PROYECTO FIN DE CARRERA

Título:	Sistema de Búsqueda de Respuestas para un Servicio de Bots a partir de Foros de Discusión
Autor:	César Monteserín Gutiérrez
Tutor:	Miguel Coronado Barrios
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Gregorio Fernández Fernández
Vocal:	Mercedes Garijo Ayestarán
Secretario:	Carlos Ángel Iglesias Fernández
Suplente:	Ignacio Soto Campos

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

PROYECTO FIN DE CARRERA

SISTEMA DE BÚSQUEDA DE RESPUESTAS PARA UN SERVICIO DE BOTS A PARTIR DE FOROS DE DISCUSIÓN

Alumno: D. César Monteserín Gutiérrez

Tutor: D. Miguel Coronado Barrios

Ponente: D. Carlos Ángel Iglesias Fernández

2011

A mis padres, hermano y Cristina, por haber hecho que todo haya sido más fácil.

Resumen

Esta memoria presenta al sistema GSI QA, un sistema de búsqueda de respuestas o *Question-Answering*. El objetivo del sistema es comprender las preguntas del usuario, buscar esa información, recuperarla, extraerla y presentarla, todo ello en lenguaje natural. Para alcanzar dicho cometido, se ha trabajado con herramientas de procesamiento del lenguaje natural (NLP) y con algoritmos de aprendizaje de patrones o *pattern learning*. La información con la que se trabaja puede provenir de archivos de texto plano, PDF o HTML. El sistema se ha centrado en el campo de las definiciones o explicaciones de un concepto a través de sus propiedades o cualidades. Además, el sistema ha sido integrado en un sistema de evaluación de conocimientos para un proyecto centrado en personas con cierta discapacidad.

Palabras clave: Sistema Question-Answering, QAS, Sistema de búsqueda de respuestas, Moodle, Foro, Tesauro, SKOS, PLN, Procesado de lenguaje natural, Aprendizaje de patrones, Recuperación de información, Extracción de información.

Abstract

This Project describes the GSI QA system, a Question-Answering system. The system's aim is to understand user questions in order to search, retrieve, extract and finally present an appropriate answer. Every mentioned step is processed in natural language. Its development has been achieved by using natural language processing tools (NLP tools) and algorithms of pattern learning. The information managed by the system is handled by using different kind of formats such as plain text, PDF or HTML. The system is focused on the field of definitions or explanations of a concept through its properties or qualities. In addition, it has also been integrated into a system of evaluation of knowledge, mainly applied for helping people with certain disabilities.

Keywords: Question-Answering system, QAS, Bot, Moodle, Forum, Thesaurus, SKOS, NLP, Natural Language Processing, Pattern Learning, Information Retrieval, Information Extraction.

Agradecimientos

El camino llega a su fin, una etapa que se acaba y otra que comienza sin apenas uno darse cuenta. Son muchos los culpables de que esto ocurra, y para ellos es esta página.

En primer lugar, agradecer la oportunidad brindada por el Grupo de Sistema Inteligentes (GSI) para la realización de este proyecto. Dentro de este grupo no me olvido de los doctorando: Álvaro, José Ignacio, José Javier, Geovanny y Paco, ni tampoco de los becarios: Jesús, José Luis, Adriano, Alberto, Juan Antonio, Alejandro, Juan Fernando y Felipe. Todos habéis hecho que el ambiente de trabajo haya sido el idóneo.

Dentro del GSI han resaltado dos personas. Carlos, no llegará el día que deje de estarte agradecido por hacerme partícipe de este grupo y, lo que considero más importante, por hacerme ver que un docente también puede ser un amigo. Miguel, gracias por guiarme, por saber motivarme cada día y estar ahí siempre que lo he necesitado. A los dos, muchas gracias por devolverme las ganas de seguir aprendiendo.

A lo largo de la carrera he compartido grandes momentos y experiencias imborrables con Estefanía, Alberto, Silvia, Ruth, Noelia, Juan, Ismael, Pablo, etc. Gracias a todos por esos recuerdos en forma de viajes, convivencias y gestos.

A Gonzalo y a César porque han sido imprescindibles desde el primer día de universidad. Una gran cantidad de vivencias y recuerdos hacen que sigamos teniendo la misma relación de siempre.

Al trío formado por Óscar, Kike y Sergio que consiguen que cada encuentro sea un cúmulo de buenos momentos. Óscar, gracias por dar tanto significado a la palabra amistad.

A mi gran amiga Paloma, un ser maravilloso capaz de crear momentos inolvidables.

Los últimos serán los primeros. Gracias Sandro por escucharme, aguantarme y aconsejarme, pero sobre todo agradezco que seas mi hermano. Por último, agradecer al mejor modelo que se puede tener para crecer, por estar ahí día tras día, por no dejar que vaya hacia atrás ni para coger impulso, y por infinitas razones más, gracias mamá.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

"Todo comienzo tiene su encanto."
— Johann Wolfgang von Goethe

En este primer capítulo se realiza una introducción al Proyecto Fin de Carrera. Se presenta la estructura y organización de la memoria, la motivación y necesidad de llevar a cabo este proyecto, el marco o línea de investigación en el que se encuentra y los objetivos marcados y conseguidos.

1.1. Estructura de la memoria

Con el fin de introducir a los lectores a la presente memoria se comienza presentando brevemente la organización de sus capítulos. Esta memoria está compuesta por siete capítulos y cada uno de ellos está, a su vez, dividido en distintas secciones y apartados:

- Primer capítulo (1): se presenta este Proyecto Fin de Carrera a través de los motivos de llevarlo a cabo, del contexto o marco en el que se encuentra y de los objetivos marcados.
- Segundo capítulo (2): se proporciona una visión general sobre el mundo de los sistemas de búsqueda de respuestas o *Question Answering*. Además, se introducen los estados del arte de la Web semántica y del procesamiento del lenguaje natural.
- Tercer capítulo (3): se revisan diferentes herramientas utilizadas en este PFC, así como un estudio, comparativa, y elección de las herramientas de procesamiento natural del lenguaje más importantes hasta la fecha.
- Cuarto capítulo (4): se presentan los resultados correspondientes a la parte de análisis con el correspondiente estudio de los distintos escenarios y casos de uso.
- Quinto capítulo (5): se detalla el diseño de la arquitectura del sistema realizado. Se revisan las distintas fases del sistema y la integración en un entorno educativo.
- Sexto capítulo (6): se muestran las distintas pruebas realizadas y los resultados de éstas. Además, se comparan los resultados tomando distintos caminos y también con sistemas del estado de arte.
- Séptimo capítulo (7): se aporta la conclusión del trabajo realizado, se citan las posibles mejoras del sistema y los trabajos futuros recomendados.

1.2. Motivación

La información en Internet no cesa en su propósito de extenderse cada segundo que pasa. Este gran desarrollo tiene como gran beneficiario a cada usuario que desea echar un vistazo, ojear, leer, profundizar o investigar sobre un tema determinado. Además, esta información se puede almacenar en un gran número de formatos electrónicos, como por ejemplo, documentos PDF o XML y cada vez más, en libros de papel escaneados convertidos a PDF. Este crecimiento, que a priori, sólo muestra ventajas para los usuarios finales, está provocando que éstos se vean desbordados cada vez que desean encontrar una información concreta. La localización de toda la información realmente relevante, acerca de un tema concreto, sigue

siendo una labor lenta y tediosa mientras lo idóneo sería obtener la información deseada de forma concisa, clara y precisa con sólo realizar la pregunta en cuestión. La aparición de los buscadores (sistemas que intentan localizar los documentos que en un momento dado son de nuestro interés) ha facilitado dicha tarea, sin embargo, los resultados hasta la fecha no son lo suficientemente satisfactorios y sólo facilitan en algunos aspectos la búsqueda de la información.

Por esta razón, desde hace ya varios años, grandes empresas, investigadores y universidades están trabajando para encontrar una solución que solvente el problema. A través de múltiples investigaciones se han desarrollado varias aplicaciones y programas, que constituyen un género en sí mismo, a los cuáles se les ha denominado QAS (*Question Answering Systems*) internacionalmente, también conocidos en castellano como SBR (Sistema de Búsquedas de Respuestas). Estos sistemas tienen como objetivo localizar, extraer y presentar al usuario exclusivamente la información requerida, liberando de la ardua práctica de inspeccionar numerosos documentos irrelevantes. Tal es su importancia a día de hoy que desde hace varios años se llevan celebrando conferencias y concursos internacionales sobre este tema. Entre ellos se encuentra TREC [1], una *workshop* que proporciona la infraestructura necesaria para llevar a cabo pruebas de recuperación de información y que cuenta con la asistencia y participación de investigadores de todo el mundo para presentar sus propuestas más avanzadas.

Por lo tanto, la principal motivación de este PFC es la de buscar, recuperar, extraer y presentar respuestas concretas a preguntas concretas, centrándose en el ámbito de las definiciones: presentaciones de los conceptos a través de sus propiedades. Además este proyecto intenta ser el punto de partida hacia un futuro libro navegable, del cual se pueda extraer y etiquetar todo tipo de información relevante como, por ejemplo: definiciones, respuestas específicas, ejemplos, listas de conceptos, ejercicios, etc.

El hecho de que el trabajo desarrollado pueda ser utilizado en el sistema educativo, tanto para ayudar a los profesores en la elaboración de FAQs dinámicas, exámenes de seguimiento o algún otro tipo de ejercicio, como a los alumnos para ayudarles a extraer información de libros de estudio y/o para hacerse rápidamente resúmenes, es otro refuerzo a la motivación de este proyecto.

1.3. Marco del proyecto

El entorno o marco de este PFC ha sido llevado a cabo dentro del comienzo de una nueva línea de investigación de procesamiento de lenguaje natural en el Grupo de Sistemas Inteligentes (GSI) del Departamento de Ingeniería Telemática (DIT) de la Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT) perteneciente a la Universidad

Politécnica de Madrid (UPM).

Anteriormente a este proyecto, se llevaron a cabo dos PFC concernientes al área de los Bots Conversacionales. Brevemente se describe un *Chatter Bot* o Bot conversacional como un ente artificial que simula mantener una conversación con un ser humano. El primer proyecto [3], realizado por Alejandro Marqués Rodríguez, consistió en el desarrollo de un Bot conversacional que fuera capaz en dar soporte de ayuda a partir de preguntas frecuentes (FAQ) de la página oficial de Orange España. El segundo de ellos [4], realizado por el tutor de este proyecto, Miguel Coronado Barrios, consistió en construir una plataforma para un sistema de Bots, además del desarrollo de una herramienta para generar automáticamente el lenguaje o dialecto que comprenden los Bots, AIML (*Artificial Intelligence Markup Language*).

Dichos proyectos se centran en dotar al sistema de Bots la capacidad de contestar las preguntas de los clientes en los sectores de telefonía móvil y turísticos. Con esto que se acaba de señalar, se observa que para cada temática hay que generar manualmente el código AIML para que el Bot sepa contestar.

Ante estos antecedentes y con el propósito de dotar de mayor inteligencia a los Bots, se ha desarrollado un sistema de búsqueda de respuestas que ayuda a encontrar respuestas de manera automática.

1.4. Objetivos

El objetivo principal de este proyecto ha sido diseñar y desarrollar un sistema de búsqueda de respuestas de tipo definición, en el lenguaje de programación orientado a objetos, Java.

A continuación se citan los diferentes objetivos marcados para este proyecto:

- Estudio del arte de los sistemas de búsqueda de respuestas.
- Elaboración de un tesauro del lenguaje de programación Java de manera formal y aceptada internacionalmente.
- Diseño e implementación de un etiquetador automático de conceptos a partir de un tesauro.
- Diseño e implementación de un sistema de búsqueda de respuestas, en concreto, definiciones, a partir de los conceptos de un tesauro.
- Diseño e implementación de un módulo capaz de entender las preguntas de los usuarios en lenguaje natural.

- Integración del sistema desarrollado dentro de un entorno de educación, generando mensajes de respuesta a las preguntas de los alumnos.

Capítulo 2

Estado del arte

"Los avances más importantes, en el ámbito de los sistemas de búsqueda de respuestas, están todavía por realizar."

— Investigadores del área de búsqueda de respuestas, 2008 [5]

En este capítulo se presenta el estado de arte de los sistemas de búsqueda de respuestas. Asimismo se presentan las diferentes técnicas del procesamiento del lenguaje natural, una visión general de un grupo de sistemas de búsqueda de respuestas con especial atención al sistema que ha servido al autor de esta memoria para adentrarse en este mundo, una breve presentación de la Web semántica y una amplia descripción del modelo escogido para la elaboración de los tesauros.

2.1. Contexto histórico

Desde que las computadoras son usadas para buscar información, se ha intentado evolucionar hacia un sistema ideal que pueda ser capaz de interpretar preguntas y descifrar las respuestas en documentos que están escritos en lenguaje natural [6]. El crecimiento de la necesidad de poder acceder fácilmente a la información ha hecho que la comunidad científica concentre sus esfuerzos en este gran mundo.

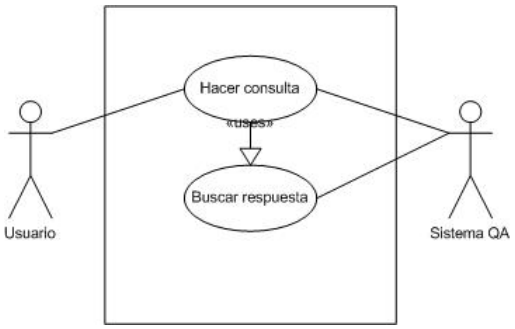

Figura 2.1: Caso de uso de partida

En el año 2008, tuvo lugar un importante debate [5] para discutir en qué punto de madurez se encontraban los sistemas de *Question-Answering*. Se concluyó con que todavía faltaban por llegar los grandes avances en este campo hacia un sistema verdaderamente eficaz. Por lo que a día de hoy, no hay ningún sistema que cumpla los requerimientos antes expuestos, salvo para determinadas preguntas. Por ello, se van a mostrar los intentos desde tres puntos de vista: la recuperación de información, la extracción de información y los sistemas pregunta-respuesta.

2.1.1. Recuperación de información

La recuperación de información, en inglés *Information Retrieval* (IR), es el primer paso dentro de este mundo, búsqueda de respuestas, tan complejo. Se puede decir que este es el punto que más avanzado se encuentra, pudiendo recuperar un basto conjunto de información relacionada con una pregunta concreta, en tan sólo unos milisegundos. Como ejemplos podemos nombrar a potentes motores de búsqueda de información como Google[1] o Bing[2].

En este ámbito se encuentran con dos técnicas fundamentales: la recuperación de fragmentos o *passage retrieval* (PR) y las aplicaciones de procesamiento del lenguaje natural o *natural language processing* (NLP). La primera técnica se basa en medir la relevancia de fragmentos o pasajes con la pregunta realizada. Esto tiene una gran ventaja frente a la búsqueda de documentos completos, puesto que si utilizáramos éstos últimos, el motor puede

[1] http://www.google.com/
[2] http://www.bing.com/

decidir que lo encontrado tiene muy poca relevancia y no mostrarlo, mientras que la búsqueda por fragmentos es más específica evaluando y mostrando los fragmentos más relevantes para que el usuario no tenga que buscar en todo el documento. La segunda técnica, utilización de herramientas NLP, solamente ha conseguido mejoras sustanciales haciendo uso de sus niveles de complejidad más bajos: tokenización y etiquetado de cada *token*. Tanto estas herramientas como los diferentes conceptos que engloban se estudiarán más adelante.

A continuación se enuncian los factores que determinan la eficacia del uso de IR [7]:

- Precisión: determina el porcentaje de acierto de documentos relevantes observando el número total de documentos recuperados.

$$precisión = \frac{(documentos\ relevantes)\ \cap\ (documentos\ recuperados)}{documentos\ recuperados} \tag{2.1}$$

- Sensibilidad o recall: determina el porcentaje de documentos relevantes recuperados a partir de todos los documentos relevantes posibles.

$$sensibilidad = \frac{(documentos\ relevantes)\ \cap\ (documentos\ recuperados)}{documentos\ relevantes} \tag{2.2}$$

- Especificidad: determina la proporción de documentos no relevantes que han sido recuperados.

$$especificidad = \frac{(documentos\ no\ relevantes)\ \cap\ (documentos\ recuperados)}{documentos\ no\ relevantes} \tag{2.3}$$

- Factor F o F-score: determina la efectividad de una prueba. Tiene en cuenta, tanto la precisión de la medida como la sensibilidad. La puntuación es máxima al llegar a 1 y mínima en 0.

$$F = 2 \cdot \frac{precisión \cdot recall}{precisión + recall} \tag{2.4}$$

  La fórmula que se acaba de presentar es la más usada y también se la conoce como F_1-score. Esta fórmula proviene de la fórmula general del F-score.

$$F = (1 + \beta^2) \cdot \frac{precisión \cdot recall}{(\beta^2 \cdot precisión) + recall} \tag{2.5}$$

  Siendo β un valor positivo que determina el número de veces que se considera que el recall es más importante que la precisión.

2.1.2. Extracción de información

Se puede definir la extracción de información o Information Extraction (IE) como la actividad de extraer automáticamente información estructurada o semiestructurada desde

documentos legibles por el ordenador. Una aplicación típica de IE es el escaneado de documentos escritos en lengua natural y rellenar una base de datos con la información extraída. Entre las principales tareas se encuentran con las siguientes [8]:

- Reconocimiento de entidades de nombre o Named Entity Recognition (NER): clasifica elementos del texto en categorías predefinidas como nombres de personas, organizaciones, valores monetarios, etc.
- Resolución de correferencia o Coreference Resolution (CR): se centra en identificar distintos sintagmas nominales que se refieren al mismo objeto.
- Extracción de terminología o Automatic Terminology Extraction (ATE): identifica y extrae candidatos a términos de los textos explorados.
- Extracción de relaciones o Automatic Relation Extraction (ARE): detecta y clasifica relaciones semánticas.

Lo que se pretende conseguir es generalizar cualquier frase, se observa mejor con un ejemplo:

– JUAN trabaja para MOVISTAR $\equiv$ PERSONA trabaja para ORGANIZACIÓN

2.1.3. Pares pregunta-respuesta

Los sistemas IR e IE facilitan el análisis de grandes cantidades de informaciones, pero son incapaces de dar una respuesta a una pregunta dada. Los sistemas IR, por sí solos, únicamente pueden obtener documentos completos a una pregunta dada. Y los sistemas IE son bastantes más precisos que los anteriores, sin embargo presentan serias dificultades en cuanto al tratamiento de cuestiones arbitrarias ya que el tipo de información debe ser definido antes de la implementación del sistema. Por tanto, los dos anteriores sistemas presentan carencias para poder acercarnos a nuestro propósito, por lo que el enfoque de los investigadores tornó hacia unos nuevos sistemas denominados, sistemas *Question-Answering*.

2.2. Recorrido por los sistemas QA

Los sistemas de búsqueda de respuestas o *Question-Answering Systems (QAS)* se definen como los sistemas encargados de obtener automáticamente respuestas concretas a preguntas formuladas por los usuarios. Son verdaderamente útiles a la hora de necesitar obtener un trozo de información muy específico y no se quiere perder tiempo en leer toda la documentación disponible. A continuación se muestran las características principales que deben cumplir estos sistemas [9]:

- Independiente del tiempo: además de contestar en tiempo real, el sistema debe ser capaz de responder a preguntas que se refieran a hechos recientes, del pasado, o incluso del futuro.
- Exactitud: se debe asegurar que la respuesta sea la idónea. Si no se encontrara, habría que dejar sin contestar la pregunta antes que dar una respuesta incorrecta.
- Usabilidad: la respuesta debe de ser comprensible al usuario independientemente de las fuente de información consultadas.
- Completitud: la respuesta dada debe ser completa aunque sean necesarios varios documentos para elaborarla.

Antes de meternos en el funcionamiento de varios sistemas específicos, vamos a dar una visión general de todo QAS. Cada uno de ellos se compone de tres componentes principales: análisis de la pregunta, selección del documento o fragmento y extracción de la respuesta. Estos componentes se muestran en la figura 2.2.

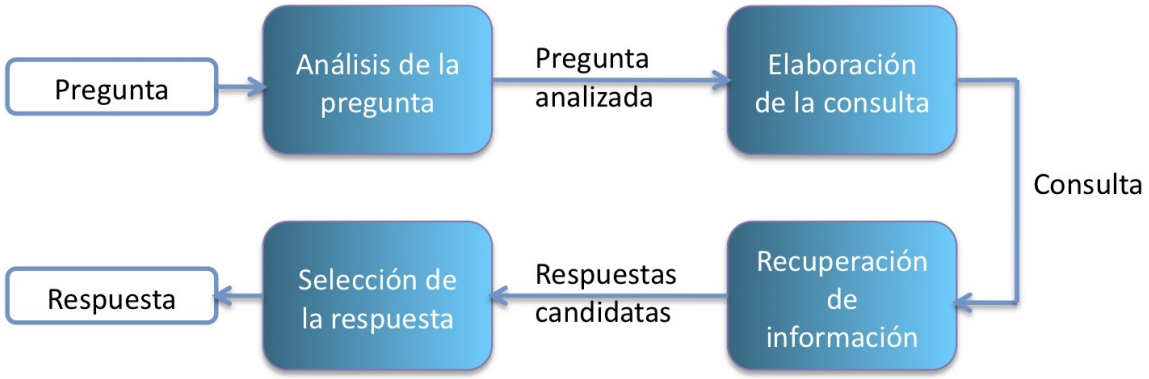

Figura 2.2: Esquema básico de un QAS

2.2.1. Análisis de la pregunta

Este es el primer paso de todo QAS. Las cuestiones formuladas son procesadas para detectar y extraer la información útil que contengan y esto se consigue a través de dos tareas: una primera evaluación para determinar el tipo de información que se espera como respuesta y, por otro lado, una selección de aquellos elementos que permitirán al sistema localizar los documentos que contengan la respuesta esperada.

El gran aporte de este módulo será el de clasificar cada pregunta en categorías, por ejemplo, la pregunta, *¿Quién es el rey de España?* espera la categoría: nombre. Para lograr

extraer esta información, hay diferentes vías de actuación. La más simple, pero también más usada por tener una gran efectividad, es el uso de patrones construidos a través de expresiones regulares. En la figura 2.3 se muestran algunos de ellos.

Name question patterns
(who|what|which) be <TARGET>
(who|what|which) be <TARGET>(in|of|on|to) <CONTEXT>
(who|what|which) be <CONTEXT>'s <TARGET>
(how|what) be <TARGET>(call|know as|name)
(how|what) do you call <TARGET>

Figura 2.3: Ejemplo de patrones de preguntas

Otro método alternativo en la clasificación de preguntas es el uso de técnicas de máquinas de aprendizaje (*machine learning*). Zhang and Lee llevaron a cabo un estudio de los diferentes métodos hasta la fecha [10]. Estos investigadores compararon árboles de decisión, cercanía de vecinos, clasificadores Bayesianos y máquinas de vectores de soporte (Support Vector Machines (SVM)) usando conjuntos de palabras y de n-gramas[3]. Determinaron que la mayor efectividad la encontraron probando con SVM, con un conjunto de pruebas y algoritmos de obtención de efectividad determinados.

2.2.2. Selección del documento

Con la información obtenida del análisis de la pregunta, el siguiente paso consiste en la selección de los posibles candidatos. Dado el gran volumen de documentos con los que se espera que estos sistemas trabajen, los responsables de realizar la selección van a ser los sistemas IR e PR. Esto desahogará al sistema para seguir trabajando con una base de datos mucho menor. Para reducir todavía más los fragmentos o documentos idóneos, se añadirá un componente de filtrado. Este componente, consulta el tipo de respuesta esperada y filtra aquellos que no contengan los elementos extraídos en el análisis de la pregunta en cuestión.

2.2.3. Extracción de la respuesta

La extracción de la respuesta es el último paso de un proceso de QA. Este módulo se encarga de llevar a cabo un análisis detallado de los textos relevantes seleccionados obtenidos

[3]subsecuencia de n elementos

en el anterior módulo para localizar y extraer la respuesta esperada. Para ello se establecerá un ranking que tendrá en cuenta la probabilidad de que la respuesta sea la correcta y se presentará la ganadora. El método más simple para construir una lista de respuestas candidatas es a través de un reconocedor de entidades del nombre, eliminando aquellos frafmentos preseleccionados en las que las entidades de los nombres no sean compatibles con el tipo de respuesta esperada. A continuación se ven los posibles métodos de realizar un ranking adecuado:

- Similitud: se fijan en la similitud del contexto de la respuesta candidata con la pregunta realizada.
- Popularidad: se mira la frecuencia con la que aparecen las respuestas candidatas.
- Relación de patrones: según los patrones de la pregunta, se dará valor a las posibles respuestas que contengan los patrones relacionados a ella.
- Validación de la respuesta: si la puntuación de la candidata no llega a un umbral, se descarta.

Se va a intentar profundizar sobre los métodos anteriormente citados.

2.2.3.1. Similitud con la pregunta

Generalmente, si una frase o fragmento posee un alto grado de semejanza con la pregunta, tiene grandes posibilidades de que la respuesta candidata sea la correcta. Para determinar esta similitud se puede afrontar desde varios puntos de actuación. Una forma simple es contar el número de palabras en común. Algo más elaborado es el uso algoritmos más complejos que sean capaces de medir la similitud teniendo en cuenta su información semántica y/o sintáctica.

2.2.3.2. Popularidad de la respuesta

Esta etapa no posee gran complejidad. El número de veces que una frase es seleccionada como una posible respuesta está directamente relacionada con la probabilidad de que esa frase sea la respuesta esperada. Pero tiene problemas de ambigüedad y redundancia. Dos posibles respuestas pueden dar el resultado esperado pero una de ellas puede contener además información que no ha sido preguntada. Para solventarlo se pueden elaborar algoritmos que clasifiquen las respuestas en función de su concretitud.

2.2.3.3. Relación de patrones

Este método es utilizado en varios sistemas como se explicará más adelante. Con la elaboración de un conjunto de patrones lo suficientemente grande, puede hacer prescindible al resto de etapas. Mientras que en la etapa de análisis de la cuestión existe la posibilidad de crear los patrones manualmente, en la parte de extracción se antoja imposible. Por ello, aquí se habla de un nuevo concepto denominado aprendizaje de patrones o, en inglés, *pattern learning*. La figura 2.4 ofrece una pequeña muestra de estos patrones.

Name answer patterns
(a|an|the)? <TARGET>is (a|an|the)? <PROPERTY>
<PROPERTY>is the <TARGET>
on <PROPERTY>, <TARGET>
the <PROPERTY>is the official <TARGET>
<TARGET><PROPERTY>:

Figura 2.4: Ejemplo de patrones de respuestas

2.2.3.4. Validación de la respuesta

Con el conjunto de posibles respuestas, se debe discriminar aquellas que por lógica tengan cabida dentro las respuestas esperadas, por ejemplo, si se pregunta por algún tipo de distancia, la respuesta no puede devolver como dato un número negativo.

2.2.4. Participantes

Ahora, para tener una visión general del uso de estos sistemas, se van a presentar los participantes de todo QAS: usuarios, preguntas, respuestas y conocimiento.

2.2.4.1. Usuarios

Los principales participantes, como no podía ser de otra manera, son los usuarios. Dentro de los usuarios se encuadran un gran número de perfiles. Se pueden citar desde los que sólo buscan una respuesta concreta como una fecha o un nombre, a los que buscan explicaciones a determinados temas. A continuación se muestran algunos ejemplos de usuarios tipo:

- Usuarios casuales: buscan respuestas concretas a preguntas casuales.

- Recopiladores de información: buscan encontrar información con mayor o menor extensión sobre un tema, por lo que se les debe de proporcionar una respuesta que puede estar basada en varios documentos.
- Periodistas y analistas de la información: este tipo de usuario necesita multitud de datos para elaborar sus artículos, por lo que el sistema seguramente deberá recopilar información de varias fuentes.

2.2.4.2. Preguntas

La experiencia en los estudios de estos sistemas ha demostrado que hay preguntas más difíciles de responder que otras, puesto que cada persona tiene una necesidad y una forma de elaborar sus preguntas. Algunos posibles tipos de preguntas:

- Preguntas de hechos: requieren una respuesta específica, como por ejemplo, fechas, nombres, cantidades, etc.

  ¿Cuál es la capital de España?

- Preguntas de definición: buscan la explicación de un hecho o concepto.

  ¿Qué es Java?

- Preguntas resumen: se espera un resumen o lista sobre un tema concreto.

  ¿Qué factores han llevado a la actual crisis mundial?

- Preguntas contextuales: éstas tienen la peculiaridad de referirse a un contexto del que ya ha realizado alguna pregunta anteriormente.

  ¿Quién o quienes han sido los responsables de ello? Con referencia a la anterior pregunta.

- Preguntas especulativas: son de muy alta complejidad y necesitan de técnicas deductivas.

  ¿Qué pasaría si el presidente del gobierno dimitiese?

2.2.4.3. Respuestas

Las respuestas pueden ser de mayor o menor longitud según la pregunta formulada. Lo verdaderamente importante es la presentación de esa respuesta. Existen dos técnicas principalmente para ese fin:

1. Por extracción: es la más simple de las dos propuestas. Una vez detectada la posible respuesta, se copia el fragmento y se presenta por pantalla.

