelado (*www.visual-paradigm.com*). La herramienta está diseñada para una amplia gama de usuarios interesados en construir sistemas de software fiables con el uso del paradigma orientado a objetos, incluyendo actividades como ingeniería de software, análisis de sistemas y análisis de negocios. Esta herramienta ha sido utilizada en el desarrollo de este proyecto para la elaboración del análisis y del diseño

5.2.6.MICROSOFT PROJECT 2003

Microsoft Project 2003 ha sido la herramienta utilizada para la realización de la planificación temporal de las tareas implicadas en el desarrollo del proyecto. Proporciona una ayuda en la asignación de recursos, asignación de tares, etc.

Se trata de un programa de administración de proyectos de *Microsoft Office System*, incluye *Microsoft Project Professional* y *Microsoft Project Server*. Juntos proporcionan una solución eficaz de Administración de proyectos empresariales (EPM, *Enterprise Project Management*) que permite a las organizaciones alinear iniciativas de negocio, proyectos y recursos con el fin de obtener mejores resultados empresariales.

5.2.7.MICROSOFT WORD 2003

Microsoft Word es el procesador de texto empleado en la redacción y maquetación de la documentación del proyecto. Se consigue de esta manera que la cantidad de información generada durante la realización del proyecto se vea adecuadamente estructurada y presentada al lector.

Microsoft Word está integrado en la paquete de ofimática *Microsoft Office*.

5.2.8.KATE (KDE ADVANCED TEXT EDITOR)

Kate es un editor de texto para KDE. Ha sido parte del paquete *kdebase* de la versión 2.2 de KDE. Es posible colocar *Kate* como un componente de edición en cualquier otra aplicación KDE. Entre otras características *Kate* incluye:

- Seguimiento de código para C++, C, PERL, y otros.
- Búsqueda y reemplazo de texto usando expresiones regulares.
- Mantener múltiples documentos abiertos en una ventana.
- Soporte de sesiones.
- Manejador de archivos.
- Emulador de un terminal, basado en Konsole.

Se ha empleado este editor de texto para todas aquellas tareas relacionadas con las herramientas propias de PLN, así como para la elaboración del fichero de configuración. Como se verá en capítulos sucesivos, los recursos de PLN utilizados tuvieron que ser modificados para adaptarlos a la necesidad concreta de esta aplicación. Dichos cambios se realizaron editando los ficheros correspondientes con Kate.

5.2.9.HERRAMIENTAS DE PLN

A continuación se mencionan las herramientas utilizadas para el desarrollo de tareas específicas al procesamiento del lenguaje natural. En las sucesivas secciones se detallan cada una de ellas.

(A) ETIQUETADOR

Un etiquetador es una herramienta que proporciona una categoría léxica a las palabras de un texto o frase que se le pasa como entrada. Su funcionamiento se puede basar en conocimiento lingüístico o en la aplicación de probabilidades.

El etiquetador utilizado en el desarrollo del proyecto ha sido TreeTagger. Un etiquetador estadístico basado en árboles de decisión y con diccionarios para el francés y el español.

(B) EXTRACTOR DE TÉRMINOS

Un extractor de términos es una herramienta de explotación de documentos para el establecimiento de un conjunto de términos relevantes, asociados al dominio específico referido en los textos.

En el sistema actual, la búsqueda de los extractores se ha realizado en función del idioma que se debía tratar. Así se han empleado dos extractores para el francés, Acabit y Fastr; y dos extractores para el español, uno de ellos desarrollado por Pablo Gamallo (miembro del grupo *Syntax* de la USC[4]) y el otro ha sido una adaptación de Fastr realizada específicamente para este proyecto.

(C) ANALIZADOR SINTÁCTICO

Un analizador sintáctico determina cómo se combinan las palabras de un texto para dar lugar a sintagmas, así como el papel estructural de cada sintagma en la frase.

En el desarrollo del proyecto se ha empleado un único analizador sintáctico, Erial. Éste analizador sintáctico se ha desarrollado en el seno del grupo COLE[5], para el idioma español. Además de aplicarle ciertas modificaciones para adaptarlo a nuestras necesidades en español, a su vez se ha adaptado para que funcionase para el francés.

5.2.10.LINUX-UBUNTU

Por ultimo cabe destacar que el sistema operativo empleado a lo largo de todo el desarrollo ha sido la distribución UBUNTU de Linux. Se ha empleado UBUNTU para todas las tecnologías y herramientas citadas, salvo en el uso de recursos Microsoft. La documentación así como la planificación temporal se llevaron acabo con el sistema operativo Windows XP Media Center Edition.

[4] http:// www.usc.es

[5] http://www.grupocole.org

UBUNTU es un sistema operativo de código abierto desarrollado en torno al kernel Linux. Se citan como principales características:

- UBUNTU se publica regularmente, una nueva versión sale cada seis meses.
- UBUNTU está totalmente comprometido con los principios de desarrollo del software de código abierto.
- Basado en Debian (una de las distribuciones tecnológicamente avanzadas y mejor soportadas), UBUNTU incluye una cuidadosa selección de los paquetes de Debian, y mantiene su poderoso sistema de gestión de paquetes que nos permite instalar y desinstalar programas de una forma fácil y limpia. A diferencia de la mayoría de las distribuciones, que vienen con una enorme cantidad de software que pueden o no ser de utilidad, la lista de paquetes de UBUNTU se ha reducido para incluir sólo aplicaciones importantes y de alta calidad.
- UBUNTU proporciona un entorno robusto y funcional, adecuado tanto para uso doméstico como profesional.

6. PLANIFICACIÓN Y PRESUPUESTO

6.1. PLANIFICACIÓN TEMPORAL

A continuación se muestran las dos planificaciones realizadas para el desarrollo de este proyecto. La primera de ellas es la planificación prevista y fue realizada antes del inicio del sistema. La segunda mostrada, es la planificación real que se realizó al finalizar el proyecto para poder observar las desviaciones producidas y las causas de las mismas.

6.1.1. PLANIFICACIÓN PREVISTA

Las fases planificadas para el desarrollo del proyecto fueron las siguientes:

- **Estudios previos**: En esta fase se produce un primer acercamiento al sistema, estudiando cuales son los objetivos que se persiguen y la posible solución a los mismos. Además se incluye en esta fase el estudio de las herramientas de PLN que se van a incorporar al sistema. Esto implica una búsqueda de las herramientas que mejor se adapten a las necesidades del sistema así como, una investigación del funcionamiento de las mismas.

- **Fase de análisis y diseño**: En la etapa de análisis se hace uso de la especificación de requisitos, los casos de uso, el diagrama de clases inicial y los diagramas de secuencia para hacer una descripción detallada del sistema, así como capturar todas las especificaciones. Durante la etapa del diseño se profundizan los conceptos del análisis, detallando los diagramas de secuencia, refinando los diagramas de clases, aplicando patrones de diseño y diseñando la base de datos.

- **Fase de implementación**: En esta fase se ha incluido el aprendizaje de las tecnologías necesarias para la implementación del sistema. Es en esta fase que se sitúa la implementación completa del sistema.

- **Fase de funcionamiento del sistema:** En la que se comprueba que el sistema funciona de la forma esperada.

- **Fase de pruebas:** En esta fase se realizan las pruebas finales del sistema que prueban situaciones críticas recogiendo los posibles errores.

- **Documentación**: Durante todo el desarrollo del sistema se ha creado la documentación del proyecto, documentando cada una de las fases realizadas.

Se muestra a continuación la planificación temporal prevista al inicio del proyecto, en la cual se consideraron 8h de trabajo diarias:

Nombre de tarea	Duración	Comienzo	Fin
− PROYECTO	**118 días**	**lun 02/10/06**	**lun 09/04/07**
− Estudios previos	**30 días**	**lun 02/10/06**	**mar 14/11/06**
Estudio del problema propuesto	8 días	lun 02/10/06	mié 11/10/06
Búsqueda e investigación de extractores	16 días	lun 09/10/06	mar 31/10/06
Búsqueda e investigación de analizadores sintácticos	16 días	lun 23/10/06	mar 14/11/06
− Fase de Análisis y Diseño	**17 días**	**mié 15/11/06**	**mar 12/12/06**
Análisis de la aplicación	8 días	mié 15/11/06	vie 24/11/06
Diseño de la aplicación	9 días	lun 27/11/06	mar 12/12/06
− Fase de Implementación	**60 días**	**lun 11/12/06**	**lun 19/03/07**
Estudio de las tecnologías	8 días	lun 11/12/06	mié 20/12/06
Desarrollo del sistema	55 días	lun 18/12/06	lun 19/03/07
Puesta en funcionamiento	4 días	mar 20/03/07	vie 23/03/07
Pruebas Finales	10 días	mar 20/03/07	lun 02/04/07
− Documentación	**118 días**	**lun 02/10/06**	**lun 09/04/07**
Documentación del problema propuesto	6 días	lun 02/10/06	lun 09/10/06
Documentación de la investigación	25 días	lun 09/10/06	mar 14/11/06
Documentación del análisis	8 días	mié 15/11/06	vie 24/11/06
Documentación del diseño	20 días	lun 27/11/06	jue 04/01/07
Manual Técnico y de Usuario	60 días	vie 05/01/07	lun 09/04/07

Figura 10 planificación temporal inicial

Se muestra a continuación el diagrama de Gantt de la planificación temporal inicial:

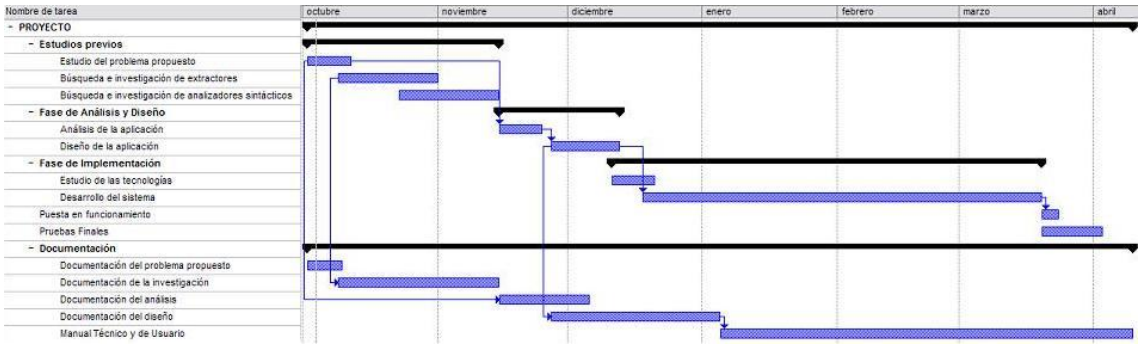


Figura 11 diagrama de gantt inicial

6.2.DURACIÓN REAL

Las siguientes figuras muestran la duración real del proyecto desarrollado:

Nombre de tarea	Duración	Comienzo	Fin
PROYECTO	**168 días**	**lun 02/10/06**	**lun 18/06/07**
Estudios previos	**54 días**	**lun 02/10/06**	**jue 21/12/06**
Estudio del problema propuesto	8 días	lun 02/10/06	mié 11/10/06
Búsqueda e investigación de extractores	26 días	lun 09/10/06	mié 15/11/06
Búsqueda e investigación de analizadores sintácticos	26 días	lun 13/11/06	jue 21/12/06
Fase de Análisis y Diseño	**23 días**	**lun 18/12/06**	**jue 25/01/07**
Análisis de la aplicación	12 días	lun 18/12/06	mié 10/01/07
Diseño de la aplicación	11 días	jue 11/01/07	jue 25/01/07
Fase de Implementación	**85 días**	**lun 22/01/07**	**mar 29/05/07**
Estudio de las tecnologías	15 días	lun 22/01/07	vie 09/02/07
Desarrollo del sistema	70 días	lun 12/02/07	mar 29/05/07
Puesta en funcionamiento	5 días	mié 30/05/07	mar 05/06/07
Pruebas Finales	10 días	lun 04/06/07	vie 15/06/07
Documentación	**168 días**	**lun 02/10/06**	**lun 18/06/07**
Documentación del problema propuesto	8 días	lun 02/10/06	mié 11/10/06
Documentación de la investigación	55 días	lun 09/10/06	lun 08/01/07
Documentación del análisis	15 días	lun 18/12/06	lun 15/01/07
Documentación del diseño	20 días	jue 11/01/07	mié 07/02/07
Manual Técnico y de Usuario	86 días	jue 08/02/07	lun 18/06/07

Figura 12 planificación temporal real

Se muestra a continuación el diagrama de Gantt de la planificación temporal real:

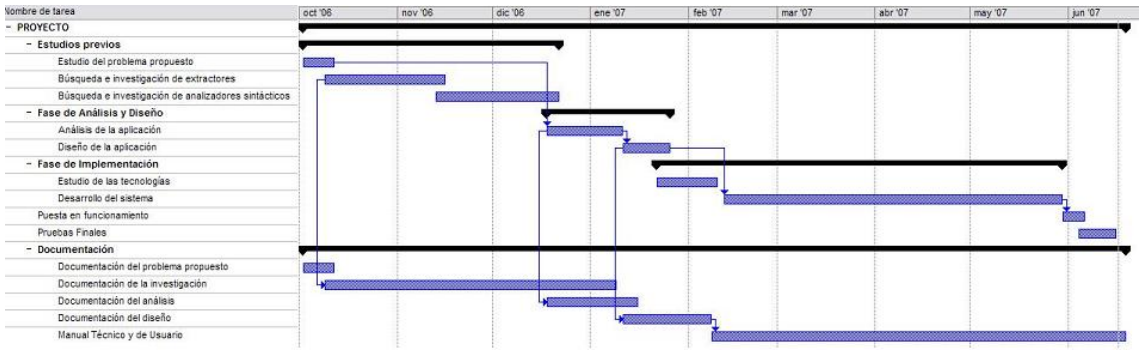


Figura 13 diagrama de gantt real

6.3.DESVIACIÓN TEMPORAL

Se puede observar que las fases de desarrollo inicialmente previstas siguieron siendo las mismas durante el desarrollo del sistema. Sin embargo se producen ciertas desviaciones en el tiempo, esto se debe a los siguientes motivos:

- La inexperiencia en la realización de planificaciones de este tipo y el optimismo a la hora de realizar la planificación inicial es una de las principales causas de las desviaciones temporales ocurridas.

- La fase de estudios previos incrementa el tiempo en las etapas de búsqueda e investigación de herramientas. Esto se debe al hecho de existir muy pocos recursos de código libre adecuados a las necesidades que se buscaban. Sumado a esto está el hecho de tener que adaptar aquellas herramientas encontradas a las exigencias específicas del sistema.

- La fase de análisis tuvo una duración mayor de la prevista debido a las nuevas especificaciones que han ido surgiendo.

- La fase de diseño se ha visto incrementada por el hecho de investigar y aplicar patrones de diseño, pues aunque se parte de una base teórica al respecto, nunca se había modelado un sistema haciendo uso de los mismos.

- La fase de implementación se ve considerablemente aumentada debido principalmente al estudio de las tecnologías, como por ejemplo, *JDOM* y *SAX* y a la necesidad de investigación de nuevas tecnologías inicialmente no

consideradas como fue el caso del API *Jena* para el manejo de ontologías en *OWL*.

6.4.PRESUPUESTO

En este apartado, se establece el presupuesto que es necesario para desarrollar el proyecto. Se ha establecido una separación de los costes de los recursos físicos, tanto a nivel de software como a nivel de hardware y a la cuantía referente a los recursos humanos.

6.4.1.RECURSOS FÍSICOS

RECURSOS DE HARDWARE

Hardware	Coste
Intel Pentium Centrino Core Duo2. 1.6 GHz; 1024MB RAM; HD: 100GHz	1099,00 €
Conexión a internet	9 meses x 20 €/mes =180€
Total (iva incluido)	1.483,64 €

RECURSOS DE SOFTWARE

Software	Uso	Coste
Micrcosoft Project 2003	Elaboración de las planificaciones temporales.	228,75€
Microsoft Word 2003	Elaboración de la documentación.	338,00€
Eclipse	Codificación de la aplicación.	0,00 €
Recursos de PLN	Para las tareas de extracción de términos y análisis sintáctico.	0,00 €
MySQL	Gestor de la base de datos.	0,00 €
Ubuntu	Sistema operativo de desarrollo.	0,00 €
	Total (iva incluido)	657,43€

RESUMEN DE LOS RECURSOS FÍSICOS

A los recursos físicos se le aplica una amortización del 25%.

