ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso académico 2010/2011

Proyecto Fin de Carrera

SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE ENTIDADES NOMBRADAS BASADO EN LINGPIPE

Autor: Daniel Álvarez Zorita

Tutores: Soto Montalvo Herranz
Eduardo García Pardo

Agradecimientos

Existe un viejo refrán español que reza que “es de bien nacido ser agradecido”, y viene a decir que agradecer es acordarse y corresponder con gratitud por un beneficio recibido. Esta enseñanza tan valiosa me la inculcaron, probablemente, mis padres en uno de sus muchos esfuerzos por educarme y aleccionarme en la vida, por lo que voy a agradecer a los que me han ayudado a lo largo de este camino que comenzó hace algún tiempo y que finaliza con este PFC.

En primer lugar, me gustaría agradecer a mis tutores Soto y Eduardo la paciencia que han demostrado conmigo. Gracias por haberme dado la oportunidad de realizar este PFC y os deseo lo mejor en vuestros proyectos e investigaciones futuras.

A mis padres, a los que jamás encontraré la forma de agradecer una vida de sacrificios, esfuerzos y amor. Gracias por el apoyo y el cariño que siempre me habéis brindado.

A mi abuela, por su ejemplo de lucha y honestidad, así como a toda mi familia, que siempre me ha alentado a lograr esta hermosa realidad.

A Esther, por su amor, paciencia y por ser donde mis fracasos y frustraciones se convierten en alegría, paz y serenidad.

Y como no, a mis compañeros y amigos, que a pesar de los malos momentos han hecho de este viaje una de las mejores experiencias de mi vida.

Un millón de gracias.

Dani

Resumen

El presente Proyecto de Fin de Carrera supone el colofón a los estudios de Ingeniería Técnica Informática de Sistemas, y tiene como objetivo el enfrentar al alumno con un sistema informático completo, como es el desarrollo de una aplicación informática.

Desde el nacimiento de la Inteligencia Artificial, se hizo necesaria una disciplina que investigara mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio de lenguajes naturales, y de esa idea surgió lo que hoy se conoce como Procesamiento del Lenguaje Natural.

Este Proyecto de Fin de Carrera se enmarca dentro del área del Procesamiento del Lenguaje Natural y, en concreto, aborda el estudio de una tarea muy importante dentro de éste, como es el Reconocimiento de Entidades Nombradas. En particular, esta rama del Procesamiento del Lenguaje Natural se encarga de identificar y clasificar elementos en un texto (denominados Entidades Nombradas) que se encuadren dentro de categorías predefinidas, como por ejemplo personas, organizaciones o lugares.

Para ello, en este Proyecto de Fin de Carrera se ha optado por extender la herramienta LingPipe (una librería Java para el reconocimiento de Entidades Nombradas), añadiendo determinadas heurísticas que mejoren los resultados de dicha herramienta. También se han incorporado nuevos idiomas a la misma mediante la creación manual de dos modelos para francés e italiano. El resultado es una aplicación capaz de identificar y clasificar Entidades Nombradas para inglés, castellano, francés e italiano. Además, se ha utilizado una API basada en Wikipedia para realizar traducciones automáticas de un idioma origen a otro destino.

Listado de acrónimos

- **API:** Del inglés *Application Programming Interface*, interfaz de programación de aplicaciones.
- **ASD:** Del inglés *Adaptive Software Development*, Desarrollo de *Software* Adaptable.
- **AUP:** Del inglés *Agile Unified Process*, Proceso Unificado Ágil.
- **EN:** Entidad Nombrada.
- **HTML:** Del inglés *HyperText Markup Language*, lenguaje de marcado de hipertexto.
- **IA:** Inteligencia Artificial.
- **IDE:** Del inglés *Integrated Development Environment*, Entorno Integrado de Desarrollo.
- **IE:** Del inglés *Information Extraction*, extracción de información.
- **IR:** Del inglés *Information Retrieval*, recuperación de información.
- **JSP:** Del inglés *JavaServer Pages*, es una tecnología Java que permite generar contenido dinámico para web.
- **MT:** Del inglés *Machine Translation*, Traducción Automática.
- **NER:** Del inglés *Named Entity Recognition*, reconocimiento de entidades nombradas.
- **PDF:** Del inglés *Portable Document Format*, formato de documento portátil.
- **PFC:** Proyecto de Fin de Carrera.
- **PLN:** Procesamiento del Lenguaje Natural.
- **POO:** Programación Orientada a Objetos.

- **PUD:** Proceso Unificado de Desarrollo.
- **QA:** Del inglés *Question Answering*, Búsqueda de Respuestas.
- **RTF:** Del inglés *Rich Text Format*, formato de texto enriquecido.
- **SGML:** Del inglés *Standard Generalized Markup Language*, lenguaje de marcado generalizado.
- **TA:** Traducción Automática.
- **TTS**: Del inglés *Text-To-Speech*, convertir texto en habla.
- **URL:** Del inglés *Uniform Resource Locator*, localizador uniforme de recursos.
- **W3C:** Del inglés *World Wide Web Consortium.*
- **WWW:** Del inglés *World Wide Web.*
- **XML:** Del inglés *eXtensible Markup Language*, lenguaje de marcas extensible.
- **XP:** Del inglés *eXtreme Programming*, Programación Extrema.

Índice general

Agradecimientos I

Resumen III

Listado de acrónimos V

1. Introducción **1**

1.1. Marco de trabajo 1

1.2. Hipótesis de partida 3

1.3. Herramientas conceptuales necesarias 5

2. Estado del arte **9**

2.1. Herramientas relacionadas con el Reconocimiento de Entidades Nombradas 9

2.2. Herramienta escogida: LingPipe 11

3. Objetivos **13**

3.1. Objetivos generales 13

3.2. Objetivos específicos 13

3.3. Objetivos personales 14

4. Sistema de Identificación y Clasificación de EN **15**

4.1. LingPipe 15

4.1.1. Métodos directos 15

4.1.2. Método basado en modelos 16

4.2. Extensión de LingPipe 18

4.2.1. Creación de modelos 19

4.2.2. Heurísticas 21

4.3. Traducción automática de EN 23

4.4. Aplicación web 25

5. Descripción informática **29**
5.1. Descripción de las metodologías más relevantes 29
5.2. Metodología empleada: Programación Extrema 31
5.3. Adaptación de la metodología XP al proyecto 33
5.3.1. Iteraciones 34
5.4. Tecnologías empleadas 41
5.4.1. Lenguaje de programación 41
5.4.2. Entornos de programación 42
5.4.3. Librerías 43
5.4.4. Sistema de documentación LaTeX 43
5.4.5. Herramientas para el reconocimiento de EN 44

6. Resultados Experimentales **45**
6.1. Descripción de los conjuntos de datos 45
6.2. Métricas de evaluación de resultados 46
6.3. Resultados experimentales del sistema NER desarrollado 47
6.3.1. LingPipe 47
6.3.2. Inclusión de heurísticas 48
6.3.3. Comparativa de resultados 50

7. Conclusiones y trabajos futuros **53**
7.1. Conclusiones generales 53
7.2. Conclusiones personales 54
7.3. Trabajos futuros 54

Anexo 1 **57**

Anexo 2 **61**

Anexo 3 **63**

Bibliografía **65**

Índice de figuras

4.1. Cotejo de resultados entre LingPipe y el sistema desarrollado. 23
4.2. Diagrama del sistema. 24
4.3. Ejemplo de noticia para el idioma Castellano. 25
4.4. Ejecución de la noticia anterior. 26
4.5. Ejemplo de consulta de algunas EN sobre la lengua inglesa. 26
4.6. Resultados sobre la consulta anterior. 27

5.1. Diagrama de clases de analizador de LingPipe. 35
5.2. Diagrama de clases referente a la creación de un modelo. 35
5.3. Diagrama de clases referente a la creación automática de ficheros de entrenamiento. 37
5.4. Diagrama de clases relativo a la traducción automática de EN. 39
5.5. Diagrama de clases referente al sistema de Reconocimiento de EN. 40

7.1. Importar el fichero NER.war en Eclipse. 58
7.2. Importar las librerías. 59
7.3. Generar fichero WAR mediante Eclipse. 59

Índice de tablas

6.1. Relación de idioma y herramientas empleadas. 46
6.2. Comportamiento en la detección y clasificación de EN de las herramientas previas en subconjunto de noticias analizado. 46
6.3. Rendimiento original de LingPipe sobre los modelos utilizados. 47
6.4. Rendimiento del sistema tras la inclusión de heurísticas sobre LingPipe para los mismos modelos. 49
6.5. Rendimiento del sistema tras añadir búsqueda en *gazetteers* a las heurísticas anteriores. 50
6.6. Rendimiento definitivo del sistema tras añadir búsqueda sobre palabras gatillo a las heurísticas anteriores. 50
6.7. Comparativa de rendimiento del sistema desarrollado frente a la herramientas FreeLing y Named Entity Tagger para el idioma Inglés. 51
6.8. Comparativa de rendimiento del sistema desarrollado frente a la herramienta FreeLing para el idioma Castellano. 51
6.9. Comparativa de rendimiento del sistema desarrollado frente a la herramienta TreeTagger para el idioma Italiano. 51
6.10. Comparativa de rendimiento del sistema desarrollado frente a la herramienta TreeTagger para el idioma Francés. 51
6.11. Comparativa de rendimiento del sistema desarrollado frente a LingPipe para los mismos modelos. 52

7.1. Listado de palabras gatillo para Inglés y Castellano. 63
7.2. Listado de palabras gatillo para Francés e Italiano. 64

Capítulo 1

Introducción

En este capítulo se describirá de manera general el problema que se pretende resolver. Asimismo se intentará arrojar luz sobre algunos conceptos que se repetirán en capítulos posteriores, facilitando así la comprensión del documento.

1.1. Marco de trabajo

El recurso más valioso de la raza humana es el conocimiento, es decir, la información. Existen en el mundo volúmenes inmensos de información en forma de lenguaje natural: libros, periódicos, informes técnicos, etcétera. Del manejo eficiente de este conocimiento depende el uso de todos los demás recursos naturales, industriales y humanos [1].

En la actualidad, la época de mayor conocimiento de la historia de la humanidad, la cantidad de información disponible en distintas lenguas y formatos es abrumadora. Tanto es así que el tamaño de las colecciones almacenadas dificulta, aunque suene paradójico, la posibilidad de encontrar información específica y útil, ya sea en un sólo documento o en un conjunto de ellos. Además, las tareas de manejo y organización de tal volumen de información resultan costosas. Es por este motivo que las computadoras juegan un papel fundamental en el procesamiento de este conocimiento.

Sin embargo, lo que es conocimiento para los seres humanos, no lo es para las computadoras. Simplemente son archivos, una secuencia de caracteres, y nada más. Una computadora puede copiar un archivo, respaldarlo, transmitirlo o borrarlo (como un burócrata que pasa los papeles a otro burócrata sin leerlos); pero no puede buscar las respuestas a las preguntas en un texto, hacer las inferencias lógicas sobre su contenido, generalizarlo o resumirlo, es decir, hacer todo lo que las personas normalmente hacemos con el texto. El motivo es que las computadoras simplemente no lo pueden entender [2].

Debido a esto, se hizo necesario el desarrollo de sistemas para procesar todo este volumen de información. Los sistemas de extracción de información (de sus siglas en inglés IE, *Information Extraction*) realizan la tarea de buscar información muy concreta en colecciones de documentos, detectar la información relevante, extraerla y presentarla en un formato susceptible a ser tratado automáticamente más tarde.
De estos aspectos se ocupa el Procesamiento del Lenguaje Natural (PLN). Es una subdisciplina de la Inteligencia Artificial y la rama ingenieril de la lingüística computacional. El PLN se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas o entre personas y máquinas, por medio de lenguajes naturales. Por tanto, el PLN trata de diseñar mecanismos para comunicarse que sean eficaces computacionalmente.

A continuación se detallan algunas aplicaciones del PLN [3]:

- Síntesis del discurso: es la producción artificial de habla humana. Un sistema usado con este propósito recibe el nombre de sintetizador de habla y puede llevarse a cabo en *software* o en *hardware*. La síntesis de voz se llama a menudo en inglés *text-to-speech* (TTS), en referencia a su capacidad de convertir texto en habla. Sin embargo, hay sistemas que en lugar de producir voz a partir de texto lo hacen a partir de representación lingüística simbólica en habla.

- Reconocimiento del habla: un sistema de reconocimiento de voz es una herramienta computacional capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso. En su desarrollo intervienen diversas disciplinas, tales como la fisiología, la acústica, el procesamiento de señales, la inteligencia artificial y la ciencia de la computación.

- Traducción automática (TA): también llamada MT (del inglés *Machine Translation*), es un área de la lingüística computacional que investiga el uso de *software* para traducir texto o habla de un lenguaje natural a otro. En un nivel básico, la traducción por computadora realiza una sustitución simple de las palabras atómicas de un lenguaje natural por las de otro.

- Búsqueda de respuestas: llamado en inglés *Question Answering* (QA), es un tipo de recuperación de la información. Dada una cierta cantidad de documentos (tales como *World Wide Web* (WWW)), el sistema debería ser capaz de recuperar respuestas a preguntas planteadas en lenguaje natural. QA es observado como un método que

requiere una tecnología de PLN más compleja que otros tipos de sistemas para la recuperación de documentos y, en algunos casos, se observa como un paso por delante de la tecnología del buscador.