2. Por generación: esta, aunque más compleja de realizar, es la que mejor percepción tiene por parte del usuario y la que hace creer al usuario que esa manteniendo una conversación. Una vez encontrada la sección de información candidata a ser presentada, se ayudan de herramientas de lenguaje para generar una respuesta estructurada.

2.3. Procesamiento del lenguaje natural

Con todo lo expuesto hasta este punto, se antoja evidente que la utilización de herramientas semánticas es indispensable en varias etapas de un sistema QA. El Procesamiento de Lenguaje Natural (PLN) o Natural Language Processing (NLP) tiene como finalidad ayudar a la interacción entre las máquinas y los humanos. Esta rama forma parte de otro gran campo, como es la inteligencia artificial. La inteligencia de estas herramientas se basa en algoritmos modernos del aprendizaje de máquinas o machine learning. Este último concepto se detalla en el apartado 2.3.1.

En esta sección se presentan algunas de las técnicas que se emplean actualmente sabiendo que cada vez existen más algoritmos NLP.

2.3.1. Aprendizaje de máquinas

Ante la imposibilidad de crear manualmente un número de reglas que abarquen los lenguajes de comunicación en su totalidad, se creó la disciplina del aprendizaje de máquinas, más conocido por su nombre inglés, *machine learning*. *Machine learning*, rama de la inteligencia artificial[4], es una disciplina concerniente al diseño y desarrollo de algoritmos que permiten a los ordenadores evolucionar en su comportamiento basado en datos empíricos, tales como un sensor o una base de datos [11]. No se pretende entrar en detalle de los diferentes algoritmos de *machine learning* pero sí se listarán algunos de ellos[5].

- Aprendizaje por árbol de decisión, o, *decision tree learning*.
- Aprendizaje por reglas de asociación, o, *association rule learning*.
- Redes neuronales artificiales, o, *artificial neural networks*.
- Programación genética, o, *genetic programming*.

[4]El día 10 de Octubre de 2011 comenzarán a impartirse, vía Web y por primera vez en la historia, las asignaturas Inteligencia Artificial e Introducción a Machine Learning por la Universidad de Stanford. Esta universidad goza de gran prestigio en estos campos y como se verá en el análisis del sistema creado por el autor de esta memoria, se han utilizado herramientas de esta universidad.

[5]No se han mejorado ni desarrollado nuevas herramientas NLP a lo largo del desarrollo de la parte práctica de este proyecto, aunque su uso ha sido fundamental.

- Programación lógica inductiva, o, *inductive logic programming*.
- Máquinas de vectores de soporte, o, *support vector machines*.
- *Clustering*.
- Redes Bayesianas, o, Bayesian networks.
- Aprendizaje de refuerzo, o, reinforcement learning.
- Aprendizaje de representación, o, *representation learning*.

Para terminar esta introducción básica a *machine learning*, y por qué es conveniente su uso para el diseño de herramientas NLP, se citan algunas de sus ventajas [12] frente a reglas producidas manualmente:

- Se centran automáticamente en los caso más comunes, mientras las reglas escritas manualmente no lo consiguen fácilmente.
- Pueden hacer uso de algoritmos de inferencia estadística para producir modelos que sean robustos ante una entrada extraña.
- La eficacia de sus algoritmos se mejora aumentando la información de entrada.

2.3.2. Tokenización

En español se puede denominar segmentación y su cometido es recoger una cadena de texto y devolver elementos más pequeños (llamados *tokens*) o palabras para su posterior uso. El resultado de su utilización se muestra en la figura 2.5.

2.3.3. Destokenización

La destokenización o reagrupación de *tokens* consiste en reasignar la posición de cada *token* a su antigua posición, tras haber sido dividido el texto en *tokens* para trabajar independientemente con cada uno de ellos. El resultado de su utilización se muestra en la figura 2.6.

2.3.4. División de frases

Conocido por su nombre en inglés *sentence splitting*, esta técnica es capaz de detectar y separar en frases a partir de un texto. El resultado de su utilización se muestra en la figura 2.7.

Cabe destacar la gran efectividad de este algoritmo frente a un texto sin peculiaridades. Las peculiaridades se muestran en diferentes situaciones:

```
StringTokenizer st = new StringTokenizer(
"this is a test");
    while (st.hasMoreTokens()) {
       System.out.println(st.nextToken());
     }
```

presenta la salida:

this

is

a

test

Figura 2.5: Ejemplo de Tokenización

```
String sentence =
  OpenNLP.retokenizer("My dog said : guau ! guau !");
System.out.println(sentence);
```

presenta la salida:

My dog said: guau! guau!

Figura 2.6: Ejemplo de Destokenización

```
String[] sentences = SentenceSplitter.split("NOTICIA: El CEO
de google.com, D.Eric, ha dimitido. ¿Qué
ocurrirá ahora?");
for(String sentence : sentences){
  System.out.println(sentence);
}
```

presenta la salida:

"NOTICIA:"

"El CEO de google.com, D.Eric, ha dimitido."

"¿Qué ocurrirá ahora'?"

Figura 2.7: Ejemplo de división de frases

- El título y el texto no están separados por ningún signo de puntuación.

  La inteligencia artificial

  La inteligencia artificial moderna se basa en algoritmos de machine learning.

- Sentencias de programación.

```
System.out.println();
```

  De los dos casos expuestos, el segundo sería fácilmente corregible a través de entrenamiento. Sin embargo, el primero caso se antoja complejo.

2.3.5. Etiquetado de palabras

Esta técnica es la más usada y útil de las que se ha visto hasta ahora. Se conoce como *Part-of-Speech Tagging (PoST)* y también como etiquetador gramatical y desambigüador de palabras. El proceso consiste en marcar cada palabra de un texto con una categoría gramatical, para ello se basa en la definición y en el contexto en el que se encuentra cada una. El resultado de su utilización se muestra en la figura 2.8.

```
echo 'El perro es el mejor amigo del
hombre' | cmd/tree-tagger-spanish
```

presenta la salida:

Palabra	Categoría	raíz
El	ART	el
perro	NC	perro
es	VSfin	ser
el	ART	el
mejor	ADJ	bueno
amigo	NC	amigo
del	PDEL	del
hombre	NC	hombre

Figura 2.8: Ejemplo de POST

2.3.6. Chunking

En el análisis de una frase, la técnica más general es el análisis superficial, conocido como *chunking*. Con ella se puede saber a qué predicado pertenece cada palabra y también es útil para dividir la frase completa en predicados, sin tener en cuenta el número de palabras que lo componen. El resultado de su utilización se muestra en la figura 2.9.

```
String[] sentence = {The, dog, is, a, pet};
String[] chunks = OpenNlp(sentence);
for(String chunk : chunks){
  System.out.println(chunk);
}
```

presenta la salida:

Palabra	Chunk
The	B-NP
dog	I-NP
is	B-VP
a	B-NP
pet	I-VP

Figura 2.9: Ejemplo de Chunking

2.3.7. Lematización

La técnica de lematización permite jugar con conjuntos de palabras en su forma más genérica, palabra por palabra. Esto permitirá comparar fragmentos de texto con un número mucho menor de combinaciones. El resultado de su utilización se muestra en la figura 2.10.

```
lemma(forgotten, chances)
```

presenta la salida:

forget chance

Figura 2.10: Ejemplo de lematización

2.3.8. Análisis sintáctico

El análisis sintáctico dentro del mundo NLP, es la combinación de dos herramientas anteriormente citadas, el chunking y el POST. La diferencia o mejora de esta nueva aplicación, es la de agrupar cada sintagma o predicado, el etiquetado PoS de cada *token* y cada *token* en una estructura de árbol. A esta herramienta se la conoce como *parser*. Un analizador de lenguaje natural es un programa que trabaja con la estructura gramatical de las oraciones, por ejemplo, qué grupos de palabras van juntos (como "frases") y qué palabras son el sujeto u objeto de un verbo. Estos programas probabilísticos, como muchos otros, utilizan el conocimiento del etiquetado manual de las frases para tratar de producir el análisis más probable de las nuevas sentencias. El resultado de su utilización se muestra en la figura 2.11.

```
String parse =
  StanfordParser.parse("My dog also likes eating sausage.");
System.out.println(parse);
```

presenta la salida:

(ROOT (S (NP (PRP$ My) (NN dog)) (ADVP (RB also)) (VP (VBZ likes) (S (VP (VBG eating) (NP (NN sausage))))) (. .)))

Figura 2.11: Ejemplo de análisis sintáctico

2.3.9. Reconocimiento de entidades del nombre

Más conocido como *Name Entity Recognition (NER)*, esta técnica consiste en la detección y clasificación de los nombres de un texto en categorías predefinidas, como pueden ser nombres de personas, organizaciones, lugares, cantidades, etc. El resultado de su utilización se muestra en la figura 2.12.

```
NER("Juan trabaja para Movistar")
```

presenta la salida:

Juan(Persona) trabaja para Movistar(Organización)

Figura 2.12: Ejemplo de NER

2.3.10. Aposición

La aposición es una construcción de dos elementos gramaticales unidos, el segundo de los cuales especifica al primero. En el mundo NLP, se encuentran herramientas capaces de relacionar estos dos elementos. El resultado de su utilización se muestra en la figura 2.13.

```
Aposition("El rey de España,
  Juan Carlos I, es un gran hombre.")
```

presenta la salida:

El rey de España, Juan Carlos I(Aposición), es un gran hombre.

Figura 2.13: Ejemplo de aposición

2.3.11. Correferencia

Cuando en una misma construcción sintáctica aparecen dos elementos correferentes, es posible llevar a cabo ciertas operaciones que entrañan la sustitución de uno de ellos. El resultado de su utilización se muestra en la figura 2.14.

```
Correference("Bill is the CEO. He is full of work.")
```

presenta la salida:

Bill is the CEO. **Bill** is full of work.

Figura 2.14: Ejemplo de correferencia

2.4. Sistemas QA específicos

En esta sección se presentan algunos de los sistemas QA que forman el estado del arte de estos sistemas. Se presentan aquellos que han sido más relevantes para el autor. Sin embargo, otras descripciones de diferentes sistemas también han sido de gran ayuda([13], [14] y [15]). A continuación, se van a presentar tres sistemas de *Question-Answering* concretos, complejos y diferentes en la manera de obtener respuestas.

2.4.1. QA Wikipedia

El sistema QA Wikipedia [16] tiene como propósito principal la búsqueda de respuestas, en español, en una fuente tan rica en contenido como es Wikipedia[6]. Este proyecto fue desarrollado por un proyectando de la Universidad Carlos III y su arquitectura se presenta en la figura 2.15.

Las distintas fases de esta arquitectura son:

1. Análisis de la pregunta.

   a) Segmentación: el sistema segmenta la pregunta introducida por el usuario para obtener sus palabras.

   b) Lematización: obtención de los lemas de cada palabra a través de la herramienta semántica Freeling[7].

   c) Expansión por sinónimos: elaboración de una lista, también lematizada, de sus sinónimos de cada palabra ayudándose del recurso OpenThesaurus[8].

[6] http://www.wikipedia.org/
[7] http://nlp.lsi.upc.edu/freeling/
[8] http://openthes-es.berlios.de/

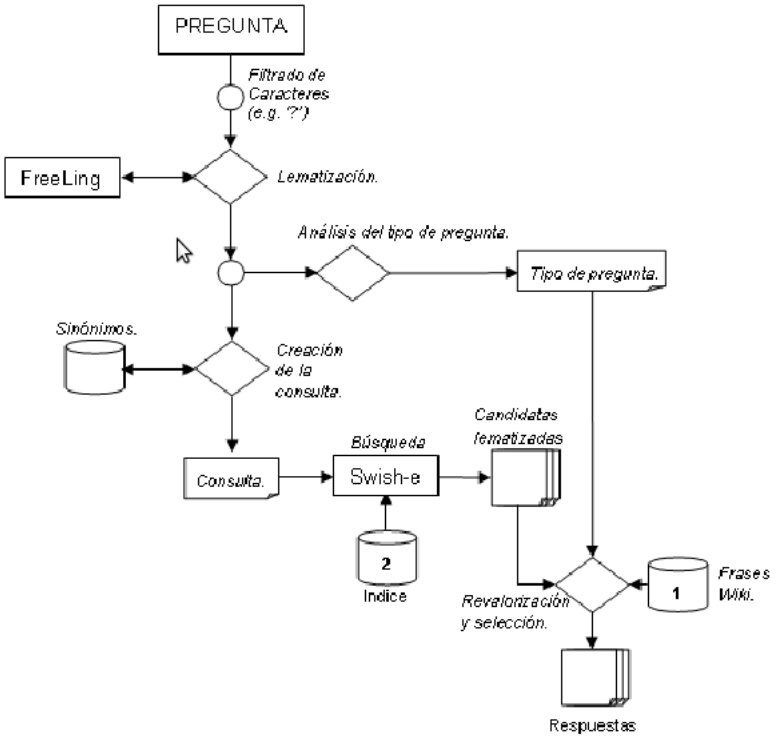

Figura 2.15: Arquitectura del sistema QA Wikipedia

d) Tipo de pregunta: módulo de detección del tipo de pregunta. El sistema es capaz de esperar tres tipos de preguntas: números, nombres propios o cualquier tipo.

e) Construcción de la consulta: consulta formada por todas las palabras de la pregunta o por sólo las más relevantes.

2. Búsqueda de información. Una vez tratada la consulta para tenerla normalizada, se lleva a cabo una búsqueda en los documentos utilizando para esta función el motor de búsqueda de Swish-e.

3. Extracción de la respuesta.

   a) Lectura del fichero indexado: se lee el fichero devuelto por Swish-e y el sistema se queda con las quince mejores respuestas.

   b) Búsqueda de equivalencias: se buscan las frases equivalentes sin lematizar.

   c) Ponderación: se pondera según la longitud de cada frase, dando más valor a las más concretas.

   d) Tipo de respuesta: se analizan gracias a Freeling para determinar las categorías de ellas y las que coincidan con el valor esperado, obtendrán mayor puntuación.

   e) Resultado: el sistema finalmente muestra las cinco mejores respuestas.

De los resultados obtenidos se consiguió alrededor de un 20 % de efectividad. Estos resultados dan una idea de que hasta ahora, la obtención de respuestas a partir de grandes cantidades de información se antoja muy complicado.

2.4.2. Ephyra

Dentro de los sistemas QA que se pueden encontrar de código abierto, OpenEphyra[9]([17], [18] y [19]) es el más consultado y referenciado. Éste es un QAS que estuvo participando varios años en la conferencia anual TREC hasta que su principal desarrollador, Nicholas Schaefler, fue escogido por IBM para ayudar en la implementación de Watson [20], probablemente el mejor sistema QA creado hasta la fecha (aunque requiera máquinas con mucho mayor potencial de procesamiento que las convencionales).

El proyecto está escrito completamente en Java, dependiente del idioma inglés y está preparado para abarcar por completo el contenido de la Web, pero también se puede modificar para trabajar con documentos locales. Este sistema parte de otro anterior, el Javelin QA [21], que también ha participado en los concursos TREC. El esquema que sigue, como el de cualquier sistema QA, se muestra en la figura 2.16.

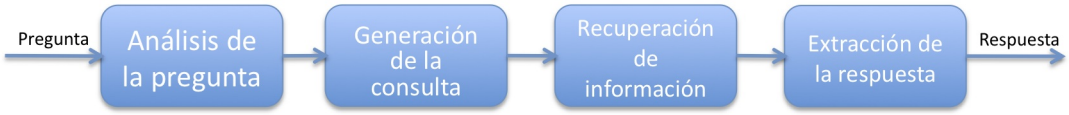

Figura 2.16: Arquitectura comprimida de OpenEphyra

Se debe tener en cuenta que no todos los bloques son independientes en su desempeño. El primer bloque, análisis de la cuestión, es independiente al resto porque no le precede ningún otro bloque. El bloque de generación de la consulta está completamente relacionado con el análisis de la cuestión para determinar la mejor manera de crear una consulta para obtener los mejores resultados del bloque de recuperación de información. Éste bloque vuelve a ser independiente y tiene como entrada la consulta y como salida unos posibles fragmentos de texto relacionados con la consulta. Por último se encuentra el bloque de extracción de la respuesta que tiene como entrada la salida del IR y, además, se ayuda de muchos recursos obtenidos en el análisis de la cuestión.

Antes de entrar con la descripción del sistema, es conveniente hacerse una idea de todos los recursos que utiliza el sistema a lo largo del proceso. La "nube" de herramientas se plasma en la figura 2.17.

Para el recorrido por cada módulo, se utiliza *"Who is the king of Spain?"* como pregunta ejemplo. En la figura 2.18 se presenta la arquitectura de OpenEphyra.

[9]OpenEphyra ha sido el punto de partida para adentrarnos en este mundo.

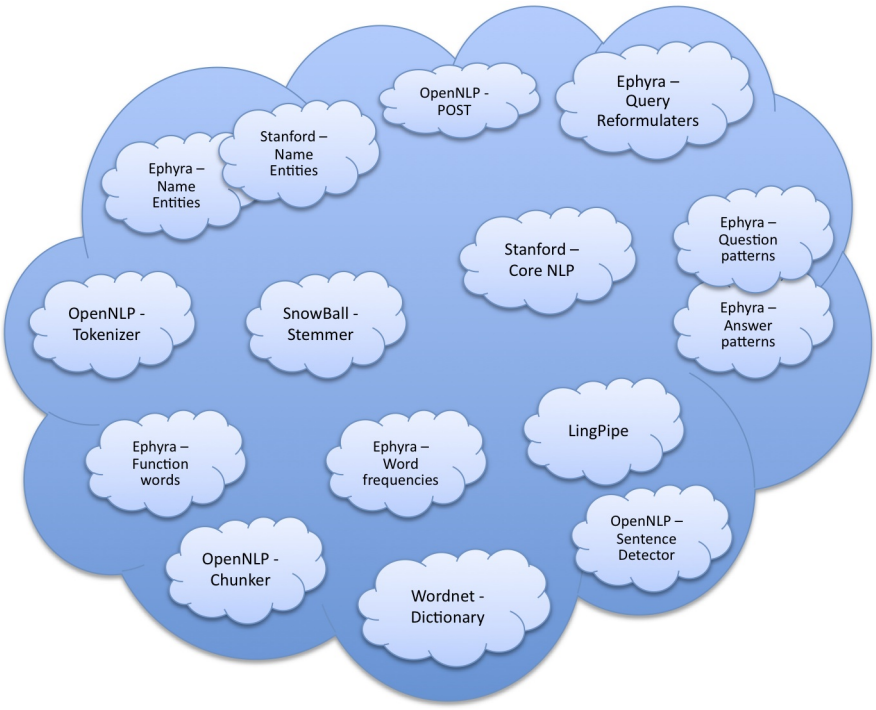

Figura 2.17: Nube de recursos de OpenEphyra

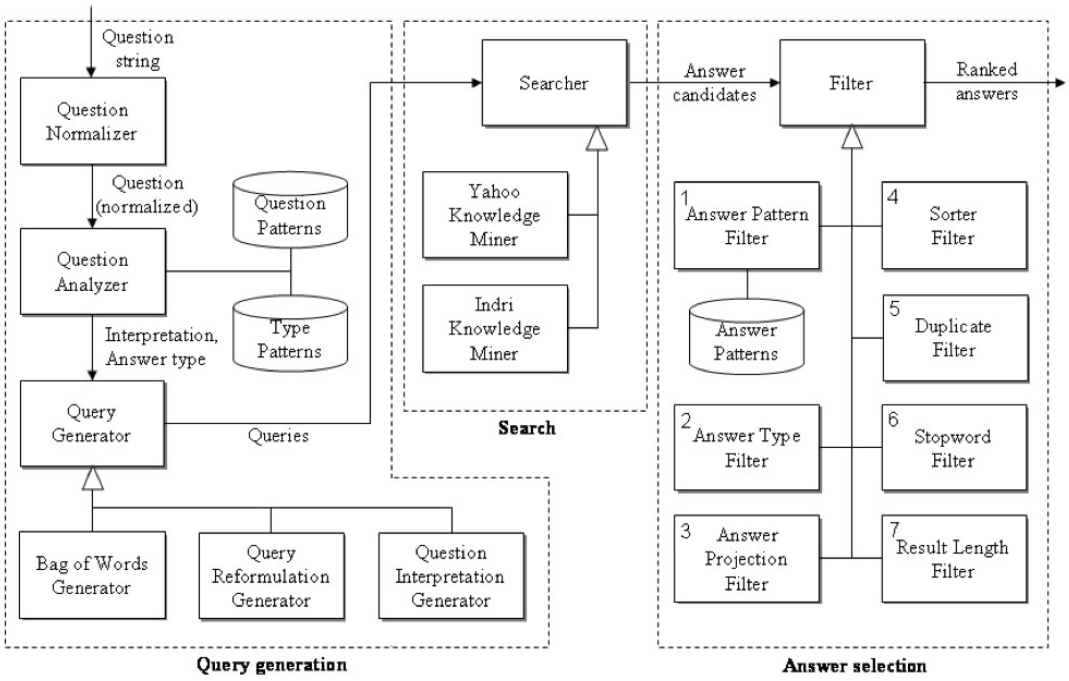

Figura 2.18: Arquitectura de OpenEphyra QA

2.4.2.1. Análisis de la pregunta

Este es el primer módulo de cualquier QA y además el más importante ya que con un análisis exhaustivo de la pregunta se puede aproximar cuál debe ser el tipo de respuesta. En el sistema OpenEphyra se siguen bastantes pasos y a continuación se van detallar uno a uno, observando qué herramientas utiliza para cada función.

1. Normalizador

   Al comienzo del sistema, una sencilla pero fundamental etapa para el desarrollo del sistema, se encuentra la normalización o estandarización de la pregunta. Principalmente, OpenEphyra sigue cuatro pasos para ello (Figura 2.19).

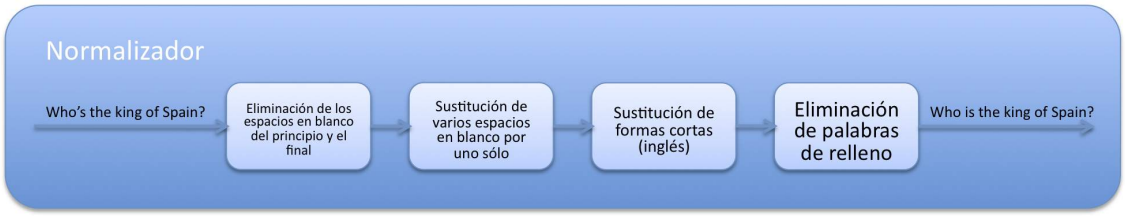

Figura 2.19: Normalización de OpenEphyra

2. Lematización

   A partir de la cuestión normalizada, el sistema transforma los nombres y verbos en singular e infinitivo, respectivamente, siguiendo el esquema de la figura 2.20.

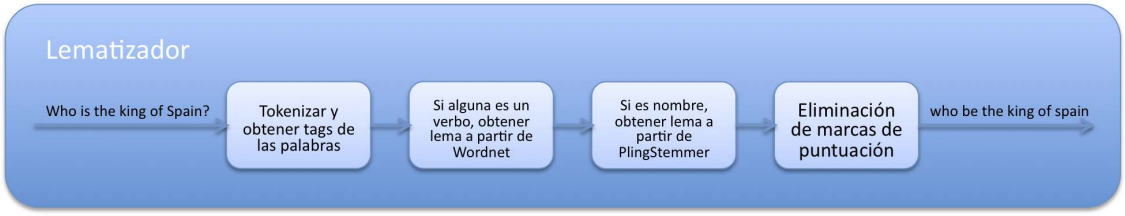

Figura 2.20: Lematización de OpenEphyra

3. Manejo de auxiliares

   Ahora vuelve a trabajar con la pregunta normalizada y se refiere a manejar los verbos auxiliares con la sustitución de los auxiliares más verbo por el verbo. No se da más detalle porque esta parte es muy dependiente del idioma, aún así, se presenta el esquema de esta parte porque se trabajará en sucesivas etapas con la salida de éste (Figura 2.21).

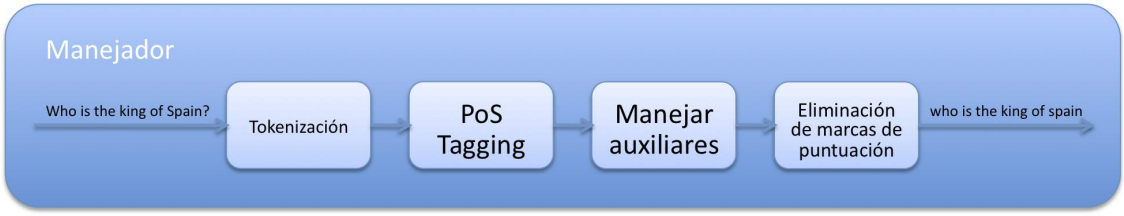

Figura 2.21: Manejo de auxiliares en OpenEphyra

4. Palabras clave

   Para saber dónde centrar la atención para obtener el posible tipo de respuesta esperado y trabajando con la salida anterior, el sistema realiza una extracción de palabras clave como muestra la figura 2.22.

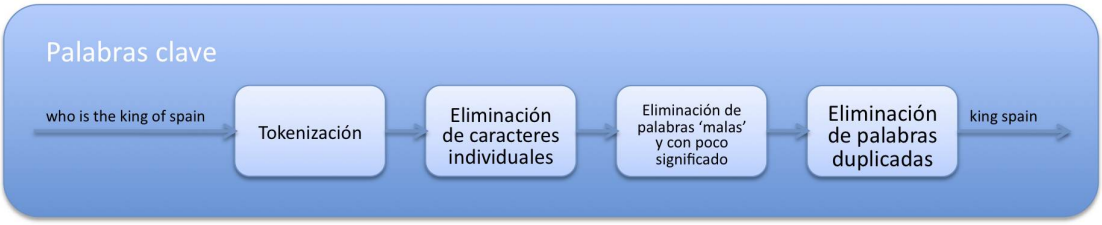

Figura 2.22: Palabras clave en OpenEphyra

5. Detectar entidades del nombre

   Una parte muy importante dentro de una pregunta son los nombres, donde es muy probable que se centre la pregunta. Teniendo en cuenta este punto, el sistema detecta los nombres utilizando las entidades de OpenNLP y del propio Ephyra. Vuelve a trabajar con la pregunta en el estado inicial como se observa en la figura 2.23.

Figura 2.23: Detección de entidades del nombre en OpenEphyra

6. Extractor de términos y marcador de frecuencias relativas

   En este módulo se obtienen unos objetos llamamos términos, que estarán compuestos por las palabras clave, su etiquetado PoS y su frecuencia relativa. Para crear este objeto

se necesita la salida del manejador de verbos, los NEs y el diccionario. El esquema se observa en la figura 2.24.

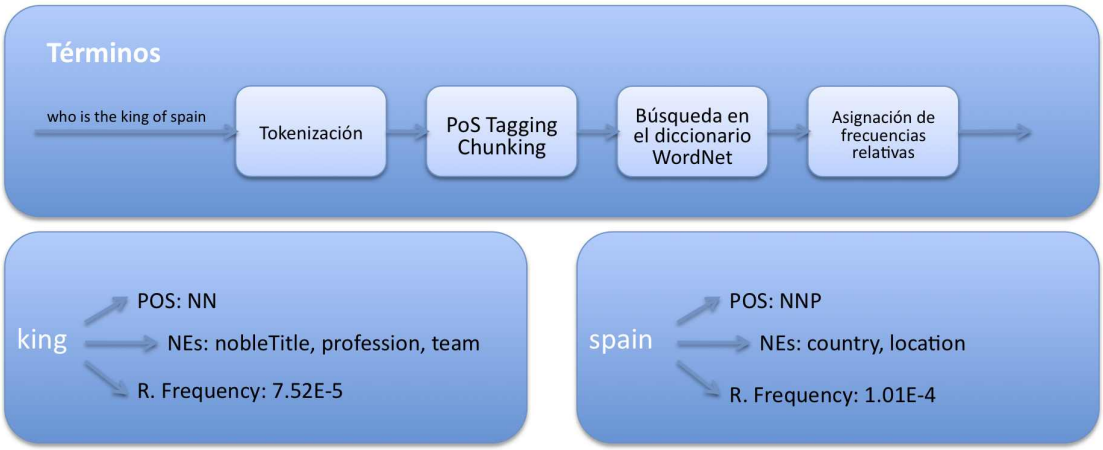

Figura 2.24: Extracción de términos en OpenEphyra

7. Foco de la pregunta

   Se analiza la pregunta a través de analizadores sintácticos para obtener el token donde recae el mayor énfasis.

Figura 2.25: Extracción del foco en OpenEphyra

   - Analizador de Stanford: se obtiene el análisis sintáctico de la manera definida por Stanford:

     (ROOT (SBARQ (WHNP (WP Who)) (SQ (VBZ is) (NP (NP (DT the) (NN king)) (PP (IN of) (NP (NNP Spain))))) (. ?)))