Recurso	Coste
Hardware	1.483,64 €
Software	657,43 €
Total (iva incluido)	535,27 €

6.4.2.RECURSOS HUMANOS

Para la elaboración del proyecto ha existido un único desarrollador, que se ha ocupado de realizar todas las fases descritas en la planificación.

Se ha estimado un coste por horas de trabajo realizado, que variará según la actividad considerada.

Fase de desarrollo	Días x (horas/día)	Coste (€/hora)	Coste total
Estudio previo	54 x 6 = 324	15 €	4.860 €
Análisis y diseño	23 x 6 = 138	18 €	2.484 €
Implementación	85 x 6 = 510	18 €	9.180 €
Documentación	168 x 2 = 336	15 €	5.040 €
		Coste final (iva incluido)	25.014,24 €

6.4.3.COSTE DEL PROYECTO

El coste total del proyecto, incluyendo los recursos físicos y humanos se resume en la siguiente tabla:

Recurso	Coste
Recursos físicos	535,27 €
Recursos humanos	25.014,24 €
Coste final del proyecto (iva incluido)	25.549,51 €

RECURSOS DE PLN

7. INTRODUCCIÓN AL PROCESAMIENTO DEL LENGUAJE NATURAL

El lenguaje es el medio de comunicación humano por excelencia, realizado a través de signos orales y/o escritos que poseen un significado. Algunos lingüistas entienden el lenguaje como la capacidad humana que conforma al pensamiento. En su forma escrita permite, además, fijar y transmitir el conocimiento.

El estudio del lenguaje puede ser enfocado desde dos puntos de vista, por un lado un enfoque centrado en su uso y por otro lado un enfoque centrado en su estructura.

El análisis del uso del lenguaje trata de lo que dicen las personas, lo que piensan que dicen y lo que significa aquello que desean comunicar. Todos estos factores están condicionados por los cambios lingüísticos y sociales que afectan a la conducta humana.

El análisis de la estructura del lenguaje concierne a la lingüística. Se centra en la comunicación escrita, la base de este proyecto. Se persigue el estudio de la estructura de los textos.

La información es, sin duda, uno de los principales factores de poder en el mundo actual. Existen toda clase de volúmenes cuya información se encuentra en lenguaje natural ha decir libros, periódicos, informes, etc. Cada uno de ellos almacenados en formato impreso o en formato digital. Pero la explotación valiosa de esa información implica la ejecución de ciertas operaciones como la búsqueda de datos, la inferencia de conclusiones y el manejo de los textos, para, por citar un ejemplo, tareas de traducción. En este punto es donde entran en juego el procesamiento del lenguaje natural y los ordenadores, capaces de procesar la información de manera más rápida que los seres humanos.

Así, podemos definir el Procesamiento del Lenguaje Natural (PLN) como la rama de las ciencias de la computación encargada del análisis, diseño e implementación de los elementos software y hardware necesarios para el tratamiento del lenguaje

natural[6]. Es decir, se trata de conseguir que las herramientas informáticas provean capacidades de análisis de los textos y de extracción de información.

7.1.NIVELES DE ANÁLISIS

Para cumplir con su objetivo un sistema de PLN necesita grandes cantidades de conocimiento acerca de la estructura del lenguaje. Concretamente en este apartado se tratan los niveles que son pertinentes a la aplicación realizada para el proyecto y paralelamente se describe en qué consiste el análisis de cada uno de esos niveles.

Conocimiento morfológico: Determina cómo son las palabras que forman el lenguaje. El análisis en este punto consiste en la asignación de una etiqueta morfosintáctica a cada una de las palabras que formen el corpus. El caso concreto empleado en el sistema ha sido TreeTagger descrito en la sección *8 etiquetación: treetagger*.

Conocimiento sintáctico: Determina cómo se combinan las palabras para dar lugar a sintagmas, así como el papel estructural de cada sintagma en la frase. El análisis en este punto consiste en obtener las relaciones entre las palabras de una misma frase. El caso concreto empleado en el sistema ha sido un analizador sintáctico superficial y robusto desarrollado en el seno del grupo COLE en el proyecto Erial y descrito en la sección *9 análisis sintáctico: erial*.

A caballo entre estos niveles se sitúan otro tipo de herramientas cuyo conocimiento implica una base morfológica y sintáctica. Se pueden citar en este nivel los extractores de términos que se presentan más adelante.

En los apartados siguientes se tratan tareas relacionadas con los distintos niveles de aplicación de técnicas de PLN, así como los recursos lingüísticos asociados. Así, tenemos una descripción del etiquetador utilizado, TreeTagger (Sección: 8), una descripción del analizador sintáctico empleado en Erial (Sección: 9), una definición del pre-procesado de los textos llevado a cabo en una etapa inicial (Sección: 10); y por último una introducción a la adquisición de terminología (Sección: 12).

[6] Es conveniente distinguir éstos de los lenguajes formales, propios del ámbito matemático, lógico o computacional.

8. ETIQUETACIÓN: TREETAGGER

NOMBRE	TreeTagger
AUTOR	Helmut Schmid
AÑO	1994
IDIOMAS	Francés y Español entre otros

Éste es un etiquetador estadístico basado en árboles de decisión, con diccionarios para el francés y el español entre otros. Incluye además un módulo de segmentación que sitúa cada palabra, con su correspondiente análisis, en una línea distinta. TreeTagger proporciona a su salida el lema y la etiqueta de la palabra examinada.

La decisión que toma su autor [2] para la implementación de esta herramienta se basa en que los etiquetadores estadísticos tienen en cuenta sólo las ocurrencias con probabilidades altas, cuando en muchos casos las probabilidades bajas se deben a ocurrencias aisladas pero correctas. El autor suple esta deficiencia en su etiquetador introduciendo medidas de suavizado en los valores pequeños.

Los árboles de decisión contemplan la extracción de un contexto de tamaño variable, permitiendo así elegir una categoría de forma más precisa. En la mayoría de los etiquetadores este hecho no se contempla y el contexto se sitúa entre 1 a 3 trigramas. La base del TreeTagger es muy similar a la de los etiquetadores de n-gramas, pero difieren en la toma de decisión de la transición correcta entre categorías.

8.1 .FUNCIONAMIENTO Y CONSTRUCCIÓN DEL ÁRBOL DE DECISIÓN

El árbol de decisión empleado por esta herramienta es binario. El recorrido del mismo en busca de la etiqueta más adecuada se realiza tomando en cada nodo el camino más probable hasta llegar a un nodo hoja. En cada nodo hoja existe información probabilística que ayudan a la toma de decisión.

Por ejemplo en una secuencia de *Determinante Adjetivo Sustantivo (DET ADJ NB)*, donde lo que se trata de averiguar es la etiqueta *NB*, el recorrido empezaría en el root buscando la etiqueta *DET* como primera etiqueta de la secuencia. En dicho nodo se tomaría la decisión de si el siguiente es un *NB*. Como no es el caso el camino nos llevaría por la transición **SI**. A continuación preguntaría por un *ADJ* la respuesta sería si, y nos llevaría por la transición correspondiente, llegando a un nodo hoja con información que devolvería como etiqueta un nombre, *NB*

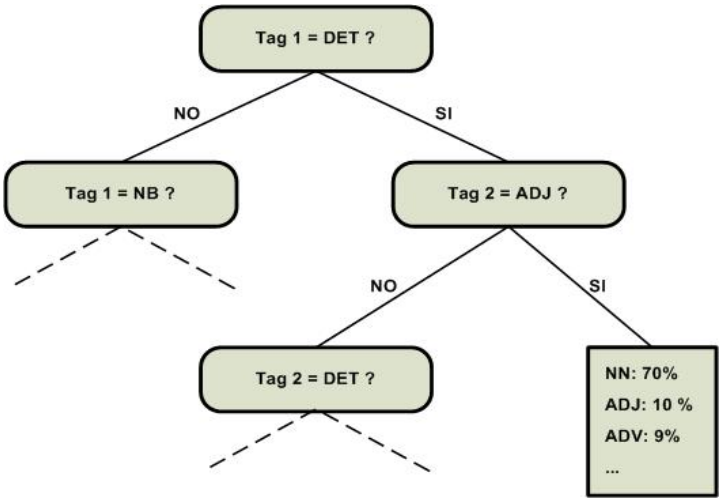


Figura 14 árbol de decisión en TreeTagger

El árbol de clasificación se construye de manera recursiva sobre un conjunto de n-gramas de entrenamiento. En cada paso se aplica un test al conjunto de n-gramas de entrenamiento que consiste en la comparación de las etiquetas anteriores a la actual. Se elige aquel test que proporciona mayor información sobre el conjunto de los n-gramas y se une al nodo actual. El conjunto de entrenamiento se divide y el proceso se vuelve a invocar. La división la define el test y genera dos subárboles que

se corresponden a las transiciones **SI** y **NO** del nodo creado. El proceso se detiene cuando alguno de los subconjuntos generados por el test es menor que un umbral dado.

8.2.LEXICÓN

El lexicón empleado por el TreeTagger está formado: por un lexicón completo llamado por el autor, *fullform lexicon* y por un lexicón de sufijos llamado, *suffix lexicon*

En un primer momento la palabra se busca en el *fullform lexicon*, si se encuentra entonces se devuelve un vector con las probabilidades asociadas. Este proceso incluye también el paso de las formas a minúscula para realizar la comparación. En caso de no encontrar la palabra se pasa a examinar el *suffix lexicon*. Este lexicón también se organiza en un árbol de decisión, pero en este caso no es binario. El árbol almacena las posibles terminaciones de las palabras. El funcionamiento es simple, se coge la última letra de la palabra y se busca a partir del nodo raíz, una vez en el nodo se coge la penúltima letra y se busca a partir de ese nodo. Así sucesivamente hasta llegar a un nodo hoja, que devolverá un vector con las probabilidades. La profundidad de este árbol no supera el valor 5.

9. ANÁLISIS SINTÁCTICO: ERIAL

En el paradigma que se aplica en este proyecto para obtener una ontología es necesario un análisis sintáctico de las oraciones del corpus. Este análisis permitirá la extracción de dependencias en forma de pares que orientarán sobre el tipo de relaciones existentes entre los elementos de los textos.

9.1 .ANALISIS SINTÁCTICO CON ERIAL

El enfoque seguido parte de un análisis sintáctico superficial. En este tipo de análisis sólo se devuelve un camino posible, es decir no tiene en cuenta ambigüedades. La herramienta elegida se ha desarrollado en el seno del proyecto Erial (*Extracción y Recuperación de Información mediante Análisis Lingüístico*)[3][4].

El sistema se desarrolló para el español, cubriendo los sintagmas nominales así como sus variantes sintácticas y morfosintácticas, incluyendo aquellas que implican el análisis de sintagmas verbales.

Como se ha comentado arriba, la forma de representar el contenido de los sintagmas se hace por medio de dependencias de las palabras que conforman los mismos. Concretamente reduce el número de palabras en una dependencia a dos, de ahí que el trabajo se realice sobre pares de dependencias. Estas dependencias incluyen los lemas de las palabras, esta normalización es muy útil puesto que establece una base común para las formas semánticamente equivalentes en las frases.

9.2.FUNCIONAMIENTO DE ERIAL

El recurso Erial es un analizador sintáctico superficial basado en una cascada de traductores finitos [5]. Concretamente la arquitectura seguida se basa en el uso de cinco capas, cuya entrada por defecto es el etiquetador-lematizador [6] MrTagoo [7] previamente pre-procesado. Como se describirá más adelante en el caso del proyecto la entrada fue proporcionada por el etiquetador-lematizador TreeTagger, previo pre-procesamiento, y después de aplicarle un conjunto de cambios para su adaptación a la entrada esperada por el analizador sintáctico.

A continuación se describe brevemente el cometido de cada etapa, pero antes es conveniente destacar que estamos tratando con reglas, cuya parte derecha deberá coincidir con un patrón para poder ser sustituida por su parte izquierda. Por ejemplo para un sintagma nominal sencillo:

Sintagma Nominal → Determinantes Sustantivo (SN --> Det N)

El analizador deberá encontrar en su entrada un conjunto de *Det* y *N* para poder sustituir este par por *SN*.

A continuación se pasa a enumerar las distintas capas de la arquitectura:

1. La primera capa mejora la salida del pre-procesador (Sección: 10) eliminando el ruido generado por ciertas construcciones sintácticas. Se puede considerar como una capa de pre-procesamiento antes de iniciar el análisis propiamente dicho. Se enumera a continuación el tratamiento de los datos en este paso:
2. Tratamiento de cifras que están total o parcialmente en formato no numérico. Por ejemplo, “cinco millones o 5 millones”.
3. Tratamiento de expresiones de cantidad. Por ejemplo, “algo más de dos millones”.
4. La segunda capa recupera sintagmas adverbiales y grupos verbales de primer nivel. Los sintagmas adverbiales están formados por secuencias de adverbios, solamente se tiene en cuenta el último adverbio de la secuencia como núcleo del sintagma. Aunque no existe ningún tipo de dependencia recuperada que

[7] Herramienta desarrollada en la Universidad de La Coruña, en el seno del grupo COLE.

trabaje con adverbios, este paso es necesario para el correcto funcionamiento de las etapas posteriores. Los grupos verbales de primer nivel son los sintagmas verbales no perifrásticos, correspondientes a formas pasivas, tanto simples ("es analizado") como compuestas ("ha sido analizado") y los sintagmas verbales de las formas activas.

5. La tercera capa detecta los sintagmas adjetivales así como los grupos verbales de segundo nivel. Un sintagma adjetival será un sintagma cuyo núcleo es un adjetivo que previamente podrá tener un adverbio que lo modifique. Los grupos verbales de segundo nivel incluyen las formas perifrásticas, conjunto de formas verbales que funcionan en unión.
6. La cuarta capa procesa los sintagmas nominales. Por definición este tipo de sintagmas tienen por núcleo un elemento que tiene una función sustantiva. Se detectan también en este punto, los sintagmas adjetivales que modifican a un sustantivo.
7. La quinta capa se encarga de la identificación de los sintagmas preposicionales, es decir los sintagmas nominales precedidos por una preposición.

9.2.1.EXTRACCIÓN DE DEPENDENCIAS

El análisis sintáctico se realiza pues en dos etapas, por un lado el reconocimiento de relaciones sintácticas y por otro lado la extracción de dependencias.

La herramienta incorpora una etapa previa que se encarga de delimitar las oraciones del conjunto a analizar. Para ello considera el fin de una sentencia al llegar a un signo de puntuación, a una conjunción, a un grupo verbal de segundo nivel que marca una forma no personal o al llegar a un relativo.

Así, una vez delimitadas las frases y las funciones sintácticas reconocidas, se procede a la extracción de los pares de dependencias formados por:

sustantivo y adjetivos que lo modifican, realmente esta extracción se realiza en la capa 4 que es donde se detectan estas construcciones.

sustantivo y su complemento nominal (*CN*), el complemento nominal puede ser un sintagma preposicional.

núcleo del sujeto y el verbo (*SUJ_*), en este caso el sujeto podrá ser activo (*A*) o pasivo(*P*) (de ahí el guión bajo), dando lugar a que el complemento del verbo sea bien un objeto directo bien un complemento agente.

verbo y sus complementos (*OD, CC, CA*): pueden ser complemento de objeto directo, complemento circunstancial o complemento agente.

núcleo del sujeto y el atributo (*ATR*), este caso ocurre ante la presencia de los verbos copulativos; "ser", "estar" y "parecer" para el español y "être" y "avoir" para el francés.

Para cada una de las dependencias extraídas, se guarda su tipo así como el lema y la etiqueta de sus componentes.