- Recuperación de información: llamada en inglés *Information Retrieval* (IR), es la ciencia de la búsqueda de información en documentos, búsqueda de los mismos documentos, la búsqueda de metadatos que describan documentos o, también, la búsqueda en bases de datos, ya sea a través de internet, intranet, para textos, imágenes, sonido o datos de otras características, de manera pertinente y relevante.

- Extracción de la información: es un tipo de IR cuyo objetivo es extraer automáticamente información estructurada o semiestructurada desde documentos legibles por una computadora.

De esta última tarea del PLN va a tratar gran parte de este trabajo. Más concretamente va a tratar de abordar el reconocimiento de Entidades Nombradas (Named Entity Recognition, NER) en documentos. Una Entidad Nombrada (EN) es una unidad de texto que representa nombres de persona, organizaciones, lugares, etc. Es decir, se puede considerar como un nombre propio que puede clasificarse en base a dichas categorías. P.Ej. "José Luis Rodríguez Zapatero", "Fondo Monetario Internacional", "Fuenlabrada", etc.

1.2. Hipótesis de partida

En este apartado se detalla el enunciado del proyecto. Éste tiene dos objetivos principales: por un lado la identificación y clasificación de EN, y por otro, la traducción de EN, de un idioma a otros, apoyándose en Wikipedia.

Para abordar la tarea de identificar y etiquetar EN se ha optado por extender y mejorar una herramienta ya existente. Para ello se han analizado diversas herramientas libres capaces de reconocer EN.

El reconocimiento de EN puede ser utilizado por sistemas que necesiten conocer y extraer información de los textos que procesan. Es muy útil en sistemas cuya funcionalidad consiste en filtrar información no deseada. Como ejemplo se puede tratar un sistema que filtre contenido pornográfico para no ser mostrado en las aulas de una escuela. En la frase: "Rocco estuvo rodando en Las Vegas junto con Pamela Anderson una película que será financiada por PlayBoy.", un sistema NER es capaz, una vez analizada la frase, de

etiquetarla de la siguiente forma: "<PER>Rocco</PER>estuvo rodando en <LOC>Las Vegas</LOC>junto con <PER>Pamela Anderson</PER>una película que será financiada por <ORG>PlayBoy</ORG>." Las entidades son delimitadas por <..></..>. En este caso las etiquetas contienen PER, LOC y ORG que hace referencia a personas (PERson), localizaciones (LOCation) y organizaciones (ORGanization). El sistema, en base a las EN detectadas, podría decidir que el contenido de la Web es inadecuado y por lo tanto lo filtraría para no ser mostrado. Como se puede observar, el reconocimiento de EN es un mecanismo muy útil que está emergiendo como un paso importante para el proceso de muchas aplicaciones basadas en el lenguaje natural [4].

Se han desarrollado sistemas muy eficientes, pero que normalmente se limitan a realizar NER en un dominio específico o en una sóla lengua. Esto permite encasillar a los sistemas para las tareas específicas de conocimiento de un lenguaje en particular.

Con este trabajo se pretende identificar y clasificar EN presentes en documentos para los idiomas Castellano, Inglés, Francés e Italiano. Esta clasificación reconoce cuatro tipos de entidades: personas, localizaciones, organizaciones y misceláneas, y permite actuar no sólo sobre un lenguaje particular, sino trasladar el conocimiento obtenido para un idioma a otras lenguas.

Este proyecto consta de las siguientes tareas:

- Estudiar y analizar diversas herramientas libres para la identificación de EN.
- Extender una de esas herramientas (LingPipe) creando un recubrimiento para reconocer entidades en cuatro idiomas: Castellano, Inglés, Francés e Italiano.
- Crear y utilizar modelos en diversos idiomas.
- Enriquecer la detección de EN mediante técnicas basadas en heurísticas.
- Enriquecer la clasificación de EN en base a *gazetteers* y "palabras gatillo".
- Enriquecer *gazetteers* utilizando Wikipedia. Para ello se han creado diversos *gazetteers* para Castellano, para posteriormente realizar una traducción automática para el resto de idiomas.
- Realizar una aplicación web que sirva como interfaz del sistema desarrollado.

1.3. Herramientas conceptuales necesarias

A continuación se detallan los conceptos más importantes que aparecerán en la memoria y que ayudarán al entendimiento de la misma:

- **Entidad Nombrada** (EN): Se refiere a las cadenas de palabras que representan nombres propios de persona, organizaciones, lugares, fechas, etc. En este proyecto se distiguen entre cuatro categorías:
  - PERSONA (PER): Las entidades de persona están limitadas a humanos identificados por un nombre, apellido, apodo o alias.
  - ORGANIZACIÓN (ORG): Las entidades de organización están limitadas a corporaciones, instituciones, agencias de gobierno, y otros grupos de gente definidos por una estructura organizacional establecida.
  - LUGAR (LOC): Las entidades de lugar incluyen nombres de lugares definidos política o geográficamente (ciudades, provincias, países, regiones internacionales, conjuntos de agua, montañas, etc.). Los lugares incluyen también estructuras hechas por el hombre como aeropuertos, autopistas, calles, fábricas y monumentos.
  - MISCELÁNEA (MISC): Corresponde al resto de EN que no están englobadas en las tres anteriores. Hay que destacar que en esta categoría también puede haber entidades de otras categorías debido a una mala clasificación.
- **Aplicación Web**: Se denomina "aplicación web" a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.
- **XML**: siglas en inglés de *eXtensible Markup Language* (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.
- **API**: interfaz de programación de aplicaciones (del inglés *Application Programming Interface*) es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

- **Gazetteer**: lista de palabras que pueden contener nombres de ciudades, países, organizaciones, etc, previamente clasificados y que sirven de apoyo al reconocimiento de EN.

- **Palabra gatillo**: Se denomina palabra gatillo a aquella que puede preceder a una entidad nombrada y proporciona información relevante sobre su clasificación (P.Ej. "**Sr.** Pedro Gómez", "**General** Stanley McChrystal", etc.

- **Tokenizar**: Consiste en el proceso de ruptura de un flujo de texto en elementos significativos llamados "tokens". Es de utilidad tanto en la lingüística (donde también se le conoce como la segmentación del "texto") y en ciencias de la computación, en donde forma parte del análisis léxico.

- **Servlet**: Los *servlets* son objetos que corren dentro del contexto de un contenedor de *servlets* (P.Ej. Tomcat) y extienden su funcionalidad. La palabra *servlet* deriva de otra anterior, *applet*, que se refería a pequeños programas que se ejecutan en el contexto de un navegador web. Por contraposición, un *servlet* es un programa que se ejecuta en un servidor.

- **Script**: Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Los *script* son casi siempre interpretados, pero no todo programa interpretado es considerado un *script*. El uso habitual de los *scripts* es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. Por este uso es frecuente que los *shells* sean a la vez intérpretes de este tipo de programas.

- **Parser**: Se traduce del Inglés como "analizador sintáctico". Es una de las partes de un compilador que transforma su entrada en un árbol de derivación. El análisis sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada. Un analizador léxico crea *tokens* de una secuencia de caracteres de entrada y son estos *tokens* los que son procesados por el analizador sintáctico para construir la estructura de datos, por ejemplo un árbol de análisis o árboles de sintaxis abstracta.

- **Serializar**: La serialización es el proceso de convertir un objeto en una secuencia de *bytes* para conservarlo en memoria, una base de datos o un archivo. Su propósito principal es guardar el estado de un objeto para poder crearlo de nuevo cuando se necesita.

- **Corpus lingüístico**: Un corpus lingüístico es un conjunto, normalmente muy amplio, de ejemplos reales de uso de una lengua. Estos ejemplos pueden ser textos (típicamente) o muestras orales (normalmente transcritas).

Capítulo 2

Estado del arte

En este capítulo se hará mención y se analizarán, de manera resumida, las herramientas más relevantes en la tarea de identificación de EN.

2.1. Herramientas relacionadas con el Reconocimiento de Entidades Nombradas

1. **Freeling** [5]: Es una herramienta de libre distribución implementada en C++. Está diseñada para ser utilizada como una librería externa desde una aplicación, y entre las utilidades que se extraen de esta herramienta destacan: Tokenización de texto, separación de sentencias, análisis morfológico, reconocimiento flexible de multipalabras y detección y clasificación de entidades nombradas. A esto se añade que está diseñada para español, catalán, gallego, inglés, portugués e italiano. Esta herramienta está basada en diccionarios.

2. **Stanford Named Entity Recognizer** [6]: Esta herramienta se centra en la detección de EN. Está implementada en Java y su código es libre. No obstante, no hay documentación para poder entrenar un clasificador propio. Incluye tres clases de entidades (PERSON, LOCATION y ORGANIZATION), pero sólamente es válida para Inglés.

3. **Rembrandt** [7]: Es un detector de EN. Está preparado para clasificar entidades que puedan tener más de un significado. Para ello utiliza un programa llamado SASK-IA, cuya función es pedir datos a Wikipedia para extraer el conocimiento con el que clasificar entidades nombradas. Como única fuente de datos, Rembrandt necesita una copia de las bases de datos de Wikipedia para los idiomas en que se quiera

trabajar, y además un servidor de bases de datos, como MySQL. Es código libre implementado en Java. Actualmente es válido para Portugués e Inglés, pero se tiene previsto actualizarlo con más idiomas en un futuro.

4. **Balie** [8]: Es un *software* de libre distribución desarrollado por la Universidad de Ottawa (Canadá). Está diseñado en Java y es válido para Inglés, Francés, Español, Alemán y Rumano. Entre sus usos destacan la tokenización de textos, el reconocimento de EN o la identificación de idioma a partir de un texto. Para el reconocimento de idioma utiliza dos técnicas:

   a) *Análisis de Frecuencia de Palabras de Parada (Stop Words Frequency Analysis).* Consiste en dado un texto, contar el número de palabras del tipo (*the, a, of, ...*), y las asocia al lenguaje que más se aproxime.

   b) *Análisis de Frecuencia de N-gramas.* Un n-grama es una subsecuencia de n elementos de una secuencia dada. Un n-grama de tamaño 2 se denomina "bigrama" o "digrama"; de tamaño 3, "trigrama"; de tamaño 4 ó más se denomina "n-grama" o "modelo de Márkov de orden (n - 1)". Es un buen indicador del idioma al que nos enfrentamos. Por ejemplo, un texto con una alta frecuencia de bigramas "wh" es más probable que se trate de un texto en inglés que en francés. Una ventaja de la utilización de éstas técnicas es que no hace falta ningún diccionario, pero puede resultar un problema si queremos analizar textos cortos, ya que en muchos casos puede resultar ambiguo.

5. **YooName** [9]: Es la evolución de *Balie*, por lo que está implementado en Java. Es un *software* basado en el aprendizaje semisupervisado, lo que consiste en una serie de técnicas de aprendizaje automático que utiliza datos de entrenamiento, tanto etiquetados como no etiquetados. Es capaz de identificar nueve categorías distintas de EN, entre las que se encuentran: *Person*, *Organization*, *Location*, *Facility* (que se refiere a edificios o construcciones realizados por el hombre), *Product* (como vehículos, armas, etc), *Event* (conferencias, guerras, etc), *Natural object* (insecto, animal, ...), *Unit* (día de la semana, mes, porcentaje, etc) y *Miscellaneous*.

6. **MinorThird** [12]: Es una colección de clases en Java para almacenar texto, anotarlo y aprender a extraer entidades nombradas y clasificarlas. Combina herramientas

para anotar y visualizar el texto mediante trazas que facilitan la eliminación de fallos. Su código es libre.

7. **OpenNLP** [13]: Es un proyecto referente al PLN desarrollado en Java. Su código es libre y entre sus utilidades destacan la detección de sentencias, tokenización de textos o el reconocimento de EN. Dichas utilidades están disponibles para procesar textos en inglés y castellano.

8. **LingPipe** [14]: Es un *software* desarrollado en Java que cuenta tanto con una versión libre como comercial. Entre sus funciones destacan la clasificación de documentos, el reconocimiento de EN, la identificación de idiomas o *clustering* en textos. Cuenta con un modelo para textos en inglés, pero es extensible a otros idiomas.

2.2. Herramienta escogida: LingPipe

Tras analizar las herramientas expuestas en el apartado anterior, finalmente se toma la decisión de extender **LingPipe**. Se ha tenido en cuenta que la herramienta esté desarrollada en Java, ya que uno de los objetivos personales del proyecto es la introducción a dicho lenguaje, pero se aprecia que, a excepción de *FreeLing*, todas las herramientas estudiadas han sido diseñadas con Java, por lo que se han valorado otros factores además del lenguaje. Entre dichos factores, además de que el código sea libre, destacan que LingPipe es fácilmente extensible, que aporta una buena documentación y que está en constante actualización.

Capítulo 3

Objetivos

A continuación se procederá a describir cuáles son los objetivos generales y parciales que se pretenden alcanzar con el desarrollo de este PFC.

3.1. Objetivos generales

El objetivo principal de este proyecto es desarrollar una herramienta basada en LingPipe que sea capaz de identificar y clasificar Entidades Nombradas para cuatro idiomas: Castellano, Inglés, Francés e Italiano. Para ello se han de cubrir una serie de objetivos parciales:

- Extender la herramienta LingPipe, incluyendo para ello un sistema de detección de Entidades Nombradas y un sistema de clasificación de las mismas en base a *gazetteers* creados manualmente y "palabras gatillo".
- Crear modelos basados en noticias para los idiomas Francés e Italiano.
- Desarrollar un sistema de traducción basado en Wikipedia que sea capaz de traducir automáticamente *gazetteers* a diversos idiomas: Castellano, Inglés, Francés, Italiano, Alemán, Holandés y Portugués.