   - A través de lo anterior, con la ayuda de herramientas de lenguaje natural y los algoritmos apropiados se puede obtener el nodo.

8. Tipo de respuesta

   A través de una potente herramienta, MinorThird[10], y con la ayuda de los resultados anteriores (foco de la pregunta, etc.), el sistema es capaz de determinar el tipo de

[10]Colección Java para guardar, anotar, categorizar, y extraer identidades de un texto.

entidad del nombre esperado como respuesta. El camino seguido se muestra en la figura 2.26.

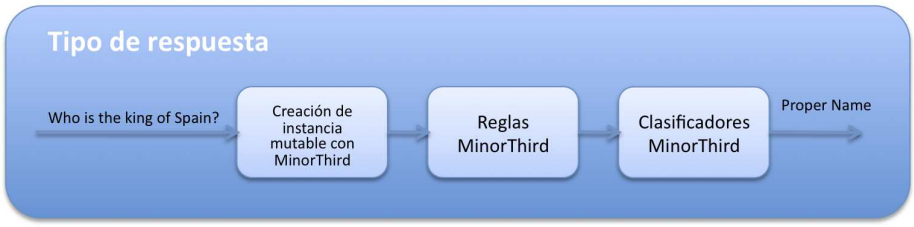

Figura 2.26: Obtención del tipo de respuesta esperado en OpenEphyra

9. Interpretación de la pregunta

   El sistema dispone de unos patrones definidos manualmente, a través de las preguntas de la TREC Workshop de diferentes años, para interpretar la pregunta. Estos patrones tendrán etiquetados el término y el posible contexto de la pregunta. La propiedad vendrá determinada por el nombre del archivo donde se encuentran los patrones correspondientes a ella. Un esquema resumen se muestra en la figura 2.27.

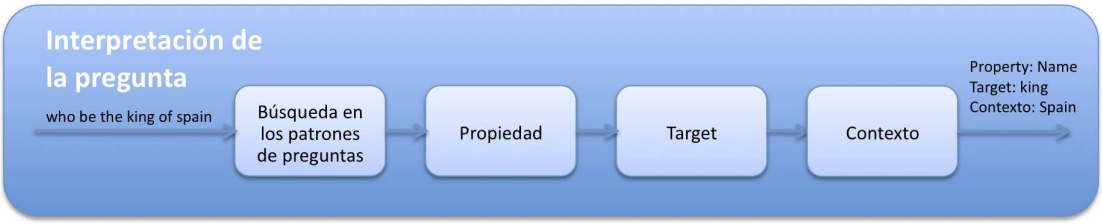

Figura 2.27: Interpretación de la pregunta en OpenEphyra

10. Expansión de términos

    A través del diccionario WordNet y una herramienta para su uso, jwnl[11], se obtienen los posibles sinónimos, hiperónimos, hipónimos, miembro de, sustancia de, parte de, tiene miembros, tiene sustancia y tiene parte, para los términos extraídos anteriormente. Se le dan pesos a los nuevos términos, pero menos que a los originales.

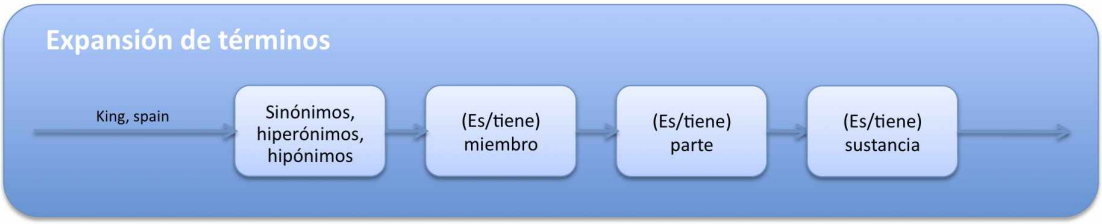

Figura 2.28: Esquema seguido por el expansor de términos en OpenEphyra

[11] http://sourceforge.net/projects/jwordnet/

Figura 2.29: Ejemplo de expansión de términos en OpenEphyra

2.4.2.2. Generación de la consulta

1. Bag of Words

   Este primer generador de consultas construye la consulta en base a las palabras clave de la pregunta. Tiene la siguiente restricción: si no se ha podido obtener el tipo de respuesta esperado, no se genera esta consulta.

Figura 2.30: Ejemplo de consulta del Bag of Words en OpenEphyra

2. Bag of Terms

   El generador 'Bag of Terms' tiene la misma restricción que el anterior pero recibe una mayor puntuación por lo que se muestra a continuación. Recupera los términos y sus expansiones, y se va a quedar con un máximo de 10 expansiones por término con la restricción de que tengan una alta puntuación. Estas expansiones suelen ser los sinónimos.

Figura 2.31: Ejemplo de consulta del Bag of Terms en OpenEphyra

3. Intérprete de la pregunta

   Recuperando información de un paso anterior, este generador de consultas tiene en cuenta los datos obtenidos en la sección de interpretación de la pregunta. Éste vuelve a tener mayor puntuación que los anteriores. Para que se lleve a cabo, se ha tenido que poder interpretar la pregunta y la consulta la formaran el target, el contexto y las palabras clave.

Figura 2.32: Ejemplo de consulta del intérprete de preguntas en OpenEphyra

4. Reformulador de la pregunta

   A través de unos patrones definidos, se puede reformular la pregunta en como se podría encontrar la respuesta en los documentos. Este generador tiene la mayor puntuación porque es muy probable que los fragmentos de texto recuperados con esta consulta, contengan la respuesta esperada. En el caso de ejemplo, las consultas generadas serían:

Figura 2.33: Ejemplo de consulta del reformulador de preguntas en OpenEphyra

2.4.2.3. Recuperación de información

OpenEphyra se ayuda de las siguientes tres utilidades para recuperar información:

- Anotadores del conocimiento: se utilizan para obtener información a partir de fuentes de conocimiento estructuradas o semi-estructuradas.
- Motores de búsqueda vía Web: también conocido como buscador es un sistema informático que busca archivos almacenados en servidores web gracias a su «spider» (o Web crawler).
- Motores de búsqueda locales: recupera información (documentos, párrafos o frases) de igual manera que los anteriores mencionados pero de manera local.

2.4.2.4. Extracción de la respuesta

La extracción de la respuesta, como se puede intuir, es el último paso dentro de este gran sistema. Trabaja con los fragmentos de texto recuperados por los buscadores de información y aplica varios filtros que se ayudan de los resultados conseguidos anteriormente. Con las puntuaciones que se han ido dando a lo largo del sistema (más puntuación a determinadas consultas, por ejemplo), y añadiendo más o menos puntuación según cumpla la restricción de cada filtro, se presentará por pantalla la respuesta con mayor puntuación (aunque se puede modificar fácilmente). Por el gran número de filtros que posee OpenEphyra (no todos activos), sólo vamos a presentar alguno de ellos.

1. Filtro de tipo de respuesta

   A partir del tipo de respuesta esperada, es decir, de la entidad del nombre esperada, se busca en cada fragmento de texto devuelto por el recuperador de información, esas entidades del nombre.

2. Filtro de patrón de respuesta

   Se busca qué respuestas han sido obtenidas por la consulta de la pregunta interpretada y se recupera interpretación. También se recuperan el término, la propiedad y las identidades del nombre de la frase. Por último se transforma la frase como patrón y si coincide con alguno de los patrones guardados, se aumenta la puntuación de esa frase.

Finalmente, después de todo este proceso, el sistema OpenEphyra es capaz de dar la respuesta correcta.

2.4.2.5. Resultados y conclusión

Los resultados de este sistema QA teniendo en cuenta 500 preguntas, en el concurso TREC número 11, son los mostrados en la tabla 2.1.

Técnica	Preguntas contestadas	Respuestas correctas	Precisión	Recall
Análisis por tipo de respuesta	361	173	0.479	0.387
Pattern learning	293	104	0.355	0.233

Tabla 2.1: Resultados de OpenEphyra en el TREC 11

2.4.3. Agente como tutor inteligente

En el moderno mundo de la inteligencia artificial, se encuentran los agentes. Éstos son conocidos como *entes* que actúan para lograr el mejor resultado o, cuando es incierto, el mejor resultado esperado [22]. En esta sección se presenta ITA (Intelligent Tutoring Agent) [23], concretamente la parte de QA. Su meta es que el agente sea capaz de dar respuestas a preguntas sobre un temario específico a los alumnos a través de una interfaz.

1. Extracción de información: rastrea los pares pregunta-respuesta que se encuentren en un libro de texto y los guarda.

2. Análisis de la cuestión: a través de un método conocido como *Vector Space Model (VSM)* y de una herramienta NLP que extrae las raíces de las palabras, compara la pregunta del usuario con las guardadas, y devuelve el par pregunta-respuesta con mayor similitud.

3. Incremento de la información: a través del indexador de texto Lucene[12], recomienda el capítulo que pueda ser de mayor ayuda para el alumno.

[12] http://lucene.apache.org/java/docs/index.html

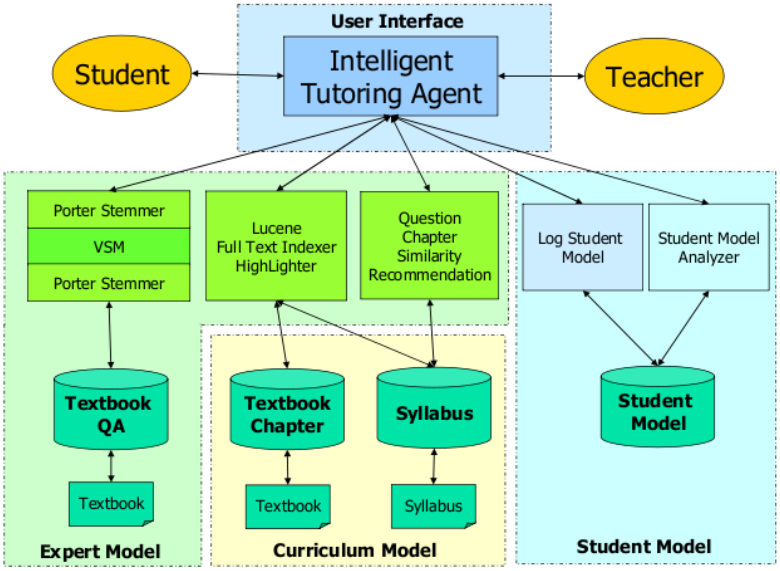

Figura 2.34: Arquitectura de ITA

4. Interfaz: utiliza MSN Messenger para comunicarse con el alumno, da la impresión de Bot conversacional sin llegar a serlo.

Puede concluirse con que la parte de QA es bastante vaga puesto que depende completamente de que el libro de texto contenga pares pregunta-respuesta en la sección de ejercicios.

2.4.4. Patrones fijos para definiciones

En el ámbito de la extracción y presentación de definiciones, existen muchas vías de actuación (una de ellas se ha presentado en la sección 2.4.3 donde partían de pares pregunta-respuesta) y una de ellas se basa en la creación de patrones fijos. Se denominan patrones fijos a aquellas reglas elaboradas manualmente, sin un aprendizaje anterior. En el laboratorio de inteligencia artificial del MIT crearon diversos patrones dividiéndolos en once campos [24], [25]. En la tabla 2.2 se detallan cada uno de ellos con sus nombres y patrones.

La evaluación de estos once patrones se llevo a cabo sobre el corpus AQUAINT[13], el cual es una agrupación de más de un millón de documentos extraídos de diferentes periódicos. Los resultados tienen en cuenta el número de *nuggets*[14] recuperados de cada concepto. En la tabla 2.3 se presentan los resultados de precisión de cada patrón.

En cuanto a los resultados del factor F, en este caso, se ha decidido que es cinco veces más importante el *recall* que la precisión. Con esta consideración, obtuvieron un factor F que rondaba el 0.3.

[13]Corpus oficial del concurso TREC.

[14]En este caso, un *nugget* es una propiedad relacionada con el concepto a definir. Por ejemplo: metal es un *nugget* de oro.

Nombre	**Patrón**	**Casos**
Copular	NP_1 be NP_2	$t \in NP_1$, $n \in NP_2$
Become	NP_1 become NP_2	$t \in NP_1$, $n \in NP_2$
Verb	NP_1 v NP_2	$v \in biography - verb\ t \in NP_1$, $n \in NP_2$
Appositive	NP_1, NP_2	$t \vee n \in NP_1$, $n \vee t \in NP_2$
Occupation	NP_1 NP_2	$n \in NP_1(occupation)$, $t \in NP_2$
Parenthesis	NP_1 (NP_2)	$t \in NP_1$, $n \in NP_2$
Also-known-as	NP_1, (also) known as NP_2	$t \vee n \in NP_1$, $n \vee t \in NP_2$
Also-called	NP_1, (also) called NP_2	$n \in NP_1$, $t \in NP_2$
Or	NP_1 , or NP_2	$t \in NP_1$, $n \in NP_2$
Like	NP_1 (such as|like) NP_2	$n \in NP_1$, $t \in NP_2$
Relative clause	NP (which|that) VP	$t \in NP$, $n \in VP$

Tabla 2.2: Patrones fijos del MIT (NP = predicado nominal, VP = predicado verbal, t = concepto, n = nugget)

Patrón	**Precisión (%)**
Also-called	85.71
Also-known-as	80.00
Occupation	69.35
Or	67.74
Relative clause	66.67
Like	64.60
Appositive	60.00
Copular	35.37
Parenthesis	34.91
Verb	26.09
Became	25.00

Tabla 2.3: Precisión de los patrones del MIT

2.5. Web semántica

Es de obligación presentar la Web semántica por estar ínfimamente relacionada con la gestión de información textual. Se conoce al término "Web semántica" como una una Web extendida, en el sentido de estar dotada de mayor significado en la que el usuario podrá encontrar respuestas a sus preguntas de manera más rápida y sencilla. Como avanzamos al comienzo de este capítulo, tenemos acceso a grandes cantidades de recursos y en numerosos idiomas, sin embargo, esto ha originado que nos encontremos con una sobrecarga de información y heterogeneidad de fuentes de información que provocan dificultades en la interoperabilidad. La Web semántica se ayuda de los metadatos, datos que describen otros datos, para conseguir dotar a la Web la habilidad de razonar.

2.5.1. Estándares de la Web Semántica

En esta sección se realiza un resumen de los estándares más usados que ayudan a la Web a convertirse en una infraestructura global en la que es posible compartir, y reutilizar datos y documentos entre diferentes tipos de usuarios:

- RDF (Resource Description Framework): proporciona un modelo de datos para la descripción de recursos y relaciones entre recursos, es decir, datos sobre recursos (metadatos). Facilita la codificación, el intercambio y el rehuso de datos. Además resta complejidad en la interoperabilidad entre aplicaciones y agentes mediante convenios sintácticos y semánticos. RDF trabaja convirtiendo las declaraciones de los recursos o datos en una sintaxis de la forma sujeto-predicado-objeto de tal manera que el sujeto tiene una propiedad predicado cuyo valor es objeto.

- SPARQL: es el lenguaje de consulta sobre RDF, permite buscar en los recursos de la Web Semántica utilizando distintas bases de datos.

- OWL: proporciona un lenguaje para definir ontologías (esquemas conceptuales) estructuradas que pueden ser utilizadas a través de diferentes sistemas.

- SKOS: proporciona un modelo para representar la estructura básica y el contenido de esquemas conceptuales como listas encabezamientos de materia, taxonomías, esquemas de clasificación, teshauros y cualquier tipo de vocabulario controlado.

2.5.2. SKOS: Modelo relacional de conceptos

En la realización de este proyecto ha sido esencial disponer de una estructura de información que relacione conceptos de un tema en concreto. Por ello, el sistema final se ayuda

de un árbol de conceptos o tesauro representado por el modelo SKOS. En la figura 2.35 se presenta un esquema-resumen que será desgranado a lo largo de la sección.

2.5.2.1. Introducción

SKOS (Simple Knowledge Organization System) es una iniciativa del W3C como aplicación RDF que proporciona un modelo de representación de la estructura básica y el contenido de esquemas conceptuales como listas encabezamientos de materia, taxonomías, esquemas de clasificación, tesauros y cualquier tipo de vocabulario controlado.

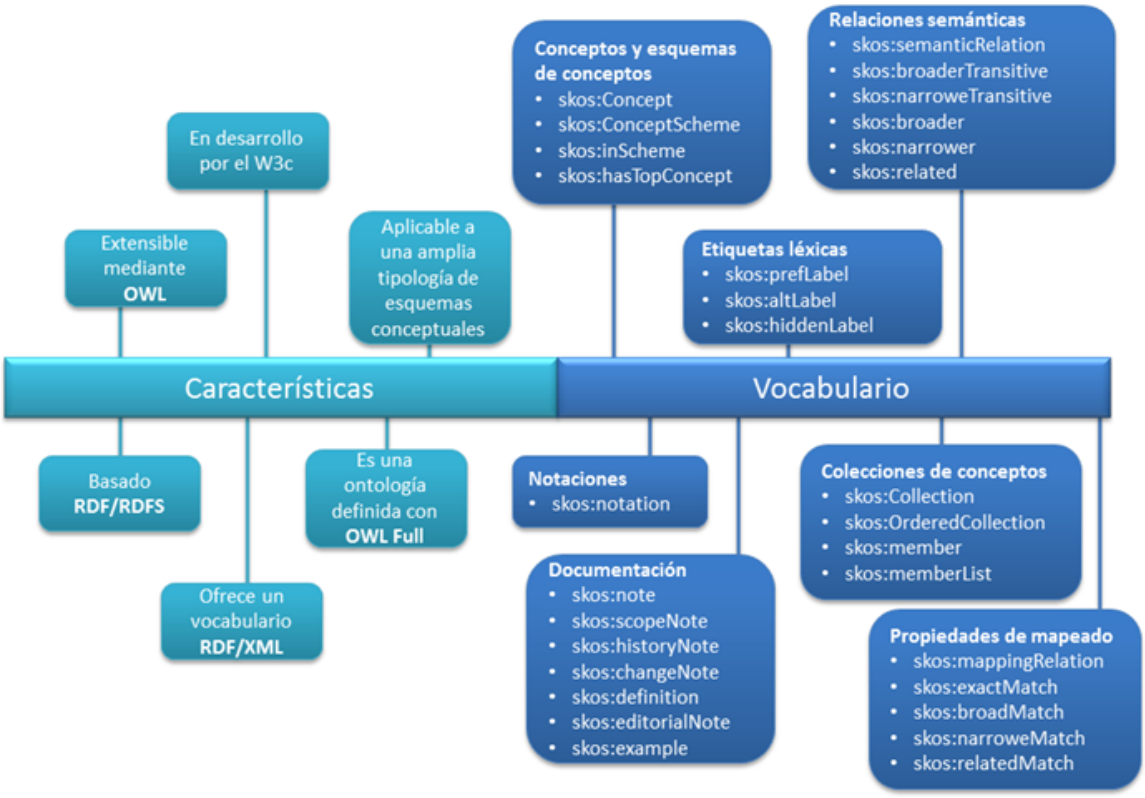

Figura 2.35: Resumen de SKOS

En el modelo SKOS, todos los conceptos están identificados a través de referencias URI[15] Las etiquetas de cada concepto pueden estar escritas en uno o varios idiomas y cada uno de los conceptos puede documentarse y estructurarse gracias a diversas relaciones semánticas de las que dispone SKOS. La principal cualidad de este modelo es proporcionar la capacidad de recorrer diferentes conceptos de diversos esquemas y construir colecciones ordenadas o árboles de conceptos. Además permite establecer relaciones entre las etiquetas asociadas a los conceptos.

[15]Uniform Resource Identifier es una cadena de caracteres corta que identifica inequívocamente un recurso.

La creación de este modelo tiene como fin ser un paso intermedio entre el bajo nivel de estructuración de la mayor parte del contenido de la Web y la rigurosa descripción de las ontologías definidas con OWL.

2.5.2.2. Modelo

El modelo de SKOS organiza los conceptos en estructuras que carecen de semántica forma y que no pueden considerarse hechos. Este modelo es simplemente una manera de representación que proporciona un mapa intuitivo de las relaciones entre cada de uno de los conceptos dentro de un dominio específico [26].

Las relaciones entre los conceptos pueden ser jerárquicas o asociativas, pudiéndose ampliar la tipología de relaciones. Como se revisará más adelante, los conceptos también pueden agruparse en colecciones que, al igual que los conceptos, pueden etiquetarse y ordenarse. SKOS proporciona la posibilidad de que conceptos de diferentes esquemas puedan relacionarse entre sí empleando relaciones jerárquicas, asociativas o de equivalencia exacta.

2.5.2.3. Conceptos y esquemas de conceptos

El primer paso para llevar a cabo la construcción de un esquema de conceptos es definir qué conceptos forman ese esquema y la posición en la jerarquía que ocupan. En la tabla 2.4 se presenta cómo se hace en SKOS.

Etiqueta	Función
skos:Concept	Concepto
skos:ConceptScheme	Esquema de conceptos
skos:inScheme	Pertenencia a un esquema de conceptos
skos:hasTopConcept	Concepto principal

Tabla 2.4: Etiquetas SKOS para conceptos y esquemas

2.5.2.4. Sinónimos

Un mismo concepto puede llamarse de distintas maneras y para ello SKOS dispone de tres etiquetas. Estas etiquetas pueden asociarse, en cualquier idioma y tantas veces como se quiera, excepto la etiqueta preferente, que sólo puede ser usada una vez por idioma. En la tabla 2.5 se recogen estas tres etiquetas, denominadas léxicas.

Etiqueta	Función
skos:prefLabel	Nombre preferente (único por idioma)
skos:altLabel	Nombres alternativos. Ej: sinónimos, acrónimos
skos:hiddenLabel	Nombres ocultos. Ej: posibles errores tipográficos

Tabla 2.5: Etiquetas léxicas de SKOS

2.5.2.5. Relaciones semánticas

Siempre que se defina un esquema de conceptos, cada concepto de este esquema tiene una relación con, al menos, otro concepto. Se pueden distinguir dos tipos de relaciones: jerárquicas y asociativas. Una relación de jerarquía indica que hay conceptos más amplios o generales que otros. Por el otro lado, una relación asociativa indica que hay conceptos que están en el mismo nivel que otros. Estas relaciones son de gran ayuda a la hora de aprender sobre un esquema concreto, pudiendo recorrer el árbol de conceptos a la vez que se aprende qué relación tienen con el resto.

Dentro de las relaciones jerárquicas se encuentran las transitivas y las no transitivas. Las relaciones transitivas suceden cuando un concepto padre tiene, a su vez, un concepto padre y éste último es concepto abuelo del primero. En la tabla 2.6 se presentan todas las posibilidades de relación que proporciona SKOS.

Etiqueta	Función
skos:semanticRelation	Relación semántica de la que derivan el resto
skos:broaderTransitive	Relación genérica transitiva
skos:narrowerTransitive	Relación específica transitiva
skos:broader	Relación genérica no transitiva
skos:narrower	Relación específica no transitiva
skos:related	Relación asociativa

Tabla 2.6: Etiquetas relacionales de SKOS

2.5.2.6. Notación

Para asociar un concepto al entorno específico de un esquema conceptual, se define la notación. Una notación tiene como valor un literal tipificado (identificador). Solamente se dispone de una etiqueta para este fin, como muestra la tabla 2.7.

Etiqueta	Función
skos:notation	Identificador del concepto dentro de un entorno específico

Tabla 2.7: Etiqueta de notación de SKOS

2.5.2.7. Ampliación de información

En SKOS, también existen elementos que ayudan a ampliar la información sobre los conceptos y sus relaciones con el resto. Las etiquetas destinadas a ese fin se muestran en la tabla 2.8.

Etiqueta	Función
skos:note	Elemento del que derivan el resto
skos:scopeNote	Significado en un determinado ámbito
skos:historyNote	Cambios del significado a lo largo del tiempo
skos:changeNote	Cambios de la posición dentro del esquema
skos:definition	Definición o descripción
skos:editorialNote	Información sobre su edición y publicación
skos:example	Ejemplo de uso

Tabla 2.8: Etiquetas de documentación de SKOS

2.5.2.8. Colecciones de conceptos

SKOS permite definir agrupaciones sin que se establezcan relaciones semánticas explícitas que distorsionen las estructuras jerárquicas o asociativas del esquema conceptual. En la tabla 2.9 se muestran estos elementos.

Etiqueta	Función
skos:Collection	Colección de conceptos
skos:OrderedCollection	Colección ordenada de conceptos
skos:member	Miembro de una colección
skos:memberList	Lista de miembros de una colección

Tabla 2.9: Etiquetas de colecciones de conceptos de SKOS

2.5.2.9. Relaciones de correspondencia

Al leer el título de esta sección, el lector puede llegar a pensar que estas relaciones tienen la misma función que las relaciones semánticas, pero esto no es así. Estas relaciones

indican la correspondencia existente entre un par de conceptos pertenecientes a esquemas conceptuales diferentes.

En la tabla 2.10 se detallan este tipo de relaciones.

Etiqueta	**Función**
skos:mappingRelation	Relación de mapeado
skos:closeMatch	Relación de correspondencia cercana
skos:exactMatch	Relación de correspondencia exacta (mismo concepto)
skos:broadMatch	Relación de correspondencia genérica
skos:narrowerMatch	Relación de correspondencia específica
skos:relatedMatch	Relación de correspondencia asociativa

Tabla 2.10: Etiquetas de correspondencia de SKOS

Capítulo 3

Estudio de herramientas

"No había nada que no pudiera construir, en especial, si tenía herramientas."
— Daniel Defoe

En este capítulo se estudian brevemente las herramientas de extracción de texto plano y un conjunto de herramientas de procesamiento del lenguaje natural (NLP). Las herramientas NLP estudiadas están consideradas dentro del conjunto de las más avanzadas hasta la fecha. También han sido elegidas para su estudio porque, o son utilizadas a lo largo del sistema de este proyecto, o bien porque tienen funcionalidades adaptadas al español (aunque ahora mismo son irrelevantes para el sistema de esta memoria, se cree que tendrá gran peso en los trabajos futuros de éste). Además de las presentadas en este capítulo, existen más herramientas de código abierto y comerciales.

3.1. Herramientas de extracción de texto

Un requisito esencial del procesamiento de lenguaje natural es el uso de texto plano o sin estructurar. En la estructura de este proyecto se comienza con una extracción de texto desestructurado a partir de documentos PDF o HTML. Por supuesto, también queda la opción de introducir texto plano directamente.

3.1.1. Documentos PDF

Para la extracción de texto plano de documentos PDF, el sistema se ayuda de una librería Java llamada PDFBox[1]. Apache PDFBox™ es una librería de código abierto escrita en Java para trabajar con documentos PDF. Además, Apache PDFBox está publicada bajo la licencia Apache v2.0. Las principales características de esta librería son las siguientes:

- Extracción de texto plano.
- Unión de documentos.
- Encriptación y desencriptación.
- Relleno de datos en FDF y XFDF.
- Creación de documentos PDF a partir de ficheros de texto.
- Creación de imágenes a partir de páginas PDF.
- Imprimir un documento PDF.

La principal ventaja de esta herramienta es que la extracción de texto plano la consigue sin errores. En cambio, tiene un gran inconveniente, o tal vez no se ha dedicado el tiempo suficiente a la herramienta, para la identificación de los títulos de cada apartado de un libro. Este hecho provoca errores en posteriores etapas del sistema ya que, como se detallará más adelante, el separador de oraciones no consigue realizar correctamente su función.

3.1.2. Documentos HTML

En cuanto a la extracción de texto plano a partir de documentos HTML, el sistema se ayuda de una librería Java llamada HTMLParser[2]. HTMLParser es una librería de código abierto escrita en Java para trabajar con documentos HTML. Además, HTMLParser está publicada bajo la licencia Common Public License 1.0, Librería GNU o Lesser General Public License (LGPL). Las principales características de esta librería son las siguientes:

[1] http://pdfbox.apache.org/
[2] http://htmlparser.sourceforge.net/

- Características de extracción.
  - Extracción de texto plano.
  - Extracción de links.
  - Screen scraping.
  - Extracción de recursos como imágenes y archivos de sonido.
  - Comprobación de links.
  - Monitorización de sitios Web para la comprobación de cambios.
- Características de transformación.
  - Reescritura de URLs, modificación de unos o todos los links de una página.
  - Captura del sitio Web para su almacenamiento en disco.
  - Censura, eliminando palabras o frases ofensivas.
  - Limpieza del HTML.
  - Eliminación de anuncios.
  - Conversión a XML.

Gracias al conocimiento de HTML del autor, esta herramienta ha sido utilizado para extraer exclusivamente el texto considerado como párrafos para su posterior procesado.

3.2. Herramientas semánticas

3.2.1. OpenNLP

La primera que se presenta es la herramienta OpenNLP[3] [27]. En realidad, OpenNLP es una colección de múltiples sub-herramientas que proporcionan muchas de las técnicas citadas anteriormente. Es una herramienta escrita totalmente en Java y está estacionada en Apache[4]. Las diferentes técnicas contenidas en esta herramienta son las siguientes:

- Detección o división de oraciones: disponible para el danés, alemán, inglés, holandés, portugués y sueco.
- Tokenización: disponible para el danés, alemán, inglés, holandés, portugués y sueco.
- Chunking: disponible para el inglés.