9.2.2.IMPLEMENTACIÓN DEL ANALIZADOR

Está construido a partir de un traductor de estado finito para cada una de las reglas involucradas en las diferentes capas del análisis. En la construcción de los pares sólo intervienen los lemas y las etiquetas, sin tener en cuenta la información relativa a la concordancia, es decir, no distingue entre femenino y masculino, ni entre singular y plural en las distintas formas.

En un primer momento el texto se divide en ternas de la forma forma-etiqueta-lema (salida devuelta por el etiquetador-lematizador). En un paso anterior al análisis propiamente dicho se sustituye esta terna por otra de la forma lema-etiqueta-categoría gramatical. En este punto, cabe señalar que dicha categoría gramatical se puede considerar como una etiqueta propia que asigna Erial para las fases posteriores.

Como se ha comentado anteriormente en cada capa se aplican una serie de reglas. En el código interno estas reglas están definidas mediante expresiones regulares en PERL (lenguaje empleado en el desarrollo de la herramienta). Así una vez que el sistema encuentra una correspondencia con la parte derecha de la regla sobre las ternas de entrada, la regla se reduce, sustituyendo las ternas por la terna indicada en la parte izquierda de la regla.

10.PREPROCESAMIENTO

Una de las actividades previas más importantes en el procesamiento automático del lenguaje natural es el pre-porcesamiento. En muchos casos se opta por no realizar esta tarea, obteniendo unos resultados poco deseados que afectan al comportamiento global del sistema. Se debe pensar que la base de un desarrollo es el pilar de un sistema y si desde el principio éste se tambalea, el resto de los módulos se verán afectados.

Las tareas que normalmente se llevan a cabo implican la segmentación de palabras y de frases, el reconocimiento de contracciones, el reconocimiento de fechas, la detección de nombres propios, etc. Es una labor muy compleja pues no existe una estrategia establecida que especifique el orden a seguir ni la técnica a emplear.

Además, otros factores aumentan la dificultad de esta etapa, como son el idioma a tratar y la aplicación en la que se incluirá.

El pre-procesador utilizado en este proyecto ha sido él utilizado en el proyecto Erial que tiene como objetivo desarrollar un entorno para recuperación de información basado en la utilización de técnicas de PLN, aunque se ha prescindido de ciertos módulos no esenciales

10.1.ESTRUCTURA DEL PREPROCESADOR

El pre-procesador es una herramienta modular que utiliza algoritmos generales para poder ser así utilizado en otros idiomas, de ahí que su uso se pudo extender también al francés. La herramienta realiza tareas de pre-etiquetación de nombres propios, fechas y contracciones. Además trata los numerales, las locuciones y los pronombres enclíticos.

En el caso de este proyecto se han obviado estas etapas. El propio etiquetador reconoce adecuadamente los nombres propios (por el lexicón que específicamente se creó para él) y etiqueta bien las fechas. Respecto a las contracciones y a los numerales, su tratamiento sólo se hizo necesario a nivel del análisis sintáctico (Sección: 9); ya que a nivel de adquisición de terminología la solución devuelta por

TreeTagger resultó adecuada. Asimismo el tratamiento de TreeTagger respecto a las locuciones y pronombres enclíticos respondía a las necesidades planteadas en el proyecto.

A continuación se describen los tres módulos que se han tomado en cuenta:

Filtro: Este módulo se encarga de compactar los separadores redundantes que existen en el texto, como pueden ser múltiples espacios en blanco entre palabras o al principio de frase.

Segmentador: Este módulo se encarga de identificar y separar los segmentos presentes en el texto, para que cada palabra individual y cada signo de puntuación forme un segmento. Se puede considerar un segmento como una unidad de información con significado.

Separador de Frase: Este módulo se encarga de delimitar las frases del texto, de manera general una frase acaba con un punto seguido de una mayúscula.

A estos módulos hay que sumar un pre-procesado anterior que consiste en extraer el cuerpo de los documentos para formar el corpus. Como se puede ver en el apartado *10.3. Formato de los documentos* del Manual Técnico, los documentos se almacenan en un fichero XML. Esto se menciona aquí, ya que ciertos sistemas de pre-procesamiento incluyen una etapa previa para convertir el formato original a texto plano para permitir así crear el corpus.

11.LEMATIZACIÓN: FLEMM

NOMBRE	Analyseur Flexionnel du français pour des corpus étiquetés.
AUTOR	Fiammetta Namer
AÑO	
IDIOMAS	Francés

El proceso de lematización consiste en la reducción de las diferentes formas flexivas y derivativas de una palabra a una forma canónica conocida con el nombre de lema. Se trata de buscar las variaciones morfológicas de una palabra para obtener éste y poder así agrupar las formas que de él derivan. Es un proceso necesario en muchas tareas del PLN, pues existe una relación semántica entre las palabras que derivan de un mismo lema. Por ejemplo, en actividades de recuperación de información muchos autores coinciden en que la lematización es uno de los mejores métodos para recuperar y clasificar los documentos obtenidos de una petición.[7].

11.1.FORMACIÓN DE LAS PALABRAS

La morfología es la rama de la ciencia lingüística que estudia la construcción de las palabras. Esta construcción lleva implícita la unión de unidades llamadas morfemas. Se puede así definir un morfema como la unidad mínima con significado parcial en la que se puede dividir una palabra.

Estos morfemas se pueden clasificar en (1) *raíz*, que es el morfema principal y el que aporta significado; (2) *afijos*, que son los morfemas que aportan significado adicional y *rasgos gramaticales*.

El objetivo del proceso de lematización es la consecución del lema de la palabra tratada. Se define así un lema como la forma canónica que representa un conjunto de palabras que comparten la misma raíz y sentido (o significado). La categoría léxica del conjunto de palabras que comparten un lema varía dependiendo del proceso de formación de dichas palabras:

Formación por flexión: Crea diferentes formas de un lema, manteniendo su categoría léxica y añadiendo rasgos gramaticales, *(gato / a / os / as)*

Formación por derivación: Crea nuevas palabras a partir de las existentes. Suelen suponer un cambio de categoría léxica, de significado y de lema, *(generar → generación ; relativo → relativizar).*

11.2.PRESENTACIÓN DEL PROGRAMA

Flemm es un conjunto de módulos PERL que hacen un análisis flexivo del francés basado en reglas. Es un sistema innovador en el uso de conocimientos lingüísticos y además es robusto, pues analiza las formas de las palabras desconocidas.

La entrada a Flemm es un texto previamente categorizado. El programa posee dos interfaces, una de entrada y otra de salida, que realizan una traducción de las etiquetas de entrada a un formato aceptado por el módulo interno. Una vez pasado por la interfaz de entrada secuencialmente se ejecutan las distintas operaciones del módulo interno. Estos módulos tienen como únicas fuentes de datos unas listas de excepciones y reglas, sin hacer uso de grandes diccionarios.

11.3.FUNCIONAMIENTO DEL PROGRAMA

El sistema coge como entrada parejas de la forma palabra/etiqueta y devuelve a la salida el lema de la palabra, así como el conjunto de sus variaciones flexivas.

11.3.1.MÓDULO DE TRADUCCIÓN

Este módulo se puede dividir en dos partes, una interfaz de entrada y otra de salida. La de entrada transforma las etiquetas en una notación interna independiente, que es la que entiende la aplicación. Una vez finalizado el proceso de lematización la interfaz de salida se encarga de restituir las etiquetas originales.

Esta característica, esencial de Flemm, garantiza su modularidad e independencia del juego de etiquetas utilizadas. Sin embargo, hasta el momento estas interfaces sólo están adaptadas a las etiquetas proporcionadas por los etiquetadores Brill y TreeTagger.

El sistema controla la entrada al programa validando las etiquetas proporcionadas por el etiquetador y corrigiéndolas en presencia de algún error. Los errores detectados se pueden deber a una mala segmentación del etiquetador haciendo que aparezcan signos de puntuación residuales. El otro proceso de control lleva a cabo un análisis de las etiquetas asignadas a las palabras para comprobar su consistencia.

11.3.2.MÓDULO DE ANÁLISIS

Este es el módulo encargado de calcular el lema y las trazas morfológicas. Se descompone en tres operaciones.

Detección de la terminación de las palabras

El sistema incorpora un conjunto de paquetes que contienen las reglas de recorte y que se activan según la categoría léxica. Las palabras se examinan de derecha a izquierda tomando en consideración todos aquellos conjuntos que potencialmente, puedan proporcionar una solución. Una vez que se tiene un grupo de

bases, se analiza cual de ellas es la más frecuente y es la que se toma en consideración. En caso de contar con dos posibles lemas, ambos son almacenados.

Reducción de los alomorfos

Se llama alomorfo a cada una de las realizaciones fonológicas de un morfema. Para estos casos Flemm aplica un conjunto de reglas sobre el par base/terminación.

Cálculo de la información flexiva

Este paso se realiza a la par que los dos anteriores en el momento de la distinción del par base/terminación. Flemm proporciona las trazas funcionales principales:

- género y número para los adjetivos, determinantes y participios.
- numero para los sustantivos
- género, número y persona para los pronombres
- número, persona, tiempo modo y grupo para los verbos.

Las trazas se codifican según el estándar recomendado para el francés por el consorcio Multext[8]. Se añade, además, una nueva traza no perteneciente al estándar para almacenar el grupo del verbo.

11.3.3.FORMATO DE LOS RESULTADOS

Flemm en combinación con TreeTagger genera un fichero XML, que es él que se ha tratado para proporcionar la entrada al extractor de terminología Acabit (Sección: *13 acabit*)

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<FlemmResults>
<FlemmResult>
        <InflectedForm>Un</InflectedForm>
        <Category original-tagger='DET:ART'>DET(ART)</Category>
        <Analyses>
                <Analyse>
                        <Lemme>un</Lemme>
```

[8] Multilingual Text Tools and Corpora

```
                        <Features>
                                <Feature name='catmultext' value='D'/>
                                <Feature name='type' value='a'/>
                                <Feature name='pers' value='3'/>
                                <Feature name='gend' value='m'/>
                                <Feature name='nb' value='s'/>
                                <Feature name='case' value='-'/>
                                <Feature name='poss' value='-'/>
                                <Feature name='quant' value='-'/>
                        </Features>
                </Analyse>
        </Analyses>
</FlemmResult>