3.2. Objetivos específicos

Algunos de los objetivos específicos necesarios para la realización de este trabajo son:

- Aprender el manejo básico de un *Integrated Development Environment* (IDE) profesional.
- Identificar y aplicar una *Application Programming Interface* (API) para parsear y escribir documentos XML.

- Identificar y aplicar una *Application Programming Interface* (API) para acceder al contenido de las bases de datos de Wikipedia.
- Introducción a la tecnología *JavaServer Pages* (JSP) para la creación de una aplicación Web que sirva como interfaz de la herramienta.
- Aprendizaje de LaTeX para creación de documentos formales.

3.3. Objetivos personales

Además de los objetivos ya citados, hay una serie de objetivos personales que se detallan a continuación:

- Introducción y aprendizaje de conocimientos sobre PLN.
- Introducción al lenguaje Java.
- Aprender a estructurar un proyecto utilizando el Paradigma de Orientación a Objetos (POO).
- Aprender a desarrollar un proyecto utilizando la metodología de Programación Extrema (XP).

Capítulo 4

Sistema de Identificación y Clasificación de EN

En esta sección se expondrá el sistema desarrollado basado en LingPipe para la identificación y clasificación de EN. Dicho sistema consiste en la extensión de la herramienta LingPipe mediante la inclusión de ciertas heurísticas que mejoren los resultados originales de la herramienta. Además, se ha realizado una extensión de idioma, basada en la creación de modelos de forma manual para francés e italiano y en la traducción automática de un idioma a otro mediante una API para Wikipedia.

4.1. LingPipe

LingPipe [14] es una herramienta basada en el procesamiento de textos. Entre sus funciones destaca la detección de EN, por lo que se aprovechará esta utilidad para el desarrollo de este proyecto.

Para el reconocimiento de EN LingPipe se basa fundamentalmente en dos técnicas: la primera implica el entrenamiento de un modelo estadístico, mientras que la segunda se basa en métodos más directos como el apoyo en diccionarios o expresiones regulares. A continuación se expone un pequeño resumen de la funcionalidad ofrecida por LingPipe mediante ambas técnicas.

4.1.1. Métodos directos

Los métodos basados en diccionarios que utiliza LingPipe se sustentan en el algoritmo Aho-Corasick.

Algoritmo Aho-Corasick

El algoritmo de coincidencia de cadenas de texto Aho-Corasick [19] es un algoritmo de búsqueda inventado por Alfred V. Aho y Margaret J. Corasick. Es un tipo de algoritmo de búsqueda en diccionarios que localiza los elementos de un conjunto finito de cadenas (diccionario) dentro de un texto de entrada. Concuerda con cualquier patrón, por lo que la complejidad del algoritmo es lineal a la longitud de los patrones, más la longitud del texto buscado, más el número coincidencias que proporcione la salida. Se debe tener en cuenta que debido a que todas las coincidencias se localizan, puede haber un número de coincidencias cuadrático si coincide cada subcadena (por ejemplo, el diccionario contiene a, aa, aaa, aaaa y la cadena de entrada es aaaa).

Clases disponibles para crear diccionarios propios

LingPipe proporciona una serie de clases que pueden utilizarse para incluir diccionarios propios o ciertas expresiones regulares. Entre esas clases se encuentran:

1. La clase *RegExChunker*: Se utiliza para añadir determinados tipos de patrones que se quieran incluir, como por ejemplo direcciones de correo.

2. La clase *DictionaryChunker*: Esta clase se utiliza para añadir palabras que puedan resultar ambiguas o palabras que sean dificiles de clasificar. P.Ej. El grupo de rap *50 Cent*.

3. La clase *ApproxDictionaryChunker*: Esta clase se parece mucho a la clase anterior, pero con la peculiaridad de que no busca coincidencias exactas, si no que busca aproximaciones a las palabras introducidas en el diccionario, asignándolas una distancia con respecto a la "original". Si encuentra una palabra exacta a la del diccionario, la distancia sería 0, mientras que si no lo es las distancias pueden variar (1, 2, ...).

4.1.2. Método basado en modelos

Los modelos son datos serializados que son utilizados por LingPipe para clasificar EN. Para crear dichos modelos se necesitan dos ficheros de entrenamiento, cuyos datos deben estar etiquetados con todas las entidades de interés y sus tipos. Por otra parte, los datos de entrenamiento deben coincidir con los datos que serán analizados en el sistema de ejecución. Por ejemplo, si un sistema se entrena con datos basados en noticias y se ejecuta sobre un documento médico el rendimiento se degrada, ya que será adaptado a las "pistas" proporcionadas por las noticias con las que se ha entrenado el modelo. LingPipe

proporciona en su página[1] dos modelos distintos para la detección de EN en inglés. Por un lado ofrece un modelo basado en noticias entrenado con el corpus MUC-6 (*the sixth in a series of Message Understanding Conferences*) [2] y, por otro lado, un modelo basado en biología entrenado con el corpus GeneTag (*tagged corpus for gene/protein named entity recognition*)[3].

LingPipe utiliza, basándose en modelos, tres métodos de reconocimiento de EN distintos. El primero, llamado *First-Best Named Entity Chunking*, consiste en clasificar las EN de un texto según el mejor resultado obtenido por LingPipe. Para dicha clasificación se utiliza la clase *RunChunker*, la cual proporciona un método para serializar el modelo y otro método que lee de la entrada estándar un texto y devuelve una lista de EN. En dicha lista se incluye el tipo de EN, la posición de inicio y la posición de fin dentro de la cadena de texto. A continuación se muestra un ejemplo de ejecución de LingPipe sobre un fragmento de texto biológico utilizando el modelo anterior mediante la clase *RunChunker*:

Entrada: *"p53 regulates human insulin-like growth factor II gene expression through active P4 promoter in rhabdomyosarcoma cells."*

Salida: *[0-3:GENE@-Infinity, 20-54:GENE@-Infinity, 81-92:GENE@-Infinity]*

Como se puede observar, LingPipe devuelve la posición de inicio y de fin de la entidad que identifica seguido del tipo, por lo que en este caso reconoce como genes las palabras *p53*, *insulin-like growth factor II gene* y *P4* promoter. Un segundo método es el llamado *N-Best Named Entity Chunking*, que consiste en devolver los n mejores resultados que pueda obtener LingPipe. A continuación se muestra un ejemplo para la misma entrada y mismo modelo anterior:

Salida:

0 -182.732574 [0-3:GENE@-Infinity, 20-54:GENE@-Infinity, 81-92:GENE@- Infinity]
1 -183.398420 [0-3:GENE@-Infinity, 14-54:GENE@-Infinity, 81-92:GENE@- Infinity]
2 -185.113308 [0-3:GENE@-Infinity, 20-54:GENE@-Infinity, 74-92:GENE@- Infinity]
3 -185.732465 [0-3:GENE@-Infinity, 20-54:GENE@-Infinity, 81-83:GENE@- Infinity]
4 -185.779154 [0-3:GENE@-Infinity, 14-54:GENE@-Infinity, 74-92:GENE@- Infinity]
5 -186.398310 [0-3:GENE@-Infinity, 14-54:GENE@-Infinity, 81-83:GENE@- Infinity]

[1] http://alias-i.com/lingpipe/web/models.html
[2] http://cs.nyu.edu/cs/faculty/grishman/muc6.html
[3] http://www.biomedcentral.com/1471-2105/6/S1/S3

6 -187.479685 [0-3:GENE@-Infinity, 20-54:GENE@-Infinity]
7 -188.145531 [0-3:GENE@-Infinity, 14-54:GENE@-Infinity]

Los *chunks* o fragmentos (EN) son ordenados por valor descendente (log base 2). El mejor resultado tiene un valor de -182.7, mientras que el siguiente tiene un valor de -183.4, etc.

El tercer y último método se denomina *Confidence Named Entity Chunking*, y consiste en ordenar los fragmentos por probabilidad, es decir, en primer lugar se mostrará el fragmento que tenga mayor probabilidad de estar correctamente clasificado, después el segundo y así sucesivamente. Véase un ejemplo de ejecución para la misma entrada y modelo anteriores:

Salida:

Rank	*Conf*	*Span*	*Type*	*Phrase*
0	*0.9999*	*(0, 3)*	*GENE*	*p53*
1	*0.7328*	*(81, 92)*	*GENE*	*P4 promoter*
2	*0.6055*	*(20, 54)*	*GENE*	*insulin-like growth factor II gene*
3	*0.3817*	*(14, 54)*	*GENE*	*human insulin-like growth factor II gene*
4	*0.1395*	*(74, 92)*	*GENE*	*active P4 promoter*
5	*0.0916*	*(81, 83)*	*GENE*	*p4*
6	*0.0088*	*(74, 83)*	*GENE*	*active P4*
7	*0.0070*	*(20, 49)*	*GENE*	*insulin-like growth factor II*
8	*0.0044*	*(14, 49)*	*GENE*	*human insulin-like growth factor II*

Por tanto hay un 99.99 % de probabilidades de que *p53* sea un gen, cerca de un 74 % de que *p4 promoter* sea un gen, y así sucesivamente.

El desarrollo de este PFC se basa en la extensión de LingPipe mediante esta técnica basada en modelos, y más concretamente mediante el método *First-Best Named Entity Chunking*. El objetivo es mejorar los resultados obtenidos por LingPipe mediante la inclusión de heurísticas y, a su vez, realizar una extensión de idioma de manera que la herramienta sea válida para los idiomas inglés, castellano, italiano y francés.

4.2. Extensión de LingPipe

Como se ha comentado en la anterior sección, se ha extendido LingPipe a partir del método *First-Best Named Entity Chunking*, ya que dicho método aporta, en principio, la

mejor clasificación para cada EN que detecte.

En este caso se contaba con diversos documentos en formato XML, por lo que se han desarrollado una serie de clases para transformar texto XML desde un fichero con extensión xml a texto plano en un fichero con extensión txt. En base a que se contaba con los documentos a analizar en ficheros de texto, se ha ampliado la funcionalidad de LingPipe para que, además de leer de la entrada estándar, pueda leer el documento desde un fichero de texto.

También se han añadido dos maneras distintas de generar la salida. La primera corresponde a la creación de una nueva clase para generar la salida en un fichero con formato XML, mientras que la segunda consiste en mostrar la salida en cuestión como un texto plano, lo que corresponde a la parte visual que se muestra desde la interfaz.
Véase un ejemplo de salida como texto plano:

<PER>Barack Obama </PER>fired his top commander in <LOC>Afghanistan </LOC>last night over a series of overtly contemptuous remarks <PER>General Stanley McChrystal </PER>.

4.2.1. Creación de modelos

Uno de los objetivos de la extensión de esta herramienta es la incorporación de nuevos idiomas para la identificación y clasificación de EN en noticias, más concretamente para los idiomas castellano, francés e italiano. Para ello se requiere la creación de tres modelos, uno para cada lengua.
Para generar un modelo mediante LingPipe son necesarios dos ficheros de entrenamiento. Cada fichero debe contener un número indeterminado de noticias, teniendo en cuenta que cuantas más líneas contengan los ficheros mejor será el modelo resultante, suponiendo que los datos de estos ficheros son correctos. A continuación se muestra un ejemplo del formato del fichero de entrenamiento con el cual se generará el modelo:

```
El O
Abogado B-PER
General I-PER
del I-PER
Estado I-PER
, O
Daryl B-PER
Williams I-PER
, O
subrayó O
hoy O
la O
necesidad O
de O
tomar O
medidas O
. . .
```

Como puede apreciarse, cada *token* se corresponde con una línea. Si se trata de una EN debe llevar marcada a su derecha un prefijo seguido de su tipo, ya sea PER, ORG o LOC, de modo que si es la primera o única palabra que forma la entidad, el prefijo será "B-", mientras que si la entidad se compone de más de una palabra, el resto de las palabras que formen la entidad tendrán el prefijo "I-". En cambio, aquellos *tokens* que no sean EN se marcarán como "O".

Para entrenar el modelo en castellano se ha dispuesto de dos ficheros de entrenamiento para noticias, extraídos de la *Conference on Computational Natural Language Learning*[4] (ConNLL), mientras que los ficheros de entrenamiento para los idiomas Francés e Italiano se han generado de manera manual. Para ello se han etiquetado a mano diversas noticias en formato XML, más concretamente 39 noticias para el modelo Italiano (27 para un fichero de entrenamiento y 12 para el otro) y 43 para el modelo Francés (32 y 11 respectivamente). Una vez se han etiquetado correctamente dichas noticias, se han creado una serie de clases con el objetivo de generar automáticamente dichos ficheros de entrenamiento en su formato correspondiente, a partir de los documentos XML que previamente se han etiquetado manualmente.