[3] http://incubator.apache.org/opennlp/
[4] http://www.apache.org/

- Etiquetado PoS: disponible para el danés, alemán, inglés, holandés, portugués y sueco.
- Análisis sintáctico: disponible para el inglés.
- Detección de entidades del nombre: disponible para el inglés, español y holandés.
- Resolución de la correferencia: disponible para el inglés.

Alguna de estas características, como el Part of Speech Tagger, se consigue gracias al uso del paquete de aprendizaje MaxEnt[5], el cual caracteriza a los eventos estadísticamente y selecciona aquellos que tengan la máxima entropía.

3.2.2. Stanford

El grupo de trabajo de Stanford del área del procesamiento del lenguaje natural[6] tiene unas características similares a la herramienta OpenNLP. Las distintas sub-herramientas proporcionadas por Stanford creadas estadísticamente y que cubren la mayoría de problemas lingüísticos computacionales. Como OpenNLP, éste también está escrito totalmente en Java.

En este caso, también tiene la ventaja de ser de código abierto bajo la licencia de acceso público general GNU[7]. Esta licencia permite el uso de estas herramientas en investigaciones, proyectos de software gratuitos, etc.

Las diferentes técnicas contenidas en esta herramienta se detallan a continuación:

- Herramienta general: es una herramienta que engloba diferentes técnicas NLP para el inglés. Incluye la tokenización, etiquetado PoS, reconocedor de entidades del nombre, analizador sintáctico y resolución de correferencia.
- Tokenizador: un rápido tokenizador para el inglés.
- Analizador sintáctico: implementa varios analizadores sintácticos a partir de diferentes modelos.
- Etiquetador PoS: como en OpenNLP, esta técnica se ayuda del concepto de máxima entropía, y está disponible para el inglés, el árabe, el chino y el alemán. Para el inglés garantiza una probabilidad de acierto entorno al 97 % sobre un corpus en concreto y un 88 % sobre texto desconocido.
- Reconocedor de entidades del nombre: se trata de un modelo condicional y está disponible para el inglés y el alemán.

[5] http://www.cs.princeton.edu/ schapire/maxent/
[6] http://nlp.stanford.edu/
[7] http://www.gnu.org/licenses/gpl-2.0.html

3.2.3. Freeling

Esta herramienta, llamada Freeling[8] [28], es la más completa que se encuentra para el español. Es una librería, de código abierto (bajo licencia LGPL[9]), orientada a desarrolladores que ofrece varios servicios de análisis del lenguaje y está pensada para llamar a sus módulos para tu propia aplicación. Ha sido desarrollada en la Universidad Politécnica de Cataluña por el centro TALP[10].

Una de la grandes cualidades que se puede citar es que la versión de distribución está disponible en múltiples idiomas (inglés, español, catalán, gallego, italiano, portugués, asturiano y galés). En español el diccionario contiene alrededor de 550.000 palabras que corresponden a más de 76.000 combinaciones de lemas-PoS, gracias a la base de datos de EuroWordNet[11]. Freeling destaca por sus múltiples características [29] o *features* que pasamos a nombrar:

- Tokenización o segmentación.
- Divisor de frases.
- Análisis morfológico.
- Tratamiento de los sufijos.
- Reconocedor flexible de multi-palabras.
- Detector de nombre propios.
- NER: capaz de agrupar nombres por categorías.
- Detector de fechas, números, magnitudes físicas, etc.
- Etiquetador de palabras: Está basado en un trigrama[12] de cadenas de Markov. *(Eficiencia por encima del 95%)*
- Resolución de correferencia.
- Análisis sintáctico superficial o por dependencias.
- Desambiguación: a partir del análisis morfológico obtenido, se vuelve a procesar el texto a través de los algoritmos de Brants [30] que implementa el algoritmo de Viterbi para los modelos de Markov de segundo orden. Gracias a este, se consigue un mayor acierto ya que se tiene en cuenta el contexto semántico y la posición en la oración.

[8] http://nlp.lsi.upc.edu/freeling/
[9] Lesser General Public Licence
[10] Centro de Tecnologías y Aplicaciones del Lenguaje y del Habla
[11] http://www.illc.uva.nl/EuroWordNet/
[12] Un n-grama es una subsecuencia de n elementos (palabras) de una secuencia determinada.

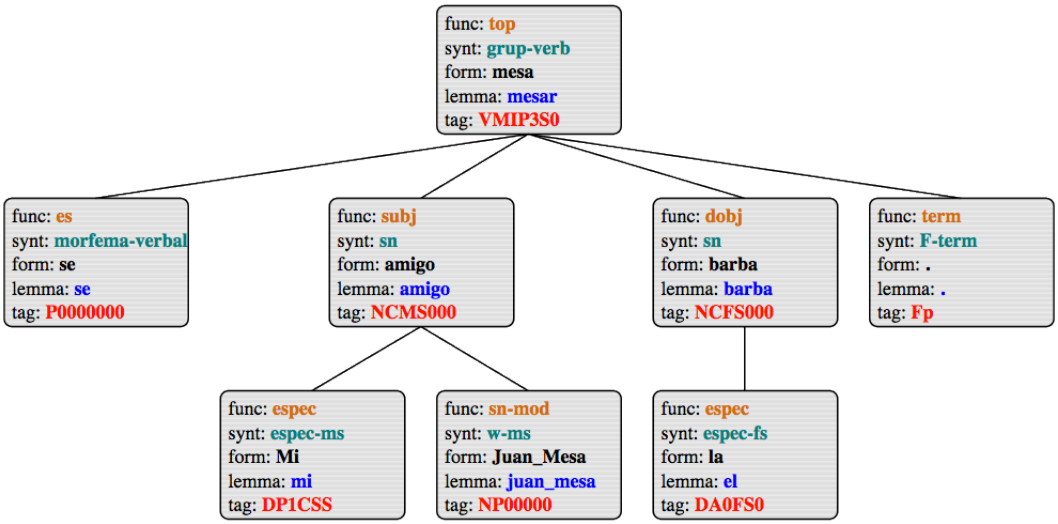

Figura 3.1: Ejemplo visual del analizador de Freeling

Por todas estas características se antoja muy útil para el uso en las distintas partes de un QAS en español, además recientemente se ha implementado un wrapper en Java del código original en C++.

Por último, se muestra un ejemplo del analizador de Freeling con la frase *'Mi amigo Juan Mesa se mesa la barba."* en la figura 3.1.

3.2.4. TreeTagger

Esta herramienta,la cual fue usada en la figura 2.8 para la demostración del resultado esperado en el uso de la técnica de PoS-Tagging, ha sido desarrollada en el Instituto de Lenguaje Computacional de la Universidad de Stuttgart[13].

El TreeTagger [17] o etiquetador en árbol, funciona de forma parecida a un n-grama pero con un árbol de decisión binario. Un etiquetador n-grama deduce las probabilidades de la clase de una secuencia de palabras según los modelos de Markov de la siguiente manera (en el caso de un modelo de Markov de segundo orden):

$\Rightarrow p(w_1w_2...w_n, t_1t_2...t_n) = p(t_n|t_{n-2}t_{n-1})p(w_1w_2...w_{n-1}, t_1t_2...t_{n-1})$

Sin embargo, TreeTagger, halla las probabilidades siguiendo el camino hacia esa palabra o nodo. En el siguiente ejemplo, se revisa el caso de utilizar tres palabras para el cálculo de probabilidades:

$\Rightarrow$ *La bella mujer.* $\rightarrow p(NN|DET, ADJ)$ En el caso de querer calcular la probabilidad de que *mujer* sea un nombre, un ejemplo de actuar es ver si la palabra justo anterior es un adjetivo y la anterior a esta es un artículo. Para esta combinación la probabilidad de que *mujer* sea un nombre es bastante alta como muestra el siguiente árbol:

[13]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

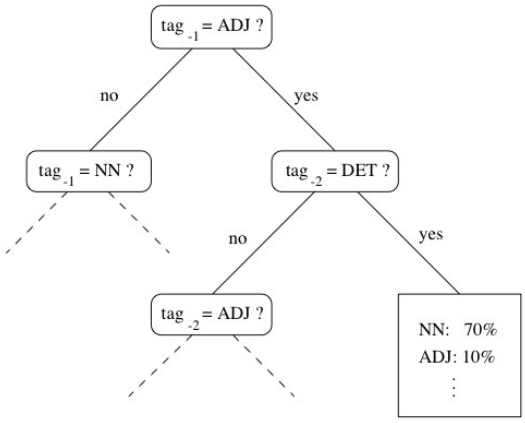

Figura 3.2: Ejemplo visual del algoritmo utilizado por TreeTagger

Esta herramienta comprende dos grandes del procesamiento del lenguaje natural:

- Etiquetado PoS: está disponible para el alemán, inglés, francés, italiano, español, holandés, búlgaro, ruso, griego, portugués, chino, swahili, latín, estonio y francés antiguo. Además, es adaptable a otros idiomas si el léxico y un corpus de entrenamiento etiquetado manualmente está disponible. Garantiza una probabilidad de acierto entorno al 96 %.
- Lematizador: disponible para los mismos idiomas que el POST.
- Chunking: disponble para el inglés, alemán, y francés.

Finalmente, cabe destacar que está disponible un *wrapper* Java para esta herramienta.

3.2.5. Comparativa y elección

Para concluir este capítulo se comparan las distintas herramientas NLP presentadas para tener una visión general de cuál puede ser la más útil para el trabajo deseado. La comparativa está realizada con las características que dispone cada herramienta por defecto.

La primera tabla (tabla 3.1) compara las diferentes herramientas con el inglés como idioma.

La segunda tabla (tabla 3.2) compara las diferentes herramientas con el español como idioma.

En relación a la elección de las herramientas NLP, se han utilizado indistintamente cada una de ellas excepto en referencia a la técnica de Part of Speech. Para esta técnica, debido a la importancia que tiene en el desarrollo del proyecto, se ha dado preferencia a la fiabilidad sobre la rapidez. Mientras que OpenNLP y Stanford NLP ofrecen grandes herramientas para esta técnica, son menos fiables que el camino escogido de analizar sintácticamente una

Funcionalidad	OpenNLP	Stanford NLP	Freeling	TreeTagger
Tokenización	✓	✓	✓	✗
Retokenización	✓	✗	✗	✗
Detección de frases	✓	✗	✓	✗
Etiquetado PoS	✓	✓	✓	✓
Chunking	✓	✓	✓	✓
Lematización	✗	✗	✓	✓
Análisis sintáctico	✓	✓	✓	✗
NER	✓	✓	✓	✗
Correferencia	✓	✓	✓	✗
Lenguaje	Java	Java	C++	Perl
Wrapper para Java	—	—	✓	✓

Tabla 3.1: Comparativa de herramientas NLP para el inglés

Funcionalidad	OpenNLP	Stanford NLP	Freeling	TreeTagger
Tokenización	✗	✗	✓	✗
Retokenización	✗	✗	✗	✗
Detección de frases	✗	✗	✓	✗
Etiquetado PoS	✗	✗	✓	✓
Chunking	✗	✗	✓	✗
Lematización	✗	✗	✓	✓
Análisis sintáctico	✗	✗	✓	✗
NER	✓	✗	✓	✗
Correferencia	✗	✗	✓	✗
Lenguaje	Java	Java	C++	Perl
Wrapper para Java	—	—	✓	✓

Tabla 3.2: Comparativa de herramientas NLP para el español

oración y posteriormente extraer las etiquetas PoS. Para ilustrar esta situación se facilita el ejemplo en la tabla 3.3 donde el término "studies" es etiquetado de manera errónea por el etiquetador PoS pero de manera correcta combinando el análisis sintáctico.

Oración	The field of machine learning studies the design of computer programs.
PoS	DT NN IN NN VBG **NNS** DT NN IN NN NNS .
Análisis+PoS	DT NN IN NN VBG **VBZ** DT NN IN NN NNS .

Tabla 3.3: Comparación entre PoS y Análisis+PoS

Capítulo 4

Análisis

"Poca observación y muchas teorías llevan al error. Mucha observación y pocas teorías llevan a la verdad."

— Alexis Carrel

En este capítulo se describe una de las fases más importantes del desarrollo del software, el análisis de los requisitos y de los distintos escenarios. Para contextualizar cada uno de los requisitos que se presentan, se realiza un análisis detallado de los Casos de Uso que pueden darse, mediante el lenguaje unificado de modelado (UML, Unified Modeling Language), que permite especificar, construir y documentar un sistema mediante lenguaje gráfico. El resultado de esta evaluación consistirá en una especificación completa de requisitos, cuyo cumplimiento por parte del módulo deberá asumirse durante la fase de diseño.

4.1. Casos de uso

Esta sección pretende identificar los casos de uso del sistema, con el objetivo de obtener una especificación completa de la utilización del mismo que permita establecer el listado completo de requisitos a cumplir. Se utilizan los diagramas UML de casos de uso junto a la descripción informal de los mismos, que permiten establecer los actores que interactúan con el sistema y las relaciones entre ambos. Estos diagramas darán lugar a tablas con la especificación completa de los casos de uso, cuya principal utilidad será la de establecer los requisitos de usuario.

4.1.1. Diccionario de los actores

En la tabla 4.1 se recogen los actores primarios y secundarios que intervienen en los casos de uso que se detallan más adelante.

Identificador del actor	Papel	Descripción
ACT-1	Usuario	Usuario del sistema
ACT-2	Administrador	Administrador del sistema
ACT-3	Sistema QA	Sistema QA cargado en el sistema.
ACT-4	Bot	Bot cargado en el sistema.

Tabla 4.1: Diccionario de los actores

4.1.2. Módulo de adaptación de entrada

- CU-1: Añadir fuentes de información (tabla 4.2)

  El sistema de QA es totalmente dependiente de los archivos de entrenamiento y de, al menos, un tesauro y un libro. Estos archivos pueden ser añadidos por cualquier administrador que tenga acceso al sistema. El resumen de este caso se presenta en la tabla 4.2.

- CU-2: Adaptar las fuentes de información (tabla 4.3)

  El sistema de QA necesita trabajar con texto plano, por lo que si el libro está en HTML o PDF, éste es convertirlo a texto plano. El resumen de este caso de uso se presenta en la tabla 4.3.

La representación de estos casos de uso se representa con la figura 4.1.

Identificador	CU-1
Nombre	Añadir fuentes de información.
Descripción	Permite mejorar la eficiencia del sistema QA a través de los archivos de entrenamiento y además que se etiquete un libro gracias al tesauro y al libro.
Actores	Administrador
Precondiciones	-
Flujo normal	1. El administrador añade el libro de texto que quiere etiquetar. 2. El administrador añade el tesauro de conceptos. 3. El administrador añade los archivos de entrenamiento.
Flujo alternativo	-
Poscondiciones	-

Tabla 4.2: Caso de uso número 1

Identificador	CU-2
Nombre	Adaptar las fuentes de información.
Descripción	Se adaptan los libros de texto HTML y PDF para el sistema QA.
Actores	Sistema QA
Precondiciones	Los libros de texto tienen que estar en HTML o PDF.
Flujo normal	1. Se adaptan los libros HTML y PDF para trabajar con texto plano.
Flujo alternativo	-
Poscondiciones	-

Tabla 4.3: Caso de uso número 2

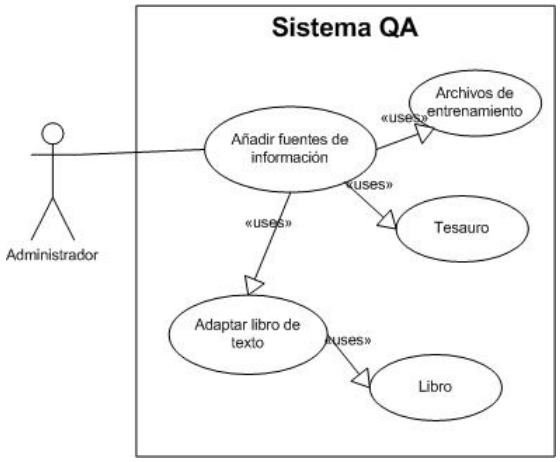

Figura 4.1: Representación de los casos de uso del módulo de adaptación de entrada

4.1.3. Módulo de etiquetado de los conceptos

El principal caso de uso de este módulo es el etiquetado de conceptos, en el cual un archivo de texto plano, mediante una etapa de procesado, pasará a tener todos los conceptos etiquetados.

- CU-3: Etiquetado de conceptos (tabla 4.4)

Identificador	CU-3
Nombre	Etiquetado de conceptos.
Descripción	Se etiquetan los conceptos, a partir de un tesauro y un libro de texto plano.
Actores	Sistema QA
Precondiciones	Los libros de texto tienen que estar cargados.
Flujo normal	1. Recuperación de los conceptos a partir de un tesauro simple o modelado en SKOS. 2. Etiquetado de los conceptos recuperados en un libro de texto plano.
Flujo alternativo	-
Poscondiciones	-

Tabla 4.4: Caso de uso número 3

La representación de este caso de uso se representa con la figura 4.2.

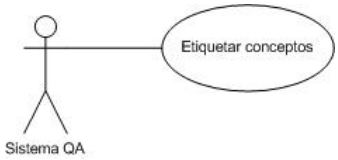

Figura 4.2: Representación del caso de uso del etiquetado de conceptos

4.1.4. Módulo de etiquetado de definiciones

En el módulo de etiquetado de definiciones se trabaja a partir de los patrones extraídos de unos archivos de entrenamiento y del texto plano con los conceptos etiquetados.

- CU-4: Etiquetado de definiciones (tabla 4.6)

Identificador	CU-4
Nombre	Etiquetado de definiciones.
Descripción	Etiquetado de definiciones a partir de los patrones extraídos.
Actores	Sistema QA
Precondiciones	1. Los diversos patrones deben estar cargados. 2. Los conceptos están etiquetados.
Flujo normal	1. Puntuación de las oración. 2. Si la puntuación supera un umbral, se etiqueta la oración como definición.
Flujo alternativo	No se etiqueta la oración como definición.
Poscondiciones	-

Tabla 4.5: Caso de uso número 4

La representación de este caso de uso se representa con la figura 4.3.

4.1.5. Módulo de la aplicación semántica

En este módulo se encuentran los casos de uso principales de un sistema QA. El usuario puede resolver sus dudas sobre un tema en concreto realizando preguntas al sistema QA.

- CU-5: Hacer consulta al sistema QA

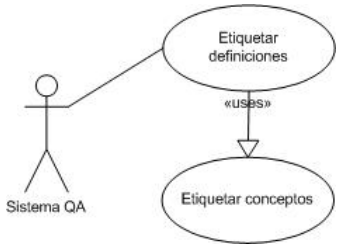

Figura 4.3: Representación del caso de uso del etiquetado de definiciones

Identificador	CU-5
Nombre	Hacer consulta al sistema QA
Descripción	Permite que un usuario pueda resolver sus dudas sobre un tema en concreto realizando preguntas al sistema QA.
Actores	Usuario y Sistema QA
Precondiciones	El libro etiquetado y el tesauro están cargados.
Flujo normal	1. El usuario realiza una pregunta. 2. El sistema QA detecta que la pregunta es de tipo definición. 3. El sistema QA obtiene y presenta la respuesta.
Flujo alternativo	El sistema QA detecta que la pregunta no es de tipo definición y no presenta respuesta.
Poscondiciones	-

Tabla 4.6: Caso de uso número 5

- CU-6: Buscar la respuesta

  Cuando se ha hecho una consulta al sistema QA, éste la busca en el libro etiquetado. La descripción de este caso de uso se encuentra en la tabla 4.7 y su representación en la figura 4.3.

Identificador	CU-6
Nombre	Buscar respuesta
Descripción	El sistema QA busca la respuesta consultada por el usuario.
Actores	Sistema QA
Precondiciones	1. El libro etiquetado y el tesauro están cargados. 2. La pregunta es de tipo definición.
Flujo normal	1. El sistema QA busca las frases que han sido etiquetadas como definición al concepto consultado.
Flujo alternativo	-
Poscondiciones	-

Tabla 4.7: Caso de uso número 6

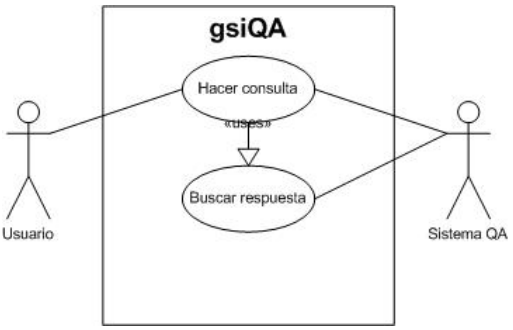

Figura 4.4: Representación de los casos de uso de la aplicación semántica

4.1.6. Sistema QA en entorno educativo

En el proyecto EduWai se ha incorporado el uso del sistema QA y del Bot a un entorno de educación vía Web. El usuario realiza su pregunta en el foro y de manera transparente puede obtener respuesta por entes artificiales como son un Bot y un sistema QA.

- CU-7: Preguntar en el foro de un entorno educativo

Identificador	CU-7
Nombre	Preguntar en el foro.
Descripción	El usuario realiza una pregunta en un foro y, de manera transparente, recibe una respuesta por un Bot o un sistema QA.
Actores	Usuario, Bot y sistema QA.
Precondiciones	El usuario ha entrado en el foro correspondiente.
Flujo normal	1. El usuario crea un nuevo hilo en el foro. 2. El Bot escucha la pregunta. 3. El Bot encuentra una respuesta satisfactoria. 4. Se genera un nuevo post de respuesta con la contestación del Bot.
Flujo alternativo	1. El Bot no encuentra una respuesta satisfactoria. 2. La pregunta la escucha el sistema QA. *a*) Si el sistema obtiene una respuesta satisfactoria, se genera un nuevo post de respuesta con la contestación del sistema QA. *b*) Si no hay respuesta, no se genera nuevo post.
Poscondiciones	-

Tabla 4.8: Caso de uso número 7

La representación de estos casos de uso se representa con la figura 4.5.

4.2. Captura de requisitos

La captura de requisitos recoge las especificaciones generales que debe cumplir el sistema.

- Texto plano: la búsqueda de las respuestas debe hacerse sobre texto plano, sin estructurar.

Figura 4.5: Representación de los casos de uso del entorno educativo

- Automatizado: los algoritmos de búsqueda de las respuestas deben ser generales y automatizados para abarcar cualquier caso.
- Tesauro formal: tanto los conceptos, como las posibles formas en las que se pueden presentar, como las relaciones entre ellos, tienen que estar representados con un estructura formal y aceptada internacionalmente.
- Interacción: el sistema final debe comprender la pregunta del usuario y presentar una respuesta si la encontrase.
- Español como idioma: debe funcionar para el español, ya que los usuarios potenciales tienen como lengua materna ese idioma.
- Integración con un entorno de educación: el sistema debe funcionar de manera transparente en un entorno de educación.

Capítulo 5

Diseño arquitectónico

"Programar sin una arquitectura o diseño en mente es como explorar una gruta sólo con una linterna: no sabes dónde estás, dónde has estado ni hacia dónde vas."

— Danny Thorpe

En este capítulo se describe, de manera detallada, la fase de diseño del proyecto. Esta etapa es la más importante y definitoria de cualquier proceso de desarrollo software para que el sistema o producto final sea el deseado. El diseño consiste en aplicar un conjunto técnicas y metodologías con el fin de obtener un resultado lo suficientemente detallado para que cualquier persona dedicada al desarrollo de software sea capaz de realizarlo.

En ciertas ocasiones, se suele asociar el término "diseño" a una interfaz gráfica, pero en software, el diseño implica un proceso específico gracias al cual se deben satisfacer los requisitos del sistema de desarrollo.

Dentro del diseño del software de la solución, se engloban tanto el diseño de todos los componentes del proyecto y las clases que los componen como el desarrollo de elementos esenciales para este sistema, tales como la elaboración de un tesauro con las especificaciones de SKOS y de los patrones de aprendizaje del conocimiento.

5.1. Descripción global del sistema

Con el propósito de dar una visión general del sistema de búsqueda de respuestas desarrollado, el sistema GSI QA, se describe brevemente la arquitectura global del sistema final. En las siguientes secciones se describen tanto el modelo funcional como el estructural de cada módulo. Lo primero que hay que destacar es que el sistema está pensado para estar integrado dentro de un entorno Web dedicado a educación o *e-learning*. La prueba de concepto consiste en dar soporte y mayor inteligencia al sistema GSI Bot. El esquema simplificado de la arquitectura se muestra en la figura 5.1.

La arquitectura se divide en cinco módulos:

- Módulo de adaptación del texto de entrada: conversión de archivos HTML y PDF a texto plano.
- Módulo de etiquetado de conceptos: etiquetado de los conceptos a partir de un tesauro o árbol de conceptos.
- Módulo de etiquetado de definiciones: etiquetado de las posibles definiciones de los conceptos etiquetados.
- Módulo de la aplicación semántica: módulo encargado de entender y comprender las preguntar planteadas por un usuario, extraer esa información y presentarla.
- Módulo de interfaz Web: engloba la inclusión del sistema en un entorno educativo.

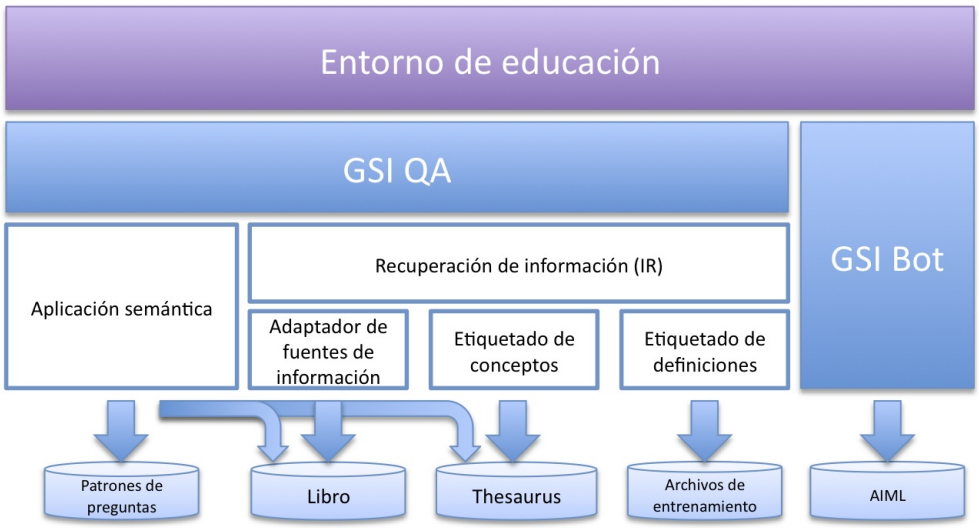

Figura 5.1: Arquitectura global con el entorno educativo

De todos los elementos representados en el diagrama, es el módulo de etiquetado de definiciones en que se aloja la inteligencia del sistema, y es por tanto al que se ha dedicado

la mayor carga de trabajo. En el capítulo 6, se presentan las pruebas realizadas y la eficiencia de este módulo.

Actualmente, y debido a la mayor madurez de los modelos de datos en lengua inglesa para las herramientas empleadas, el sistema trabaja con textos en inglés y realiza la conversión al español a través del traductor de Google[1].

La incorporación de un servicio de traducción disminuye la eficiencia del sistema. No obstante, la eficiencia del sistema con la incorporación de este servicio es mayor que la que ofrece el sistema con los modelos de datos de las herramientas de NLP para el castellano.

El proyecto está completamente escrito en el lenguaje de programación Java salvo la integración con el entorno de educación, en el que se ha tenido que trabajar con PHP.

El resumen de los distintos procesos del sistema GSI QA se muestra en la figura 5.2 y se describe en el siguiente listado:

1. A partir de un documento en HTML o PDF, el módulo adaptador de fuentes de información extrae la información en texto plano.

2. Con la información contenida en el tesauro se extrae un listado con los sinónimos que pueden obtener los conceptos relevantes. El módulo etiquetador de conceptos se encarga de reconocer si alguna de estas formas está presente en los documentos de entrada y las etiqueta.

3. Utilizando las herramientas de NLP y los distintos patrones de definiciones se aplica el algoritmo de etiquetado de definiciones sobre el documento con los conceptos etiquetados para obtener una nueva versión de aquel documento en el que están marcadas las definiciones de cada concepto. Pudiendo haber diferentes definiciones de un mismo concepto.

4. Una vez finalizado este proceso, la aplicación semántica puede hacer uso de los resultados obtenidos para responder a las consultas que le realicen los usuarios sobre los conceptos con los que trabaja.