<FlemmResult>
        <InflectedForm>euro</InflectedForm>
        <Category original-tagger='NOM'>NOM</Category>
        <Analyses>
                <Analyse>
                        <Lemme>euro</Lemme>
                        <Features>
                                <Feature name='catmultext' value='N'/>
                                <Feature name='type' value='c'/>
                                <Feature name='gend' value='-'/>
                                <Feature name='nb' value='s'/>
                                <Feature name='case' value='-'/>
                                <Feature name='typesem' value='-'/>
                        </Features>
                </Analyse>
        </Analyses>
</FlemmResult>
…
```

12.ADQUISICIÓN DE TERMINOLOGÍA

En esta sección se tratan aspectos característicos de la adquisición de términos. En las sucesivas secciones se habla de los extractores concretos utilizados.

El rápido avance existente en todas las ramas de la ciencia, así como la creación de nuevas áreas de conocimiento, conlleva la aparición constante de nuevos términos. De un modo similar, las formas de comunicación hacen que un mismo término se vea modificado para satisfacer los distintos dominios en los que puede ser empleado. Se aprecia entonces que el ámbito de uso de los recursos terminológicos se amplía día a día. Se adaptan a las necesidades de las empresas y podemos encontrarlos en tesauros, en sistemas de indexación automática, en sistemas de ayuda a la redacción, en ontologías, etc.

La investigación relacionada con este tema exige que los sistemas de gestión de información, empleados tanto en las empresas como en las instituciones, sean eficaces. Para producir, difundir, gestionar y explotar los documentos de manera óptima estos sistemas necesitan hacer uso de recursos terminológicos.

12.1.DEFINICIÓN DE TÉRMINO

Existen varias acepciones para término, a continuación se exponen algunas.

La norma ISO 1087 (1990) define término como "La designación de un concepto definido en un lenguaje especializado mediante una expresión lingüística. Un término puede estar formado por una sola palabra o por dos o más palabras".

(Fleber,1987) da la siguiente definición: *"un término es un representante lingüístico de un concepto en un dominio de conocimiento"*. Esta definición se basa en la *Théorie Générale de la Terminologie* nacida a finales de los años treinta y formulada por Eugène Wüster. Esta doctrina defiende que el conocimiento está recortado en dominios estables, en los cuales cada uno equivale a una red fija de conceptos, siendo los términos los representantes lingüísticos de esos conceptos.

Hoy en día esta teoría se cuestiona por con el continuo desarrollo de sistemas de explotación terminológica. Es decir, dado un dominio no existe un único conjunto de términos que represente el saber de dicho domino.

Se puede también definir un término como *"el resultado de un proceso de análisis terminológico cuya última tarea es la decisión"*. Es decir, el proceso de análisis devuelve una lista de palabras que deberán ser examinadas para determinar si se trata realmente de términos relevantes en el dominio trabajado. Esta lista de palabras se conoce como términos candidatos [8]. Es interesante en este punto establecer la diferencia entre un término candidato y un término, obteniendo otra definición:

El concepto de término candidato (también colocación) se refiere a un grupo de palabras cuyo significado global se deduce de las unidades que lo componen, por ejemplo: Alcalde de París.

El concepto de término se refiere a una colocación que tiene propiedades sintácticas y representa una realidad en un dominio de conocimiento, por ejemplo: Inteligencia Artificial.

Se trata, pues, de un trabajo de construcción y no de un proceso de descubrimiento de la terminología. Existen, sin embargo, dos características deseables en los términos reconocidos:

- Deben poseer estructura léxica específica en el dominio, y estable en el corpus.
- Deben proporcionar una utilidad demostrable en la aplicación final en la que se va a utilizar el término. En otras palabras, han de ser funcionales para la aplicación que los vaya a explotar.

12.2.INGENIERÍA TERMINOLÓGICA

Aunque bien está saber el significado de *término*, para construir una herramienta que explote los recursos terminológicos se hace necesaria la definición de una metodología. Se trata de beneficiarse del análisis de los recursos terminológicos para poder extraer una serie de principios y reglas de adquisición. Se exponen a continuación ciertas características esenciales sobre esta actividad y que servirán de base para la elaboración de las herramientas:

- Como cabe esperar, la forma de explotar estos sistemas parte del análisis de las fuentes textuales digitales. La construcción del corpus es esencial, el conjunto de textos ha de ser pertinente al dominio tratado.
- Las estructuras léxicas deben estar convenientemente definidas en el corpus de referencia.
- Esta tarea debe llevarla a cabo un analista (lingüista o terminólogo) y no el experto del dominio. El analista funciona de mediador entre el experto y la aplicación.
- La validación de la herramienta es doble, por un lado el experto juzga si las descripciones del analista son correctas y por otro lado el usuario final validará la aplicación en la cual se engloban los recursos terminológicos.

Visto la cantidad masiva de datos a procesar para obtener unos resultados adecuados, la tarea del análisis terminológico tiene que ser necesariamente llevada a cabo mediante herramientas informáticas. El uso de estas aplicaciones minimiza la tarea tediosa asociada con el subrayado manual de los términos. De las características anteriores se deducen los siguientes elementos a tener en cuenta a la hora de elaborar dichas aplicaciones:

- Son principalmente herramientas de ayuda y no un fin en si mismo. Los analistas explotarán dichas herramientas en las distintas fases de su trabajo.
- Construir una terminología supone estructurar los términos. Por lo tanto, es necesario por un lado aplicaciones que identifiquen los términos y por otro lado aplicaciones que ayuden a estructurar

Estas herramientas tienen que ser robustas para admitir textos provenientes de cualquier dominio, sin necesidad de retocar ninguna línea de código

12.3.UN POCO DE HISTORIA

Los trabajos existentes en adquisición de terminología son relativamente recientes. En muchos casos se considera el origen de dichos sistemas en el momento de aparición de la aplicación *TERMINO*. Este sistema fue desarrollado al final de la década de los ochenta, en la Universidad de Québec en Montreal, por Pierre Plante.

Se puede considerar a TERMINO como la primera herramienta dedicada exclusivamente a la construcción de bases terminológicas. Efectúa un análisis morfo-sintáctico del texto que se le pasa como entrada, es a partir de este análisis que detecta términos candidatos. Se distingue principalmente de sus antecesores en que además de hacer un tratamiento sintáctico del corpus, integra un interfaz de validación de los términos candidatos.

Al principio de los 90 otras herramientas ven la luz, se destacan tres aplicaciones que son ANA, LEXTER y Acabit. Se caracterizan por trabajar con el idioma francés y en un contexto industrial. Es interesante ver como la adquisición de terminología nace en ese ambiente. La justificación la encontramos en la necesidad de trabajar con un vocabulario especializado y altamente técnico para el desarrollo de aplicaciones de traducción o búsqueda de información. Es por ello que los trabajos expuestos anteriormente fueron financiados por empresas y no por universidades. Otro factor determinante viene dado por la no disponibilidad de corpora especializados por parte de las universidades al tratarse de información confidencial.

Hoy en día, y gracias al desarrollo constante de Internet, la accesibilidad a corpora especializados es más abierta. Cada vez son menores los obstáculos en la investigación en este campo en el seno de las universidades. Las corrientes actuales se centran en la construcción de redes léxicas especializadas para los motores de búsqueda Web.

12.4.EXTRACCIÓN AUTOMÁTICA DE TÉRMINOS

La extracción de términos, ya sea de forma automática o asistida por ordenador, es un medio eficaz de explotar los documentos y establecer el conjunto relevante de términos asociados con el dominio tratado en el corpus textual.

Visto esto, se puede definir la extracción de términos como "la lectura anotada de un corpus textual y la selección de candidatos a términos junto con su contexto".

En muchas tareas relacionadas con el PLN se hace necesaria la utilización de términos, para extraer parte del conocimiento latente en los documentos. Como se ha visto en los apartados anteriores, es necesario el uso de herramientas que automaticen esta tarea y permitan obtener la terminología presente en los textos a procesar.

Es posible caracterizar un término, con independencia del dominio y de la lengua haciendo uso de información estadística y lingüística.

Existe una tipología en la que se pueden dividir las herramientas que ayudan al reconocimiento de terminologías.

1. Adquisición de términos: Extracción a partir del corpus, analizando los términos candidatos. Es decir, palabras o conjuntos de palabras susceptibles de ser reconocidas como términos por parte del analista. Herramientas que se han utilizado en la elaboración de este proyecto.
2. Estructuración de términos: Necesaria para crear una estructura a partir de la colección de términos obtenida. Se cuentan aquí las herramientas para clasificación automática de términos y para la detección de relaciones entre éstos. Suelen trabajar estableciendo una unión entre los términos que aparecen frecuentemente juntos y por lo tanto se establece un grado de proximidad semántica entre ellos.
3. Alineación de términos: Se trata de herramientas y técnicas para alinear términos a partir de corpora bilingües y paralelos. Usualmente el proceso se descompone en dos fases, la primera es la adquisición monolingüe y la segunda es la adquisición multilingüe. En este segundo caso, se suelen emplear técnicas estadísticas de ocurrencia de los términos y técnicas del posicionamiento en la frase. Normalmente se obtienen mejores resultados con la alineación a nivel de sintagmas que en la alineación a nivel de palabras.

12.4.1.PROBLEMÁTICA ASOCIADA

Existe una gran diversidad de colectivos profesionales interesados en la extracción automática de términos. Se pueden citar lingüistas, traductores, intérpretes, periodistas, informáticos, etc. Su interés en esta materia viene del hecho de que supone una etapa básica para la creación de glosarios, de diccionarios terminológicos y de sistemas expertos, así como para la indexación de textos entre otros.

La fase clave en un sistema de adquisición es la búsqueda de las unidades terminológicas. Dicha fase se puede descomponer en tres etapas: (1) Reconocimiento de términos; (2) Reconocimiento de términos complejos; (3) Determinación de la pertenencia del término al dominio en cuestión.

Estas etapas asocian una problemática. En primer lugar detectar donde empieza y acaba un término complejo. En segundo lugar decidir si un determinado segmento es terminológico o libre, y en último lugar concretar si la unidad pertenece al vocabulario común o al especializado. Es por lo tanto necesario asentar unas bases que delimiten al término en un dominio concreto y un área de especialidad.

Esta es la razón por la que la mayoría de los sistemas de adquisición son semi-automáticos. De este modo al final del proceso de extracción lo que se obtiene es una lista de términos candidatos y no una lista de términos. Será entonces cuando el experto aporte sus conocimientos. Para facilitar la tarea, muchas aplicaciones acompañan los términos con sus contextos de aparición y con información adicional como la frecuencia.

12.4.2.APROXIMACIONES PARA LA EXTRACCIÓN

APROXIMACIONES AUTOMÁTICAS

Sistemas estadísticos: Seleccionan los términos en función de su poder discriminatorio entre los documentos o basándose en una ley estadística.

Sistemas lingüísticos: Se basan principalmente en patrones sintácticos. Estos procedimientos dan mejores resultados, pero presentan el inconveniente de funcionar con grandes volúmenes de conocimiento, diccionarios y gramáticas que deben ser descritas anteriormente.

Sistemas mixtos: Utilizan las dos aproximaciones anteriores, consiguiendo buenos resultados.

APROXIMACIONES MANUALES

Son las que se utilizan con más frecuencia a pesar de ser de mala calidad. Los términos son incompletos e inconsistentes (dependen mucho de la persona que los determina).

SISTEMAS ESTADÍSTICOS

Los sistemas que utilizan una aproximación estadística basan su funcionamiento en la detección de dos o más unidades léxicas que ocurren frecuentemente juntas. Se considera que esta concurrencia no es casual. Se utiliza para ello la medida conocida como información mutua, que calcula una cierta forma de independencia de las palabras que componen la colocación.

$$IM(x, y) = \log_2 [P(x,y) / P(x)P(y)]$$

El problema asociado a este tipo de aproximación es que los términos que ocurren con una frecuencia muy baja no son detectados, pero tiene la ventaja de que al utilizar información numérica son relativamente independientes de la lengua utilizada.

SISTEMAS LINGÜÍSTICOS

Surgen por necesidad de suplir el problema de los anteriores respecto a los términos que ocurren con poca frecuencia. Tratan para ello de utilizar conocimiento lingüístico enfocado de dos maneras:

- Específica para el término: Detectan la secuencia típica para los términos, por ejemplo Nombre-Adjetivo, Nombre-Preposición-Nombre, etc. Es decir hacen uso de expresiones regulares y técnicas de autómatas.
- Genérica del lenguaje: En este caso se recurre a un análisis de más alto nivel, mediante técnicas de PLN que devuelven estructuras básicas de la frase, como por ejemplo sintagmas nominales o sintagmas verbales.

En ambos casos es necesario utilizar algún tipo de sistema que devuelva la categoría morfológica de las palabras, por ejemplo un etiquetador.

12.4.3.EVALUACIÓN DE UN SISTEMA DE EXTRACCIÓN

La evaluación es un proceso fundamental para medir el sistema, pero es una tarea también importante para detectar fallos o para realizar comparaciones con otros sistemas [9].

Si se considera que existen dos clases de ejemplos, los que son términos y los que no lo son, se pueden crear cuatro grupos:

Verdaderos Positivos (VP): Términos correctamente reconocidos por el sistema.

Falsos Negativos (FN): Términos que el sistema no reconoce.

Falsos Positivos (FP): Candidatos que no son términos pero el sistema los reconoce como tal.

Verdaderos Negativos (VN): Candidatos que no son términos y que el sistema reconoce adecuadamente.

Se presenta a continuación la matriz de confusión que refleja estos grupos:

	CLASE PREDICHA	
CLASE REAL	Término	No Término
Término	VP	FN
No Término	FP	VN

Si cogemos N como el número de ejemplos usados tenemos que *N= VP + FN + FP + VN*, de esta manera se puede calcular la tasa de acierto y de error:

Tasa de acierto = (VP + VN) / N

Tasa de error = (FP + FN) / N

Existen situaciones en las que los errores no se valoran de la misma manera, por ejemplo en algunos casos puede interesar recoger muchos términos a cambio de recoger como términos cosas que no lo son. En este caso un falso positivo es menos costoso que un falso negativo. En la matriz anterior podemos incorporar los costes y beneficios, obteniendo la matriz correspondiente.

	CLASE PREDICHA	
CLASE REAL	Término	No Término
Término	B_{VP}	C_{FN}
No Término	C_{FP}	B_{VN}

Combinando ambas matrices podemos obtener los valores a continuación:

Beneficio = $VP * B_{VP} + VN * B_{VN}$

Coste = $FP * C_{FP} + FN * C_{FN}$

COBERTURA Y PRECISIÓN

La cobertura (*recall*) mide la proporción de términos correctamente reconocidos respecto al total de términos reales. Se puede considerar como la capacidad de detección.

cobertura = $VP / (VP + FN)$

La precisión es la relación entre los términos predichos y de esos cuales son realmente términos. Se puede considerar como la capacidad de discriminación.

precisión = $VP / (VP + FP)$

Existe, en general, una relación inversa entre estas dos medidas. Si aumenta la cobertura del sistema, es decir, cada vez se reconocen más términos, disminuye la precisión. También se reconocen como términos estructuras que no lo son. Si disminuye la cobertura, ya no se reconocen tantos términos, aumenta la precisión pues lo que se reconoce es correcto.

ESPECIFICIDAD Y SENSIBILIDAD

La sensibilidad (tasa de VP) indica el número de términos reconocidos.

sensibilidad = $VP / (VP + FN)$

La especificidad indica cuantos falsos términos se han clasificado como tales

especificidad = $VN / (VN + FP) = 1 - [FP / (FP + VN)]$

SILENCIO Y RUIDO

El silencio es la proporción de términos no reconocidos del total de términos predichos. Se trata de evaluar los que están presentes en el texto, pero no detectados por el sistema

silencio = FN / (VP + FN) = 1 - cobertura

El ruido es la proporción de falsos positivos respecto al total de términos predichos. Se trata de evaluar los que se rechazaron respecto a los aceptados.

ruido = FP / (VP + FP) = 1 - precisión

Como la cobertura y precisión se relacionan de manera inversa. Una disminución del silencio implica conseguir más falsos positivos, por lo tanto aumentará el ruido. Inversamente, disminuyendo el ruido se consiguen más falsos negativos.

13.ACABIT

NOMBRE	Automatic Corpus-based Acquisition of Binary Terms.
AUTOR	Béatrice Daille
AÑO	1994
IDIOMAS	Francés

13.1.DESCRIPCIÓN

Acabit tiene por objetivo preparar la tarea del terminólogo proponiéndole una lista ordenada de términos candidatos a partir de un corpus que previamente tiene que ser tratado y estructurado.

La idea básica de este extractor es la de combinar conocimiento lingüístico con conocimiento estadístico. Acabit ejecuta su funcionalidad en dos etapas. En la primera, y gracias a un conjunto de patrones predefinidos, extrae un conjunto de términos simples y sus variaciones. En la segunda etapa aplica una clasificación a los términos según una medida estadística.

13.1.1.TÉRMINOS BASE Y SUS VARIACIONES

La extracción que aquí se presenta trata de descubrir, estructurar e inferir relaciones conceptuales entre términos del francés [10]. Las relaciones conceptuales que veremos más adelante se deducen a partir de tipos específicos de variaciones de los términos que pueden ser sintácticas o morfosintácticas.

Como se ha comentado en el apartado anterior la adquisición consta de dos etapas. En la primera de ellas se extraen términos base y sus variaciones. La variación terminológica en los textos es un conocido fenómeno que ocurren entre un 15% y 35% de los textos, dependiendo del dominio reflejado en los mismos y de los tipos de variantes manejadas.

Los términos base definidos en Acabit siguen una gramática basada en expresiones regulares:

- *Nb1 Adj (Sustantivo1 Adjetivo)*
- *Nb1 (Prep (Det)) Nb2 (Sustantivo1 (Preposición (Determinante))) Sustantivo2*
- *Nb1 à Vinf (Sustantivo1 à Verbo en infinitivo)*

Se presenta a continuación la clasificación de las variaciones de los términos base tenidas en cuenta por Acabit:

Variaciones sintácticas débiles. Modifican las funciones de las palabras en el término base.

- Variación de la preposición.
- Variación entre el artículo y la preposición.
- Variaciones flexionadas: Introducen pequeñas modificaciones en el interior del término base. Pueden ser de tipo gráficas, en las cuales se definen las variaciones que tienen en cuenta la diferencia entre mayúsculas y minúsculas así como las variaciones que introducen un guión entro dos sustantivos.

Variaciones sintácticas: Se encuentran en este grupo las variaciones por coordinación y las que introducen un nuevo elemento.

- Combinación de términos por la coordinación
  Nb1 Prep Nb3 + Nb2 Prep Nb3 → Nb1 et Nb2 Prep Nb3
  envío de correo + recepción de correo → envío y recepción de correo.
- Combinación de término por la surcomposición
  Nb1 Prep Nb2 + Nb1 Prep Nb3 → Nb1 Prep Nb2 Prep Nb3
  resultado del trimestre + resultado en la empresa → resultado del trimestre en la empresa.
- Introducción de un modificador adjetival o adverbial
  Nb1 Prep Nb 2 → Nb1 Adj/Adv Prep Nb2.
  asistencia por teléfono → asistencia técnica por teléfono.

Variaciones morfosintácticas: Modifican la estructura interna de los términos base.

- Modificación de la preposición por un afijo
  crisis después de la guerra → crisis de la post-guerra
- Derivación morfológica: Uso de un adjetivo relacional. Este tipo de adjetivos se distingue de los calificativos en que permiten expresar una relación que

normalmente se denota con una preposición. Es decir, el adjetivo relacional es un argumento del sustantivo al que acompaña.

crecimiento de la economía → crecimiento económico.

Las ocurrencias de los términos base y sus variantes se almacenan en forma de pares. Estos pares son dos palabras lematizadas y son los extremos de un mismo término.

PATRÓN Nb1 (Prep (Det)) Nb2	Secuencia extraída del corpus	9
(resultado, empresa)	Resultado de la empresa	6
	Resultado trimestral de la empresa	3
	Resultado financiero de la empresa	1

Como se puede observar, el extractor se centra en la detección de nombres compuestos binarios. Este hecho se debe fundamentalmente a que éstos son los más numerosos en textos especializados. Además, cualquier nombre compuesto de longitud mayor que dos se puede descomponer en forma binaria.

13.1.2.RELACIONES CONCEPTUALES

La mayoría de las variaciones conservan el mismo valor semántico que el término base del que proceden. Pero ocurren casos, en los que se da lugar a un término con valor semántico distinto obteniendo un nuevo término base.

Interesa entonces definir una serie de funciones que permitan identificar las relaciones conceptuales entre términos base.

Sintácticas:

- Relación de hiperonimia: Dado un término base con estructura Nb1 Adj ó *Nb1 (Prep (Det))Nb2* si existe un adjetivo relacional que lo modifica, el término base será hiperónimo de su variante.

  Spec(resultado de la empresa) = resultado financiero de la empresa

- Relación de antonimia: Dado un término base con estructura *Nb1 Adj,* si existe un adverbio de negación que lo modifica, se presenta una relación de antonimia entre el término base y su variante.

  Anti(marketing) = no-marketing

Morfosintácticas: Aparecen distinciones semánticas si dos términos se relacionan morfológicamente. Dados dos términos w_1 w_2 y w_1' w_2' se dice que están morfológicamente relacionadas si ocurre alguno de los siguientes casos.

1. Si w_1 y w_1' son cabeza de sus respectivos términos y son idénticos; entonces, w_2 y w_2' son expansiones de sus respectivos términos y se relacionan semánticamente por el uso de un afijo.
2. Si w_1 y w_1' son cabeza de sus respectivos términos y se relacionan semánticamente por el uso de un afijo; entonces, w_2 y w_2' son expansiones de sus respectivos términos y son idénticos.
3. Si w_1 y w_1' son cabeza de sus respectivos términos y w_2 y w_2' son expansiones de sus respectivos términos; entonces bien w_1 y w_2' son idénticos y w_2 y w_1' se relacionan semánticamente por el uso de un afijo o al contrario.

El uso de los afijos en francés proporciona pistas sobre el tipo de función que se puede aplicar a los términos y devuelve el tipo de relación semántica que ocurre entre ellos. Estas funciones pueden ser de antonimia, de conjunto o de actor ("transporte → transportista").

13.1.3.APLICACIONES DE MEDIDAS ESTADÍSTICAS

En la segunda etapa de la extracción es cuando se aplican medidas estadísticas, para clasificar los términos candidatos en orden decreciente de pertinencia.

Las medidas aplicadas por la autora son la frecuencia, el criterio de verosimilitud y la información mutua; al cubo. Esta última medida es obtenida por la autora con el fin de favorecer los pares más frecuentes.

El criterio de verosimilitud se basa en una matriz que guarda una relación del número de ocurrencias *(a, b, c, d)* de un término *(x y)* y sus elementos.

13.2.FUNCIONAMIENTO DEL EXTRACTOR

El programa de extracción toma como entrada un corpus que previamente ha sido etiquetado, lematizado y estructurado. La lematización se lleva a cabo con el programa Flemm (Sección: 11).

En un primer momento se extraen los términos base y sus variantes a partir de la gramática de expresiones regulares. Esta técnica hace uso de autómatas finitos asociados a las reglas. La gramática hace uso de la información que le proporciona el etiquetador. Las diferentes ocurrencias de un término se almacenarán en forma de pares.

En un segundo paso se lleva a cabo la identificación de las relaciones conceptuales. Este paso se divide en dos etapas, la primera se centra en la identificación de las relaciones conceptuales sintácticas y la segunda en la identificación de las relaciones conceptuales morfosintácticas vistas. El algoritmo en ambos casos es el mismo.

Llegado este punto es necesario explicar la presencia de unas reglas especiales que usa Acabit, reglas por sufijo. Estas reglas siguen la siguiente nomenclatura:

$A \rightarrow N : -S + M$

Donde:

A → *N* representa la estructura del término base Nb1 Adj, pero se podría aplicar igualmente a la estructura Nb1 Nb2.
S es un sufijo que debe borrarse, bien del adjetivo bien de la cabecera dependiendo de si estamos ante el caso de Nb1 Adj ó de Nb1 Nb2. Obteniendo en este caso un item R.
M es el segmento que se debe concatenar a R para obtener un sustantivo.

Estas reglas son las que van a permitir identificar los adjetivos relacionales y las funciones tratadas en el apartado 13.1.2. .

Algoritmo

1. Se examina cada candidato a término base.
2. Se aplican las reglas por sufijo a cada uno de esos candidatos, generando todos los posibles casos.
3. Se busca en el conjunto de términos candidatos si existen los casos generados en el paso 2.
4. En caso de éxito en el paso anterior se añade ese nuevo par al conjunto de candidatos.

El algoritmo aplica un filtro en el paso 2 para poder prescindir de aquellos casos generados y que son incorrectos. Para ello se hace una comprobación en dos sentidos, por un lado se hace un recorrido del corpus para comprobar que el caso está contenido en el mismo y por otro lado se comprueba que el candidato generado pertenece a la expansión de un término candidato. Se evita así la sobre-generación y se consigue menor ruido.

13.2.1.SALIDA DE LA EXTRACCIÓN

El formato de salida de Acabit es un fichero XML que agrupa los términos candidatos después del agrupamiento morfológico.