[4] http://www.cnts.ua.ac.be/conll/

4.2.2. Heurísticas

Se observó que los resultados resultaban muy variables según el modelo, ya que las estadísticas de aciertos resultaron, en general, proporcionales a su tamaño. Por ello, se han incluido una serie de heurísticas con el fin de mejorar los resultados obtenidos por LingPipe. Entre las heurísticas añadidas a la herramienta se incluyen:

- Se comprueba que LingPipe no clasifique una misma EN de diferentes maneras a lo largo del mismo documento.
- Se comprueba que LingPipe no deje sin etiquetar una EN que previamente ha identificado en otra parte del texto.
- Se ha desarrollado un sistema de detección de EN en función de si una palabra o conjunto de ellas comienzan o no por mayúsculas. Para ello se ha diseñado un método de filtrado donde se contempla que las precedencias de la potencial entidad no sean signos de puntuación, como por ejemplo '.', ';', ':', etc, ya que la palabras que se encuentren tras alguno de esos símbolos probablemente no serán ENs. De igual manera se comprueba que el sistema no identifique la palabra que da comienzo a una noticia como una EN a no ser que, o bien su clasificación sea distinta a "MISC" o bien dicha posible EN esté compuesta por más de una palabra.
- Se ha observado que en ocasiones LingPipe identifica determinadas palabras que no comienzan por mayúscula como una EN, por lo que se comprueba que LingPipe no identifique y clasifique dichas palabras como tal.
- Se han creado *gazetteers* de personas, lugares y organizaciones en los cuatro idiomas, para lo que se ha incluido un sistema de búsqueda de EN en *gazetteers*. De esa manera, cuando el sistema de detección de entidades que se ha incluido localice una posible EN en el texto, realizará la búsqueda en dichos ficheros. En caso de que un *gazetteer* contenga la EN en cuestión, ésta se clasificará como corresponda.
- Como apoyo a la identificación de EN pertenecientes a la categoría de persona se ha creado un *gazetteer* auxiliar de nombres propios, ya que se observó que en numerosas ocasiones se nombran personajes que no son populares y que, por lo tanto, al no estar incluídas en el *gazetteer* principal de personas, el sistema puede errar en su clasificación. De esta forma, si tras realizar la búsqueda de la posible EN en el *gazetteer* de personas sigue ser clasificada por el sistema, se comprueba que dicha EN contenga al menos un nombre de ese nuevo gazetteer, en cuyo caso será clasificada como "PERSON". P.ej. si aparece la EN *"Angelo Balducci"* en una noticia,

prácticamente con toda seguridad el sistema la va a etiquetar como "MISC", ya que dicha EN no está contenida en ningún *gazetteer*. En cambio, la nueva heurística comprueba si *"Angelo"* está contenido en el nuevo listado de nombres de personas, en cuyo caso el sistema clasificará la EN como "PERSON".

- También se ha creado un fichero contenedor de palabras gatillo para personas de igual manera para los cuatro idiomas (ver Anexo 3). Esto se debe a que se percibió que ciertas EN pertenecientes a la categoría de persona podían estar precedidas por una de estas palabras, que además tienen la peculiaridad de comenzar por mayúscula (P.Ej. *Mr. Obama*), característica que no compartirían otro tipo de palabras gatillo para el resto de categorías. Por ello se ha creado un nuevo sistema de búsqueda de dichas palabras gatillo, de forma que si el sistema de detección de EN localiza una posible entidad que no pertenece a ningún *gazetteer*, comprueba que dicha entidad contenga una de las palabras gatillo incluidas en el fichero, en cuyo caso se etiquetará como persona.

- Si LingPipe no consigue identificar una EN, se comprueba el caso de que el sistema desarrollado sea capaz de identificarla pero no de clasificarla (debido a que no pertenezca a ningún *gazetteer* ni contenga una palabra gatillo), en cuyo caso se clasificará la EN como MISC.

Así, el sistema resultante funcionaría de la siguiente manera: en primer lugar se llama a LingPipe (haciendo uso de las heurísticas diseñadas para tal efecto) y se almacena el listado de EN identificadas y clasificadas. Una vez almacenada la lista de posibles EN proporcionada por LingPipe, el sistema desarrollado detectaría todas las posibles EN que contuviera la noticia y procedería a clasificarlas, cotejando finalmente los resultados con LingPipe. A continuación se expone el proceso de clasificación de EN desarrollado: en primera instancia el sistema procedería a buscar las ENs identificadas previamente en los *gazetteers* incluídos. En el caso de que la búsqueda resultara satisfactoria el sistema las clasificaría según corresponda, mientras que en caso contrario se procedería a buscar en el fichero auxiliar de nombres propios y, posteriormente, en el fichero contenedor de palabras gatillo. Una vez llegado a este punto, si la búsqueda es positiva en alguno de los ficheros anteriores el sistema etiquetará la EN como "PERSON". Finalmente, si tanto la búsqueda en *gazetteers* como en los listados de nombres propios y palabras gatillo es infructuosa, el sistema clasificará dicha EN como "MISC". Una vez clasificadas dichas EN, se procede a cotejarlas con las obtenidas por LingPipe de la siguiente manera:

- En caso de que LingPipe no sea capaz de identificar una EN y el sistema desarrollado sí la identifique, el sistema será el encargado de etiquetarla y viceversa.

- En caso de que tanto LingPipe como el sistema identifiquen una EN, se comprueba lo siguiente: si LingPipe clasifica una EN como "MISC" y el sistema la clasifica como "PERSON", "LOCATION" u "ORGANIZATION", será el sistema el encargado de etiquetarla. En caso contrario será LingPipe quien la etiquete.

En la Figura 4.1 se muestra con más claridad cómo se cotejan los resultados.

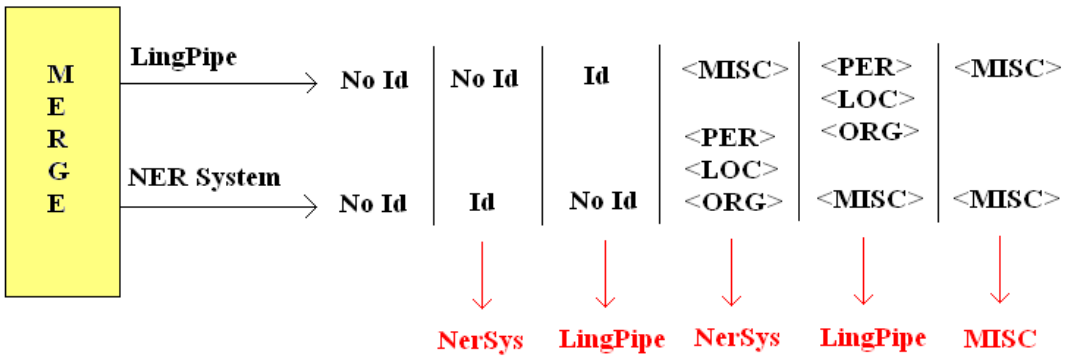

Figura 4.1: Cotejo de resultados entre LingPipe y el sistema desarrollado.

También puede apreciarse en la Figura 4.2 un sencillo diagrama del sistema completo.

4.3. Traducción automática de EN

La creación manual de *gazetteers* puede llegar a ser algo tedioso, por lo que otro de los objetivos de este PFC consiste en desarrollar un sistema de traducción automática de EN utilizando una API basada en Wikipedia. De esta manera no sólo se facilita el trabajo a la hora de crear nuevos *gazetteers* en distintos idiomas, sino que además se favorece la exportabilidad de la herramienta a la hora de añadir nuevas lenguas en un futuro.

De inicio deben crearse manualmente al menos *gazetteers* en un idioma, para posteriormente traducir esos mismos *gazetteers* al resto de lenguas que se pretenden añadir a la herramienta. Por ello, originalmente se han creado diversos *gazetteers* en castellano que se han traducido de manera automática al francés, inglés e italiano.
La tarea de traducción consiste en lanzar consultas a Wikipedia de las EN que se desean traducir, indicando el idioma desde el que se realiza la consulta. Para ello la API accede al contenido de las bases de datos de Wikipedia y devuelve una ristra de traducciones de la consulta en formato XML.

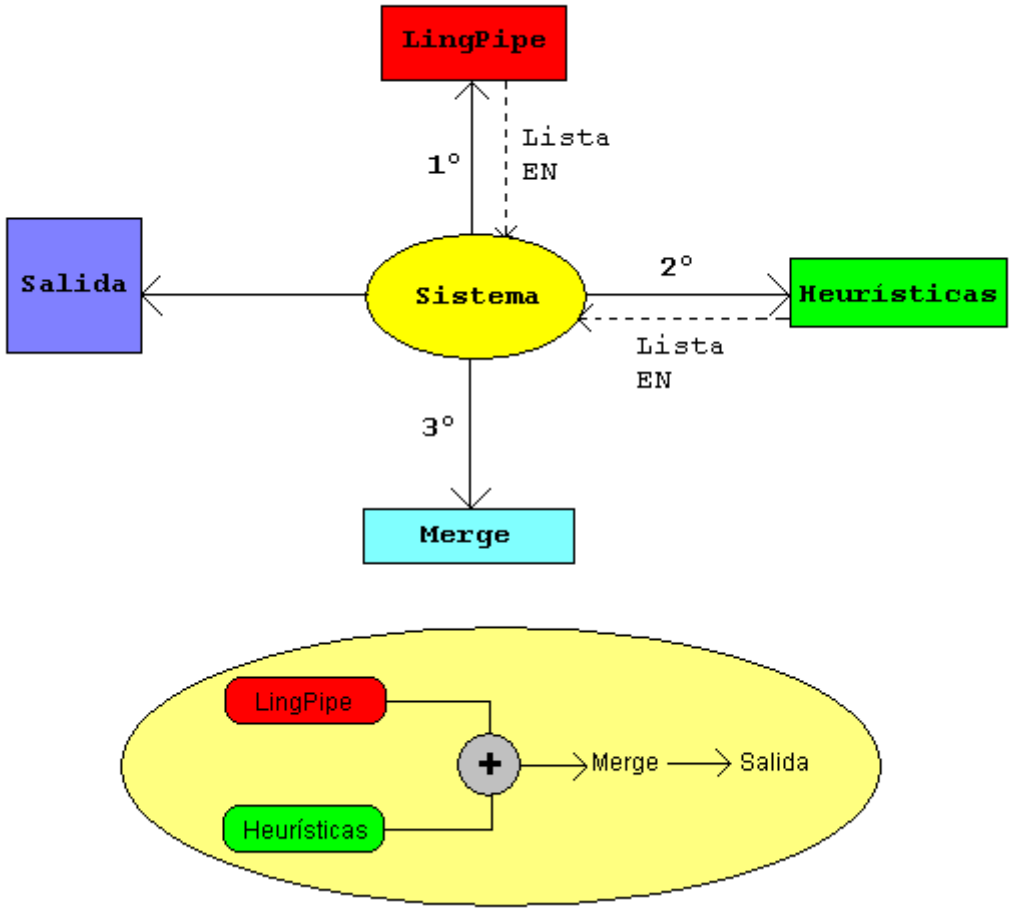

Figura 4.2: Diagrama del sistema.

Véase un ejemplo para la consulta en Inglés de la palabra *England*:

<page pageid="9316" ns="0" title="England">
<langlinks>
<ll lang="af" xml:space="preserve">Engeland</ll>
<ll lang="ak" xml:space="preserve">Ngyiresi</ll>
<ll lang="als" xml:space="preserve">England</ll>
<ll lang="an" xml:space="preserve">Anglaterra</ll>
<ll lang="ang" xml:space="preserve">Englaland</ll>
...

Como se puede observar, la salida proporciona, para cada idioma del que se devuelve traducción, un identificador de la lengua en cuestión y su traducción correspondiente. Por tanto, es necesario parsear la salida para filtrar todo contenido no deseado, de manera que se obtenga tan sólo la traducción de la consulta en los idiomas buscados.

Para este fin se han incluido en el sistema de traducción automática de EN los siguientes idiomas: Castellano, Inglés, Francés, Italiano, Holandés, Alemán y Portugués, de manera que se aporten traducciones para cualquiera de estas lenguas tanto de origen como destino.

4.4. Aplicación web

Para proporcionar un manejo intuitivo y simple de la herramienta se ha desarrollado una aplicación web sencilla que sirva como interfaz de la aplicación. En ella se incluyen tanto la funcionalidad de reconocimiento de EN como la de traducción automática de EN.

En la Figura 4.3 se muestra un ejemplo de ejecución del sistema desarrollado.