[1] http://code.google.com/p/google-api-translate-java/

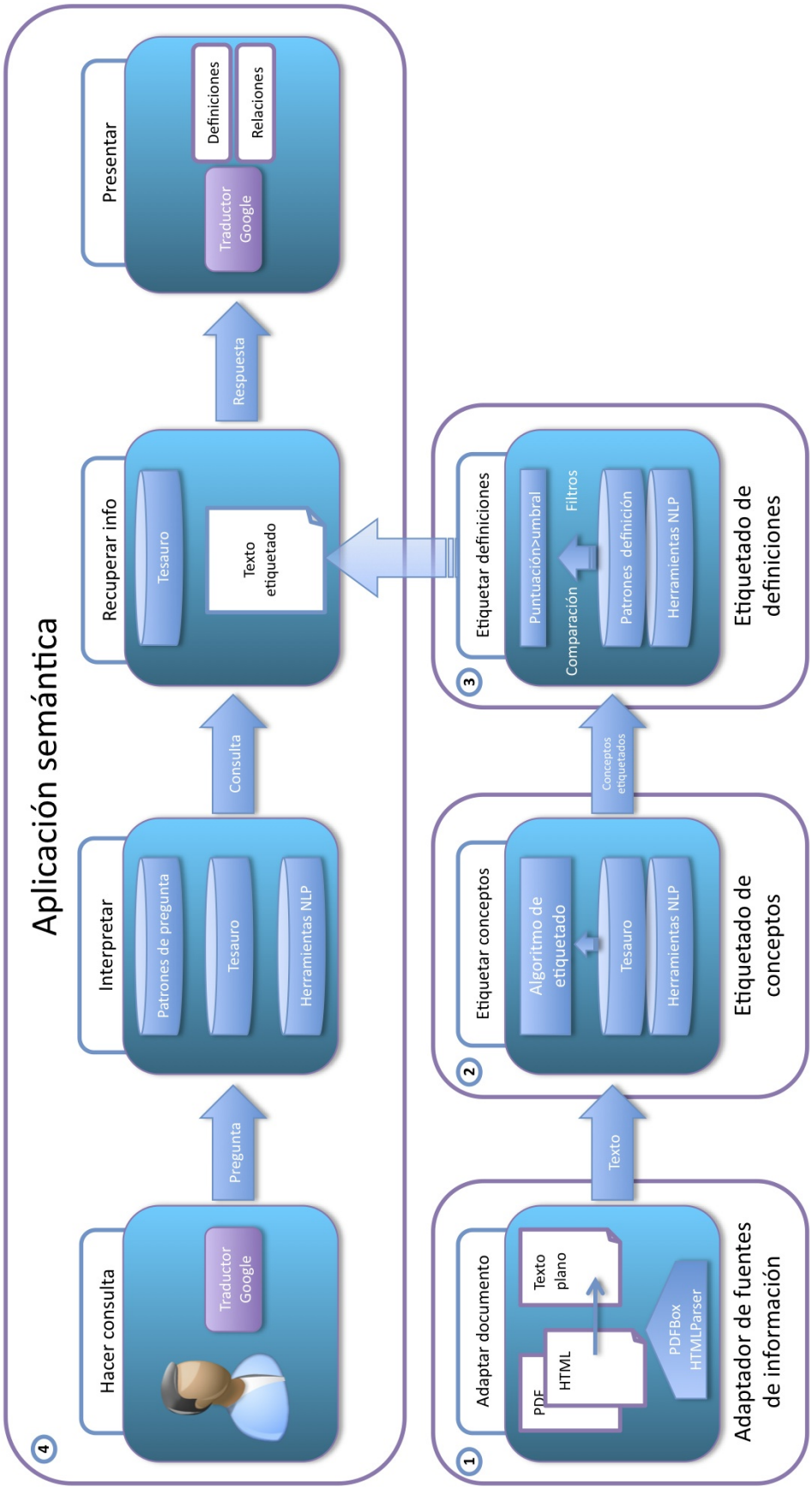

Figura 5.2: Arquitectura global del sistema GSI QA

5.2. Módulo de adaptación de entrada

El módulo de adaptación de entrada (figura 5.3) prepara el texto de los documentos de los que se desea extraer información. Un requisito esencial del procesamiento de lenguaje natural es el uso de texto plano o sin estructuras. Para ello, este módulo se encarga de convertir texto estructurado en texto plano, a partir de documentos HTML o PDF. Además, se tiene la opción de introducir texto plano directamente.

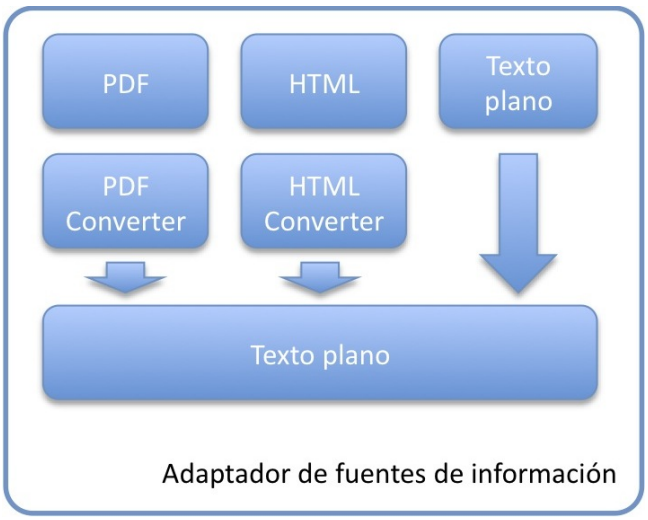

Figura 5.3: Adaptador de fuentes de información

5.2.1. Modelo funcional

La funcionalidad principal de esta primera etapa es la de convertir archivos de texto estructurado en archivos de texto plano. Esta funcionalidad se divide en las siguientes:

- Eliminar las etiquetas y otras cadenas de formato o estilo presentes en el documento HTML.
- Filtrado de información irrelevante como los menús y otros elementos de navegación en HTML.
- Conversión de archivos PDF a archivos de texto plano.
- Preparar la información para su correcto procesado en posteriores etapas.

5.2.2. Modelo estructural

La adaptación de entrada posee una arquitectura de entrada/salida en el que la entrada es el archivo codificado y la salida el texto plano contenido en esos archivos.

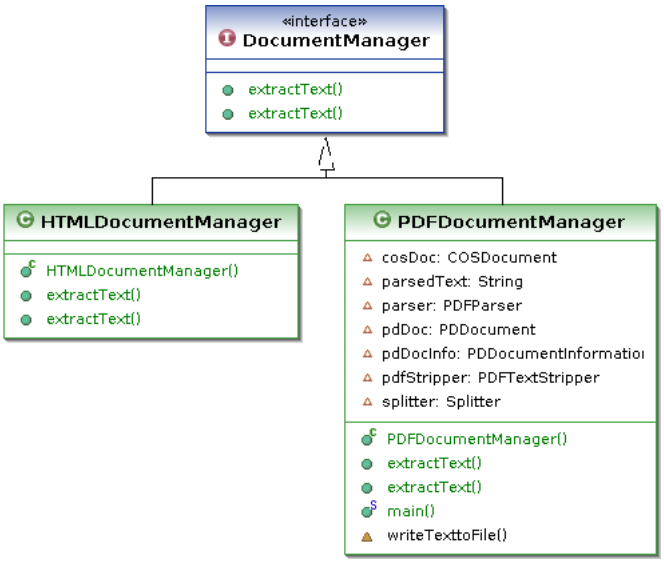

Figura 5.4: Diagrama de clases del adaptador de fuentes de información

La estructura de este módulo se muestra en el diagrama de clases de la figura 5.4.
El resumen de la estructura de este módulo es el siguiente:

- Conversión de archivos HTML: el objeto `HTMLDocumentManager` se ayuda de la herramienta HTMLParser (sección 3.1.2) para crear el método correspondiente para la extracción de texto a partir del fichero HTML o de la ruta de acceso a éste.

- Conversión de archivos PDF: el objeto `PDFDocumentManager` se ayuda de la herramienta PDFBox (sección 3.1.1) para crear el método correspondiente para la extracción de texto a partir del fichero PDF o de la ruta de acceso a éste.

La ventaja de utilizar estas herramientas es la posibilidad de trabajar con texto plano, sin embargo, en el área de la búsqueda de definiciones, se necesita un postprocesado del texto para solventar los inconvenientes con algunos elementos:

- Fragmentos de código de programación: las herramientas NLP no tienen un detector de código. Esto provoca errores a la hora de obtener las frases contenidas en un texto.

- Títulos: las herramientas NLP tienden a solapar los títulos con el primer párrafo del texto. Se muestra el inconveniente con este elemento en tabla 5.1.

- Todo elemento que no se considere párrafos de texto (listas, pies de página, etc.): las herramientas NLP cometen un mayor número de errores al trabajar con estos elementos, que además aportan menor proporción de definiciones que los párrafos de texto.

Fragmento HTML	For loop In computer science a for loop is a programming language statement which allows code to be repeatedly executed. A for loop is classified as an iteration statement.
Fragmento texto plano	For loop In computer science a for loop is a programming language statement which allows code to be repeatedly executed. A for loop is classified as an iteration statement.
Oraciones obtenidas	1. For loop In computer science a for loop is a programming language statement which allows code to be repeatedly executed. 2. A for loop is classified as an iteration statement.

Tabla 5.1: Ejemplo de fallo de la herramienta que divide el texto en oraciones

En el trabajo con los archivos PDF, estos inconvenientes se han solventado manualmente debido a la complejidad de manejo de su codificación. En cambio, con los archivos HTML se han solventado los problemas extrayendo exclusivamente los fragmentos de texto contenidos en los párrafos.

5.3. Módulo de etiquetado de conceptos

El etiquetado de conceptos es la siguiente etapa dentro del preprocesado del documento. Este paso es necesario para posteriormente identificar las definiciones o explicaciones correspondientes a esos conceptos, de manera que los algoritmos de etiquetado de definiciones -que se explican en el apartado 5.4.1- sólo busquen definiciones de aquellos conceptos que han sido identificados y etiquetados por este módulo. Cabe puntualizar que el etiquetado podría ocupar tiempo real de la aplicación final pero se ha elegido este camino por el deseo de tener un libro totalmente etiquetado en un futuro. En esta sección se presentan las distintas etapas de este módulo, como muestra la figura 5.5.

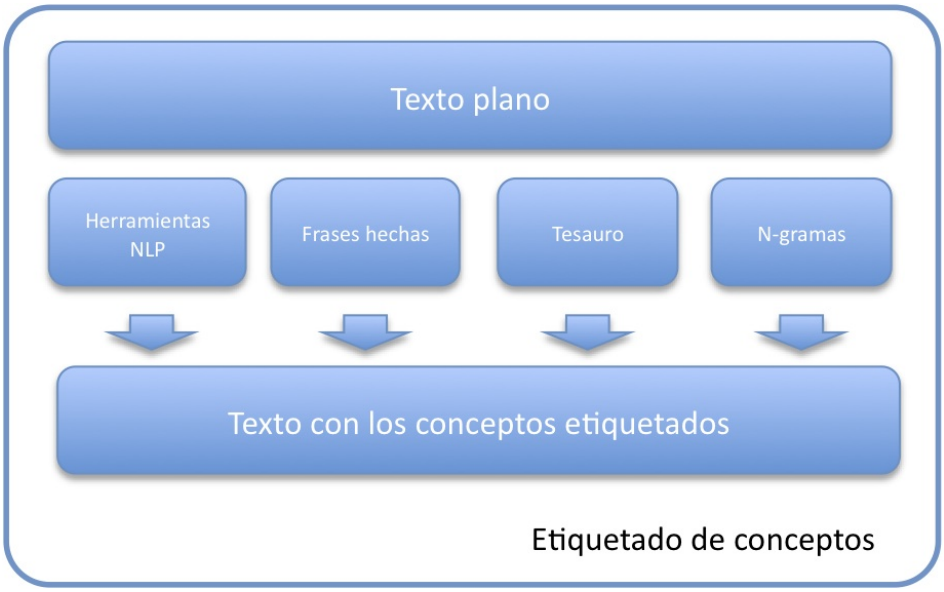

Figura 5.5: Visión general del etiquetado de conceptos

5.3.1. Modelo funcional

Este módulo es el encargado de etiquetar los conceptos definidos en un tesauro en el texto plano obtenido por el adaptador de fuentes de información. Las funcionalidades que el módulo de etiquetado proporciona, se encuentran en la siguiente lista:

- Recuperación de conceptos a partir de un tesauro simple o modelado en SKOS.
- Etiquetado de los conceptos recuperados en el libro de texto plano.

5.3.2. Modelo estructural

5.3.2.1. Etapa de recuperación de los conceptos

La etapa de recuperación de los conceptos es la encargada de recuperar todas las formas posibles en las que un concepto se puede presentar. El sistema necesita una fuente de información estructurada sobre los conceptos del libro que se quiere etiquetar. La gran ventaja en esta etapa del etiquetado es disponer de un conjunto de sinónimos o *synsets* para encontrar todas las formas posibles en las que un concepto se puede mostrar. Por ejemplo, dado el concepto: *for loop*, del tesauro se extraen todas las distintas formas en que puede denotarse dicho concepto: *for loop, for statement, for structure* y *for keyword*.

Un requisito del proyecto es que el tesauro tiene que estar representado con una estructura formal, y por ello, se ha elegido el lenguaje SKOS ya que proporciona los requisitos deseados:

- Estructura formal.

- Acceso al conjunto de sinónimos o *synsets*.
- Acceso a las relaciones entre conceptos con el objetivo de proporcionar valor añadido a los usuarios.

Sus ventajas y características se han presentado en la sección 2.5.2.

Cabe destacar que existe la posibilidad de crear un *dummy* o un tesauro simple en un archivo de texto con la forma mostrada en la figura 5.6. La desventaja de este tipo de tesauro es que se pierde la información añadida por SKOS.

```
Concepto:sinónimo;sinónimo;sinónimo;.....
```

Figura 5.6: Ejemplo de un tesauro

En esta etapa, cuyo diagrama de clases se muestra en la figura 5.7, es necesario la creación de un objeto, denominado `concept`, el cual proporcione la información concerniente a su nombre y a la lista de las posibles formas en las que se puede encontrar su nombre. El objeto `concept`, en el caso de SKOS, proporciona más información de la requerida en esta etapa, por lo que se volverá a citar en el módulo de la aplicación semántica. Para recuperar

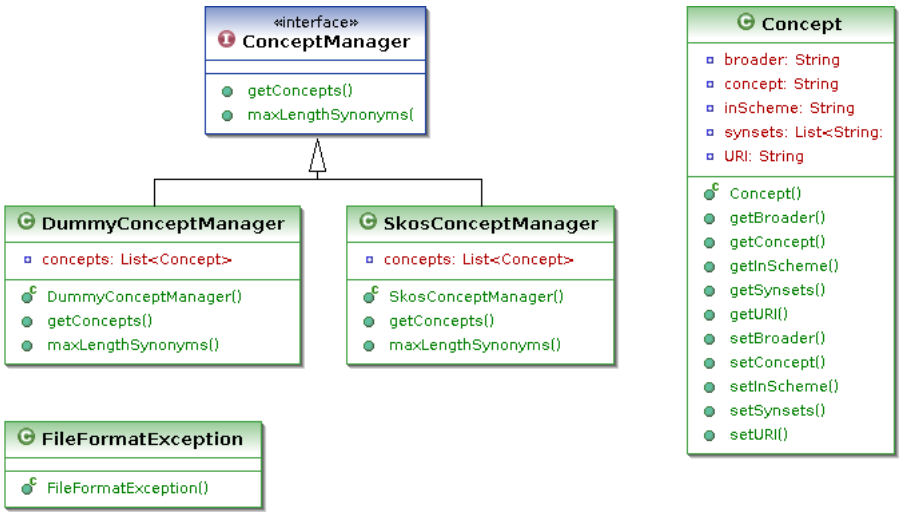

Figura 5.7: Diagrama de clases del manager de conceptos

la información deseada, se han creado los siguiente dos objetos:

- `SkosConceptManager`: es el objeto encargado de trabajar con tesauros creados en SKOS. En la recuperación de información se ha trabajado con la librería SKOS API [2] que proporciona acceso a todas las etiquetas contenidas en el modelo SKOS.

[2] http://skosapi.sourceforge.net/

- `DummyConceptManager`: es el objeto encargado de trabajar con tesauros creados con la estructura mostrada en la figura 5.6.

5.3.2.2. Etapa de etiquetado

Una vez se ha extraído el texto y se tienen los conceptos de los que se desea extraer información, se procede a su etiquetado. Para este fin se ha desarrollado un algoritmo de n-gramas[3] que comprueba si un elemento o un conjunto de elementos del texto es un concepto o no. Esta función la realiza el método `annotate` del objeto `IndexConceptDocumentAnnotator`. Se le llama así a este objeto porque actualiza el índice del recorrido del texto cada vez que encuentra un concepto o una frase hecha del inglés. El diagrama de clases que representa a esta etapa se muestra en la figura 5.8.

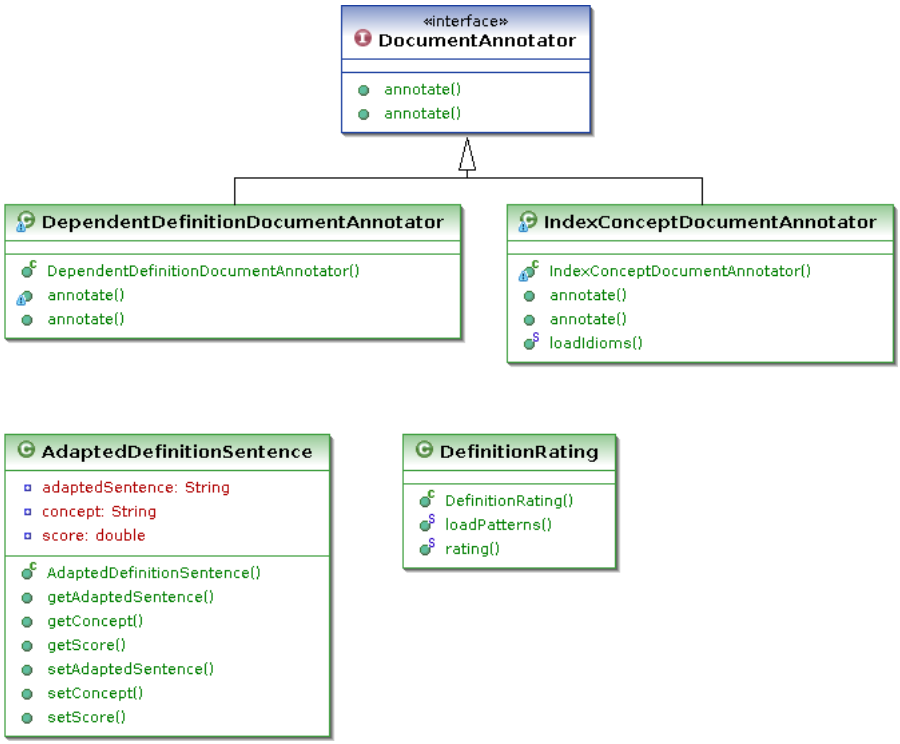

Figura 5.8: Diagrama de clases del anotador de información

Las distintas etapas y el orden de ejecución de cada una se detalla a continuación con la ayuda de la figura 5.9:

1. Se tokeniza el texto con la herramienta OpenNLP y se lematiza con la ayuda las herramientas Wordnet[4] (para los verbos) y PlingStemmer[5] (para los nombres).

[3] Un n-grama es una subsecuencia de n elementos (palabras) de una secuencia determinada.
[4] http://wordnet.princeton.edu/
[5] http://www.mpi-inf.mpg.de/yago-naga/javatools/doc/javatools/parsers/PlingStemmer.html

2. Se comienza desde el primer elemento del texto agrupando tantos elementos como marque la máxima longitud del número de elementos del conjunto de sinónimos del tesauro. Se comienza con la máxima longitud para eliminar los casos de conceptos que estén contenidos en otros.

3. Se comprueba si ese grupo de elementos coincide con algún elemento de un conjunto auxiliar de frases hechas del inglés o *idioms*. Si coincide, se actualiza el índice y se vuelve al segundo paso.

4. Finalmente se comprueba si ese grupo de elementos coincide con algún elemento del tesauro ya lematizado. Si coincide, se etiqueta ese elemento o grupo de elementos, se actualiza el índice y se vuelve al segundo paso. Si no coincide, se disminuye en una unidad el tamaño del grupo de elementos que se comprueban y se vuelve al tercer paso.

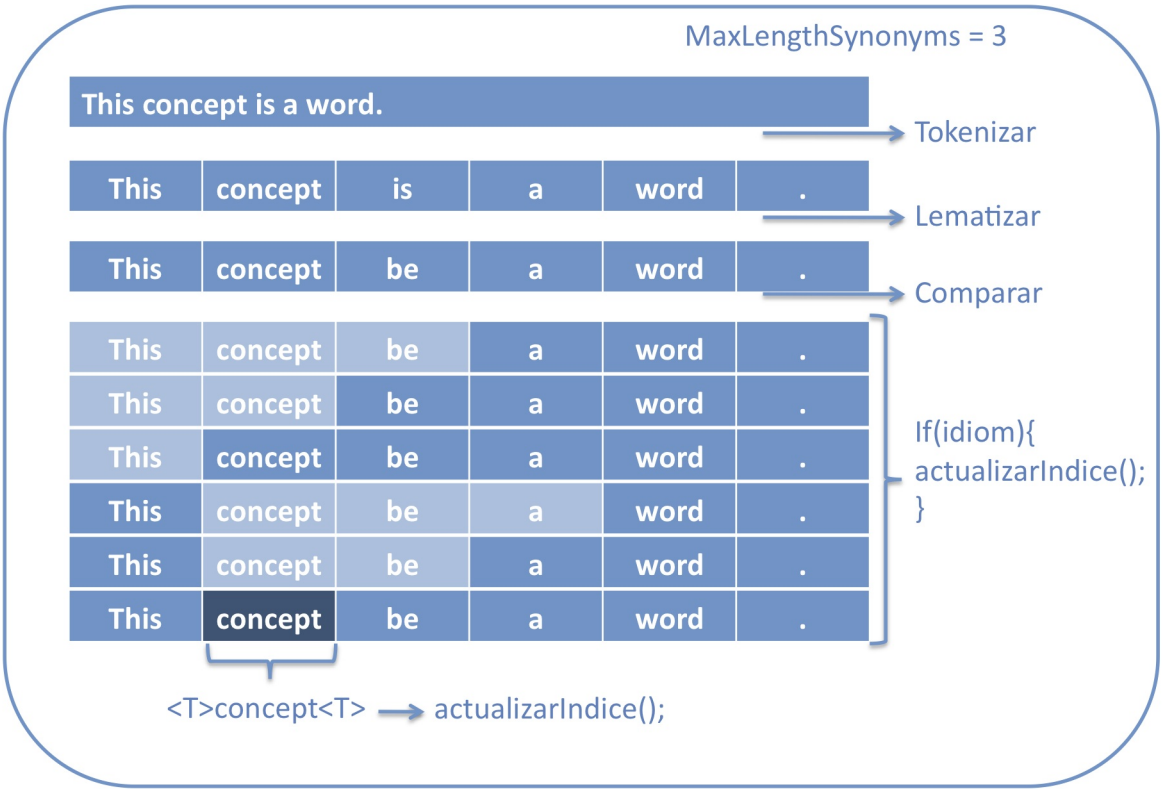

Figura 5.9: Algoritmo de etiquetado de los conceptos

Los conceptos se etiquetan con la estructura mostrada en la figura 5.10

- *ámbito*: ámbito o marco del árbol de conceptos o tesauro.
- *concepto*: nombre preferente del concepto etiquetado.

- *CONCEPTO*: forma en la que se ha encontrado el concepto.

<*ámbito*Concept:*concepto*>*CONCEPTO*</*ámbito*Concept:*concepto*>

Figura 5.10: Etiquetado de un concepto

5.3.3. Un caso específico: tesauro de Java

Para este proyecto se ha llevado a cabo la realización de un tesauro del lenguaje de programación de Java. El tesauro está basado en una ontología conocida como JLOO [31] (*Java Learning Object Ontology*). En la figura 5.11 se muestra un esquema de conceptos dentro de la estructura de JLOO.

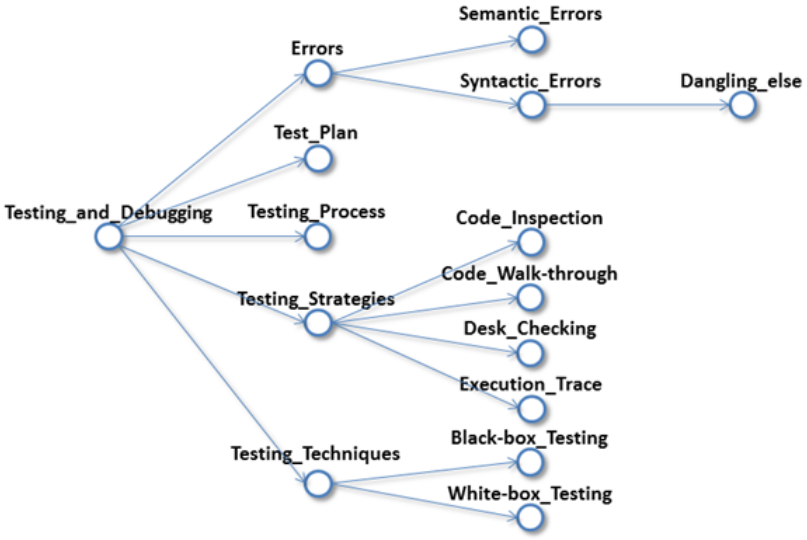

Figura 5.11: Esquema de conceptos dentro de JLOO

Utilizar esta ontología tiene una serie de ventajas para el proyecto:

- Su objetivo principal es ayudar en el aprendizaje del lenguaje de programación Java.
- Sigue las directrices marcadas por el Computing Curricula CC2001 de la ACM y del IEEE/CS [32], donde se define la estructuración ideal de los planes de estudio.
- Es fácilmente realizable en SKOS.

Por otro lado, JLOO, proporciona las siguientes contribuciones:

1. Definición de las unidades de conocimiento atómicas de los cursos de introducción a Java, y sus relaciones entre ellas.

2. Reusabilidad y posibilidad de compartir esas unidades.

3. Permite diferentes estrategias a un entorno de *e-learning* para elegir dinámicamente el uso de JLOO como guía de estudio.

4. Consigue que la curva de aprendizaje sea menos pronunciada.

Por último, se muestra un ejemplo en la figura 5.12, con el concepto *for*. En ella se representan las etiquetas`skos:prefLabel` y `skos:altLabel` que componen en el conjunto de sinónimos, `skos:broader` que proporciona el concepto más amplio y `skos:inScheme` que proporciona el esquema de conceptos al que pertenece.

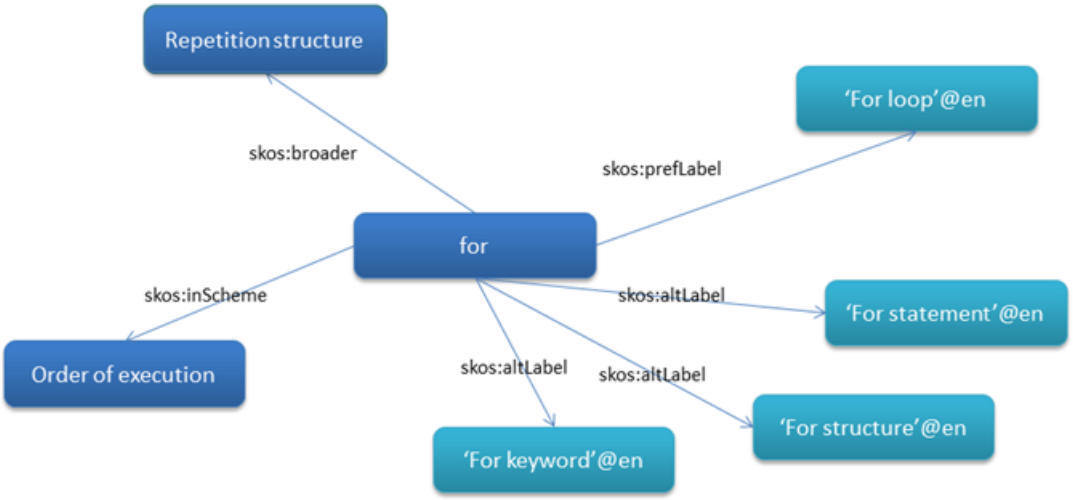

Figura 5.12: Ejemplo de la información contenida por un concepto

5.4. Módulo de etiquetado de definiciones

En esta sección se citan conceptos referentes al aprendizaje de patrones y a la creación de nuevos patrones en el ámbito de la búsqueda definiciones, por lo que se recomienda al lector la lectura del apartado 5.4.1.

5.4.1. Aprendizaje de patrones

5.4.1.1. Introducción

En la búsqueda de definiciones asociadas a un concepto se puede actuar desde varios puntos de vista, como pueden ser los patrones fijos, algoritmos de *machine-learning*, aprendizaje de patrones, etc. Mientras en la etapa de comprensión de la pregunta, se han utilizado

patrones fijados manualmente, esta metodología es descartada a la hora de extraer respuestas por la gran cantidad de formas en las que se pueden mostrar. Por ello, para este proyecto, se ha optado por la rama del aprendizaje de patrones o *pattern learning*. En esta sección se describe cómo se han generado los patrones para la extracción de respuestas, y en nuestro caso particular, definiciones.