```
<?xml version="1.0" encoding="UTF-8"?>
<LISTCAND>
<SETCAND new_ident="212" loglike="1941.642" freq="43">
        <CAND old_ident="212">
        <NPN freq="43">
                <BASE> <TERM> milliard de euro </TERM>
                </BASE>
        </NPN>
        </CAND>
</SETCAND>

<SETCAND new_ident="2" loglike="1834.543" freq="34">
        <CAND old_ident="2">
        <NPN freq="34">
                <BASE> <TERM> billet vert </TERM>
                </BASE>
        </NPN>
        </CAND>
</SETCAND>

<SETCAND new_ident="385" loglike="1683.783" freq="30">
        <CAND old_ident="385">
        <NPN freq="30">
                <BASE> <TERM> taux de intÃ©rÃªt </TERM>
                </BASE>
        </NPN>
        </CAND>
</SETCAND>

<SETCAND new_ident="64" loglike="1561.722" freq="24">
        <CAND old_ident="64">
        <NA freq="24">
                <BASE> <TERM> banque central </TERM>
                </BASE>
        </NA>
        </CAND>
</SETCAND>
```

Cada una de las etiquetas de este XML encierra un significado:

SETCAND: agrupa el conjunto de términos candidatos para una familia de términos base.

CAND: agrupa cada uno de los candidatos y guarda en un atributo la frecuencia de aparición en el corpus.

NPN, **NA**: guarda la estructura del patrón del término.

TERM: guarda el tipo del término, en este caso un término base.

MODIF: guarda el tipo del término, en este caso una variación de un término base.

COORD: guarda el tipo del término, en este caso la coordinación de dos términos base.

Cada uno de estos datos se almacena en distintos ficheros a partir de los cuales se crea el XML. En estos archivos la información se almacena teniendo siempre presente cual es el primer lema y el último lema del par que forma el término candidato. En estos ficheros se puede encontrar tanto la frecuencia de aparición de cada uno de los lemas como la frecuencia del par en el corpus, además también se almacena todo los resultados de las medidas estadísticas. Esto puede resultar de interés si se pretende manejar directamente la información de estos ficheros sin tratar el archivo XML de salida. En el caso de este proyecto se ha optado por tratar la salida estándar y no cada uno de los ficheros. Esta decisión se toma partiendo del hecho que XML es un estándar sobre él que se han basado la mayor parte de los módulos del sistema.

14.FASTR

NOMBRE	Filtrage et Acquisition Syntaxique de TERmes
AUTOR	Christian Jacquemin
AÑO	1997
IDIOMAS	Francés

En este apartado se mencionan la descripción y las características de Fastr, un analizador sintáctico robusto dedicado al reconocimiento de términos y sus variantes. Integra dos formas de procesamiento de los datos. Por una parte permite una indexación de un corpus a partir de una lista de términos predefinidos pasados como argumentos de entrada y por otra parte permite una indexación libre de un corpus, esto es, una extracción de terminología. En lo que a este proyecto se refiere se ha utilizado la indexación libre en la explotación de esta herramienta.

14.1.TÉRMINOS BASE Y SUS VARIANTES

La extracción que aquí se presenta, parte de un análisis de la morfología flexional de las palabras, por ser esta la más fácil de modelizar. El tema central de la investigación llevada a cabo por el autor [12] se centra en el estudio de la sintaxis de los términos así como de sus variantes.

La arquitectura de este recurso se descompone en dos niveles. Un nivel de descripción de los términos el **nivel morfológico** y un nivel de representación de las variaciones, el **nivel sintáctico**. Las variantes tenidas en cuenta por Fastr combinan variaciones morfológicas combinadas con variaciones sintácticas. El nivel morfológico, se divide a su vez en dos subconjuntos uno dedicado a la explotación de las palabras y el otro a la explotación de los términos. Es decir, se almacena información sobre cada una de las palabras analizadas y sobre los términos obtenidos. El nivel sintáctico opera sobre la estructura sintáctica de los términos y sus variantes. En ambos niveles los términos se generan a partir de un conjunto de reglas.

El objetivo del nivel morfológico consiste en la detección de términos a partir de un conjunto de reglas predefinidas. Se muestra a continuación un ejemplo de una regla para un término formado por dos sustantivos.

N1 --> N2 N3

Una vez aplicada una regla, entra en juego el nivel sintáctico que trata de aplicar un conjunto de meta-reglas sobre las reglas del nivel morfológico, obteniendo así variantes de los términos y por consiguiente nuevos términos.

Reglas del nivel morfológico.

Estas reglas guardan información sobre el lema y la categoría de las palabras que componen el término. Se generan así un conjunto de ecuaciones que indican las restricciones sobre los mismos. El sistema en primer lugar efectúa un análisis que filtra las reglas posibles y recupera los términos que coinciden con los patrones de esas reglas.

Supóngase el término resultado financiero
Para cada palabra se aplica una *regla de palabra sencilla*:
Palabra 'resultado':
< cat > = 'N'
Para cada término se aplica una *regla de término*:
Regla: N1 --> N2 A3
<N1 lexicalización > N2
<N2 lemma> = resultado
<A3 lemma> = financiero

Reglas del nivel sintáctico, meta-reglas.

Jacquemin destaca que las palabras de una misma categoría léxica poseen un esquema de frase que constituye su entorno prototipo. Es decir, la colocación o posicionamiento de una palabra de cierta categoría en una frase es habitualmente la misma. Así generalmente un adjetivo se coloca después que el sustantivo al que modifica y un artículo se sitúa antes del sustantivo al que determina. Basándose en este hecho Jacquemin define las transformaciones de los términos que dan lugar a las meta-reglas.

Las meta-reglas se componen de un origen que se casa con una regla inicial definida en el nivel morfológico y un destino que dará lugar a la variante del término. Este es un proceso incremental, se generan inicialmente un conjunto de términos y sus variaciones. Estas variaciones dan lugar a nuevos términos que se vuelven a introducir en el proceso para buscar más variantes y términos, hasta que ya no se generan nuevos términos.

Fastr trabaja con tres tipos de reglas para generar las variantes:

- Coordinación: *(X1 --> X2 X3) = X2 C4 X5 X3*
  resultado financiero --> resultado y beneficio financiero
- Permutación: *X1 --> X2 X3 = X3 X2*
  asistencia recomendada ---> recomendada asistencia
- Inserción: *(X1 -->X2 X3) = X2 X4 X3*
  asistencia por teléfono --> asistencia técnica por teléfono

Para entender mejor el funcionamiento se describe a continuación un ejemplo. Sea **beneficio financiero** un término canónico reconocido por una regla morfológica. En el momento de aplicar una meta-regla para descubrir una variante, se captura una coordinación: **resultado y beneficio financiero**. Gracias a esa variante se ha descubierto un nuevo término que será **resultado financiero**.

Se establece una relación conceptual entre los términos y sus variantes, de ahí la importancia del cálculo de las mismas. Además esta relación varía en función del tipo de meta-regla aplicada.

El autor parte del supuesto de que dados dos términos que aparecen coordinados, éstos comparten un mismo esquema semántico, es decir, pertenecen a una misma clase semántica.

Por otro lado, parte del supuesto de que un término y su variante definidos por inserción establecen una relación taxonómica entre ellos, es decir , una clasificación jerárquica de los términos. Se establecen por lo tanto relaciones de hiperonimia.

14.2.RECONOCIMIENTO DE TÉRMINOS

Como se ha comentado anteriormente, el análisis de un corpus por parte de Fastr encadena una fase morfológica y una fase sintáctica. Las palabras y los términos detectados en la primera fase se reciclan en forma de reglas para dar paso a la segunda fase con un mecanismo de producción de variantes a partir de meta-reglas.

Fastr recibe como entrada un texto bruto, a continuación lo etiqueta mediante TreeTagger (Sección:8). En el caso de este proyecto se proporciona directamente el texto etiquetado con TreeTagger, esto se debe a que se ha incluido una fase previa al análisis léxico que pre-procesa el texto (Sección: 10). La primera etapa de la extracción es el análisis morfológico que filtra la entrada que proporciona TreeTagger y genera una colección de términos candidatos que responden a la gramática definida para las reglas morfológicas. A partir de esta lista, cada palabra contenida en un término se convierte en una *regla de palabra sencilla* y cada término se convierte en una *regla de término* (ambas estructuras en el apartado 14.1). Y, por último, se ejecuta el proceso de análisis sintáctico que generará las variantes a partir de los términos filtrados de la etapa anterior y las meta-reglas definidas para esta etapa.

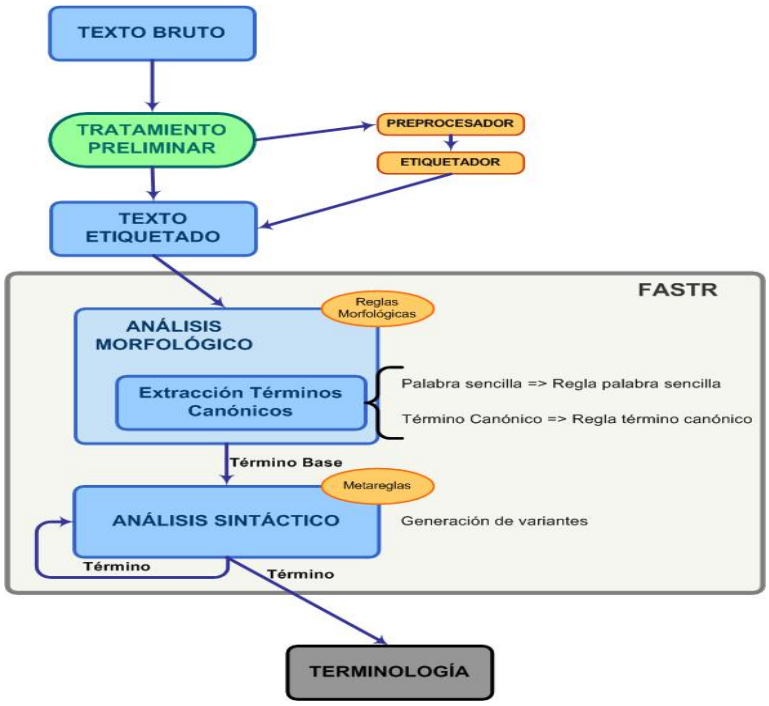


Figura 15 arquitectura de fastr

ONTOLOGÍAS

15.ONTOLOGÍAS

El término ontología se ha empleado desde los albores de la historia en el campo de la filosofía y del conocimiento. Décadas atrás cobró especial importancia en ámbitos como la biblioteconomía y la documentación. Las ontologías se usaron por primera vez en informática en el campo de la Inteligencia Artificial (IA), para poder facilitar la compartición y reusabilidad del conocimiento. A partir de los años 90 se convirtieron en una línea de trabajo muy popular dentro de varias corrientes de investigación relacionadas con la IA. Así, podemos destacar la ingeniería y representación del conocimiento y el procesamiento del lenguaje natural. En nuestros días, el estudio de las ontologías se ha extendido a otros ámbitos como la integración inteligente de información, los sistemas de información cooperativos o el comercio electrónico. Pero probablemente el ámbito más conocido de aplicación de las ontologías en nuestros días, es el *World Wide Web* a través de lo que se conoce como Web Semántica.

La representación del conocimiento es un tema de estudio en la informática que podemos encontrar ya en la década de los 60. Su objetivo fundamental es encontrar la forma de almacenar el conocimiento de manera que resulte operativo para la máquina, buscando en cada instante cumplir una serie de premisas:

- Hacer un modelado adecuado de la realidad. Las características que se elijan para el modelo tienen que ser relevantes para el uso que se quiera hacer de la representación.

- Conseguir el mayor grado posible de eficiencia para la computación. Se espera manipular el conocimiento de una forma útil en un tiempo razonable.

- Realizar un modelo útil como medio de expresión. Puesto que se está hablando de una representación de un dominio, debe de ser un medio de comunicación satisfactorio entre los expertos del mismo.

15.1.DEFINICIÓN DE ONTOLOGÍA

Existen diferentes acepciones para el término *ontología*. Se puede definir como una especificación de una conceptualización compartida [13], esto es, un marco común o una estructura conceptual sistematizada y de consenso no sólo para almacenar información, sino también para poder recuperarla. Por lo tanto se puede decir que una ontología define un vocabulario común para aquellas personas que necesiten compartir una información sobre un dominio.

De este modo, una ontología incluye un conjunto de términos relevantes en el dominio, la especificación de significado de cada uno y la relación existente entre ellos. Todo ello para lograr la comprensión de un área de conocimiento. Se trata pues de organizar la información en unas estructuras formalizadas. En éstas se podrán referenciar los datos relevantes en el dominio y estudiar los vínculos entre esos datos. Por lo tanto, las ontologías no sólo son una estructura que almacena información sino que contienen referencias para poder hacer deducciones sobre los datos guardados.

Llegados a este punto es importante destacar que los conceptos y relaciones presentes en una ontología, además de pertenecer a un dominio, son elementos compartidos y consensuados. Es decir, que la noción de ontología captura conocimiento aceptado por un grupo de personas y no sólo por un individuo. Además, se trata de desarrollar ontologías formales, esto es, procesables por los ordenadores.

15.1.1.ONTOLOGÍA VS TAXONOMÍA Y ONTOLOGÍA VS TESAURO

En general, no está muy clara la diferencia entre taxonomía y ontología. Los expertos en el tema distinguen las ontologías superficiales que son principalmente taxonomías que describen el dominio de manera poco profunda. De las que modelan el dominio más exhaustivamente y en las que la semántica del dominio está más restringida.

Por otro lado, tanto los tesauros como las ontologías son herramientas que sirven para estructurar conceptualmente determinados ámbitos del conocimiento por medio de vocabularios controlados. La diferencia entre los tesauros y las ontologías radica en la complejidad ya que estas últimas introducen un mayor nivel de

profundización semántica y proporcionan una descripción lógica y formal que puede ser interpretada tanto por las personas, como por las máquinas, mientras que los tesauros sólo pueden ser interpretados por humanos. Las ontologías permiten, además, la interoperabilidad entre sistemas distintos.

15.1.2.SEMÁNTICA VS SINTAXIS

En la búsqueda de documentos (o de páginas Web) a través por ejemplo de un buscador web como *Google*[9], la consulta del usuario es procesada recuperando aquellos documentos en los cuales aparecen los elementos constituyentes de la consulta. Sin embargo, esto puede aportar una serie de errores:

POLISEMIA: Se realizan consultas a través de un término. Los resultados devueltos incluyen documentos con el término solicitado, pero con un significado distinto.

SINONIMIA: Se realiza una consulta que no devuelve ningún documento, esto es debido a que en los textos aparece un sinónimo del término de la consulta y no el propio término.

MULTILINGUISMO: Se realiza una consulta con términos en un determinado idioma que no está presente en los textos. Sin embargo, el término está presente en otro lenguaje.

El problema en estos tres casos es que el motor de búsqueda no identifica de forma precisa las cosas en las que se está interesado puesto que al no poder hacer inferencia sobre la consulta no recupera textos relevantes. Este procedimiento de búsqueda localiza los términos de la consulta tal y como se han introducido. Este tipo de búsqueda se conoce como **búsqueda sintáctica**.

Ahora bien, planteado el problema se divisa cual es la solución y la importancia que cobran en este punto las ontologías. Pues sería interesante que los motores de búsqueda tuvieran asociados anotaciones formales que identificasen de forma inequívoca los conceptos principales en el conjunto de textos de búsqueda. De esta forma, mediante la utilización de una ontología que estructurase esos conceptos, se

[9] http://www.google.com

podrían suplir los inconvenientes citados, ya que la consulta navegaría en primer lugar a través de la ontología par encontrar los conceptos relacionados con su ámbito.

15.1.3.BENEFICIOS Y COSTES DE LAS ONTOLOGÍAS

Una vez ubicado el concepto de ontología es interesante plantear cuales son los beneficios en la utilización de ontologías.

- Proporcionan una forma potente y eficiente de representar y compartir el conocimiento dentro de un área, utilizando para ello un vocabulario común.
- Permiten usar un formato de intercambio de conocimiento. Este hecho se ve cada vez más potenciado por la utilización de estándares y lenguajes de definición de ontología.
- Proporcionan un protocolo específico de comunicación.
- Son una importante fuente de conocimiento, esto implica la reusabilidad del mismo. Pues es cada vez más común la integración de ontologías en el desarrollo de sistemas.
- Permiten analizar el conocimiento del dominio.
- Separan el conocimiento del dominio, del conocimiento operacional. De este modo, permiten hacer independientes las técnicas y algoritmos para solucionar un problema, del conocimiento concreto del problema.
- La combinación de distintas ontologías aportan apoyo en la búsqueda de documentos, en la clasificación de textos, integración de bases de datos, etc.

Existen también una serie de desventajas a la hora de utilizar y/o desarrollar ontologías

- A pesar del amplio número de áreas de aplicación es necesario tener en cuenta que la aplicación de una ontología está muy sujeta al dominio tratado. En estos momentos, resultaría imposibles e inabordable hacer una ontología genérica que abarcase un gran número de dominios al mismo tiempo.

- Cada ontología está muy sujeta a la interpretación que el desarrollador haga de ella. Son, por el momento, unas estructuras muy subjetivas.