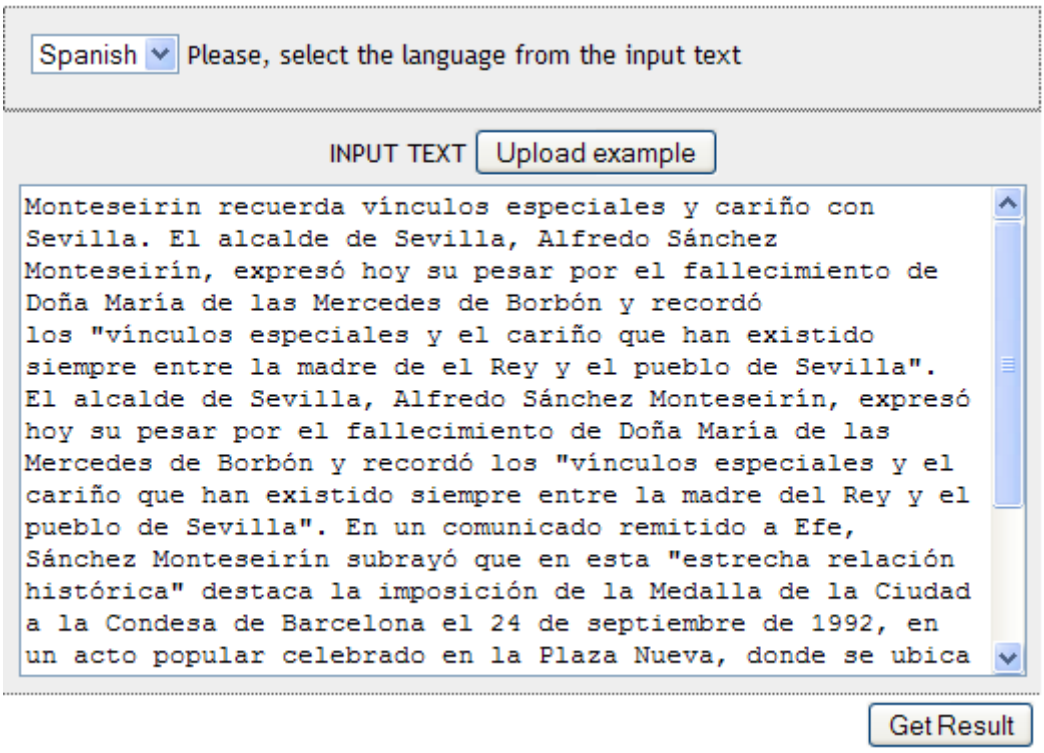

Figura 4.3: Ejemplo de noticia para el idioma Castellano.

Como se puede observar, la interfaz permite tanto seleccionar el idioma de la noticia a analizar como cargar un ejemplo de prueba.

En la Figura 4.4 se muestra la salida generada por la herramienta de forma visual.

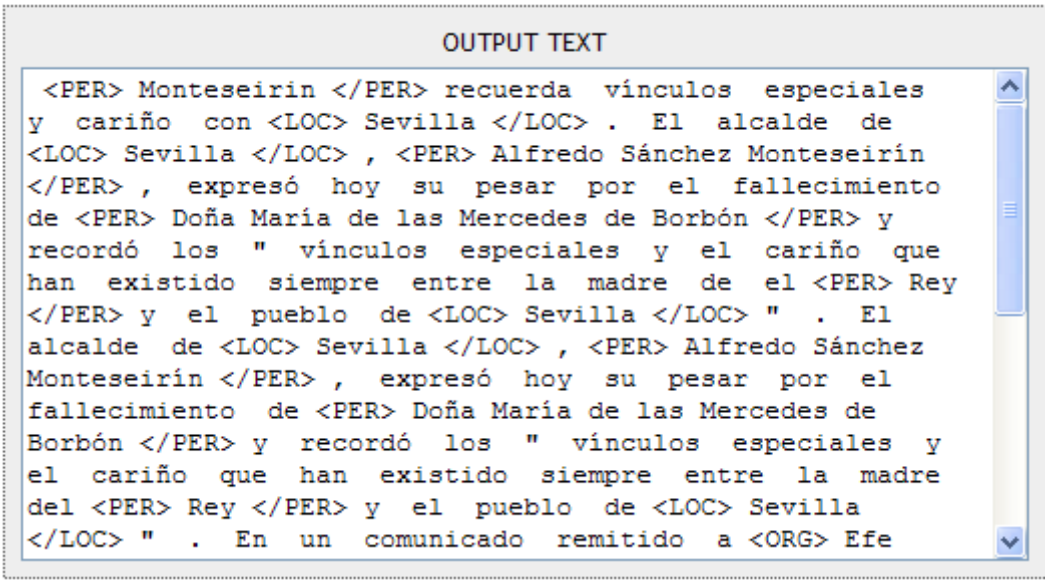

Figura 4.4: Ejecución de la noticia anterior.

Véase ahora un ejemplo (Figura 4.5) de traducción automática de EN. En él se quieren traducir ciertas EN desde el idioma Inglés al idioma Alemán.

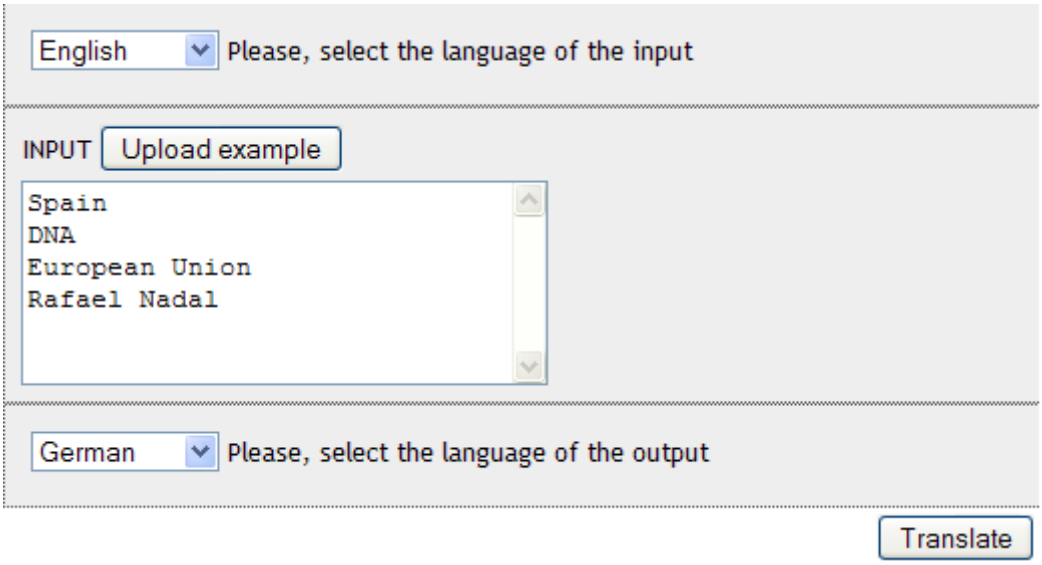

Figura 4.5: Ejemplo de consulta de algunas EN sobre la lengua inglesa.

Como se puede observar, se ha señalado el idioma desde el que se quiere traducir (en este caso inglés) y el idioma de destino (alemán). Es importante que la EN que se quiera traducir tenga entrada en Wikipedia tanto en el idioma origen como en el de destino, ya que de no ser así no se encontrará traducción para esa entidad, en cuyo caso se mostrará el mensaje "No hay traducción". De igual manera, es importante introducir la EN de la misma forma que figura su entrada en Wikipedia. Por ejemplo, *DNA* en Inglés

o Ácido desoxirribonucleico en Castellano. En la Figura 4.6 se muestra la salida obtenida por la consulta.

Figura 4.6: Resultados sobre la consulta anterior.

Capítulo 5

Descripción informática

En este capítulo se abordará la descripción informática del proyecto, para lo cual se tratarán resumidamente algunas de las metodologías de programación más comunes y las tecnologías utilizadas para el desarrollo de la aplicación, así como la adaptación de la metodología empleada al proyecto en curso.

5.1. Descripción de las metodologías más relevantes

Para el correcto desarrollo de todo proyecto informático, el programador o diseñador de *software* debe seguir una metodología apropiada a sus intereses con el fin de agilizar el proceso, controlar el desarrollo y mejorar los resultados, de manera que le proporcione un marco de trabajo adecuado. Por ello, en esta sección se hará un resumen de las metodologías de programación más importantes, entre las que se encuentran las siguientes:

- **Incremental [15]:** Provee una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto *software* reservando el resto de aspectos para el futuro.
  Los principios básicos sobre los que se sustenta esta metodología son:

  - Una serie de mini-Cascadas se llevan a cabo, donde todas las fases de la cascada modelo de desarrollo se han completado para una pequeña parte de los sistemas, antes de proceder a la próxima incremental, o
  - Se definen los requisitos antes de proceder con lo evolutivo, se realiza una mini-Cascada de desarrollo de cada uno de los incrementos del sistema, o
  - El concepto inicial de *software*, análisis de las necesidades y el diseño de la arquitectura y colectiva básicas se definen utilizando el enfoque de cascada, seguida por iterativo de prototipos, que culmina en la instalación del prototipo final.

- **Espiral [15]:** En esta metodología la atención se centra en la evaluación y reducción del riesgo del proyecto dividiendo el proyecto en segmentos más pequeños que proporcionan más facilidad de cambio durante el proceso de desarrollo, así como ofrecen la oportunidad de evaluar los riesgos durante todo el ciclo de vida.
  Cada viaje alrededor de la espiral atraviesa cuatro cuadrantes básicos:

  1. Determinar objetivos, alternativas y desencadenantes de la iteración.
  2. Evaluar alternativas, identificar y resolver los riesgos.
  3. Desarrollar y verificar los resultados de la iteración.
  4. Plan de la próxima iteración.

- **Proceso Unificado de Desarrollo Software (PUD) [16]:** Es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational o simplemente RUP. El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos.

- **Desarrollo ágil de software [17]:** Se entiende como desarrollo ágil de *software* a un paradigma de desarrollo de *software* basado en procesos ágiles. Los procesos ágiles de desarrollo de *software*, conocidos anteriormente como metodologías livianas, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados. Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto.
  Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación y defienden que el *software* funcional es la primera medida del progreso, lo que hace que sean criticados y tratados como "indisciplinados" por la falta de documentación técnica.
  Algunas metodologías ágiles de desarrollo de software son: *Adaptive Software Development* (ASD), *Agile Unified Process* (AUP), *Programación Extrema* (XP), etc.

- **Programación Extrema (XP) [18]:** Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la Programación Extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable

del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Se puede considerar la XP como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del *software*.

Por lo general, las metodologías clásicas pueden provocar inconvenientes como el retraso en la entrega del producto, mayor precio que el presupuestado originalmente o, en el peor de los casos, problemas de calidad debido al incumplimiento de los objetivos iniciales. No obstante, las metodologías ágiles pueden corregir estas contrariedades mediante un desarrollo incremental del producto y un crecimiento adaptativo, según las necesidades que van surgiendo.

5.2. Metodología empleada: Programación Extrema

XP consiste en la estrecha colaboración entre el cliente y el creador y la comunicación cara a cara. Primeramente se fija una versión sencilla de la herramienta con las mínimas funciones y un desarrollo básico para que, posteriormente, mediante reuniones periódicas, se vaya probando lo realizado y mejorando con nuevas opciones. Por tanto, el producto se va adaptando "sobre la marcha" a las necesidades del cliente, dando el producto por finalizado cuando se alcanza lo que el cliente pretendía en un principio [18].

Los valores originales de la XP son: simplicidad, comunicación, retroalimentación (*feedback*) y coraje. Un quinto valor, respeto, fue añadido en la segunda edición del libro *Extreme Programming Explained: Embrace Change*. Los cinco valores se detallan a continuación:

- **Simplicidad:** La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento, ya que un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores, hacen que la complejidad aumente exponencialmente.

- **Comunicación:** Para los programadores el código comunica mejor cuanto más simple sea, debido a que si el código es complejo hay que esforzarse para hacerlo inteligible. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método.

- **Retroalimentación (*feedback*):** Al tener al cliente integrado en el equipo de desarrollo, es más fácil una comprobación regular y una mejora constante de la aplicación.

- **Valentía:** Este punto puede llegar a parecer fuera de lugar comparando con los anteriores, pero es un atributo que deben tener los gerentes a la hora de aceptar la programación en parejas, ya que pueden pensar que la productividad vaya a reducirse a la mitad al estar sólo la mitad de los programadores escribiendo código. Por tanto, hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la productividad.

- **Respeto:** Tanto entre programadores (con una serie de normas en el desarrollo del código) como entre los creadores y el cliente.

Además de los valores, la XP debe cumplir con una serie de características:

- **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras.

- **Pruebas unitarias** continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.

- **Programación en parejas**: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera (el código es revisado y discutido mientras se escribe) es más importante que la posible pérdida de productividad inmediata.

- Frecuente **integración del equipo de programación con el cliente** o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

- **Corrección** de todos los errores antes de añadir nueva funcionalidad.

- **Refactorización del código**, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.

- **Propiedad del código compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- **Simplicidad en el código**: la XP apuesta por realizar algo simple y más adelante cambiarlo si se requiere, antes que realizar algo complejo y quizás no utilizarlo nunca.

5.3. Adaptación de la metodología XP al proyecto

A continuación se detalla el proceso realizado mediante XP adaptado particularmente a este proyecto:

Este trabajo ha sufrido un **desarrollo iterativo e incremental**, debido a que según se iban alcanzando objetivos previamente fijados se iban proponiendo nuevas ideas y funcionalidades que añadir al proyecto, de manera que se ha evolucionado en el desarrollo de algo básico a lo que se propuso como objetivo final.

A lo largo del desarrollo del proyecto se han realizado diversas **pruebas unitarias**, ya que se ha probado el correcto funcionamiento para cada nuevo módulo que se ha ido incluyendo al producto final, así como una serie de estadísticas para comprobar en qué porcentaje se han aportado mejoras a lo que se había hecho anteriormente. Por tanto, una característica implícita a lo anterior que se ha llevado a cabo en la realización de este proyecto es la **corrección de errores** antes de añadir nueva funcionalidad.

En este caso particular, al tratarse de un PFC, los tutores toman el papel del cliente, ya que son ellos quienes marcan los requisitos del proyecto, imponen los objetivos y modifican las distintas versiones del producto adaptando las nuevas funcionalidades de la herramienta a los objetivos fijados en un principio. Por tanto, la **integración del equipo de programación con el cliente** es total, realizándose pequeñas entregas a través de todo el proceso y reuniones periódicas entre alumno y tutores.