Dentro de las modalidades de *pattern learning*, en el desarrollo de este sistema, se ha enfocado en 6 puntos de vista distintos. Cada uno de ellos está pensando para diferentes grados de precisión y sensibilidad o *recall* (apartado 2.1.1). Para la elaboración de ellos se ha hecho uso de las herramientas de procesamiento de lenguaje natural, utilizando un gran número de las técnicas disponibles.

5.4.1.2. Creación de patrones

A continuación se presentan los seis tipos de reglas o patrones (reglas que dan lugar a patrones) y en la sección de experimentación y evaluación se detallarán los resultados de cada uno. En esta presentación se va a seguir un orden marcado por el grado de precisión, de menor precisión a mayor. A priori, los patrones menos precisos recuperarán una gran cantidad de definiciones, siendo válidas solamente un número reducido de ellas. Al contrario que con los patrones más precisos, éstos recuperarán un reducido número de definiciones pero con una alta probabilidad de cada una de ellas lo sea. Para comprender cada tipo de patrón se recomienda seguir el árbol de la figura 5.13 donde se representan, por niveles, los distintos tipos de etiquetas que se utilizan para la creación de los patrones.

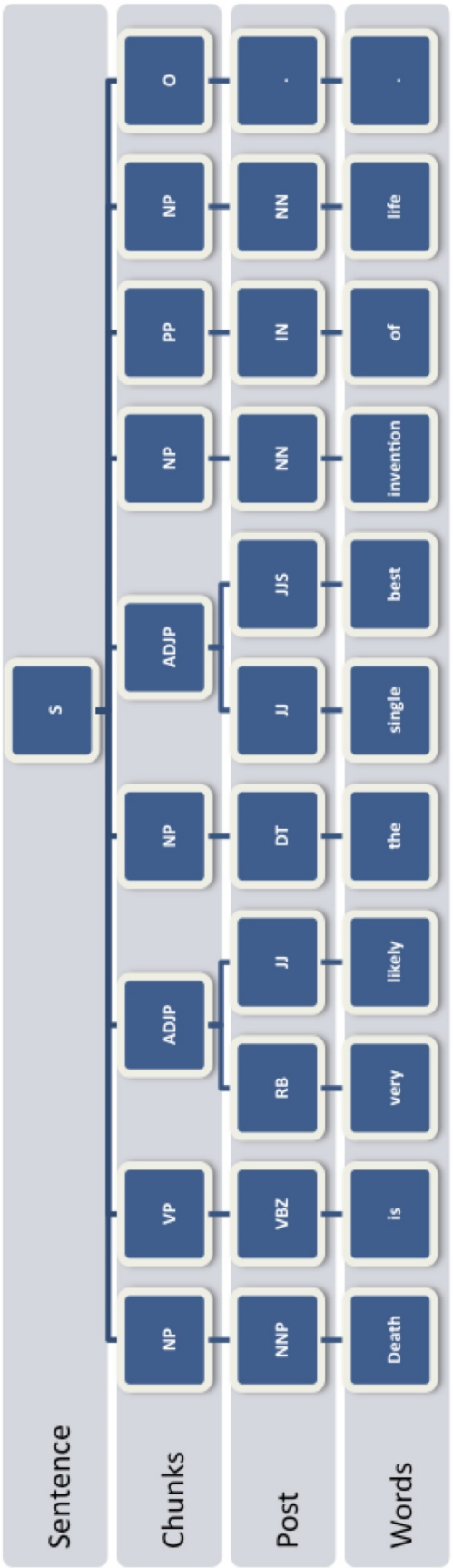

Figura 5.13: Ejemplo de las etiquetas utilizadas en los patrones

1. **Patrones Chunks suaves o *Soft Chunks*:** son patrones que valoran exclusivamente el orden de los predicados dentro de una oración. Para ello se ha hecho uso de la herramienta OpenNLP en la modalidad de *chunking* y se ha desarrollado un simple algoritmo para eliminar las etiquetas que informan de la posición de cada palabra dentro de los predicados y de agrupar las etiquetas iguales consecutivas en una sola. Las etiquetas del *chunking* se encuentran en el apéndice A.2.

   Los patrones *SoftChunk* tienen la siguiente estructura:

   ```
   Predicados <Concepto> Predicados
   ```

   Con el propósito de ilustrar la estructura de este patrón, se muestra un ejemplo en la tabla 5.2.

Oración	Death is very likely the single best invention of life.
Chunking	<T>B-VP B-ADJP I-ADJP B-NP I-NP I-NP B-PP B-NP O
Soft Chunking	<T>VP ADJP NP PP NP O

Tabla 5.2: Ejemplo de patrones *SoftChunk*

2. **Patrones Chunk:** son patrones que valoran el orden de los predicados dentro de una oración y además la posición de cada dentro de cada predicado. Para ello se ha hecho uso de la herramienta OpenNLP en la modalidad de *chunking*. Las etiquetas del *chunking* se encuentran en el apéndice A.2.

   Los patrones Chunk tienen la siguiente estructura:

   ```
   Posiciones en un predicado <Concepto> Posiciones en un predicado
   ```

   Con el propósito de ilustrar la estructura de este patrón, se muestra un ejemplo en la tabla 5.3.

Oración	Death is very likely the single best invention of life.
Chunking	<T>B-VP B-ADJP I-ADJP B-NP I-NP I-NP B-PP B-NP O

Tabla 5.3: Ejemplo de patrones Chunk

3. **Patrones Post suaves o *Soft Post*:** son patrones que tienen en cuenta la función de cada palabra dentro de la oración pero de manera reducida. Para este tipo de patrones, las etiquetas PoS (Apéndice A.1) se han reducido escogiendo solamente la primera letra

de cada etiqueta oficial. Además, se agrupan las etiquetas iguales consecutivas en una sola. Para este fin se ha utilizado la herramienta NLP de Stanford ya que ofrece los resultados más fiables y un algoritmo auxiliar para la reducción. Las 21 etiquetas resultantes, de las 40 oficiales, quedan reflejadas en la tabla 5.4.

Etiqueta	**Función**
C	Conjunción o nombre cardinal
D	Determinante
E	*There* existencial
F	Palabra extranjera
I	Preposición
J	Adjetivo
L	Marcado de listas
M	Verbo modal auxiliar
N	Nombre
P	Predeterminante o pronombre
R	Adverbio o partícula adverbial
S	Símbolo
T	*To*
U	Interjección
V	Verbo
W	Wh-(determinante o pronombre o adverbio)
.	Marca de final de oración
,	Coma
:	Dos puntos
(	Separador izquierdo
)	Separador derecho

Tabla 5.4: Etiquetas PoS reducidas

Los patrones *SoftPost* tienen la siguiente estructura:

```
Funciones reducidas <Concepto> Funciones reducidas
```

Con el propósito de ilustrar la estructura de este patrón, se muestra un ejemplo en la tabla 5.5.

4. **Patrones Post:** son patrones que tienen en cuenta exclusivamente la función de cada palabra dentro de la oración. Para este fin se ha utilizado la herramienta NLP de

Oración	Death is very likely the single best invention of life.
Post	<T>VBZ RB JJ DT JJ JJS NN IN NN .
Soft Post	<T>V R J D J N I N .

Tabla 5.5: Ejemplo de patrones SoftPost

Stanford ya que es la más efectiva de las que se tenía acceso. Las etiquetas PoS se encuentran en el apéndice A.1.

Los patrones Post tienen la siguiente estructura:

```
Funciones <Concepto> Funciones
```

Con el propósito de ilustrar la estructura de este patrón, se muestra un ejemplo en la tabla 5.6.

Oración	Death is very likely the single best invention of life.
Post	<T>VBZ RB JJ DT JJ JJS NN IN NN .

Tabla 5.6: Ejemplo de patrones Post

5. **Patrones literales suaves o *Soft Literal*:** son patrones que mezclan las etiquetas PoS con los verbos en su forma lematizada o raíz en vez de su función. Se le está dando importancia al verbo en este tipo de patrón. Las etiquetas PoS se encuentran en el apéndice A.1.

   Los patrones *Soft Literal* tienen la siguiente estructura:

   ```
   Funciones(verbos lematizados) <Concepto> Funciones(verbos lematizados)
   ```

   Con el propósito de ilustrar la estructura de este patrón, se muestra un ejemplo en la tabla 5.7.

Oración	Death is very likely the single best invention of life.
Soft Literal	<T>be RB JJ DT JJ JJS NN IN NN .

Tabla 5.7: Ejemplo de patrones *Soft Literal*

6. **Patrones literales:** son patrones que solamente tienen en cuenta las palabras sin ningún tipo de modificación.

   Los patrones literales tienen la siguiente estructura:

   ```
   Palabras <Concepto> Palabras
   ```

Con el propósito de ilustrar la estructura de este patrón, se muestra un ejemplo en la tabla 5.8.

Oración	Death is very likely the single best invention of life.
Literal	<T>is very likely the single best invention of life .

Tabla 5.8: Ejemplo de patrones literales

Por último, se detalla el algoritmo desarrollado para obtener las distintas longitudes de los patrones. Se ha utilizado un concepto conocido como *cropping* que proviene del inglés y significa cortar, segar o esquilar. Este concepto ha sido usado en la Universidad de Singapur [15] para extraer patrones con los elementos vecinos, pero en este caso, utilizaban varios heurísticos para su creación, combinando PoS, Chunking y literales en un mismo patrón.

En el sistema de esta memoria, el algoritmo de *cropping* es realizado por el método con el mismo nombre y recorta la frase etiquetada según la longitud elegida. El resultado del algoritmo se muestra en la tabla 5.9.

Frase original	Death is very likely the single best invention of life.
Patrón SoftLiteral	<T>be RB JJ DT JJ JJS NN IN NN .
Patrón SoftLiteral L=1	<T>be .*
Patrón SoftLiteral L=2	<T>be RB .*
Patrón SoftLiteral L=3	<T>be RB JJ .*
Patrón SoftLiteral L=4	<T>be RB JJ DT .*
Patrón SoftLiteral L=5	<T>be RB JJ DT JJ .*

Tabla 5.9: Ejemplo de *cropping*. '.*' = cualquier elemento

5.4.1.3. Archivos de entrenamiento

Con el fin de obtener los diferentes patrones presentados en la sección anterior y dotar a cada uno de ellos de una efectividad porcentual, han sido creados dos archivos de entrenamiento. El primero de ellos, el archivo de definiciones válidas, crea los distintos patrones y les dota de puntuaciones positivas. El segundo, el archivo de frases que no representan definiciones, dota a los patrones ya creados de puntuaciones negativas en el caso de coincidencia.

La efectividad de los patrones utilizados para el sistema desarrollado en este proyecto se ha basado en 500 frases de tipo definición y 1000 frases que no representan a definición. No hace falta demostrar que, como todo sistema que se basa en entrenamiento, la efectividad del sistema aumentará al incrementar el conjunto de prueba.

En la tabla 5.10 se muestran los dos tipos de frases con los que trabaja el sistema para su entrenamiento y cómo es el etiquetado de las mismas para su procesamiento. En cada una de ellas se etiqueta un concepto por frase. En la primera frase de la tabla 5.10 se comprueba que es de tipo definición porque tiene etiquetado el concepto que se está definiendo, *Death*. En cambio, la segunda frase, no es de tipo definición puesto que el concepto etiquetado no se está definiendo, *life*.

Definición	<T>Death<T>is very likely the single best invention of life.
No definición	Death is very likely the single best invention of <T>life<T>.

Tabla 5.10: Muestra de los archivos de entrenamiento

Nota: Si en una misma frase se encuentra un mismo concepto escrito de diferentes maneras, se etiquetan todas porque representan al mismo concepto.

5.4.2. Modelo funcional

Este módulo es el encargado de etiquetar las correspondientes definiciones a los conceptos etiquetados en el texto obtenido por el conversor de documentos. Las funcionalidades que este módulo de etiquetado de proporciona, se encuentran en la siguiente lista:

- Aprendizaje de patrones o reglas a partir de definiciones contenidas en archivos de entrenamiento.
- Puntuación a las posibles definiciones según efectividad de los patrones.
- Filtrado de aquellas posibles definición con pocas posibilidades de ser una definición.
- Etiquetado de las definiciones correspondientes a los conceptos etiquetados a partir de los patrones extraídos, de las puntuaciones y del filtrado.

5.4.3. Modelo estructural

La estructura de este módulo se muestra en el diagrama de clases de la figura 5.8. A partir de la interfaz `DocumentAnnotator`, el uso del `DependentDefinitionDocumentAnnotator` consigue el etiquetado de la información deseada y su nombre se basa en la dependencia que tiene con el texto con los conceptos etiquetados.

5.4.3.1. Extracción de patrones

Esta etapa se lleva cabo independientemente del resto y será utilizada en el etiquetado de definiciones. El diagrama de clases queda reflejado en la figura 5.14.

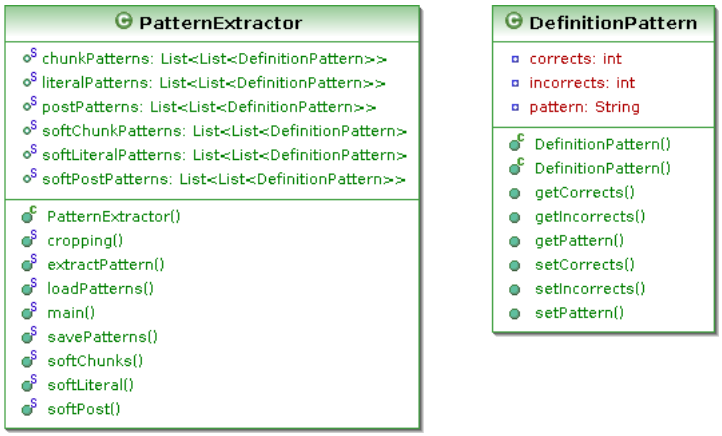

Figura 5.14: Diagrama de clases del extractor de patrones

La clase `PatternExtractor` es la encargada de extraer todos los patrones presentados en el apartado 5.4.1. Los patrones básicos se obtienen con el método `extractPattern` y los otros tipos con sus respectivos métodos: `softLiteral`, `softChunk` y `softPost`. Para obtener patrones con diferentes longitudes se llama al método `cropping`. Cada patrón es un objeto `DefinitionPattern` que contiene la cadena de texto del patrón y los aciertos y errores obtenidos por unos archivos de entrenamiento. Estos patrones proporcionan la probabilidad de que una frase con un concepto sea una posible definición.

5.4.3.2. Etiquetado de definiciones

El módulo de etiquetado de definiciones es el módulo que mayor esfuerzo investigador ha acaparado y que necesitará más mejoras en el futuro para mejorar su fiabilidad y así poder prescindir del ser humano en la etapa de validación. Esta etapa es la encargada de etiquetar las definiciones a partir del texto los conceptos etiquetados, y por ello el objeto que realiza esta función se denomina `DependentDefinitionDocumentAnnotator`. El método encargado del etiquetado es `annotate`.

Los pasos seguidos para etiquetar una frase como definición se muestran en la figura 5.16 y son los siguientes:

1. División del texto en frases con el uso de la herramienta OpenNLP.
2. Creación de tantas frases como conceptos distintos tenga cada una.
3. Puntuar cada frase con las posibilidades de que sea una definición. Este paso tiene a su vez distintas etapas:
   a) Filtrado: se utilizan filtros del tipo "blanco/negro", es decir, o dotan a la respuesta de total confianza o bien se le quita por completo. En la tabla 5.11 se muestran los filtros de tipo negro.

Filtro	Descripción	Ejemplo
El verbo es esencial	La posible definición debe contener, al menos, un verbo.	Java in the world!
Ejemplo de uso	La posible definición se refiere a un caso concreto para explicar un ejemplo.	This class is a dog.
Negación	La posible definición proporciona justamente lo que no es el concepto.	A cat is not a vegetable.

Tabla 5.11: Filtros para las posibles definiciones

b) Puntuación de los patrones elegidos: la puntuación de cada posible definición se obtiene de la media de las puntuaciones de los patrones con los que haya coincidido. Por supuesto, se pueden elegir qué patrones cargar según se desee dar mayor importancia a la precisión, a la sensibilidad o a ambas.

4. Decisión: a la hora de decidir si una posible definición lo es o no, se debe elegir qué umbral de confianza se quiere. Si una posible definición obtiene una puntuación mayor a ese umbral, se marca como definición para su posterior etiquetado.

5. Etiquetado: si la frase ha superado el umbral de confianza, ésta se etiqueta con la estructura presentada en la figura 5.15.

<*ámbito*Definition:*concepto*>*DEFINICIÓN*</*ámbito*Definition:*concepto*>

Figura 5.15: Etiquetado de una definición

- *ámbito*: ámbito o marco del árbol de conceptos o tesauro.
- *concepto*: nombre preferente del concepto del que se ha encontrado una definición.
- *DEFINICIÓN*: frase que cumple los requisitos para ser etiquetada como definición.

5.5. Módulo de la aplicación semántica

Un sistema de búsqueda de respuestas es totalmente funcional cuando es capaz de proporcionar una respuesta a una pregunta dada. Para ello se necesita interpretar y comprender la pregunta del usuario con el fin de deducir qué respuesta espera recibir. Una visión general de este módulo se presenta en la figura 5.17.

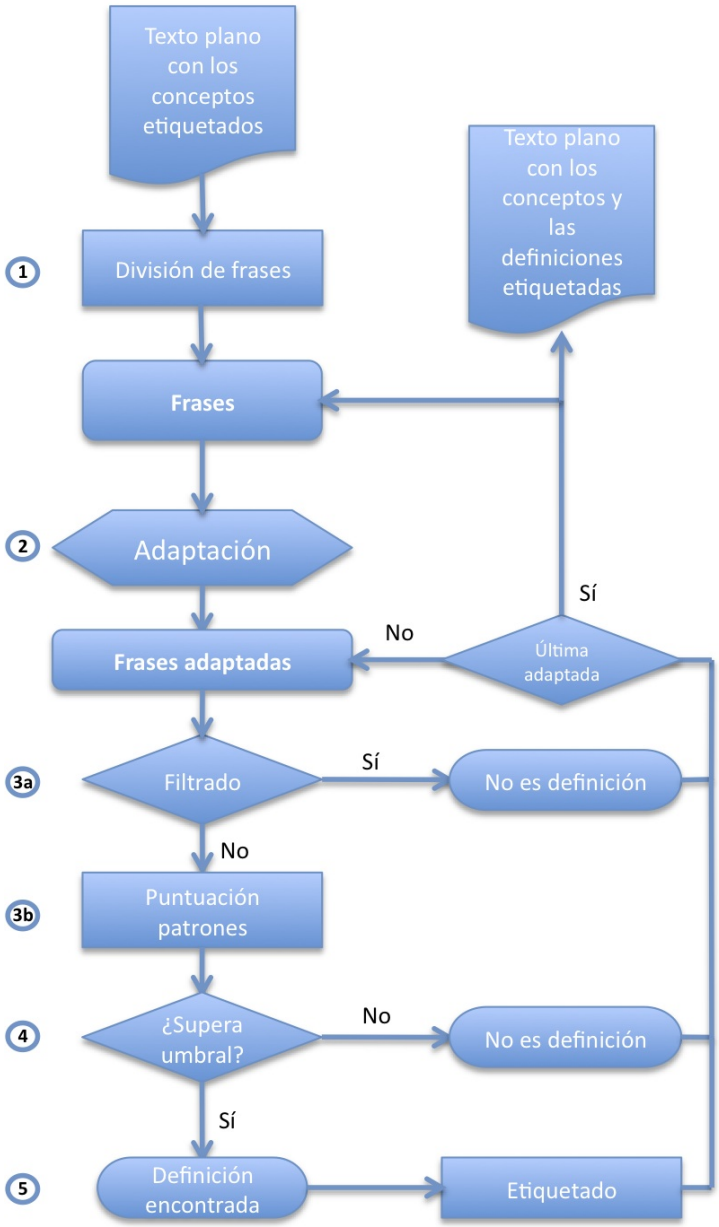

Figura 5.16: Pasos seguidos para el etiquetado de definiciones

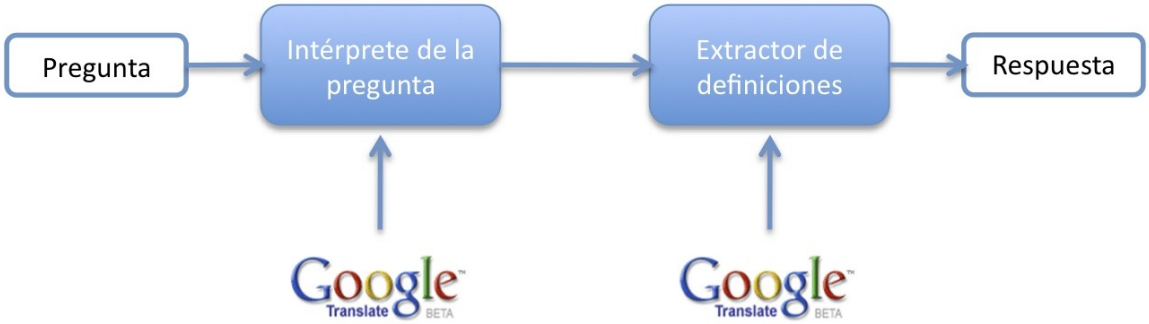

Figura 5.17: Visión general de la aplicación semántica

5.5.1. Modelo funcional

Las diversas funcionalidades de este módulo son las de cualquier sistema de búsqueda de respuestas:

- Interpretar si el usuario está preguntando por una definición o explicación.
- Averigüar si el concepto por el que se consulta, está contenido en el tesauro.
- Recuperar las definiciones etiquetadas de ese concepto en el libro.
- Proporcionar valor añadido a la respuesta con las relaciones que el concepto posee en el tesauro.

Tal como muestra la figura 5.17, el sistema interpreta la pregunta y el extractor de definiciones, en el caso de que la respuesta esperada sea una definición sobre un concepto del tesauro, extrae las posibles definiciones del concepto consultado.

5.5.2. Modelo estructural

Principalmente, la aplicación semántica consta de tres módulos:

1. **Intérprete de la pregunta:** Este módulo es el encargado de deducir qué respuesta está esperando recibir el usuario. El intérprete es capaz de extraer la pregunta a través del texto que ha escrito el usuario. Esto demuestra que el usuario no tiene por qué escribir solamente la pregunta, si no también un texto auxiliar. Un ejemplo que ilustra una posible consulta por un alumno se muestra en la figura 5.18.

   En el ejemplo se ha detectado la pregunta y, por lo tanto se extrae para trabajar con ella. A continuación, se traduce con el traductor de Google y se compara con una serie de patrones fijos de tipo definición para deducir sobre qué concepto se está preguntando.

Hola,
tengo unas dudas sobre un concepto en concreto.
Alguien me puede explicar, ***¿qué es un bucle?***
Muchas gracias por su tiempo.
Un saludo.

Figura 5.18: Ejemplo de consulta por un alumno

2. **Extractor de definiciones:** el extractor de definiciones trabaja si se ha averigüado sobre qué concepto se ha preguntado. Si la respuesta es satisfactoria, se procede a comparar el concepto con el conjunto de sinónimos guardados en el tesauro o árbol de conceptos.

   Si la búsqueda ha tenido éxito, se rastrea el documento anteriormente etiquetado y se extraen y guardan las posibles definiciones para su posterior presentación.

3. **Presentación de la respuesta:** ésta es la última etapa de la aplicación y consiste en presentar la respuesta de la pregunta realizada. En este punto, el sistema tiene guardadas las posibles definiciones o explicaciones sobre el concepto en cuestión y además, gracias al tesauro elaborado en SKOS, se añade una última explicación que se ayuda de las relaciones que tiene dentro del tesauro. Por último, la respuesta es traducida al español gracias al traductor de Google.

   La salida final del sistema tiene la siguiente estructura:

   a) Definiciones o explicaciones del concepto extraídas de un libro.

   b) Relaciones del concepto con el resto de concepto del tesauro.

5.6. Interfaz de usuario: Moodle

El sistema QA de la presente memoria, gsiQA, toma como interfaz de usuario la interfaz Web de la plataforma Moodle[6], centrada en el entorno educativo. Más concretamente, se hace uso del módulo de foros de discusión proporcionado por Moodle, el cual está ideado para que los alumnos de un sistema educativo planteen sus dudas, y éstas sean resueltas por otros alumnos o por los profesores que imparten la asignatura.

5.6.1. Elección y características

El entorno de educación elegido ha sido Moodle ya que en las fechas de realización de este proyecto, Moodle es el entorno de *e-learning* que da soporte a la Escuela Técnica Superior

[6]Moodle es un ambiente de educación virtual. http://moodle.org/

de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid (UPM).

La importancia que posee Moodle en la UPM[7] se detalla con las siguientes plataformas diseñadas para su uso:

- Titulaciones Oficiales: Plataforma que aloja los espacios virtuales de las asignaturas (troncales, obligatorias, optativas y de libre configuración) de los estudios conducentes a títulos oficiales de la UPM, de grado, postgrado y doctorado, en las modalidades a distancia (*e-learning*) y de apoyo a las enseñanzas presenciales (*b-learning*).

- Titulaciones Propias: Plataforma que aloja los espacios virtuales de los cursos y asignaturas de los estudios conducentes a títulos propios de la UPM, de grado o postgrado, en las modalidades e-learning o b-learning.

- Punto de Inicio: Plataforma que aloja los espacios virtuales de cursos de áreas básicas para el autoestudio y la autoevaluación de los alumnos de nuevo ingreso en estudios de las titulaciones oficiales de la UPM.

- Plan de Formación de la UPM: Plataforma que aloja las ofertas formativas del Plan de Formación de la UPM del su personal Docente e Investigador y de Administración y Servicios.

- Puesta a Punto: En este espacio web los profesores podrán encontrar los materiales que se ofrecen a los estudiantes para que en régimen de autoestudio puedan fortalecer la adquisición de competencias transversales y el desarrollo de determinadas capacidades que complementen su formación para el ejercicio profesional.

- ADA-Madrid: Plataforma que aloja los espacios virtuales de las asignaturas de libre configuración del convenio ADA-Madrid, para el intercambio de asignaturas de las seis universidades públicas de la Comunidad de Madrid (Alcalá, Autónoma de Madrid, Carlos III de Madrid, Complutense de Madrid, Politécnica de Madrid y Rey Juan Carlos.)

- Formación Externa: Plataforma que aloja las ofertas formativas destinadas a alumnos de programas de internacionalización o de cooperación para el desarrollo.

- Cursos de verano de la UPM: Plataforma de apoyo a la impartición de los cursos de verano de la UPM.

[7] http://moodle.upm.es/

5.6.2. Escenario propuesto

Con el propósito de mejorar el proceso de resolución de dudas así como la disminución del tiempo de respuesta entre alumnos y tutores, se propone la integración de bots conversacionales y del sistema QA, detallado en la presente memoria, dentro de la plataforma Moodle. Concretamente, en el módulo de foros.

Este escenario pretende, por un lado crear una primera fuente de resolución de dudas que sea capaz de dar respuesta a aquellas consultas más sencillas o frecuentes sobre el temario de una asignatura, gracias a las respuestas generadas por un bot conversacional o por un sistema QA, de manera transparente. Por otro lado, si alguno de los dos sistemas dispone de la respuesta, elimina la demora en los tiempos de respuesta de los foros de discusión. ya que la respuesta es instantánea.

La tecnología de bots conversacionales permite desarrollar sistemas capaces de mantener una conversación coherente en lenguaje natural sobre un tema determinado para lo cual han sido programados de antemano. En este escenario, esta capacidad se ve mermada ya que la única función habilitada es la de responder preguntas transparentemente.

5.6.3. Caso de estudio

El siguiente caso de estudio ilustra el escenario propuesto:

Alejandro es un alumno de del Grado en Ingeniería de Tecnologías de Redes y Servicios de Telecomunicación en la UPM. Comienza conociendo la plataforma Moodle ya que varias asignaturas hacen uso de ella y le surgen dudas concernientes a conceptos que se encuentran en el temario. Decide realizar sus consultas en los foros de discusión ya que es el lugar apropiado para no molestar al profesor.

Nada más crear el nuevo hilo en el foro, Alejandro observa que su consulta ya tiene una respuesta. Accede a ella y comprueba que un sistema artificial ha contestado a su pregunta y no tiene que esperar más tiempo. Sólo queda que el profesor correspondiente u otro alumno opinen sobre la respuesta dada.

5.6.4. Descripción de la solución

Para la realización de este estudio se ha desarrollado el sistema GSI QA, que proporciona explicaciones a conceptos concretos y se utiliza para aumentar la inteligencia del bot GSI Bot. Estos dos sistemas han sido integrados en la plataforma Moodle de tal manera que sean capaces de contestar las consultas de los alumnos, de forma transparente, a través de los foros de discusión.

1. Crear nuevo hilo: el alumno crea un nuevo hilo con su consulta en el foro de la asig-

natura correspondiente. Un ejemplo se muestra en la imagen 5.19.