- A la hora de integrar ontologías podrían surgir problemas de integridad en los términos. Pues en ciertos ámbitos un términos podría tener un significado y en otro uno completamente distinto. A modo de ejemplo, la palabra 'análisis' podría referirse a uno de sangre en un dominio de medicina o a una actividad a realizar durante el ciclo de vida de un producto software.

- Adicionalmente también podrían aparecer sinónimos, es decir, diferentes términos para referirse al mismo concepto. La sinonimia incluye de forma natural problemas de ambigüedad en el análisis de los textos.

- Otro problema que está relacionado con las suposiciones es la pérdida de conocimiento ligado al sentido común. Es posible olvidar en la ontología algo importante porque se asume que es evidente y que todo el mundo la conoce.

15.1.4.APLICACIÓN DE LAS ONTOLOGÍAS

Las ontologías, como modelo de representación de conocimiento, son una herramienta ampliamente aceptada y utilizada en la práctica mayoría de las áreas científicas. La capacidad de aplicarlas en un entorno informático para recabar información, integrarlas en una herramienta y el apoyo que aportan al usuario, les conceden una amplia popularidad.

En particular, están especialmente recomendadas en las situaciones en las que sea necesario:

- Establecer comunicación entre personas y organizaciones con el objetivo de unir diferentes áreas de investigación.

- Permitir que distintos sistemas software puedan comunicarse utilizándolas cómo pasarela. De esto modo, la ontología define una plataforma semántica independiente del lenguaje particular considerado en cada momento.

- Aumentar los beneficios a la hora de reutilizar software basado en conocimiento ya que las ontologías facilitan la construcción de este tipo de software de forma fiable y específica.

15.1.5.TIPOS DE ONTOLOGÍAS

Las ontologías se pueden clasificar teniendo en cuenta diferentes criterios:

ALCANCE DE SU APLICABILIDAD

- Ontologías de dominio: Proporcionan un vocabulario necesario para describir un dominio dado. Incluyen términos relacionados con los objetos del dominio y sus componentes.

- Ontologías de tarea: Proporcionan un vocabulario para describir términos involucrados en los procesos de resolución de problemas los cuales pueden estar relacionados con tareas similares en el mismo dominio o en dominios distintos. Incluyen nombres, verbos, frases y adjetivos relacionados con la tarea.

- Ontologías generales: Representan los datos generales que no son específicos de un dominio. Son las ontologías de nivel más alto ya que describen conceptos generales (espacio, tiempo, materia, objeto, etc)

- Ontologías específicas: Son ontologías especializadas que describen los conceptos para un campo limitado del conocimiento o una aplicación concreta.

GRANULARIDAD DE LA CONCEPTUALIZACIÓN

- Ontologías terminológicas: Especifican términos que son usados para representar conocimiento en el universo de discurso. Suelen ser usadas para unificar vocabulario en un dominio determinado.

- Ontologías de información: Ofrecen un marco para el almacenamiento estandarizado de información.

- Ontologías de modelado del conocimiento: Especifican conceptualizaciones del conocimiento. Poseen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen (términos y semántica).

AGENTES QUE VAYAN A EMPLEAR LA ONTOLOGÍA.

- Ontologías lingüísticas: Se vinculan a aspectos lingüísticos tales como los gramaticales, semánticos y sintácticos, destinados a su utilización por los seres humanos.

- Ontologías no lingüísticas: Aquellas ontologías destinadas a ser empleadas por robots o agentes inteligentes.

- Ontologías mixtas: Combinan las características de las anteriores.

NIVEL DE ABSTRACCIÓN Y RAZONAMIENTO LÓGICO PERMITIDO

- Ontologías descriptivas: Estas ontologías incluyen descripciones, taxonomías de conceptos, relaciones entre los conceptos y propiedades, pero no permiten inferencias lógicas.

- Ontologías lógicas: Estas ontologías permiten inferencias lógicas mediante la utilización de una serie de componente como la inclusión de axiomas, etc.

15.2. DESCRIPCIÓN DE UNA ONTOLOGÍA

Se puede describir una ontología utilizando distintos tipos de lenguajes:

- Empleando el lenguaje natural dando lugar a ontologías altamente flexibles y cercanas al usuario.

- Empleando una forma estructurada y restringida del lenguaje natural.

- Empleando un lenguaje formal, como por ejemplo OWL)

- Empleando un lenguaje artificial perfectamente definido (semántica formal, teoremas y pruebas de completitud).

15.2.1. COMPONENTES DE UNA ONTOLOGÍA

Gruber[13] propuso utilizar técnicas de IA, *frames* y lógica de primer orden para modelar ontologías. Propuso además cinco componentes básicos dentro de una ontología y que se presentan a continuación:

Conceptos: Se pueden definir como las clases básicas que se pretenden formalizar. Representan las ideas relevantes en el modelo. Los conceptos pueden ser clases de objeto, métodos, planes, estrategias o procesos de razonamiento. En una

ontología se suelen organizar de forma jerárquica y se les pueden aplicar mecanismos de herencia. A modo de ejemplo en una ontología sobre la Humanidad posibles clases serían Persona y País.

Relaciones: Representan las interacciones y enlaces entre los conceptos del dominio. Forman la taxonomía del dominio. Las uniones que establecen son del tipo subclase-de, parte-de, conectado-a, etc. Por ejemplo, una Persona es subclase_de Humano.

Funciones: Son casos especiales de relaciones donde se identifican elementos mediante el cálculo de una función que implica a varios elementos de la ontología. Por ejemplo, Madre_de: Persona -> Mujer.

Instancias: Son entidades pertenecientes a una determinada clase, es decir, con los elementos concretos de las clases. Por ejemplo, España es una instancia de País.

Axiomas o reglas de inferencia: Sirven para modelar sentencias que son siempre ciertas. Son teoremas sobre relaciones que deben cumplir los elementos de una ontología. Se utilizan para representar conocimiento que no puede modelarse con los otros tres componentes.

15.3.LENGUAJES DE DEFINICIÓN DE ONTOLOGÍAS

Las ontologías requieren ser expresadas a través de un lenguaje lógico y formal. Se han desarrollado numerosos lenguajes para este fin. Algunos lenguajes que están basados en la lógica de predicados, ofrecen potentes primitivas de modelado, y otros basados en *frames* tienen un mayor poder expresivo a costa de disminuir su capacidad de inferencia. Dentro de los lenguajes de ontologías podemos destacar los siguientes: (las referencias web de estos lenguajes se recogen en la sección de bibliografía).

KIF

Knowledge Interchange Format. Fue uno de los primeros lenguajes de representación de conocimiento. Fue diseñado como un formato de intercambio de conocimiento y desarrollado por el grupo de trabajo de Interlingua de la Universidad de Standford en 1992. Se pretendía que fuera un estándar que solucionara los problemas derivados de la heterogeneidad de lenguajes dentro del campo de la Representación del Conocimiento y que, a su vez, posibilitara el intercambio de conocimiento entre diferentes sistemas de información.

Está basado en la lógica de predicados con extensiones para definir términos. Busca ser un lenguaje capaz de representar la mayoría de los conceptos y distinciones actuales de los lenguajes más recientes de representación de conocimiento.

OIL

Ontology Interchange Language. Es un lenguaje de especificación de ontologías orientado a la Web, se tomó como punto de partida RDF Schema y se añadieron características necesarias para enriquecerlo y convertirlo en un lenguaje que junto con DAML, forman la base de OWL. En OIL, una ontología es una estructura formada por varios componentes organizados en tres niveles: el nivel de objetos (instancias), el primer meta-nivel o nivel de definición de la ontología y el segundo meta-nivel o contenedor de ontología en el cual se definen las propiedades de una determinada ontología.

OIL es un a apuesta de Europa para un lenguaje estándar de especificación de ontologías. En el mismo momento se estaba trabajando en EEUU sobre una propuesta parecida llamada DAML. En diciembre del 2000 se convino trabajar en conjunto y unir los lenguajes creados, que pasaron a llamarse DAMIL+OIL

FLOGIC

Frame Logic.[14] Fue creado por el Departamento de Ciencias de la computación de la Universidad estatal de Nueva York. Inicialmente se desarrolló como un aproximación orientada a objetos y a la lógica de primer orden, para ser utilizado en bases de datos deductivas y orientadas a objetos. Más tarde evolucionó

hacia un lenguaje de representación de ontologías. Acepta identidad de objetos, herencia, métodos de consulta, etc.

RDF

Resource Description Framework. Es un lenguaje creado por el W3C y está orientado a la creación de meta-datos para la descripción de recursos web. Está muy relacionado con XML pues RDF hace posible la especificación de información semántica, mediante la utilización de XML. Proporciona un modelo de datos para describir la semántica de la información de una forma accesible por la máquina. Un objeto se describe mediante sus propiedades.

RDF SCHEMA

Resource Description Framework Schema (RDFS). Es un lenguaje que se utiliza en la declaración de esquemas RDF. El modelo de datos de RDFS está basado en *frames* y proporciona un mecanismo para la definición de relaciones entre propiedades y recursos.

OWL

Web Ontology Language. Es un lenguaje etiquetado semántico para publicar y compartir ontologías en la Web. Se trata de una recomendación del *W3C*, y puede usarse para representar ontologías de forma explicita. Permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos. Es una extensión de RDF con mayor expresividad.

CONCLUSIONES

16.DIFICULTADES ENCONTRADAS

A lo largo del desarrollo se han ido encontrando diversas dificultades que se han superado con éxito. Los principales problemas surgieron durante la fase de estudio y durante la fase de implementación. A continuación se detallan cuales han sido los más relevantes y como se han podido solucionar:

- Cabe destacar la ardua tarea de investigación llevada a cabo para encontrar, entender y poner en marcha las distintas herramientas de PLN. En una persona sin experiencia la búsqueda de este tipo de recursos es de suma dificultad, pues se desconocen los sitios donde buscar información sobre los mismos y donde obtenerlos. A esto se suma el hecho de que al trabajar con dos idiomas poco explotados en el campo de PLN, como son el francés y el español (sobre todo para éste), el encontrar aplicaciones dedicadas a los mismos es tarea difícil. Por otra parte, a pesar de encontrar en ciertos casos herramientas potencialmente útiles, una vez se instalan y se entienden, se descubre que no interesan para los requisitos planteados.

- Si bien se encontraron herramientas, éstas cumplían en parte los objetivos. Esto da lugar a tener que adaptarlas unas a otras y adaptarlas al propio sistema para poder así, cumplir con los requisitos planteados. Esto supuso un considerable aumento en el tiempo de desarrollo.

- El tamaño del corpus ha sido sin lugar a duda uno de los factores determinantes en la eficiencia del sistema. Al tener que utilizar el corpus en las tareas más relevantes, sus debilidades se trasladan a todos los módulos del sistema. Se ha contado, con un conjunto de noticias de un mismo periódico, en los idiomas francés y español. El número de estas noticias era de 65 para cada uno de las lenguas, lo que implica que en algunas tareas los resultados obtenidos fuesen confusos y poco relevantes pues el corpus es demasiado pequeño.

- Al problema anterior hay que sumar el no estar en posesión de un corpus de entrenamiento para entrenar la herramienta TreeTagger. Esta herramienta posee un lexicón propio, tanto para el francés como para el español, pero el

número de palabras desconocidas era muy elevado. Esto se suplió añadiendo esas palabras al lexicón de forma manual.

- Otra dificultad añadida fue la codificación de caracteres de los textos y la codificación aceptada por las herramientas de PLN. La mayoría de las herramientas trabajan con la codificación ISO-8859-15 (ó latin-1), sin embargo, Acabit necesita una conversión de los textos proporcionados a codificación UTF-8. A esto hay que sumar que Java trabaja con UTF-8, por lo que todas las operaciones efectuadas de dentro a fuera (y de fuera a dentro) de la máquina virtual Java deben pasar previamente una conversión a la codificación oportuna.

- Son destacables las dificultades surgidas debido a la inexperiencia y desconocimiento de las tecnologías empleadas. Inicialmente el proceso de implementación se vio ralentizado por la falta de experiencia con el lenguaje de programación Java, así como con todos los API's utilizados. Destacando, sobre todo, la manipulación de la ontología con la librería Jena.

- En cuanto al desarrollo de la interfaz surgieron problemas de incompatibilidad con el editor visual de Eclipse (*VisualEditor*) y la versión de Eclipse. Este hecho se solucionó optando por otra plataforma de desarrollo para implementar la interfaz gráfica, la plataforma *NetBeans*. Esto supuso tener que trasladar todo el código generado en *NetBeans* al código creado en Eclipse, invirtiendo en ello un tiempo considerable.

17.CONCLUSIONES

En obligada referencia al contexto de globalización de los canales de información, el tratamiento de los contenidos económicos constituye un tema de especial interés tanto por su impacto social como por el carácter crítico en la toma de decisiones asociadas. En este punto, el proceso de análisis de datos está condicionado por la avalancha en el flujo generativo, así como por su acceso masivo y universal, que dificulta un seguimiento manual de los mismos. A ello hay que unir la complejidad técnica intrínseca al tratamiento de informaciones en las que los datos están fuertemente condicionados por matices, opiniones, aclaraciones y correcciones emitidas por actores sociales, políticos y económicos que, a menudo, resultan decisivos en la elaboración de estrategias de planificación.

Tras referir el ámbito de trabajo, el objetivo es la puesta en marcha de un protocolo de análisis de datos que elimine, en lo posible, el factor humano en la toma de decisiones y, por tanto, la subjetividad en la interpretación por parte del actor. Una tarea de este tipo requiere de la concurrencia de técnicas y especialistas en diferentes áreas. En general, podemos referirnos a los ámbitos del procesamiento del lenguaje natural (PLN), la lingüística computacional, la inteligencia artificial (IA) y la economía de mercado; cuyas aportaciones podemos organizar en campos de trabajo multidisciplinares que actualmente resultan perfectamente reconocibles.

Así, el marco habitual de actuación son las estrategias de recuperación de información (RI) , de extracción de información (EI) y de búsqueda de respuestas (BR). Las primeras proporcionan al usuario una lista de documentos en una colección dada, quizás priorizada en relación a algún criterio que refleje su impacto en el contexto de la pregunta; en los que éste pueda consultar los datos de su interés. Las técnicas EI constituyen un refinamiento adicional cuyo objetivo es señalar una información pertinente, indicando expresamente su localización en los propios documentos. Finalmente, las estrategias BR no sólo localizan un dato determinado para el usuario, sino que también son capaces de extraerlo y/o sintetizarlo a partir de diferentes fuentes; y quizás auxiliado por un proceso de diálogo interactivo con éste.

La inclusión de uno u otro grupo de estrategias depende en la práctica de dos factores. Por un lado del grado de estructuración reconocible en el texto y, por otro, su grado de especialización. La explotación eficiente de estos recursos constituye el centro de este trabajo, lo que desde un punto de vista técnico se traduce en la puesta en marcha de un protocolo de adquisición/representación automática de conocimiento. Dicha estructura lógica reflejará fielmente las dependencias semánticas que constituyen los conceptos contenidos en los documentos analizados, lo que permitirá establecer criterios objetivos para la generación de índices de clasificación/localización en bibliotecas digitales, así como para la gestión de éstos en entornos EI/RI/BR, todo ello evitando valoraciones subjetivas derivadas de una interpretación del documento por parte del actor. Se garantiza, de este modo, que la clasificación, gestión y acceso a la información sean resultado exclusivamente del significado extraíble de los documentos que lo contienen, lo que redundará en su eficacia y calidad.