Por el mismo motivo que se ha comentado anteriormente, en este caso no existe la **programación en pareja**, ya que el único desarrollador es el alumno y no se puede aplicar esta característica. No obstante, en ocasiones se comentarán y compartirán partes del código con los tutores. Otra característica que no tiene sentido para este caso en particular es la **propiedad compartida del código**, ya que al ser un único programador no es necesario compartirlo.

También se realiza una **refactorización** constante del proyecto, debido a que se actualiza y mejora la herramienta en distintas fases del proyecto, sin que esto produzca un retroceso en el código ya implementado.

Por último, aunque no por ello menos importante, se debe mencionar que el diseño

cumple con la característica referente a la **simplicidad en el código**, ya que al haber seguido un desarrollo incremental, se ha partido de un código simple para posteriormente adaptarlo a los objetivos finales, de manera que no se ha realizado nada demasiado complejo que finalmente no haya sido utilizado.

5.3.1. Iteraciones

En este apartado se expondrá una división y planificación orientativa del trabajo realizado a modo de "cuaderno de bitácora" organizado en forma de hitos. Durante el resto de la sección se hablará de **cliente** cuando se quiera referir a los **tutores** de este PFC.

Hito 1: Encontrar una herramienta adecuada

El objetivo es seleccionar una herramienta sobre NER libre y adecuada al propósito de este PFC para posteriormente extenderla. Para ello, el cliente propone una serie de herramientas que hay que analizar.

Es importante verificar el mayor número de herramientas posibles.

Hito 2: Extender LingPipe

Tras analizar las herramientas propuestas por el cliente se opta por extender LingPipe. Debido a sus características (libre, implementada en Java, extensible, etc.) parece la herramienta más apropiada.

Una vez decidida la herramienta sobre la cual se trabajará, el cliente propone realizar un estudio sobre su funcionalidad y manejo. Este punto es de vital importancia para el desarrollo del resto del proyecto. En la Figura 5.1 se muestra el diagrama de clases referente al analizador de LingPipe utilizado.

Hito 3: Crear un modelo para Castellano

Una vez asimilado el funcionamiento de LingPipe, el cliente propone generar un modelo para noticias en Castellano aprovechando la funcionalidad de esta herramienta. Para ello, se parte de un fichero de entrenamiento y otro de evaluación proporcionado por la *Conference on Computational Natural Language Learning* (ConNLL).

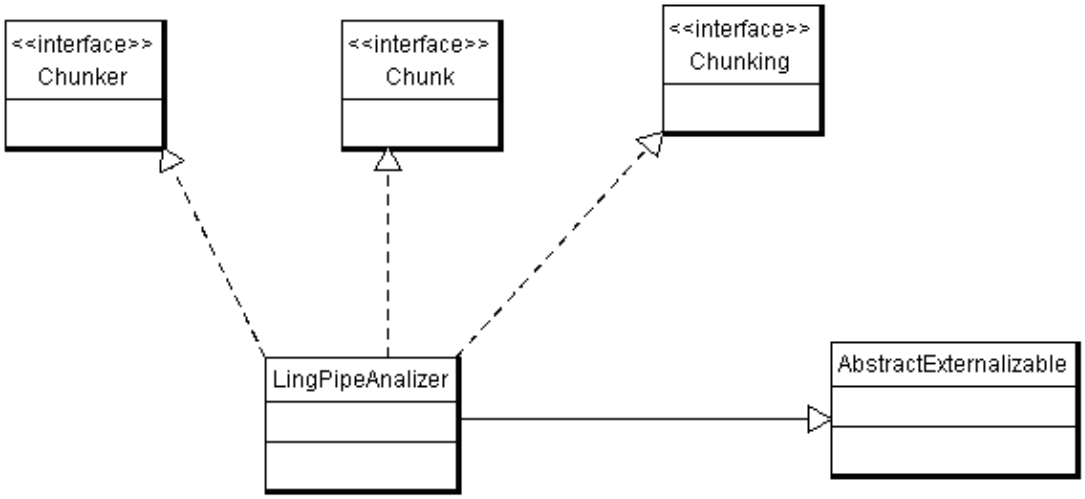

Figura 5.1: Diagrama de clases de analizador de LingPipe.

Una vez creado el modelo se verifica que se ha generado correctamente mediante el análisis de diversos documentos en Castellano. En la Figura 5.2 se puede observar el diagrama de clases para generar un modelo a partir de dos ficheros de entrenamiento.

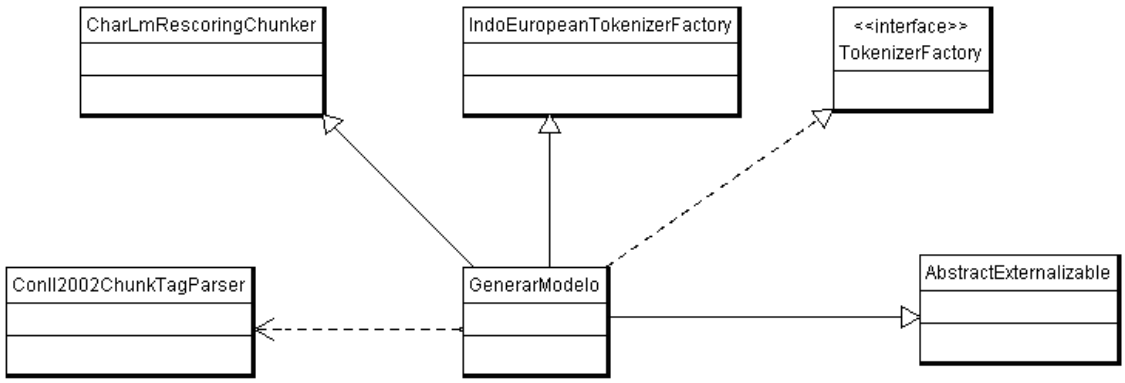

Figura 5.2: Diagrama de clases referente a la creación de un modelo.

Hito 4: Analizar documentos en Castellano e Inglés

Se propone el análisis de varias noticias en Castellano e Inglés. Para ello el cliente proporciona una serie de documentos en ambos idiomas previamente etiquetados por otra herramienta disponible para NER y en formato XML.

Para la realización del análisis es necesario parsear el texto de los documentos XML

proporcionados con el fin de generar ficheros de texto planos a partir de ficheros XML. Para ello se utiliza la API basada en Java SAX[1]. Una vez transformados los ficheros XML a ficheros de texto plano, se debe crear una nueva clase para la tarea inversa, es decir, escribir el texto plano de la salida generada por LingPipe en un fichero con formato XML.

Una vez analizados los documentos se hace una comparativa de resultados entre las noticias aportadas originalmente por el cliente y esas mismas noticias etiquetadas por LingPipe mediante una estadística de aciertos.

Hito 5: Crear ficheros de entrenamiento para Italiano y Francés

En este hito el cliente propone la creación de ficheros de entrenamiento para Italiano y Francés, con el objetivo de generar dos nuevos modelos para dichas lenguas. Al igual que en el hito anterior, proporciona diversos documentos XML para ambos idiomas previamante etiquetados.

En esta ocasión se impone una carga de trabajo más manual. Se deben etiquetar a mano correctamente todas las noticias que se quieran utilizar para, posteriormente, crear una serie de clases que generen automáticamente dichos ficheros de entrenamiento en su formato correcto a partir de los documentos XML que previamente se han etiquetado manualmente.

Una vez creados los dos modelos, se realiza una estadística del etiquetado de LingPipe para noticias en ambos idiomas. También se realiza una estadística de aciertos para los documentos XML aportados por el cliente y, finalmente, se comparan los resultados. En la Figura 5.3 se puede apreciar el diagrama de clases referente a la generación automática de ficheros de entrenamiento a partir de documentos XML.

Hito 6: Diseñar heurísticas para etiquetar EN

Tras realizar las primeras estadísticas de los aciertos de LingPipe se observa que los resultados son diferentes dependiendo del tamaño del modelo. Por ello el cliente propone realizar una serie de heurísticas que, a priori, puedan mejorar los resultados obtenidos en las primeras fases del PFC.

Se observó que en los modelos más pequeños es probable que se dejen EN sin anotar, o que incluso cierta EN sea anotada en una parte del texto y en otra no, entre otras cosas.

[1]Descripción en el apartado 5.4.3

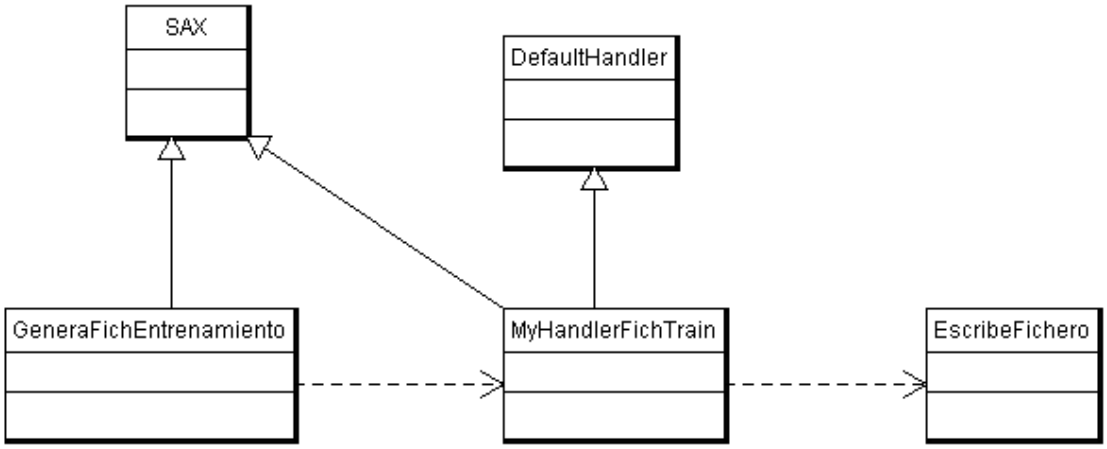

Figura 5.3: Diagrama de clases referente a la creación automática de ficheros de entrenamiento.

Por ello, entre las heurísticas realizadas en esta fase se incluyen:

- Un sistema de detección de potenciales EN.
- Se comprueba que LingPipe no etiquete una misma EN de distintas maneras a lo largo del mismo documento. Para ello se ha tomado el convenio de etiquetar durante todo el documento dicha EN como se haya clasificado en primera instancia.
- Se comprueba que LingPipe no deje sin anotar una EN que previamente ha sido anotada en el texto.
- Se comprueba que LingPipe no identifique como EN palabras que no contengan mayúscula.
- Se comprueba que el sistema no identifique la primera palabra de cada noticia como una EN, a no ser que su clasificación sea distinta de "MISC" o que el sistema detecte que pueda tratarse de una EN con más de una palabra.
- Una vez que se ha obtenido la salida proporcionada por LingPipe, se conteja si nuestro sistema identifica alguna EN que LingPipe no haya identificado, en cuyo caso se etiquetará dicha EN como "MISC".

Después de añadir éstas heurísticas a LingPipe, se realiza de nuevo una estadística de aciertos para los mismos documentos, comparándose los resultados.

Hito 7: Crear gazetteers para castellano

El cliente propone un nuevo mecanismo de clasificación de EN. Se propone la identificación de EN en base a una serie de *gazetteers* para la clasificación de personas, lugares y organizaciones.
Para ello se realiza, de manera manual, un *gazzetter* para personas[2], para lugares[3] y para organizaciones[4][5], además de un *gazetteer* auxiliar de nombres propios de persona.

Hito 8: Incluir sistema de búsqueda en gazetteers

Una vez se dispone de los correspondientes *gazetteers* en castellano, se propone incluir un sistema de búsqueda de EN en *gazetteers*.

El objetivo es realizar búsquedas eficientes de EN en ficheros relativamente grandes, por lo que se opta por implementar y añadir una búsqueda dicotómica a la herramienta. De esta manera, en esta fase se incluye una nueva heurística que consiste en comprobar si una posible EN pertenece a alguno de los *gazetteers* disponibles, ampliando así las técnicas incluidas en anteriores fases.

Una vez terminado el proceso, se realiza de nuevo una estadística y se comprueba con los anteriores resultados obtenidos para castellano.

Hito 9: Traducir automáticamente los gazetteers para castellano a francés, italiano e inglés

Recopilar listados de palabras a mano para construir *gazetteers* resulta muy costoso, por lo que el cliente propone un sistema para traducir automáticamente *gazetteers* de un idioma destino a otro origen. El objetivo de esta fase es traducir los *gazetteers* recopilados en fases anteriores a inglés, francés e italiano. Para ello se propone la utilización de una API basada en Java para acceder al contenido de Wikipedia (BlikiEngine[6]).

La salida que proporciona esta API debe parsearse, ya que para este caso sólo resulta útil la información del idioma del que se busca traducción, más concretamente la entrada del artículo. Por tanto, se debe filtrar toda información no deseada, para lo que se utiliza de nuevo el parseador SAX.

[2]http://www.lalistawip.com/
[3]http://www.meteo24.com/es/Cities,World,6,45.html
[4]http://money.cnn.com/magazines/fortune/global500/
[5]www.elpais.com
[6]Descripción en la sección 5.4.3

Tras traducir automáticamente los nuevos *gazetteers* desde los originales en castellano, se realiza, como en la anterior fase, una nueva estadística para cada idioma y se comparan resultados. En la Figura 5.4 se muestra el diagrama de clases relativo a la traducción automática de EN.