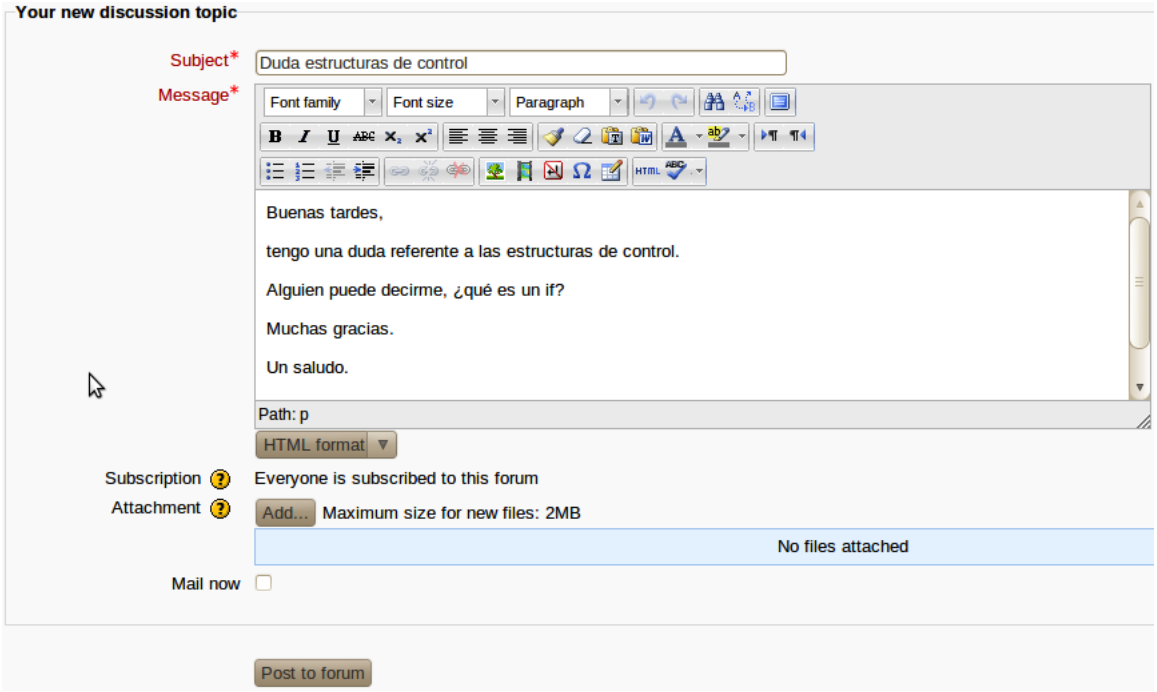

Figura 5.19: Ejemplo de consulta en el foro de Moodle

2. El bot escucha la pregunta: la pregunta realizada es enviada a un bot, llamado Tutor GSI. Si la respuesta generada por el bot está etiquetada sin errores, se crea un nuevo post (mensaje) de respuesta en el foro.

3. El sistema QA escucha la pregunta: si la respuesta generada por el bot contiene errores, la consulta es escuchada por el sistema QA, GSI QA. Si este sistema genera una respuesta positiva, se crea un nuevo post de respuesta en el hilo creado.

4. Si tanto la respuesta del bot como la del sistema QA contienen errores, no se hará nada.

En el caso de que se obtuviera alguna respuesta por parte de cualquiera de los sistemas, el alumno comprueba que nada más crear el hilo, ya tiene un primer mensaje de respuesta como muestra la figura 5.20.

Por último, se muestra, en la figura 5.21, la respuesta obtenida en el caso de que el sistema GSI QA sea el que conteste.

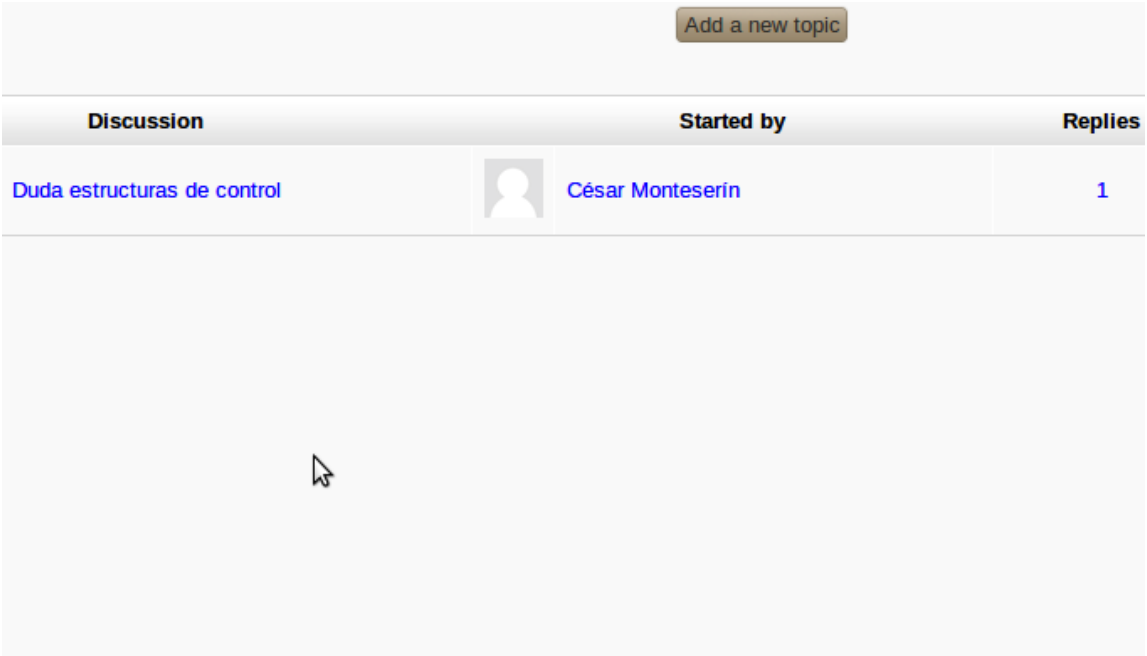

Figura 5.20: La consulta ha obtenido respuesta

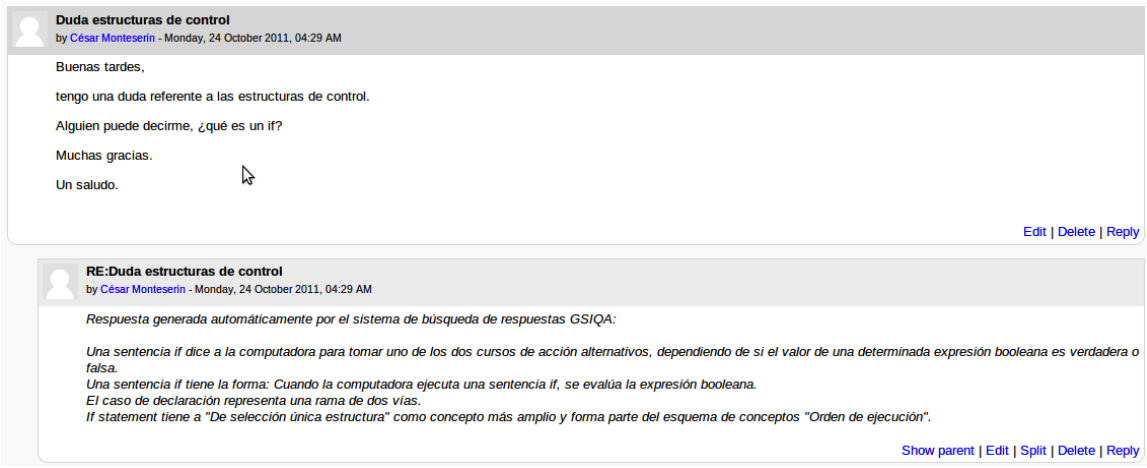

Figura 5.21: Ejemplo de contestación por parte del sistema GSI QA

Capítulo 6

Experimentación y validación

"El mundo exige resultados. No le cuentes a otros tus dolores del parto. Muéstrales al niño."

— Indira Gandhi

En la sección 5.4.1 se han presentado los diferentes patrones que se utilizan para recuperar información. En este capítulo se muestra el número de reglas o patrones obtenido de los archivos de entrenamiento, al mismo tiempo que los errores cometidos. Por otro lado, en la sección de validación, se presentan los resultados obtenidos a partir de dos libros de texto y usando diferentes tipos de patrones. Por último se comparan esos resultados con los de otros sistemas del estado del arte.

6.1. Experimentación

En la fase de experimentación se han utilizado los dos archivos de entrenamiento presentados en la sección 5.4.1.3. Éstos están formados por 500 frases de tipo definición y 1000 frases que no representan una posible definición. En este apartado, se presenta el número de patrones o reglas obtenidos de ellos, además de sus errores, para diferentes longitudes. Las diferentes longitudes representan los elementos vecinos escogidos para la creación de ellos. Un ejemplo de patrón de longitud tres (tres vecinos a cada lado del concepto, si es posible) es el siguiente:

Patrón de longitud 3	etiqueta etiqueta etiqueta CONCEPTO etiqueta etiqueta etiqueta

En el proceso de experimentación se han escogido longitudes desde un sólo vecino hasta cinco. En total, se dispone de treinta patrones ya que hay seis tipos de patrones con cinco longitudes diferentes.

6.1.1. Número de patrones

En este apartado se presenta la evolución del número de patrones extraídos a lo largo del incremento de información de los archivos de entrenamiento. Este estudio es de gran utilidad para saber si para cada longitud de los patrones, los archivos de entrenamiento han sido suficientes o no.

En la figura 6.1 se muestran los resultados de esta evolución para los seis tipos de patrones y para cinco longitudes diferentes. Las conclusiones de estos resultados, basadas en la pendiente de las distintas gráficas, se presentan en la tabla 6.1.

	L=1	**L=2**	**L=3**	**L=4**	**L=5**
SoftChunks	✓	✓	✗	✗	✗
Chunks	✓	✓	✗	✗	✗
SoftPost	✓	✗	✗	✗	✗
Post	✓	✗	✗	✗	✗
SoftLiteral	✗	✗	✗	✗	✗
Literal	✗	✗	✗	✗	✗

Tabla 6.1: Validez de un patrón por su evolución

En general, la conclusión de la tabla 6.1 y la figura 6.1 es que hay pocos tipos de patrones que no vayan a seguir aumentando en número con la adicción de nuevas definiciones, aunque

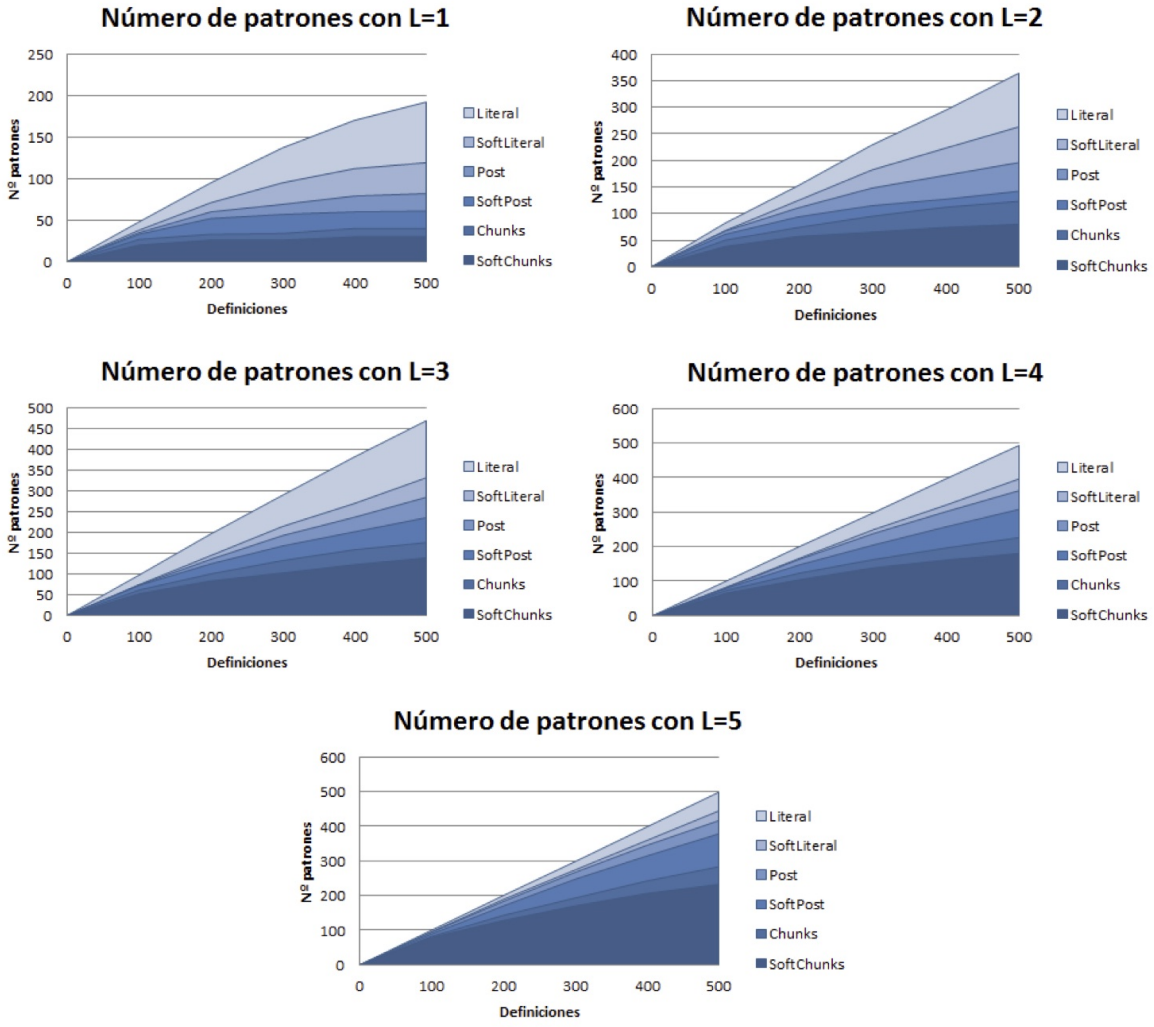

Figura 6.1: Número de patrones obtenidos para las distintas longitudes

muestren una cierta tendencia a suavizarse. Por ello, es conveniente que los archivos de entrenamiento sean mayores para que el aprendizaje de patrones sea mejor.

6.1.2. Número de errores

El número de errores proporciona datos sobre la eficiencia de cada tipo de patrón. Si un tipo de patrón comete muchos errores, quiere decir que su sensibilidad o *recall* es muy alta (recupera de todo) pero que su precisión es muy baja. En cambio si un patrón no comete prácticamente errores, quiere decir que su precisión es muy alta porque de las definiciones recuperadas, la mayoría son correctas. En este proyecto se ha decidido que un tipo de patrón es preciso cuando el número de errores es menor al 5 %.

Figura 6.2: Número de errores obtenidos agrupados por longitud

Se presentan los resultados de los archivos de entrenamiento desde dos puntos de vista diferentes. El primero de ellos (figura 6.2) presenta el número de errores agrupando los diferentes patrones según la longitud de ellos. Se comprueba que para la longitud igual a dos, comienzan a ser útiles algunos tipos de patrones. El segundo de ellos (figura 6.3) presenta el número de errores agrupando los diferentes patrones según el tipo. Se comprueba que la precisión de cada tipo de patrón aumenta con el incremento de su longitud.

Se concluye este apartado de la misma manera que el anterior, mostrando, en este caso, los patrones que se consideran válidos para el requisito de obtener un número de errores menor al 5 %. Para el archivo de entrenamiento utilizado, el número de errores tiene que ser menor a 50. La tabla de validación se presenta en la tabla 6.2.

Comparando las tablas 6.1 y 6.2, se comprueba que con los archivos de entrenamiento disponibles, no hay ningún tipo de patrón que cumpla los dos requisitos. En cambio, estos

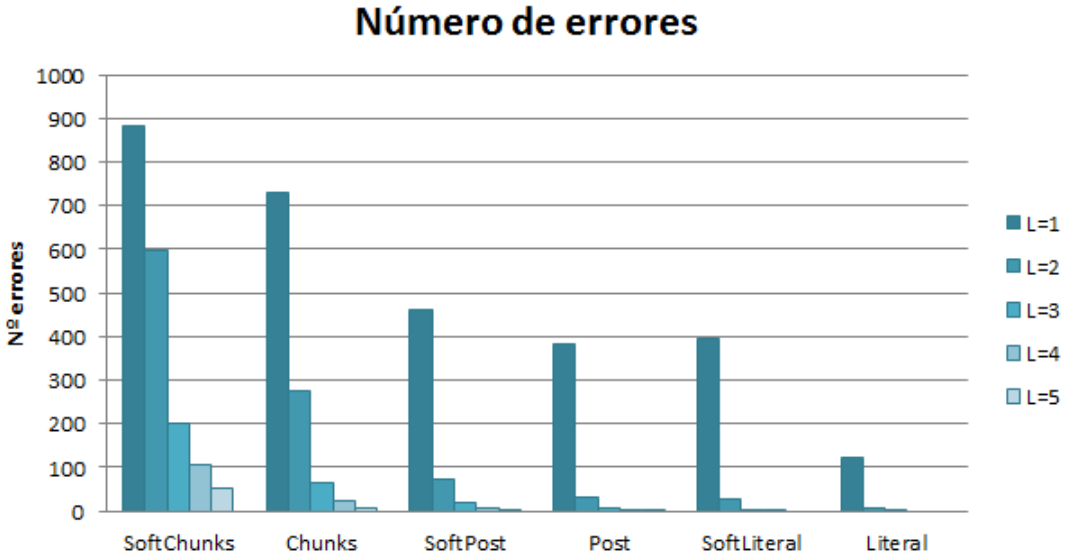

Figura 6.3: Número de errores obtenido agrupados por tipo

	L=1	L=2	L=3	L=4	L=5
SoftChunks	✗	✗	✗	✗	✗
Chunks	✗	✗	✗	✓	✓
SoftPost	✗	✗	✓	✓	✓
Post	✗	✓	✓	✓	✓
SoftLiteral	✗	✓	✓	✓	✓
Literal	✗	✓	✓	✓	✓

Tabla 6.2: Validez de un patrón por su precisión

resultado han reafirmado las suposiciones hechas a priori: los tipos de patrones con un menor número de etiquetas y vecinos poseen una gran sensibilidad, y los que poseen un mayor número de etiquetas y vecinos poseen una gran precisión.

6.2. Validación

Las pruebas de validación en la ingeniería de software son el proceso de revisión de que el sistema de software producido cumple con las especificaciones y que cumple su cometido. A su vez, la validación es el proceso de comprobar lo que se ha especificado es lo que el usuario realmente quería. En el caso del sistema desarrollado en esta memoria, al ser un proyecto de investigación, se comprueban cuáles han sido los resultados en términos de efectividad.

Como se ha mencionado al comienzo del capítulo, se ha tratado de validar los patrones aprendidos con dos libros de texto diferentes, tanto en su temática como en su longitud. Se tendrán en cuenta las diferentes características de evaluación de la recuperación y extracción de información, en el ámbito de las definiciones: definiciones correctas, definiciones incorrectas, definiciones no detectadas, precisión, sensibilidad o *recall* y factor F o *F-score*.

A lo largo de todas las mediciones se ha definido una definición como una frase que contiene el término que se define y la propia definición o explicación del concepto a través de sus propiedades o cualidades. Todas las frases que no posean esta estructura han sido descartadas y no han sido considerados en los resultados.

Un último aspecto a tener en cuenta es que, obviamente, al no tener los mismos datos de prueba las comparativas son complicadas, y se muestran para evaluar el sistema, pero sería necesario poder evaluar todos los sistemas con las mismas pruebas.

6.2.1. Libro de texto sobre Java

El primero de los libros en el que se ha procedido a la validación de los patrones aprendidos, consiste en un libro de texto sobre Java codificado en HTML. Gracias al uso de la técnica de división de frases u oraciones de la herramienta OpenNLP, se obtienen un total de 13.505 oraciones. Actuando como se ha detallado en el capítulo 5 de diseño, se han obtenido los resultados basados en dos tipos de patrones. En primer lugar, se muestran los resultados logrados por los patrones Post con longitudes que varían entre dos y cinco y que ofrecen una combinación perfecta entre precisión y sensibilidad en las tablas 6.3 y 6.4.

En segundo lugar, se muestran los resultados conseguidos con los patrones Chunk en las tablas 6.5 y 6.6. Se comprueba que este tipo de patrón es menos preciso y que su sensibilidad también es menor.

Definiciones manuales	149
Definiciones automáticas	268
Aciertos	86
Errores	182

Tabla 6.3: Resultados (Post) libro Java (1)

No detectadas	63
Precisión	32.09 %
Recall	57.72 %
F-score	41.25 %

Tabla 6.4: Resultados (Post) libro Java (2)

Definiciones manuales	149
Definiciones automáticas	300
Aciertos	52
Errores	248

Tabla 6.5: Resultados (Chunk) libro Java (1)

No detectadas	97
Precisión	17.33 %
Recall	34.89 %
F-score	23.16 %

Tabla 6.6: Resultados (Chunk) libro Java (2)

Por último, hay que tener en cuenta que recuperar definiciones a partir de libros de programación, es tremendamente complicado por las peculiaridades de éstos. Se tiende a explicar los conceptos a través de ejemplos, por lo que en muchas ocasiones, el sistema interpreta como definiciones a esos ejemplos.

6.2.2. Capítulo de introducción a la Inteligencia Artificial

Para la segunda prueba de validación se ha escogido un capítulo de introducción a un libro de inteligencia artificial. Este capítulo tiene como objetivo ser un punto de partida para todos aquellos que deseen comenzar a aprender ese campo, y por ello, tiene un gran número de definiciones. En este caso, se ha hecho un estudio exhaustivo de todos los tipos de patrones. El capítulo consta de 533 oraciones extraídas por el divisor de frases de la herramienta OpenNLP de las que 67 son definiciones. Un número elevado de definiciones pero que, según un estudio realizado por la Universidad de Amsterdam en la búsqueda de respuestas para el campo de la medicina [2], el porcentaje de definiciones en un texto de ese campo es del 18 %. Este dato dota de gran importancia a las definiciones dentro de libros enciclopédicos o científicos.

Los diversos tipos de patrones evaluados para este capítulo de texto han sido los seis tipos creados en la sección 5.4.1 y tres más:

- PostL2L5: según la experimentación realizada, este patrón es el mejor combinando la

precisión y la sensibilidad.

- Todos: todos los tipos de patrones.
- Válidos: todos los tipos de patrones validados según su precisión, a partir de la experimentación.

En la figura 6.4 se muestran los resultados de cada tipo de patrón para las tres medidas del campo de la recuperación de información.

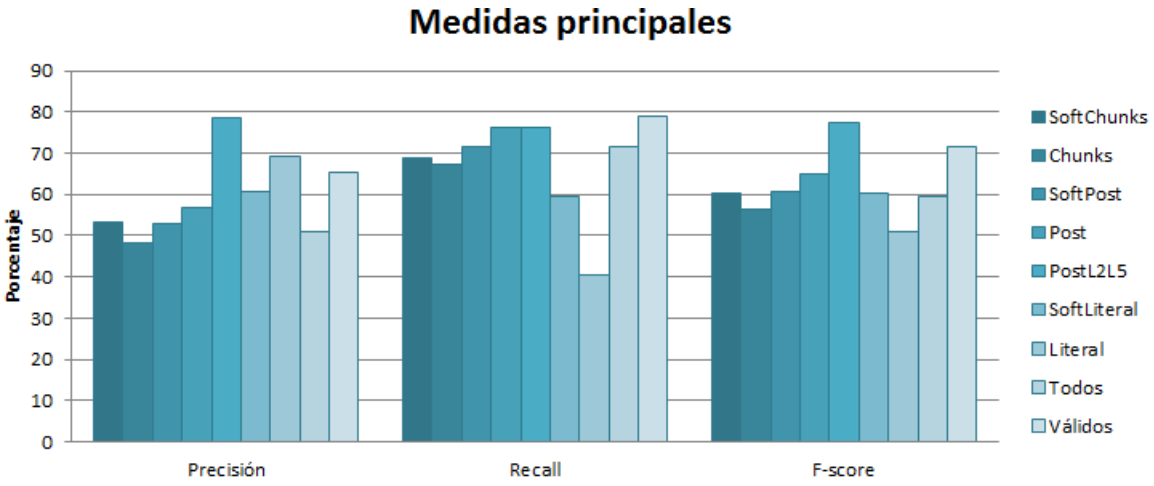

Figura 6.4: Medidas principales sobre el capítulo de IA

Por otro lado, en la figura 6.5 se muestran los resultados de cada tipo de patrón para otras medidas que también son relevantes.

6.2.3. Comparativa de resultados

Todo estudio realizado con unos resultados obtenidos, necesita una comparación con el resto de sistemas del estado del arte. Con los datos extraídos de diferentes artículos se presentan varias comparativas con diferentes valores de la constante β dentro del factor F o *F-score*. Es preciso recordar que el *F-score* tiene su valor máximo en 1 y su mínimo en 0. A lo largo de las comparativas, se han escogido los mejores valores del sistema descrito en esta memoria.

La primera comparativa se realiza con un sistema desarrollado en la Universidad de Lisboa [33]. Esta es la comparativa que más fielmente puede calificar al sistema GSI QA ya que, este sistema considera que una definición es una frase donde se encuentran el concepto y su definición unidos por el verbo ser. Tiene como características principales que ha sido probado tanto para el portugués y el alemán y utiliza para ello varios algoritmos del aprendizaje de máquinas o *machine learning*.

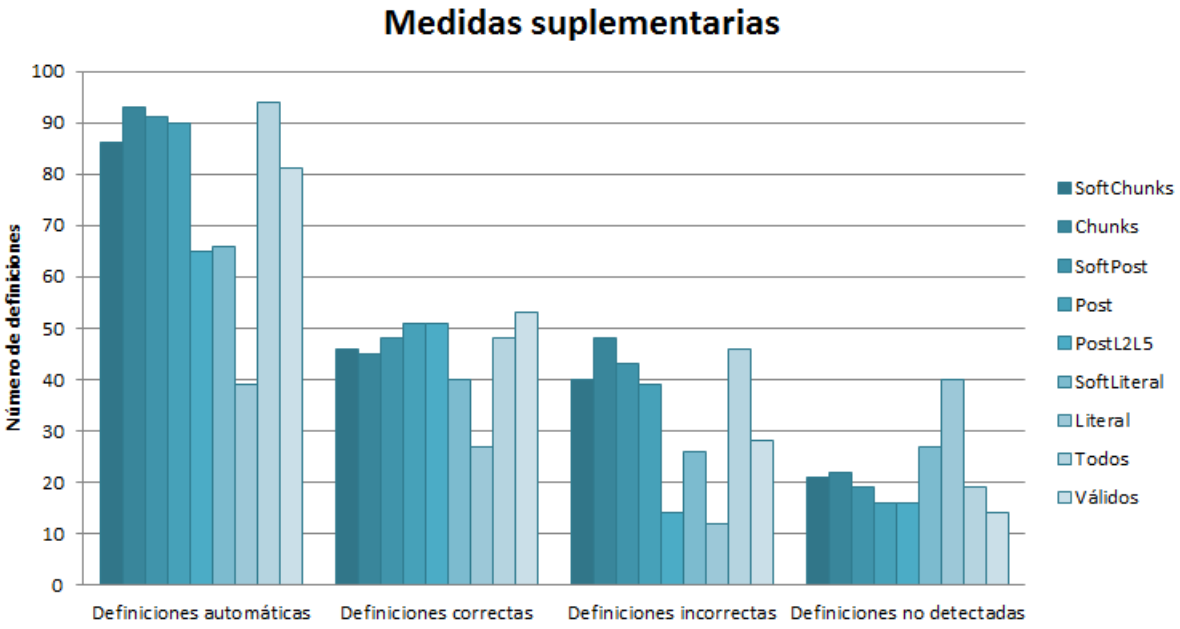

Figura 6.5: Medidas suplementarias sobre el capítulo de IA

En la tabla 6.7 se muestran los resultados de esta comparativa (escogiendo los mejores valores de la Universidad de Lisboa), concluyendo que el sistema de esta memoria es levemente peor. Varias de las posibles causas es porque en este proyecto el concepto de definición es más amplio y los idiomas son diferentes al igual que los algoritmos.

Sistema	U. Lisboa (Portugués)	U.Lisboa (Alemán)	GSI QA (Inglés - Java)	GSI QA (Inglés - IA)
F_1 **score**	0.77	0.95	0.56	0.751

Tabla 6.7: Comparativa del F-score con β=1

La segunda comparativa se realiza con el sistema OpenEphyra (estudiando en la sección 2.4.2) y con un sistema de la Universidad de Singapur [15]. OpenEphyra es un sistema centrado en las preguntas que esperan un nombre como respuesta, por lo que los resultados en el área de las definiciones son bajos. La Universidad de Singapur utiliza (como en este proyecto) una etapa de aprendizaje de patrones pero utilizando un gran número de heurísticos por lo que los patrones resultantes son diferentes a los del sistema de esta memoria.

La tabla 6.8, muestra (para β igual a 3) que los dos sistemas de la comparativa tienen peores resultados. Una posible razón es que estos resultados son los correspondientes a diversos concursos TREC, donde las posibles definiciones se pueden encontrar en frases sin el concepto. Esto sucede al utilizar pronombres en vez del nombre del concepto. Esta puntualización no ha sido considerada en nuestro sistema.