En un marco como el descrito, el trabajo que se presenta constituye la formalización y validación de un protocolo de prototipado que incluye todas las fases en la adquisición del conocimiento a partir de texto plano y que no se limita a permitir la experimentación sobre entornos idiomáticos, de conocimiento y de recursos, diferenciados. En esta línea, y con objeto de establecer un entorno unificado de experimentación, se ha buscado activamente la concurrencia de técnicas y herramientas comunes a todos los contextos de trabajo previsibles; con una especial atención a la procura de una flexibilidad real de esta arquitectura inicial.

Sin querer ser una contribución meramente formal, la propuesta se encuadra dentro de los objetivos de un proyecto de investigación (TIN2004-07246-C03-01), eminentemente práctico, cuya finalidad es proveer tecnología útil en el acceso y gestión de información económica en entornos multilingües. Es en este ámbito donde esta propuesta adquiere su verdadera dimensión, por cuanto el tratamiento de datos en lenguaje natural sobre una variedad idiomática, no necesariamente afín ni en el léxico ni en la sintaxis, sólo puede pasar por la generación y consideración de estructuras semánticas que reflejen sin deformaciones el sentido de los textos y de las preguntas que los interrogan y que, en consecuencia, permitan el trasvase de uno a otro contexto lingüístico con la mayor objetividad.

Más explícitamente, y en contraste con propuestas anteriores, nuestra acercamiento garantiza la objetividad en el proceso de extracción, por cuanto éste

se hace depender únicamente de la estructura gramatical del documento. En particular, la resolución de posibles ambigüedades resultantes de una débil cobertura formal en este punto, se resuelve recurriendo a un modelo iterativo de aprendizaje cuya estructura de cálculo son las propias dependencias sintácticas y cuyo objetivo es descartar las interpretaciones del texto no viables. La existencia y unicidad de un punto fijo en este proceso de razonamiento es garantía de su corrección y, desde un punto de vista más práctico, de la fidelidad del procedimiento seguido.

El texto es el canal de comunicación más poderoso, justificado por lo que consideramos un acceso universal y sencillo a los documentos escritos. Esta simplicidad es, sin embargo, aún un reto abierto en el ámbito de las aplicaciones informáticas. Se ha llegado, de hecho, a la paradoja de disponer del medio de preservar cantidades ilimitadas de información, pero cuya consulta por parte del usuario no especializado no ha evolucionado en la misma medida, falta de flexibilidad en su uso y de efectividad computacional. Esta propuesta quiere, en este sentido, ser una humilde contribución en este avance.

18.LÍNEAS FUTURAS DE TRABAJO

A continuación se plantean una serie de posibles ampliaciones del sistema desarrollado:

- Crear una interfaz que permita la manipulación del sistema a través de la línea de comandos, sin necesidad de utilizar la interfaz gráfica. Esto se realizaría, por ejemplo, añadiendo un fichero en el cual se incluirían las tareas que se quieren llevar a cabo, así como los parámetros necesarios para ejecutar cada una de ellas. Se piensa en esta ampliación para incorporar este sistema en una plataforma de mayor envergadura.

- Mejorar el sistema de traducción incorporando una herramienta más potente y menos dependiente que el traductor de Google, ya que éste necesita una conexión a Internet para su funcionamiento.

- Implementar una estrategia que hiciese coincidir en la medida de lo posible las estructuras de las distintas ontologías. De este modo ya no sería necesario un traductor, estableciendo una ontología siempre en la base, se podría transitar de un idioma a otro. La idea sería similar a la estructura implementada en la base de datos *EuroWordNet*.

- Incorporar un mayor número de herramientas de adquisición de terminología y de análisis sintáctico que exploten los idiomas actuales (francés y español).

- En el entorno concreto trabajado, sería de mucha utilidad contar con la opinión de un experto en el dominio. Esto cobra especial relevancia, a la hora de manipular la ontología creada a partir de la relación de dependencias, pues un usuario experto podría manipular los ficheros que controlan la creación de la ontología de una forma más eficiente.

- En cuanto a la interfaz gráfica, una posible ampliación sería mejorar el diseño de la misma.

- También a nivel gráfico, sería interesante incluir un menú que permitiese manejar los ficheros de gestión de la ontología por dependencias a través de la interfaz.

- Como se ha mencionado con anterioridad, una posible ampliación se situaría incorporando nuevos formatos para los documentos que se introduzcan en el sistema.

- La implementación del patrón DAO se hace pensando en la posibilidad de añadir otras fuentes de datos a parte de la base de datos, como por ejemplo podrían ser fichero XML. Así, pensando en utilizar un almacenamiento distinto para el análisis sintáctico y la generación de la ontología por dependencias, el uso de patrón DAO facilitaría esta conversión, pues bastaría implementar cada una de las interfaces correspondientes.

APÉNDICE TÉCNICO

19.LUCENE

Lucene es una herramienta que permite tanto la indexación cómo la búsqueda de documentos. Creada bajo una metodología orientada a objetos e implementada completamente en Java, no se trata de una aplicación que pueda ser descargada, instalada y ejecutada sino de un API flexible, y potente, a través de la cual se pueden añadir capacidades de indexación y búsqueda a cualquier sistema que se esté desarrollando.

19.1.CONCEPTO

Originalmente escrita por Doug Cutting, en Septiembre de 2001 pasó a formar parte de la familia de código abierto de la fundación Jakarta. Desde entonces, debido a su mayor disponibilidad, ha atraído a un gran número de entidades interesadas en su desarrollo, incluso empresas como Hewlett Packard, FedEx, etc. lo usan, o al menos lo han evaluado.

Existen otras herramientas, a parte de *Lucene*, que permiten realizar la indexación y búsqueda de documentos, pero dichas herramientas han sido optimizadas para usos concretos, lo que implica que el intentar adaptar dichas herramientas a un proyecto específico sea una tarea realmente difícil. La idea que engloba *Lucene* es completamente diferente, ya que su principal ventaja es su flexibilidad, que permite su utilización en cualquier sistema que lleve a cabo procesos de indexación

19.2.CARACTERÍSTICAS

A continuación se detallan algunas características que hacen de *Lucene* una herramienta flexible y adaptable:

INDEXACIÓN INCREMENTAL VS INDEXACIÓN POR LOTES

El término de *indexación por lotes* se utiliza para referirse a aquellos procesos de indexación, en los cuales, una vez que ha sido creado el índice para un conjunto de documentos, el intentar añadir algunos documentos nuevos es una tarea difícil por lo que se opta por *reindexar* todos los documentos de nuevo. Sin embargo, en la indexación incremental se pueden añadir documentos a un índice ya creado con anterioridad de forma fácil. *Lucene* soporta ambos tipos de indexación.

ORIGEN DE DATOS

Muchas herramientas de indexación sólo permiten indexar ficheros o páginas web, lo que supone un serio inconveniente cuando se tiene que indexar contenido almacenado en una base de datos. *Lucene* permite indexar tanto documentos y páginas web como el contenido procedente de una base de datos.

CONTENIDO ETIQUETADO

Algunas herramientas, tratan los documentos como simples flujos de palabras. Sin embargo, *Lucene* permiten dividir el contenido de los documentos en campos y así poder realizar consultas con un mayor contenido semántico. Es decir, se pueden buscar términos en los distintos campos del documento concediéndole más importancia según el campo en el que aparezca.

TÉCNICA DE INDEXACIÓN

Existen palabras tales como *"a", "unos", "el", "la"*, etc. que añaden poco significado al índice, son palabras poco representativas del documento. Al eliminar estas palabras del índice se reduce considerablemente la complejidad espacial y temporal del proceso. Estas palabras están contenidas en lo que se denomina *lista de parada*.

ELECCIÓN DEL IDIOMA

Tal y como ya se indicó con anterioridad, *Lucene* trabaja con *listas de parada*, las cuales son proporcionadas por el programador. La *lista de parada* contendrá aquellas palabras, de un idioma, que se considera no aportan información para la indexación. Por lo tanto al utilizar una lista de este tipo para un determinado idioma, el analizador del índice, segmentará el texto de acuerdo a la misma, obteniendo resultados más precisos que si no se utilizase ninguna lista. Además existen implementaciones de analizadores de índice para idiomas concretos.

19.3.INDEXACIÓN DE DOCUMENTOS

Para buscar grandes cantidades de texto rápidamente, se debe primero poner en un índice ese texto y convertirlo en un formato que permita buscarlo de manera eficaz, eliminando ineficaces procesos secuenciales de exploración. Esta estrategia de la conversión se conoce como *indexación*, y su salida es un *índice*.

A continuación se hace una breve descripción de las clases utilizadas por *Lucene* para la indexación de documentos:

INDEXWRITER

Es el componente central del proceso de la indexación de direcciones. Esta clase crea un nuevo índice y agrega documentos a un índice existente.

DIRECTORY

Representa la localización de un índice de *Lucene*. Es una clase abstracta que permite que sus subclases, dos de las cuales se incluyen en *Lucene*, almacenen el índice mientras se modifica.

ANALYZER

Antes de que se introduzca texto en un índice, éste debe pasar a través de un analizador, implementado en la clase *Analyzer*. El *Analyzer*, especificado en el constructor de *IndexWriter*, se encarga de extraer *tokens* del texto para ser introducidos en un índice y de eliminar el resto. Si el contenido que se introduce en

un índice no es texto plano, debe primero ser convertido a dicho formato. *Analyzer* es una clase abstracta, pero *Lucene* tiene varias implementaciones para ella.

DOCUMENT

Representa una colección de campos, *fields*. Se puede pensar en él como en un documento virtual, un conjunto de datos, tales como una página Web, un correo electrónico, o un archivo de texto que se desea hacer recuperable en un tiempo posterior. Los campos de un documento representan al documento mismo o a los meta-datos asociados a ese documento. La fuente original (por ejemplo un expediente de la base de datos, un documento de Word o un capítulo de un libro) de los datos del documento son inaplicables a *Lucene*. Los meta-datos tales como autor, título, el tema o fecha modificada, se introducen en un índice y se almacenan por separado como campos de un documento. Así pues, para cada documento de texto, se crea una nueva instancia de la clase *Document*, rellenamos los campos, *Field* (que se explica a continuación), y agregamos ese documento al índice, indexando eficazmente ese documento.

FIELD

Cada *Document* en un índice presenta uno o más campos, contenidos en la clase *Field*. Cada campo corresponde a una porción de datos en relación a los cuales se pregunta o se recupera el índice durante la búsqueda. Lucene ofrece cuatro tipos de campos:

Keyword: No se analiza, se almacena e indexa en el índice literalmente. Este tipo es conveniente para los campos cuyo valor original se debe preservar en su totalidad, tales como URLs, rutas del sistema de ficheros, fechas, nombres, números de seguridad social, números de teléfono, etc.

UnIndexed: No se analiza ni se indexa, pero su valor es almacenado en el índice. Este tipo es conveniente para los campos que se necesita exhibir con los resultados de la búsqueda (tales como una clave primaria del URL o de la base de datos), pero cuyos valores nunca se buscarán directamente.

UnStored: Opuesto a *UnIndexed*, este campo es analizado e indexado pero no es almacenado en el índice. Es conveniente para introducir en un índice una

gran cantidad de texto que no se necesita recuperar en su forma original, tal como cuerpos de páginas Web, o cualquier otro tipo de documento del texto.

Text: Este campo se analiza y se indexa, pero no siempre es almacenado

19.4.BÚSQUEDA DE DOCUMENTOS

La búsqueda es el proceso mediante el cual se examinan palabras en el índice para encontrar aquellos documentos en los que aparecen dichas palabras. La calidad de la búsqueda es típicamente descrita utilizando métricas de *precisión* y *cobertura*. Donde la *cobertura* se refiere a la fracción de los documentos relevantes que fue devuelta; y la *precisión* a la fracción de documentos devueltos que son relevantes para la consulta. Sin embargo, se debe considerar un gran número de otros factores al hablar de búsquedas. Como la velocidad y la capacidad búsqueda para cantidades grandes de texto. El soporte para consultas simples y de multi-término, los comodines, el filtrado y ordenamiento del resultado son también importantes, al igual que una sintaxis amistosa para incorporar esas consultas.

A continuación se hace una breve descripción de las clases utilizadas por Lucene para la búsqueda de documentos:

Search

La clase *Searcher* es una clase abstracta y constituye la base para cualquier búsqueda en un índice de documentos. Declara métodos que son implementados por sus subclases, como por ejemplo, *IndexSearcher*, proporcionando así la forma de acceder y recuperar información indexada.

Term

Un término es la unidad básica para buscar, y se implementa en la clase *Term*. Similar al objeto *Field*, consiste en un par de elementos: el nombre del campo y el valor de ese campo.

Query

La clase *Query,* al igual que *Searcher* es una clase abstracta siendo *QueryParser* su subclase más importante, pero además existen otras como *RangeQuery*, para buscar por rangos de valores; *PrefixQuery*, para buscar dentro de una cadena; *BooleanQuery*, para búsquedas por diversos criterios; *PhraseQuery*, para añadir *Terms* en el orden deseado y *TermQuery*, para emparejar los documentos que contienen campos con valores específicos.

Hits

Es una clase que permite recuperar los resultados de la búsqueda.

20.OWL

OWL es un lenguaje creado para la *Web Semántica* y desarrollado por el *W3C Ontology Working Group* para publicar y compartir ontologías en la Web.

OWL es un lenguaje XML que se aventaja de la universalidad de la sintaxis de XML. Está basado en la sintaxis de RDF/XML. OWL ofrece un medio para crear ontologías. Se diferencia de RDF/RDFS en que es un lenguaje de ontologías. Proporciona formas de describir clases y propiedades, al igual que RDF/RDFS. También proporciona la capacidad de comparar clases mediante operaciones de identidad, opuesto, cardinalidad, etc.

20.1.SUBLENGUAJES DE OWL

OWL proporciona tres sub-lenguajes, de creciente expresividad, que ofrecen distintas capacidades de expresión y destinados a distintas comunidades de desarrolladores.

- **OWL Lite**: Es el sub-lenguaje menos complejo. Destinado a aquellos usuarios que necesitan una jerarquía de conceptos sencilla.
- **OWL DL**: Permite una expresividad mayor. Está fundado en la lógica descriptiva y por lo tanto confiere a OWL DL su adaptación al razonamiento automatizado.
- **OWL Full**: Es la versión más compleja de OWL, pero es aquella que ofrece mayor nivel de expresividad. Está destinado a aquellas situaciones en las que es más importante tener un alto nivel de capacidad de descripción a cambio de no garantizar la completitud y la decisibilidad de los cálculos relativos a la ontología.

Existe entre estos lenguajes una dependencia de naturaleza jerárquica. Toda ontología OWL Lite es una ontología OWL DL; y toda ontología OWL DL es una ontología OWL Full.

20.2.ESTRUCTURA DE UNA ONTOLOGÍA OWL

A continuación se presenta los elementos constituyentes de una ontología en OWL. Para ilustrarlo se acompaña de una serie de ejemplos sobre una ontología acerca del concepto humanidad.

OWL fue creado pensando en su flexibilidad. Este hecho viene dado debido a la naturaleza distribuida y dinámica de la Web. Esto, permite además, extender ontologías existentes o emplear diversas ontologías ya existentes para completar la definición de una nueva ontología.

20.2.1.ESPACIO DE NOMBRE

Para poder emplear términos en una ontología, es necesario indicar de qué vocabulario provienen. De este modo, al igual que sucede en muchos documentos XML, una ontología empieza con la declaración del espacio de nombres que se encierra en una etiqueta *rdf*.