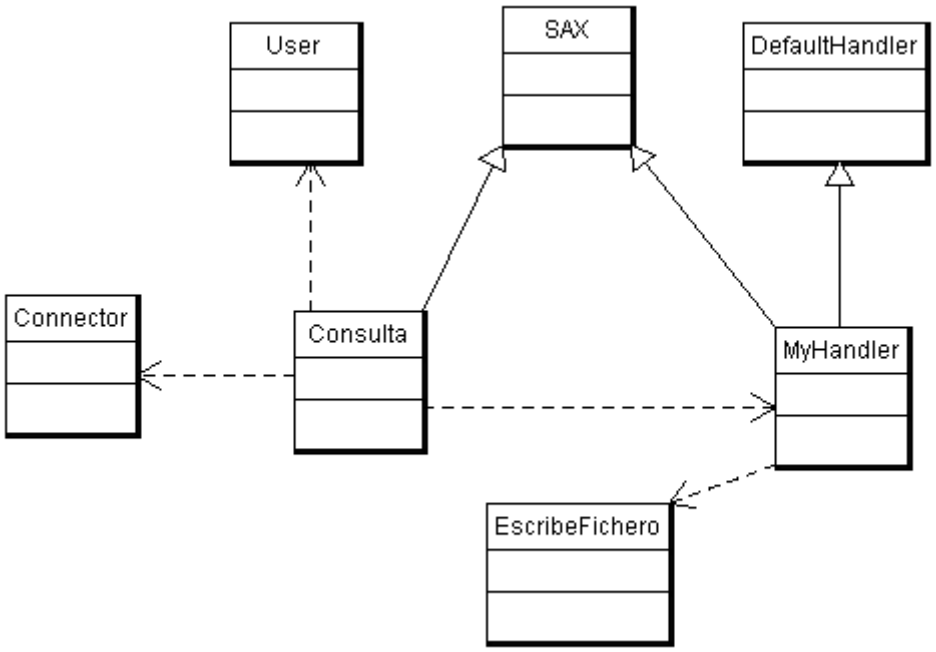

Figura 5.4: Diagrama de clases relativo a la traducción automática de EN.

Hito 10: Incluir palabras gatillo

Tras realizar diversas pruebas sobre noticias en distintas lenguas se observa que determinadas palabras que preceden a una EN proporcionan información relevante sobre su clasificación. Por ello, el cliente propone incluir una nueva heurística a la herramienta, que consistirá, como se ha hecho en alguna etapa anterior, en crear un listado manualmente de palabras gatillo en algún idioma para, posteriormente, traducirlo al resto de idiomas. En este caso el idioma de partida será el inglés, ya que una parte de la lista ha sido extraída de la Web[7].

Una vez creado el listado de palabras gatillo para los cuatro idiomas se añade un sistema de búsqueda para dichas palabras. De este modo, si tras comprobar que la EN que se está clasificando no pertenece a ningún *gazetteer*, se procede a buscar si dicha entidad

[7]https://www.harrods.com/HarrodsStore/register

contiene una palabra gatillo, en cuyo caso se etiqueta como “PER”.

Al igual que en ocasiones anteriores, tras añadirse la nueva mejora a la herramienta se procede a realizar una estadística de aciertos para los cuatro idiomas, con el fin de observar si hay mejoría con respecto a las anteriores estadísticas.

Hito 11: Crear aplicación Web

Se propone la creación de una aplicación web que sirva como interfaz de la herramienta. El cliente sugiere utilizar para ello la tecnología JSP[8]. El objetivo es dotar a la herramienta de un manejo cómodo a través de una interfaz sencilla. En este hito el sistema NER ya se encuentra completamente desarrollado, por lo que en la Figura 5.5 se puede apreciar el diagrama de clases referente a la parte de Reconocimiento de EN.

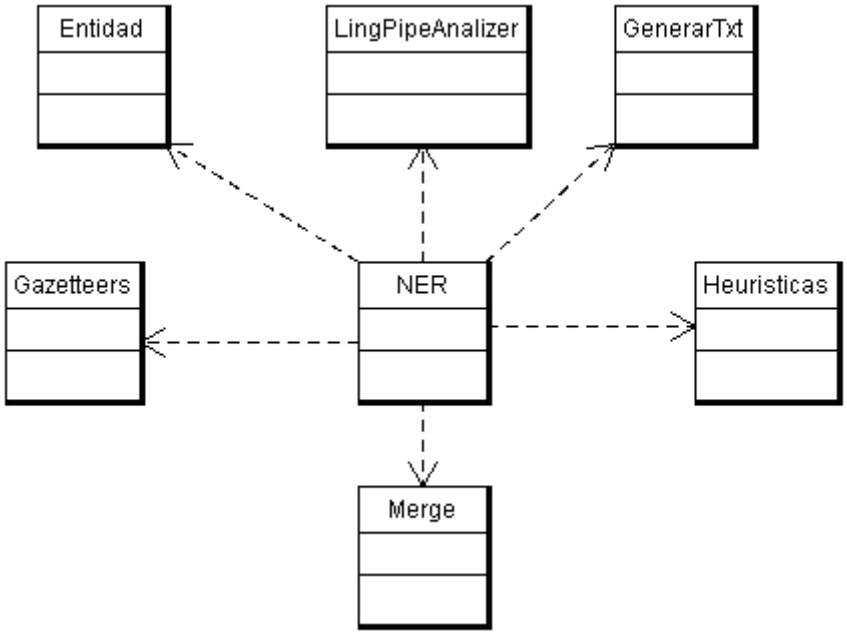

Figura 5.5: Diagrama de clases referente al sistema de Reconocimiento de EN.

Hito 12: Extender la traducción automática de EN a más idiomas

Una vez diseñada la interfaz, el cliente formula que sería interesante ofrecer alguna lengua más a la hora de traducir EN, por lo que se añaden los idiomas alemán, holandés y portugués a los cuatro ya existentes. De este modo, las lenguas disponibles para la traducción automática de EN serían castellano, inglés, francés, italiano, alemán, holandés y portugués.

[8]Descripción en la sección 5.4

5.4. Tecnologías empleadas

En esta sección se abordarán las distintas tecnologías que han sido necesarias para el desarrollo de la herramienta.

5.4.1. Lenguaje de programación

El lenguaje de programación fijado desde un principio en el enunciado del proyecto para el diseño de la aplicación es el lenguaje Java, en parte debido a su condición de pertenecer al paradigma de Programación Orientada a Objetos (POO), cuyas características se detallan a continuación.

Programación Orientada a Objetos

Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otras personas se creó la POO. Resumiendo mucho, se puede decir que consiste en una serie de normas para desarrollar código de manera que otras personas puedan utilizarlo y adelantar su trabajo. Es decir, conseguir que un código sea reutilizable [10].

El elemento fundamental de la POO es, como su nombre indica, el objeto. Se puede definir el objeto dentro del paradigma de la POO como un elemento con dos características: un estado y un comportamiento. Los objetos en programación son modelados observando objetos del mundo real. Un objeto podría ser, por ejemplo, un perro. El perro tiene un estado: nombre, color, raza, etc. y un comportamiento: ladrar, comer, llorar, dormir,etc. A las características que componen el estado de un objeto se las denomina **atributos** y las que definen el comportamiento se las conoce como **métodos**.

Java

Java [20] es un lenguaje de programación Orientado a Objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Además, es un lenguaje fuertemente tipado y que se considera robusto porque realiza verificaciones tanto en tiempo de ejecución como en tiempo de compilación.

En cuanto a la filosofía del lenguaje, Java se creó para responder a cinco principios fundamentales:

1. Debería usar la metodología de la Programación Orientada a Objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

JavaServer Pages

JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Al igual que Java, esta tecnología es un desarrollo de la compañía Sun Microsystems.
Las JSP's permiten la utilización de código Java mediante *scripts*. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de Bibliotecas de Etiquetas (*TagLibs* o *Tag Libraries*) externas e incluso personalizadas.

En cuanto a su arquitectura, JSP puede considerarse como una manera alternativa y simplificada de construir servlets. Es por ello que una página JSP puede hacer todo lo que un servlet puede hacer, y viceversa. Cada versión de la especificación de JSP está fuertemente vinculada a una versión en particular de la especificación de servlets.

El funcionamiento general de la tecnología JSP es que el Servidor de Aplicaciones interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que se presentará en la pantalla del navegador del usuario, como refleja el siguiente esquema general:

$$\text{JSP} \rightarrow ServidorAplicaciones(Servlets) \rightarrow Cliente(Navegador)$$

5.4.2. Entornos de programación

Para el desarrollo de la aplicación se ha utilizado fundamentalmente el siguiente entorno:

Eclipse

Eclipse [11] es un Entorno de Desarrollo Integrado (IDE) compuesto de librerías, herramientas y compiladores extensibles para construir, desarrollar y mantener el *software* durante su ciclo de vida. El IDE de Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.

5.4.3. Librerías

Algunas de las librerías utilizadas para este proyecto son:

SAX

Son las siglas en inglés de "Simple API for XML". Es una API que nos permite crear nuestro propio parser de XML. Aunque originalmente sólo estaba implementada en Java, actualmente también está disponible para otros lenguajes de programación.

Bliki Engine

Es una API diseñada para el lenguaje Java. Entre sus propósitos principales se incluyen la conversión de sintaxis Wikipedia a sintaxis HTML, PDF, etc. Además, también se incluye la funcionalidad inversa para HTML, es decir, convertir sintaxis HTML a sintaxis Wikipedia. No obstante, su función más importante es la de proporcionar acceso directo a los datos y contenidos de las bases de datos de MediaWiki.

5.4.4. Sistema de documentación LaTeX

LaTeX [21] es un sistema de composición de textos que está formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX(un lenguaje "de bajo nivel", en el sentido de que sus acciones son muy elementales). Está orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas.

Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con LaTeX es comparable a la de una editorial científica de primera línea.

LaTeX es *software* libre bajo licencia LPPL y presupone una filosofía de trabajo diferente a la de los procesadores de texto habituales basándose en comandos. Tradicional-

mente, este aspecto se ha considerado una desventaja, sin embargo, LaTeX, a diferencia de los procesadores de texto comunes, permite a quien escribe un documento centrarse exclusivamente en el contenido sin tener que preocuparse de los detalles del formato. Además de sus capacidades gráficas para representar ecuaciones, fórmulas, notación científica e incluso musical, permite estructurar fácilmente el documento (con capítulos, secciones, notas, bibliografía, índices analíticos, etc.) lo cual brinda comodidad y lo hace útil para artículos académicos y libros técnicos. Una de las ventajas de LaTeX es que la salida que ofrece es siempre la misma, con independencia del dispositivo (impresora, pantalla, etc.) o el sistema operativo (MS Windows, MacOS, Unix, GNU/Linux, etc.) y puede ser exportado a partir de una misma fuente a numerosos formatos tales como Postscript, PDF, SGML, HTML, RTF, etc.

5.4.5. Herramientas para el reconocimiento de EN

LingPipe

LingPipe es una herramienta de procesamiento de textos. Tiene diferentes aplicaciones, como tokenización de textos, *clustering*, detección de entidades nombradas o verificación ortográfica, entre otras. Es libre, implementado en Java y extensible.

Capítulo 6

Resultados Experimentales

En el siguiente capítulo se analizarán los resultados obtenidos como fruto de las pruebas realizadas sobre el sistema desarrollado.

6.1. Descripción de los conjuntos de datos

Para comprobar el rendimiento del sistema se ha dispuesto de un conjunto de noticias en castellano, francés, inglés e italiano previamente etiquetadas por diversas herramientas para el reconocimiento de EN. Para verificar dicho rendimiento, los resultados estadísticos obtenidos de dichas noticias han sido cotejados con los resultados obtenidos por nuestro sistema, para lo cual se ha analizado un subconjunto de diez noticias para cada idioma. Se ha elegido esta cifra debido a que se ha considerado que el número de EN contenidas en dicho número de noticias sería suficiente para reflejar de manera adecuada el comportamiento de dichas herramientas y de nuestro sistema, aunque evidentemente, un número mayor de noticias supondría una mayor precisión.

El etiquetado previo de las EN para las noticias en inglés ha sido realizado mediante la herramienta *FreeLing* [22], mientras que su clasificación ha sido obra de la herramienta *Named Entity Tagger* [23]. Para el caso de las noticias en italiano, su etiquetado y clasificación se debe a la herramienta *TreeTagger* [24], al igual que para las noticias en francés, mientras que de la clasificación y etiquetado de las noticias en castellano se ha ocupado de nuevo la herramienta *FreeLing*.
En la Tabla 6.1 se muestra un resumen de las herramientas utilizadas para cada idioma y su cometido.

En relación a los resultados obtenidos por estas herramientas cabe detallar que, por un lado, se han evaluado para cada noticia, el número de EN correctamente etiquetadas y clasificadas respecto al total de EN de la noticia, mientras que por otro lado, también se han valorado el número de EN correctas respecto al total de las detectadas por la

Idioma	Etiquetado	Clasificación
Castellano	FreeLing	FreeLing
Inglés	FreeLing	Named Entity Tagger
Francés	TreeTagger	TreeTagger
Italiano	TreeTagger	TreeTagger

Tabla 6.1: Relación de idioma y herramientas empleadas.

herramienta (se considerará correcta una EN bien etiquetada y bien clasificada). Además, una EN clasificada como "MISC" sólo se considerará correcta siempre y cuando dicha EN no encaje en las categorías "LOC", "ORG" o "PER"). Es decir, se ha calculado un porcentaje de aciertos respecto al total de EN de cada noticia y respecto a las detectadas. A continuación, en la Tabla 6.2, se puede ver un resumen del comportamiento de las herramientas anteriores en las noticias analizadas.