Sistema	OpenEphyra	U. Singapur	GSI QA (Java)	GSI QA (IA)
F_3 **score**	0.189	0.463	0.534	0.753

Tabla 6.8: Comparativa del F-score con β=3

La última comparativa se ha llevado acabo con el sistema del MIT, presentado en la sección 2.4.4. Como se analizó en esa sección, el sistema extrae definiciones con patrones fijos. Este sistema también obtiene peores resultados que GSI QA y una de las posibles causas, además de los algoritmos, es que estos resultados corresponden al concurso TREC donde el concepto de definición es distinto al nuestro como se ha comentado anteriormente. La comparativa se muestra en la tabla 6.9, donde se ha toma β igual a 5, es decir, dando cinco veces más importancia a la sensibilidad que a la precisión.

Sistema	MIT	GSI QA (Java)	GSI QA (IA)
F_5 **score**	0.309	0.56	0.751

Tabla 6.9: Comparativa del F-score con β=5

Capítulo 7

Conclusiones y trabajos futuros

"Estudia el pasado si quieres pronosticar el futuro."
— Confucio

En este capítulo se evalúan los resultados conseguidos y se exponen las conclusiones obtenidas tras la realización del proyecto. Además se presentan diferentes líneas de trabajo orientadas a mejorar el sistema de búsqueda de respuestas desarrollado en este proyecto.

7.1. Conclusiones

El proceso de maduración del procesamiento del lenguaje natural continúa a buen ritmo aunque todavía quede un largo camino por recorrer. Por esta razón en especial y otras más en particular, los sistemas de búsqueda de respuestas necesitan bastante trabajo hasta poder llegar a ser funcionales al 100 %.

Respecto al sistema desarrollado en la presente memoria, se han diseñado e implementando satisfactoriamente todas las etapas que componen un sistema de búsqueda de respuestas: análisis de la pregunta, elaboración de la consulta, recuperación de información y selección de la respuesta.

En cuanto a los resultados obtenidos en el ámbito de la recuperación de definiciones, como en todo sistema del estado del arte, no se puede considerar que, a priori, sea apto para su uso docente. Por otro lado, se destaca que el sistema obtiene resultados aceptables comparado con otros sistemas estudiados, con la consideración de que una definición necesita contener al término que define. Como aspecto negativo, el sistema obtiene resultados pobres al trabajar con textos en los que se utilicen constantemente ejemplos para explicar un concepto sin especificar, con alguna partícula del lenguaje, que se está refiriendo a un ejemplo concreto y no al concepto general.

Dentro de la estrategia del aprendizaje de patrones o *pattern learning* se ha comprobado que los patrones generados a partir de etiquetas de Part-of-Speech son los más equilibrados y con mayor efectividad final. También se concluye que el tamaño de los archivos de entrenamiento necesita ser mucho mayor para obtener unos resultados más fiables.

La integración del sistema QA, GSI QA y el bot TutorGSI, en el entorno de educación Moodle proporciona un valor añadido a los foros de discusión eliminando el tiempo de espera de respuesta a consultas donde se espera la explicación de un concepto.

7.2. Trabajos futuros

En toda línea de investigación, hay margen para la mejora y la continuación del trabajo realizado. En esta sección se plantean tanto posibles mejoras a los módulos desarrollados como las posibles líneas futuras de este proyecto.

Entre las posibles mejoras de los módulos desarrollados se encuentran:

- Comprensión de la pregunta: esta etapa del sistema es muy precaria y sólo entiende ciertos tipos de preguntas. Una posible mejora es añadir un número mayor de patrones de preguntas e integrar la etapa del análisis de la pregunta del sistema OpenEphyra (sección 2.4.2).

- Eliminar la dependencia con el traductor de Google: ahora mismo el sistema trabaja con el inglés y se puede hacer uso de él en español gracias al traductor de Google. Esta dependencia se podría eliminar cambiando las herramientas NLP utilizadas por Freeling.
- Mayor inteligencia en la búsqueda de información: ahora mismo el sistema trabaja con la técnica *pattern learning* para este fin. Para mejorar esta etapa se podrían crear nuevos patrones de aprendizaje mejores a los actuales, o bien, aplicar algún algoritmo de *machine learning* presentados en la sección 2.3.1.
- Puntuación de las respuestas: las posibles definiciones se puntúan a partir de la media de las puntuaciones que posean los patrones con los que coincide. Este algoritmo es mejorable, por ejemplo, dando mayor puntuación a los patrones de mayor longitud.
- Aumento de los archivos de entrenamiento: el sistema mejorará su eficiencia con un mayor corpus de entrenamiento. Por ello se recomienda conseguir los corpus AQUAINT[1] y AQUAINT2[2] para un entrenamiento con millones de documentos.
- Integración de las respuestas en el bot, tras un proceso de validación, para dotarle de mayor inteligencia y haga uso de las definiciones aprendidas.
- Mejorar la integración de la herramienta realizada en entornos de *e-learning*, para facilitar el trabajo a profesores y alumnos.
- Desarrollar un sistema de evaluación automática para evaluar la comprensión de conceptos.

[1] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T31
[2] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T25

Bibliografía

[1] Charles L A Clarke, Nick Craswell, and Ian Soboroff. Overview of the TREC 2009 Web Track, 2009.

[2] Erik Tjong, Kim Sang, Gosse Bouma, and Maarten De Rijke. Developing Offline Strategies for Answering Medical Questions IMIX : Interactive Multi-Modal Information Extraction. *Time*, 2005.

[3] Alejandro Marqués Rodríguez. Desarrollo de un bot de ayudaen lenguaje aimlcon acceso web y wap., 2009.

[4] Miguel Coronado Barrios. Desarrollo de una plataforma para la asistencia en el desarrollo y gestión de sistemas de bots., 2010.

[5] D Ferrucci, E Nyberg, J Allan, K Barker, E Brown, J Chu-Carroll, A Ciccolo, P Duboue, J Fan, D Gondek, and Eduard Hovy. Towards the Open Advancement of Question Answering Systems. *IBM Research Report RC24789 W0904093 IBM Research New York*, 24789, 2009.

[6] Vicedo Jos and Diego Moll. Open-Domain Question-Answering Technology: State of the Art and Future Trends ´, 2001.

[7] Wikipedia. Information Retrieval. http://en.wikipedia.org/wiki/Information_retrieval.

[8] Wikipedia. Information Extraction. http://en.wikipedia.org/wiki/Information_extraction.

[9] P. D. Lepore. Q&A on QA, September 1996.

[10] Dell Zhang and Wee Sun Lee. Question classification using support vector machines. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03*, page 26, 2003.

[11] Wikipedia. Machine learning. http://en.wikipedia.org/wiki/Machine_learning.

[12] Wikipedia. Natural Language Processing. http://en.wikipedia.org/wiki/Natural_language_processing.

[13] List Questions. QA at BBN: Answering Definitional Questions. *TREC2003*, 2003.

[14] Sasha Blair-goldensohn, Kathleen R Mckeown, and Andrew Hazen Schlaikjer. A Hybrid Approach for Answering Definitional Questions. *Program*, 2000.

[15] Hang Cui, Min-Yen Kan, and Tat-Seng Chua. Soft pattern matching models for definitional question answering. *ACM Transactions on Information Systems*, 25(2):8–es, April 2007.

[16] Jesús Fernández Benito. Sistema de question answering basado en wikipedia., 2006.

[17] G Sautter, N Schlaefer, and P Gieselmann. The Ephyra QA system at TREC 2006. *Area*, 2006.

[18] Nico Schlaefer. Pattern Learning and Knowledge Annotation for Question Answering. *Knowledge Creation Diffusion Utilization*, (September), 2005.

[19] Nico Schlaefer. Deploying Semantic Resources for Open Domain Question Answering. *Components*, (June), 2007.

[20] David Ferrucci, Eric Brown, Jennifer Chu-carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, pages 59–79, 2010.

[21] Eric Nyberg and Robert Frederking. JAVELIN: A Flexible, Planner-Based Architecture for Question Answering. *Architecture*, (June):19–20, 2003.

[22] Stuart Russell and Peter Norvig. Artificial Intelligence, A Modern Approach - 2nd Edition, 2003.

[23] Min-yuh Day, Chun-hung Lu, Jin-tan David Yang, Guey-fa Chiou, Chorng-shyong Ong, and Wen-lian Hsu. Designing an Ontology-based Intelligent Tutoring Agent with Instant Messaging. *Integration The Vlsi Journal*, pages 4–6, 2005.

[24] Yuan K Shen and Boris Katz. Answering Definition Questions. *Artificial Intelligence*, (c):1–4.

[25] Wesley Hildebrandt, Boris Katz, and Jimmy Lin. Answering Definition Questions Using Multiple Knowledge Sources. *Artificial Intelligence*.

[26] Juan Antonio Pastor Sánchez. Diseño de un sistema colaborativo para la creación y gestión de tesauros en Internet basado en SKOS. 2009.

[27] OpenNLP. OpenNLP: Herramienta NLP de código abierto escrita en Java. http://sourceforge.net/apps/mediawiki/opennlp/index.php?title=Main_Page#OpenNLP_Documentation.

[28] Xavier Carreras and Isaac Chao. FreeLing: An Open-Source Suite of Language Analyzers. *Management*, 1998.

[29] Miquel Collado, Samuel Reese, and Marina Lloberes. FreeLing 2.1: Five years of open-source language processing tools. *Language*, 2004.

[30] Thorsten Brants. TnT – A Statistical Part-of-Speech Tagger. (i):224–231, 1999.

[31] Ming-che Lee, Ding Yen Ye, and Tzone I Wang. Java learning object ontology. *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, pages 538–542, 2005.

[32] Final Report, Carl Chang, Francis Lau, and Pradip Srimani. Computing Curricula 2001 Computer Science. *Education*, (0003263), 2001.

[33] Rosa Del and Gaudio Ant. Language Independent System for Definition Extraction : First Results Using Learning Algorithms. pages 33–39, 2009.

[34] Beatriz Santorini. Part-of-speech tagging guidelines for the penn treebank project, 1990.

Apéndice A

Etiquetas NLP

En el desarrollo de este proyecto han sido esenciales las etiquetas de las diferentes técnicas de procesamiento del lenguaje natural y por ello, se van a presentar en este apéndice. En concreto, las etiquetas para el inglés. Todas las etiquetas tienen su origen en el proyecto Penn Treebank, el cual tiene como propósito el etiquetado de lenguaje natural de la estructura lingüística a través de unas etiquetas predefinidas que se presentan en las siguientes secciones.

A.1. Etiquetas del Part-of-Speech

Las etiquetas del Part-of-Speech o PoS se asignan a una sola palabra de acuerdo con su rol en la oración[34]. Los clasificadores gramaticales tradicionales estaban basados en ocho etiquetas. En cambio, a día de hoy, los sistemas trabajan con un número mayor de etiquetas que les permite ser más específicos. Las diferentes etiquetas, originadas del proyecto Penn Treebank[1], se presentan en las tablas A.1 y A.2.

Etiqueta	**Función**
CC	Conjunción
CD	Nombre cardinal
DT	Determinante
EX	*There* existencial
FW	Palabra extranjera
IN	Preposición
JJ	Adjetivo
JJR	Adjetivo comparativo
JJS	Adjetivo superlativo
LS	Marcado de listas
MD	Verbo modal auxiliar
NN	Nombre singular
NNS	Nombre plural
NNP	Nombre propio
NNPS	Nombre propio plural
PDT	Predeterminante
PRP	Pronombre personal
PRP$	Pronombre posesivo
RB	Adverbio
RBR	Adverbio comparativo

Tabla A.1: Etiquetas de PoS I

Etiqueta	**Función**
RBS	Adverbio superlativo
RP	Partícula adverbial
SYM	Símbolo
TO	To
UH	Interjección
VB	Verbo en infinitivo
VBZ	Verbo en 3ªp del pres. singular
VBP	Verbo en pres. singular (no 3ª)
VBD	Verbo en pasado
VBN	Verbo en participio pasado
VBG	Verbo en gerundio o p.presente
WDT	Wh-determinante (which...)
WP	Wh-pronombre personal (what...)
WP$	Wh-pronombre posesivo (whose...)
WRB	Wh-adverbio (where...)
.	Marca de final de oración
,	Coma
:	Dos puntos
(	Separador izquierdo
)	Separador derecho

Tabla A.2: Etiquetas de PoS II

[1] http://www.cis.upenn.edu/ treebank/

A.2. Etiquetas del Chunking

Las etiquetas del *Chunking*, los *chunks*, definen a qué tipo de predicado, sintagma o frase pertenece una palabra o un grupo de ellas. Hay dos tipos de etiquetas en este caso, las etiquetas que nos dicen a que predicado pertenecen y aquellas que nos dicen en que posición. Estas etiquetas, también originadas del proyecto Penn Treebank, se muestran en las tablas A.3 y A.4, respectivamente.

Etiqueta	**Función**
NP	Predicado nominal
PP	Predicado preposicional
VP	Predicado verbal
ADVP	Predicado adverbial
ADJP	Predicado adjetival
SBAR	Subordinada
PRT	Partícula
INTJ	Interjección
PNP	Predicado nominal preposicional

Tabla A.3: Etiquetas de Chunking de función

Etiqueta	**Función**
I-	Dentro del chunk
B-	Primero del chunk
O	No es parte del chunk

Tabla A.4: Etiquetas de Chunking de posición

Apéndice B

Manual de desarrollo

En este manual se detallan los componentes necesarios tanto para la instalación del sistema de la presente memoria como para proseguir con su desarrollo. Al ser un proyecto desarrollado en Java, se describe su instalación y la del entorno de desarrollo elegido. Se citan los pasos a seguir para conseguir que el sistema sea funcional para cualquier tipo de documento y para poder mejorar la eficiencia del sistema. Por último se detalla su integración con el entorno educativo Moodle.

B.1. Puesta a punto del entorno de desarrollo

Para trabajar en el proyecto es necesario instalar el kit de desarrollo de Java y un entorno de desarrollo integrado que soporte Java. En esta sección se indican los pasos a seguir para la instalación de Java y el entorno Eclipse en el sistema operativo Linux.

B.1.1. Instalación del kit de desarrollo de Java

En el sistema operativo Linux, es habitual que el kit de desarrollo de Java esté instalado por defecto. Para comprobarlo, se ejecuta el comando:

```
java -version
```

En el hipotético caso de que no encuentre la versión de Java, existe la posibilidad de instalar tanto el JRE como el JDK mediante el siguiente comando:

```
sudo apt-get install sun-java6-bin, sun-java6-jre, sun- java6-jdk
```

Es recomendable tener instalada la versión 6 de Java, de manera que si la versión encontrada es inferior, se recomienda ejecutar el comando de anterior pero con la opción `--reinstall`.

B.1.2. Instalación del entorno de desarrollo

El entorno de desarrollo elegido ha sido Eclipse, aunque podría haber sido cualquier otro como Netbeans. La instalación del entorno es verdaderamente sencilla haciendo uso del comando de instalador de paquetes:

```
sudo apt-get install eclipse
```

Este comando descarga los paquetes requeridos y los instala en el ordenador, pero Eclipse no tendrá un funcionamiento óptimo puesto que estará haciendo uso de la versión GNU de Java en lugar de la oficial de Sun, para corregirlo, se ejecuta el siguiente comando:

```
sudo update-java-alternatives -s java-6-sun
```

A continuación se edita el archivo de configuración JVM, `sudo -b gedit /etc/jvm` y finalmente se agrega la siguiente línea al principio del archivo: `/usr/lib/jvm/java-6-sun`.

B.2. Haciendo funcional al sistema

El sistema GSI QA es capaz de trabajar sea cual sea la temática elegida. En esta sección se presentan los pasos a seguir para mejorar su eficiencia, para elegir la temática de la que se quieren encontrar explicaciones de conceptos, para que el sistema sea capaz de responder a preguntas de los usuarios y para comprobar la eficiencia sobre el libro seleccionado.

B.2.1. Mejorar la eficiencia

A parte del umbral elegido (por debajo del cual una posible definición no será considerada como tal), la eficiencia del sistema reside en los archivos de entrenamiento. Estos archivos de entrenamiento se dividen en conjuntos de frases de tipo definición y en frases que no representan una definición.

En el momento de la entrega de esta memoria los archivos están compuestos por 500 frases de tipo definición y 1000 que no lo son. Se presupone que este número de frases para el entrenamiento es muy reducido ya que, por ejemplo, los traductores hacen uso de millones de casos para obtener una efectividad razonable.

Las distintas oraciones de los archivos de entrenamiento tienen etiquetados el concepto que se define, en el caso de las definiciones. En el otro caso, tienen etiquetadas un concepto que no se está definiendo (sólo se etiquetada un concepto por frase). Se muestra en la tabla B.1 (ya presentada en el capítulo 5 de diseño) un ejemplo del entrenamiento. El paquete `PatternExtractor` es el encargado de generar los nuevos patrones con sus respectivas eficiencias.

Definición	<T>Death<T>is very likely the single best invention of life.
No definición	Death is very likely the single best invention of <T>life<T>.

Tabla B.1: Muestra de los archivos de entrenamiento

B.2.2. Etiquetado del documento

Con el fin de etiquetar un nuevo libro, y por lo tanto extraer información de él, es indispensable la elaboración de un tesauro sobre el dominio del libro y disponer del libro en cuestión.

El paquete encargado de extraer el texto plano es el paquete `documentManager`. Si no se hace ninguna mejora sobre él, se recomienda el uso de un libro en HTML o directamente en texto plano, por lo problemas citados en la memoria con documentos PDF.

Por otro lado, el paquete encargado etiquetar el libro es el paquete `annotator`. A través de una clase sencilla se utilizan tanto las clases del `documentManager` como las del `annotator`

para obtener el libro con los conceptos y definiciones etiquetadas.

B.2.3. Aplicación semántica

Esta es la parte del sistema que es verdaderamente funcional. El paquete `qa` es el encargado de comprender si la respuesta del usuario es de tipo definición, buscar la respuesta y presentarla. El archivo de patrones de preguntas de tipo definición se puede aumentar para que el sistema comprenda un mayor número de formas en las que se puede preguntar por la definición de un concepto.

Nota importante: el traductor de Google, utilizado para que el sistema sea capaz de trabajar en español, tiene un número de consultas gratuitas limitadas. Ha pasado a ser un servicio de pago.

B.2.4. Pruebas

El sistema ya es totalmente funcional pero no se sabe la efectividad con la que trabaja. Para ello se hace uso del paquete `annotator.test` que compara un libro de texto plano con las definiciones etiquetadas automáticamente, con otro etiquetado manualmente.

Gracias a esto, se obtienen los factores de la recuperación de información presentados en la sección 2.1.1 (precisión, sensibilidad y F-score) y además el número de definiciones etiquetadas manualmente y automáticamente, el número de definiciones correctamente e incorrectamente etiquetadas y las definiciones no detectadas.

B.3. Integración con Moodle

En esta sección se describe la instalación de Moodle 2.0.3 y los cambios del código fuente de Moodle para integrar a un Bot y al sistema GSI QA en el módulo de foros de discusión.

B.3.1. Instalación

B.3.1.1. Instalación del entorno LAMP

Antes de instalar Moodle se hace fundamental la instalación del entorno LAMP (Linux, Apache, MySQL y PHP) ya que es necesario para la instalación de Moodle. Este entorno se puede instalar, en Ubuntu, a través de dos vías distintas. La primera de ellas, consiste en ejecutar el comando:

```
sudo apt-get install lamp-server^
```

El símbolo `^` es importante para indicar que se va a instalar un meta paquete, es decir, una tarea que incluye varios paquetes.

La segunda opción es a partir de la interfaz gráfica. En el gestor de paquetes Synaptic (Sistema>Administración>Gestor de paquetes Synaptic), se selecciona el el menú de editar, la pestaña "Marcar Paquetes por Tarea". Por último se selecciona LAMP Server y se hace click en el botón Aceptar del cuadro de diálogo mostrado en la imagen B.1.

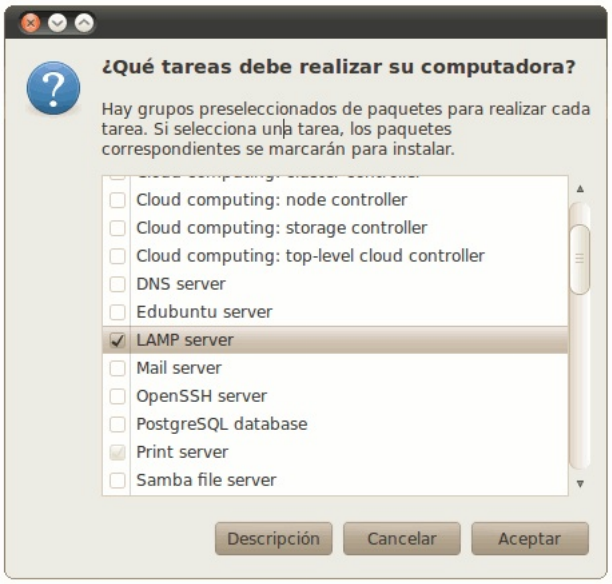

Figura B.1: Instalación del entorno LAMP

B.3.1.2. Instalación de Moodle 2.0.3

Una vez instalado el entorno LAMP, se está en disposición de instalar la plataforma Moodle. El primer paso es descargarse el paquete desde http://download.moodle.org/. En este proyecto, en concreto, se ha trabajado con la versión 2.0.3.

A continuación se debe crear adecuadamente la base de datos. Primero, se ejecuta MySQL desde un terminal y escribimos la contraseña de `root`:

```
mysql -u  root -p
```

Para la creación de la base de datos, se escribe el siguiente `script` en la consola de MySQL:

```
CREATE DATABASE moodle DEFAULT character SET utf8;
GRANT ALL privileges ON moodle.* TO 'moodleuser'@'localhost'
IDENTIFIED BY 'yourpassword';
FLUSH privileges;
quit
```

Tras este paso, se tiene que copiar la carpeta de Moodle descomprimida en la carpeta /var/www del sistema de archivos. Se necesita privilegios de superusuario, por lo que se

procede a ejecutar en un terminal `sudo nautilus` para conseguir un navegador de archivos con permisos `root`.

La instalación de Moodle se hace desde la dirección `http://localhost/moodle`. Al realizar la configuración se muestra un archivo `config.php` que se debe copiar en la carpeta de instalación de Moodle. Se acepta la licencia, y se comienza con la configuración de la base de datos. En este paso se ahorra tiempo si se pincha sobre la opción *unnatended operation*.

Por último, se pide crear un usuario administrador rellenando varios campos y se escoge el nombre para el Moodle.

B.3.2. Integración

Para la integración del bot GSI Bot y del sistema de búsqueda de respuestas GSI QA en Moodle, se ha modificado su código fuente. La modificación se ha realizado en el módulo de foros de discusión, por lo que los cambios se han llevado a cabo en la carpeta /moodle/mod/forum.

En concreto se ha añadido un archivo JSON.php para poder trabajar con el formato de intercambio de datos JSON y se ha modificado el archivo post.php. El archivo post.php es el encargado de generar respuestas en el foro, por lo que se ha modificado el método donde se crea un nuevo hilo en el foro para que antes de presentar la consulta, se cree un mensaje con la respuesta del Bot (alojado en `http://minsky.gsi.dit.upm.es/gsiBot`) o del sistema QA (alojado en Tomcat).

Apéndice C

La importancia de los patrones

Desde hace miles de años, el ser humano ha intentado, y en ocasiones con éxito, encontrar los patrones que rigen el universo. Un patrón se define como un tipo de sucesos u objetos recurrentes, a veces referidos como elementos de un conjunto de objetos. Estos elementos se repiten de una manera predecible. Puede ser una plantilla o modelo que puede usarse para generar objetos o partes de ellos, especialmente si los objetos que se crean tienen lo suficiente en común para que se infiera la estructura del patrón fundamental, en cuyo caso, se dice que los objetos exhiben un único patrón.

En el campo de la inteligencia artificial, un patrón es un diseño de una técnica simple e independiente que puede ser utilizado en un desarrollo software para añadir un aspecto de inteligencia en la toma de decisiones, en la proyección o estimación de un suceso, en el reconocimiento de patrones o en la resolución de problemas.

Una vez que se ha introducido la definición de patrón, se presentan algunos casos donde el ser humano pone sus esfuerzos para reconocerlos y a día de hoy no se han encontrado. Sin

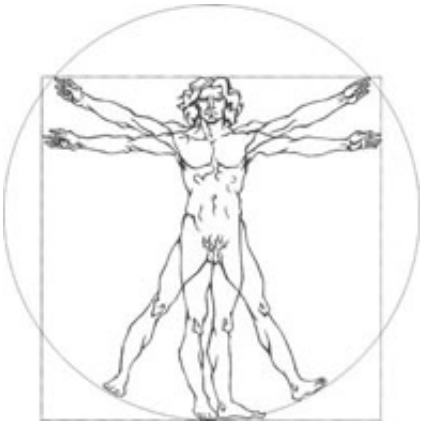

Figura C.1: Patrón o canon de la belleza

embargo, antes de presentarlos, se muestra en primer lugar uno de los casos más famosos: el patrón o canon de la belleza. En la imagen C.1 se muestra la ilustración "El hombre de Vitruvio" de Leonardo Da Vinci que data del año 1487 (más de 500 años atrás). Esta ilustración es el resultado de un estudio de este polifacético hombre sobre los patrones que rigen la belleza humana. La conclusión de ese estudio es que el cuerpo ideal de un ser humano se rige por la proporción áurea[1].

En el siglo XXI, diversos campos científicos continúan en la búsqueda de, por ejemplo, los siguientes patrones:

- Patrón de mutación del SIDA/VIH: durante el transcurso de la enfermedad del SIDA (Síndrome de Inmunodeficiencia Adquirida), el virus VIH (Virus de la Inmunodeficiencia Humana) sufre continuas mutaciones genéticas lo que hace tan difícil la creación de una vacuna que sea capaz de eliminarle. En el momento que se extraiga su patrón de mutación, se podrá elaborar un fármaco específico para su eliminación como sucede con otras enfermedades.

- Patrón de los números primos: los matemáticos de todo el mundo llevan muchos años

[1]También conocida como la divina proporción, se basa en relaciones áureas

$$\Phi = \frac{1+\sqrt{5}}{2} \tag{C.1}$$

detrás del patrón que siguen los números primos. A día de hoy su desconocimiento consigue que, por ejemplo, muchos algoritmos de criptografía sean seguros. En el caso concreto de las transacciones bancarias, a los números de la cuenta bancaria se le asocian varios números primos de gran tamaño que se operan entre sí consiguiendo que el número de la cuenta quede inaccesible.

Glosario de términos

AIML Artificial Intelligence Mark-up Language. Es un lenguaje de programación basado en XML especializado en la creación de agentes software con lenguaje natural.

Bot Nombre que recibe el programa que, conteniendo un intérprete de AIML, engloba un conjunto de archivos AIML, los parsea y almacena en memoria y acepta consultas a las que devuelve una respuesta.

Chunk Fragmento de información y es el término empleado para nombrar a los predicados en las técnicas NLP.

E-learning Educación a distancia vía electrónica.

F-score Medida que tiene en cuenta tanto la precisión como el recall en el ámbito de la recuperación de información.

FAQ Frequently Asked Questions. Preguntas más frecuentas en un campo determinado.

HTML HyperText Markup Language. Lenguaje de marcado de hipertexto.

Java Lenguaje de programación orientado a objetos creado por Sun Microsystems.

JLOO Java Learning Object Ontology. Ontología de Java cuyo objetivo es el aprendizaje de ese lenguaje.

Moodle Entorno educativo virtual.

N-grama Subsecuencia de n elementos (palabras) de una secuencia determinada

NER Named Entity Recognition. Reconocimiento de entidades del nombre.

NLP Natural Language Processing.

Patrón Diseño de una técnica simple e independiente que puede ser utilizado en un desarrollo software para añadir un aspecto de inteligencia en la toma de decisiones, en la proyección o estimación de un suceso, en el reconocimiento de patrones o en la resolución de problemas.

PDF Portable Document Format.

PHP PHP Hypertext Pre-processor.

PLN Procesamiento del lenguaje natural.

PoS Part-of-Speech. Función que toma una palabra en una oración.

Precisión Medida que determina el porcentaje de acierto de la información relevante recuperada sobre la recuperada.

QAS Question-Answering System. Sistema de búsqueda de respuestas.

Recall Medida que determina el porcentaje de acierto de la información relevante recuperada sobre la relevante.

SBR Sistema de búsqueda de respuestas.

SKOS Simple Knowledge Organization System.

Tesauro Listado de palabras o términos empleados para representar conceptos.

Thesaurus Tesauro.

Token En el campo NLP, son los elementos más básicos y pueden referirse a palabras, símbolos, etc.

TREC Concurso que proporciona la infraestructura necesaria para llevar a cabo pruebas de recuperación de información y que cuenta con la asistencia y participación de investigadores de todo el mundo para presentar sus propuestas más avanzadas.

UML Unified Modeling Language. Lenguaje unificado de modelado que permite especificar, construir y documentar un sistema mediante lenguaje gráfico.

El ingeniero D. César Monteserín Gutiérrez,

__ de _____de 2011