```
<rdf:RDF
    xmlns              = http://domain.tld/path/humanidad#
    xmlns:owl           = http://www.w3.org/2002/07/owl#
    xmlns:rdf          = http:// www.w3.org/1999/02/22-rdf-syntax-ns #
    xmlns:rdfs          = http:// www.w3.org/2000/01/rdf-schema#
    xmlns :xsd          = http:// www.w3.org/2001/XMLSchema#
```

En el primer espacio de nombre se indica el actual. En los cuatro siguientes se indican los espacios de nombre relativos a owl, rdf, rdfs y xsd. Esto es necesario para poder así emplear etiquetas propias definidas para esos lenguajes.

20.2.2.CABECERA DE LA ONTOLOGÍA

A continuación de la declaración del espacio de nombres se puede indicar la cabecera que describe el contenido de la ontología actual. Es la etiqueta *owl:Ontology* que permite indicar estas informaciones.

```
<owl:Ontology rdf:about=""
    <rdfs:comment>Ontología que describe la humanidad</rdfs:comment>
     …
```

20.2.3.LAS CLASES

Una clase define un grupo de individuos que se agrupan porque poseen características similares.

Una clase puede ser definida de cuatro maneras diferentes.

INDICADOR DE CLASE

La declaración se hace nombrando directamente la clase. Este es la única de las cuatro maneras que permite definir un nombre para la clase. En los otros casos las clases definidas se conocen como "anónimas".

```
<owl:Class rdf:ID="Humano"/>
```

ENUMERACIÓN DE LOS INDIVIDUOS

Mediante una enumeración de los individuos que pertenecen a una clase se puede definir la clase que los posee. Esto se hace mediante la propiedad *owl:oneOf*.

```
<owl:Class>
    <owl:oneOf rdf:parseType="Collection">
       <owl:Thing rdf:about="María"/>
       <owl:Thing rdf:about="Luis"/>
    </owl:oneOf>
<owl:Class>
```

A través del tipo *Collection* definido en la variable *parseType* de rdf se indica que lo que se presenta es un conjunto de elementos, en este caso de personas.

RESTRICCIÓN DE LAS PROPIEDADES

En este caso se define una clase anónima cuyas instancias deben satisfacer una determinada propiedad. Este tipo de restricciones pueden ser de valor o de cardinalidad. En el primer caso la restricción se restringe a un valor de una propiedad del individuo. En el segundo caso la restricción se enfoca al número de valores que puede tomar una determinada propiedad.

```
<owl:Class rdf:ID="FamiliaNumerosa">
  <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="#tieneNumHijos"/>
        <owl:mincardinality  rdf:datatype="&xsd;nonNegativeInteger">
            3</owl:mincardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

En este caso la clase anónima viene definida entre las etiquetas *subClassOf*. El ejemplo muestra como la clase *FamiliaNumerosa* debe de ser una subclase de una clase anónima que indica que para la propiedad *tieneNumHijos* el valor ha de ser por lo menos 3.

HERENCIA

A través de una propiedad de herencia se puede crear una nueva clase. Esto se hace a través de la etiqueta subClassOf.

```
<owl:Class rdf:ID="Hombre">
    <rdfs:subClassOf rdf:resource="#Humano"/>
</owl:Class>
```

20.2.4.LAS INSTANCIAS

Un individuo puede hacerse miembro de una clase de diversas maneras:

Mediante una declaración se indica que un determinado individuo pertenece a una clase y se indica además cuales son los valores de propiedad para ese individuo.

```
<Humano rdf:ID=”María”>
   <tienePorMadre rdf:resource=”#Rosa”/>
   <tienePorPadre rdf:resource=”#Miguel”/>
</Humano>
```

Mediante una declaración anónima para ello se procede como en el caso anterior pero obviando el nombre del individuo.

Estableciendo una relación entre el individuo y otro ya existente. Esto se consigue utilizando las propiedades, *owl:sameAs* *owl:differentFrom* y *owl:allDifferent* .

```
rdf:about=”Jose”>
   <owl:sameAs rdf:resource=”Pepe”/>
</rdf:Description>
```

20.2.5.LAS PROPIEDADES

Las propiedades permiten expresar hechos sobre las clases y sus instancias. OWL distingue entre dos tipos de propiedades:

- **Propiedades de objeto** que permiten unir instancias de clase. Este tipo de propiedad es una instancia de la clase *owl:ObjectProperty*.
- **Propiedades de tipo de dato**, que permiten enlazar individuos con valores. Este tipo de propiedad es una instancia de la clase *owl:DataTypeProperty*.

```
<owl:ObjectProperty rdf:ID=”tienePorPadre”/>
<owl:DataTypeProperty rdf:ID=” tieneNumHijos”/>
```

Es posible asignar otro tipo de características a las propiedades, entre las que encontramos la transitividad, la simetría, etc.

Especificar una propiedad no es más que restringir la relación que simboliza. Así, deben verse las propiedades como un función que hace corresponder a un individuo otro individuo o bien un valor. Por lo tanto, se pueden definir el dominio y la imagen de esta propiedad. Además se puede definir una propiedad como una especialización de una existente. Sirva a modo de ilustración el siguiente ejemplo:

```
<owl:ObjectProperty rdf:ID="vive">
    <rdfs:domain rdf:resource="#Humano"/>
    <rdfs:range rdf:resource="#País"/>
</owl:ObjectProperty>
```

donde, tanto Humano como País son clases que han sido definidas en la ontología.

BIBLIOGRAFÍA

21.BIBLIOGRAFÍA

REFERENCIAS BIBLIOGRÁFICAS

[1] Bourigault, D. & Jacquemin, Ch. *Construction de ressources terminologiques*. Ingénierie des langues, Jean-Marie Pierrel (ed.), Hermès, 2000, pp 215-233.

[2] Shmid, H. (1994) Probabilistic Part-of-Speech Tagging Using Decision Trees. In International Conference on New Methods in Languague Processing, 1994, Manchester, UK.

[3] Barcala. M, Domínguez. E, Alonso M.A., Cabrero. D, Graña. J, Vilares. J, Vilares. M, Rojo. G, Santalla. P, Sotelo. S. *Una aplicación de RI basada en PLN: el proyecto ERIAL.*

[4] Vilares, J.(2005) Aplicaciones del procesamiento del lenguaje natural en la recuperación de información en Español. Departamento de Ciencias de la computación, Universidade da Coruña.

[5] *Extracción de términos índice mediante cascadas de expresiones regulares*, in Actas de las II Jornadas de Tratamiento y Recuperación de Información (JOTRI 2003), pp. 204-211, Leganés (Madrid), Spain, 2003

[6] Graña, J. *Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural*. PhD thesis, Departamento de Computación, Universidade da Coruña, (Diciembre 2000)

[7] Namer, F. *Un analyseur flexionnel du français à base de règles.* Université de Nancy & Laboratoire LANDISCO

[8] Estopà, R., Vivaldi, J., Cabré, M.T. *Sistemes d'extracció automática de (candidats a) terms:Estat de la qüestió.* Institut Universitari de Lingüística Aplicada. Universitat Pompeu Fabra. Barcelona.

[9] Porta Zamorano, J. *Técnicas cuantitativas para la extracción de términos en un corpus*. Escuela politécnica Superior-Universidad Autónoma de Madrid. (2006)

[10] B. DAILLE, *"Conceptual structuring through term variations"*. In F. Bond, A. Korhonen, D. MacCarthy and A. Villacicencio (eds.), Proceedings ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment, 9-16, 2003b

[11] B. Daille, Du groupe prépositionnel à l'adjectif relationnel: vers une stabilité de la dénomination.

[12] Jacquemin, C. (1997). *Variation terminologique : Reconnaissance et acquisition automatiques de termes et de leurs variantes en corpus*. Mémoire d'Habilitation à Diriger des Recherches en informatique fondamentale, Université de Nantes, Nantes.

[13] Grubber, T. *"What is Ontology?",* http://www-ksl.stanford.edu/kst/what-is-naontology.html,Noviembre. (1999)

[14] Kifer,M. ; Lauser,G ;Wu,J. *"Logical Foundation of Object-Oriented and Frame-Based Languages"* Journal of the Association of Computing Machinery, Mayo 1995.

[15] Informe en el proyecto ATOLL (Atelier d'Outils Logiciels pour le Langage naturel) realizado por el INRIA (Institut National de Recherche en Informatique et Automatique).

[16] Bourigault, D., Lame, G. (2002) Analyse distributionnelle et structuration de terminologie. Revue TAL.

[17] Gavilanes Fernández,M; Villemonte de la Clergerie, É ; Vilares, M. *"Knowledge Acquisition through Error-Mining"* Aceptado en el congreso RANLP (Recent Advances in Natural Language Processing) pendiente de publicación. (Junio 2007-Rumanía).

[18] Gavilanes Fernández,M; Villemonte de la Clergerie, É ; Vilares, M. *"From Text to Knowledge"* Aceptado en el congreso EuroCast (Conference on Computer Aided Systems Theory) pendiente de publicación. Febrero 2007.

[19] Harris,Z. *Mathematical Structures of Languages*. John Wiley & Sons, New York, U.S.A. 1968.

[20] E. de la Clergerie. *Dyalog : a tabular logic programming based environment for nlp*. In Proc. Of 2nd Int. Workshop on Constraint Solving and Language Processing, Barcelona, Spain, 2005.

LIBROS CONSULTADOS

"Lucene in Action"
Erik Hatcher and Otis Gospodnetić
Manning Publications Co., 2004.

"Java 2: Interfaces gráficas y aplicaciones para Internet"
Fco. Javier Ceballos
Ra-Ma

"Introduction à la programmation en Perl (ou comment débuter en Perl)"
Sylvain Lhullier

"El lenguaje Unificado de Modelado"
Gragy Booch, James Rumbaugh, Ivar Jacobson
Addison Wesley

"Java manual de referencia"
Patrick Naughton, Herbert Shildt
McGraw- Hill

"XML : [imprescindible]"
Elliotte Rusty Harold, W. Scott Means
Madrid : Anaya Multimedia, 2005

"Patrones de diseño aplicados a Java"
Stephen Stelting, Olav Maassen
Pearson Educación, D.L. 2004

REFERENCIAS DE LA WEB

Procesamiento del lenguaje natural. Wikipedia enciclopedia libre.
http://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales.

Ontologías.
http://www.hipertexto.info/documentos/ontologias.htm

JAVA
http://java.sun.com

XML
http://www.w3.org/XML/

LUCENE
http://lucene.apache.org/

JENA2
http://jena.sourceforge.net/

OWL
http://www.w3.org/2004/OWL/

MYSQL
http://www.mysql.com/

SWING

TREETAGGER
http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

FLEMM
http://www.univ-nancy2.fr/pers/namer/

FASTR

http://www.limsi.fr/Individu/jacquemi/FASTR/

ACABIT

http://www.sciences.univ-nantes.fr/info/perso/permanents/daille/index.html

ECLIPSE

http://www.eclipse.org/

VISUAL PARADIGM

http://www.visual-paradigm.com/product/vpuml/

NETBEANS

http://www.netbeans.org/

UNBUNTU

http://www.ubuntu-es.org/

OTRAS FUENTES WEB

- Página oficial de EuroWordNet: http://www.illc.uva.nl/EuroWordNet/
- Página con manuales de perl: http://www.webtaller.com/construccion/lenguajes/info/manuales/perl/
- Página sobre jdom y sax: http://www.programacion.com/tutorial/apuntesxml/5/#java-dom-analizador
- Página sobre swing: http://www.labo-sun.com/resource-fr-essentiels-650-1-java-io-swing-interface-homme-machine.htm
- Página oficial de las conferencias TREC: http://trec.nist.gov/
- Página del ATILF (Analyse et Traitement Informatique de la Langue Française) página sobre recursos de PLN en francés : http://www.atilf.fr/
- Página del ATALA (Association pour le Traitement Automatique des Langues) página sobre recursos de PLN en francés y otros idiomas : http://www.atala.org/

- Página oficial de ant: http://ant.apache.org/index.html
- Página oficial de RDF: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
- Página oficial de KIF: http://logic.stanford.edu/kif/kif.html
- Página oficial de OIL: http://www.ontoknowledge.org/oil/
- Página oficial de RDFS: http://www.w3.org/TR/rdf-schema/

ÍNDICE DE FIGURAS

22.ÍNDICE FIGURAS