Idioma	EN Totales	EN Detectadas	EN Correctas
Inglés	204	209	135
Castellano	164	176	126
Italiano	143	157	84
Francés	169	218	68

Tabla 6.2: Comportamiento en la detección y clasificación de EN de las herramientas previas en subconjunto de noticias analizado.

6.2. Métricas de evaluación de resultados

A continuación se describen las medidas de evaluación utilizadas para la elaboración de los resultados experimentales en este proyecto:

- Precisión: La precisión calcula el porcentaje de respuestas correctas dadas por el sistema con respecto al número de respuestas totales. Se calcula con la siguiente expresión:

$$Precisión = \frac{\text{Nº de respuestas correctas dadas por el sistema}}{\text{Nº de respuestas totales}}$$

- Recall: Esta medida computa qué cantidad de información relevante ha sido extraída del texto con el que se trabaja por el sistema. Se calcula de la siguiente manera:

$$Recall = \frac{\text{Nº de respuestas correctas dadas por el sistema}}{\text{Total de posibles respuestas correctas}}$$

6.3. Resultados experimentales del sistema NER desarrollado

En esta sección se expondrán de manera detallada los resultados obtenidos por el sistema. Para ello se hará un recorrido por las distintas fases de las que ha constado el proyecto y en las que se analizarán los resultados, de manera que pueda verificarse si las heurísticas añadidas en cada etapa aportan alguna mejoría a la herramienta que se pretende extender.

6.3.1. LingPipe

En primer lugar se ha evaluado el rendimiento de LingPipe sobre los modelos utilizados para castellano, francés, italiano e inglés, los cuales se muestran en la Tabla 6.3.

Idioma	ENs totales	ENs detectadas	ENs correctas	Precisión	Recall
Inglés	316	257	145	45.8 %	56.4 %
Castellano	263	284	217	82.5 %	76.4 %
Italiano	224	104	60	26.8 %	57.7 %
Francés	301	199	118	39.2 %	59.2 %

Tabla 6.3: Rendimiento original de LingPipe sobre los modelos utilizados.

Como puede observarse los resultados varían dependiendo del idioma. Esto se debe principalmente al tamaño de los modelos, ya que cuanto mayor sea el modelo, a priori, mejor será el rendimiento, aunque también pueden influir otros factores como la dificultad de clasificar determinadas EN o la ambigüedad de algunas de ellas. Se puede apreciar que los resultados para castellano son notablemente mejores que para el resto de idiomas. Dicho modelo ha sido generado a partir de un fichero de entrenamiento y otro de evaluación proporcionado por la *Conference on Computational Natural Language Learning* (ConNLL), cuyo tamaño final es de 59.5 MB, mientras que el modelo para noticias en Inglés ha sido obtenido desde la URL de LingPipe[1], cuyo peso asciende a 15.2 MB. En cuanto a los modelos restantes (italiano y francés), han sido generados mediante el etiquetado manual de 39 noticias para el modelo Italiano y 43 para el modelo Francés, cuyos tamaños son de 8.16 MB y 7.09 MB respectivamente.

Se aprecia que el *Recall* es mayor que la Precisión en todos los casos excepto para el Castellano. Esto es debido a que en el resto de idiomas el sistema detecta menos EN en las noticias de las que en realidad tienen, por lo que el porcentaje de aciertos sobre las EN detectadas es mayor que sobre las EN totales. Mientras tanto, para el Castellano ocurre

[1] http://alias-i.com/lingpipe/web/models.html

lo contrario, ya que detecta ligeramente algunas EN más de las que en realidad hay.
También se observa que en los modelos más pequeños la diferencia entre *Recall* y precisión se acentúa, lo cual justifica lo dicho anteriormente, ya que se debe en mayor medida a la escasa detección de EN para dichos modelos.

6.3.2. Inclusión de heurísticas

Una vez se tiene conocimiento del rendimiento de LingPipe sobre determinados modelos, el objetivo es mejorarlo. De esta manera se pretende que el sistema desarrollado obtenga mejores resultados que LingPipe para un mismo modelo.
Entre las heurísticas incluidas en la primera fase del proyecto destacan las siguientes:

- Debido a que se comprobó que, para los modelos utilizados, LingPipe podía clasificar una misma EN de distintas maneras a lo largo de una noticia (o incluso etiquetarla en ciertas partes de la noticia y en otras no hacerlo), el sistema comprueba que una EN siempre se clasifique de igual forma durante la misma. Por consiguiente, también se comprueba que LingPipe no deje sin etiquetar una EN que previamente había sido detectada y clasificada.

- Como se observó en los resultados obtenidos por LingPipe, su rendimiento para modelos pequeños es peor debido, en parte, al número de EN que la herramienta deja sin clasificar. Por ello se ha desarrollado un sistema de detección de EN en función de si una palabra o conjunto de ellas comienza por mayúscula y no la precede un signo de puntuación. En este caso, siempre y cuando LingPipe no sea capaz de clasificarla, el sistema la clasificará como "MISC". De igual manera, se comprueba que el sistema no etiquete la palabra que abre la noticia, a no ser que su clasificación sea distinta de "MISC" o que el sistema detecte que puede tratarse de una EN formada por más de una palabra.

- Otra peculiaridad que se observa tras el análisis de las noticias etiquetadas y clasificadas por LingPipe es que dicha herramienta puede, en algunos casos, identificar una o varias palabras dentro del texto que no comiencen por mayúscula como una EN. Por ello, se ha añadido al sistema otra heurística que consiste en que si LingPipe identifica y clasifica una palabra con esa característica, no sea etiquetada como tal en la salida final.

A continuación, en la Tabla 6.4, se muestra el rendimiento del sistema desarrollado tras la inclusión de las heurísticas anteriores sobre LingPipe para los mismos modelos.

Idioma	ENs totales	ENs detectadas	ENs correctas	Precisión	Recall
Inglés	316	350	192	60.7 %	54.8 %
Castellano	263	278	219	83.2 %	78.7 %
Italiano	224	241	104	46.4 %	43.15 %
Francés	301	333	174	57.8 %	52.3 %

Tabla 6.4: Rendimiento del sistema tras la inclusión de heurísticas sobre LingPipe para los mismos modelos.

Como puede observarse, la Precisión en los resultados posteriores aumenta con respecto a la Tabla 6.3, ya que hay un ligero aumento del número de aciertos en comparación con LingPipe. Por contra, se aprecian peores resultados para el *Recall*, que empeora algo más cuánto más pequeño es el modelo, a excepción del Castellano, que se ve ligeramente incrementado con respecto a la tabla anterior. Esto se debe a que para el resto de idiomas el sistema detecta un mayor número de EN en comparación con la primera fase, y la diferencia se acentúa cuando se trata de un modelo pequeño, ya que el número de EN que identifica LingPipe para estos modelos es reducido. Esto indica que los resultados no tienen porqué ser necesariamente peores, sino que el total de posibles respuestas es mayor, lo que hace que disminuya el porcentaje de aciertos. En cuanto al Castellano vemos que mejora levemente, en parte debido a que el sistema detecta alguna EN menos que en la primera fase gracias a la heurística referente a no etiquetar EN que no comiencen por mayúscula.
No obstante, se puede percibir que el número de EN detectadas es algo mayor que el número total de EN que existen, debido sobre todo a que en ocasiones el sistema identifica más de una EN sobre una EN compuesta de varias palabras.

En la siguiente etapa del proyecto se han incluido *gazetteers* de personas, lugares y organizaciones para los cuatro idiomas. De esta manera, si el sistema detecta una posible EN realizará una búsqueda en dichos ficheros que, en caso de contener dicha EN, la aplicación la clasificará como corresponda según el *gazetteer* en que se encuentre. En caso de que ningun *gazetteer* la contenga y que LingPipe no la detecte, el sistema la clasificará como "MISC". En la Tabla 6.5 se contemplan los resultados obtenidos en esta fase una vez añadida la búsqueda en *gazetteers* a las heurísticas anteriores.

Como se puede percibir, la inclusión de *gazetteers* ha supuesto un índice de mejoría significativo en ambas medidas de evaluación. A excepción del Castellano, que ya presentaba unos resultados similares gracias al modelo utilizado, se observa que el índice de aciertos

Idioma	ENs totales	ENs detectadas	ENs correctas	Precisión	Recall
Inglés	316	350	243	76.9 %	69.4 %
Castellano	263	278	227	86.3 %	81.6 %
Italiano	224	241	151	67.4 %	62.7 %
Francés	301	333	235	78.07 %	70.6 %

Tabla 6.5: Rendimiento del sistema tras añadir búsqueda en *gazetteers* a las heurísticas anteriores.

ha mejorado con respecto a fases anteriores. Esto se deduce a raíz de que el número de EN detectadas es el mismo que en la etapa anterior y que sólo varía la clasificación de algunas de ellas.

Por último, en la fase final del proyecto se ha incluido un nuevo sistema de búsqueda sobre un fichero contenedor de palabras gatillo para personas a las heurísticas añadidas anteriormente. A continuación, en la tabla 6.6, se exponen los resultados definitivos obtenidos tras añadir este nuevo sistema de búsqueda sobre palabras gatillo a la aplicación:

Idioma	ENs totales	ENs detectadas	ENs correctas	Precisión	Recall
Inglés	316	350	248	78.4 %	70.8 %
Castellano	263	278	228	86.7 %	82 %
Italiano	224	241	157	70 %	65.1 %
Francés	301	333	235	78.07 %	70.6 %

Tabla 6.6: Rendimiento definitivo del sistema tras añadir búsqueda sobre palabras gatillo a las heurísticas anteriores.

En esta ocasión no se aprecian mejoras significativas en los resultados cosechados. A excepción del Inglés, que su rendimiento ha mejorado en algunos puntos, la mejoría es casi imperceptible. Se debe tener en cuenta que esta última heurística es válida únicamente para clasificar personas, y que para ello se cuentan con dos *gazetteers* adicionales, por lo que los resultados están dentro de lo esperado.

6.3.3. Comparativa de resultados

En esta sección se mostrará una comparativa entre los resultados obtenidos por el sistema desarrollado y el resto de herramientas utilizadas con el fin de comprobar si el sistema es válido y cumple con los objetivos fijados inicialmente.

Comparativa con herramientas previas

En este punto se realizará una comparativa de rendimiento entre el sistema realizado basado en LingPipe y las herramientas utilizadas previamente sobre las noticias correspondientes a los idiomas inglés, castellano, francés e italiano. Dicha comparativa se muestra en la Tablas 6.7, 6.8, 6.9 y 6.10, respectivamente.

Idioma	**FreeLing + NET**		**Sistema Desarrollado**	
Inglés	**Precisión**	**Recall**	**Precisión**	**Recall**
	66.2 %	64.6 %	78.4 %	70.8 %

Tabla 6.7: Comparativa de rendimiento del sistema desarrollado frente a la herramientas FreeLing y Named Entity Tagger para el idioma Inglés.

Idioma	**FreeLing**		**Sistema Desarrollado**	
Castellano	**Precisión**	**Recall**	**Precisión**	**Recall**
	76.8 %	71.6 %	86.7 %	82 %

Tabla 6.8: Comparativa de rendimiento del sistema desarrollado frente a la herramienta FreeLing para el idioma Castellano.

Idioma	**TreeTagger**		**Sistema Desarrollado**	
Italiano	**Precisión**	**Recall**	**Precisión**	**Recall**
	58.7 %	53.5 %	70 %	65.1 %

Tabla 6.9: Comparativa de rendimiento del sistema desarrollado frente a la herramienta TreeTagger para el idioma Italiano.

Idioma	**TreeTagger**		**Sistema Desarrollado**	
Francés	**Precisión**	**Recall**	**Precisión**	**Recall**
	40.2 %	31.2 %	78.07 %	70.6 %

Tabla 6.10: Comparativa de rendimiento del sistema desarrollado frente a la herramienta TreeTagger para el idioma Francés.

A tenor de los resultados obtenidos se puede comprobar como el rendimiento del sistema es positivo si se compara con el rendimiento de las herramientas expuestas en las tablas anteriores. Si se escrutan las estadísticas se verifica que el sistema implementado ha cosechado mejores resultados tanto en la Precisión como en el *Recall*, y que en algún caso, como es el caso del idioma Francés, la diferencia resulta notable.

Comparativa con LingPipe

A continuación, en la Tabla 6.11, se muestra la comparativa de resultados entre LingPipe y el sistema implementado para los mismos modelos:

Idioma	**LingPipe**		**Sistema Desarrollado**	
Inglés	**Precisión**	**Recall**	**Precisión**	**Recall**
	45.8 %	56.4 %	78.4 %	70.8 %
Castellano	**Precisión**	**Recall**	**Precisión**	**Recall**
	82.5 %	76.4 %	86.7 %	82 %
Italiano	**Precisión**	**Recall**	**Precisión**	**Recall**
	26.8 %	57.7 %	70 %	65.1 %
Francés	**Precisión**	**Recall**	**Precisión**	**Recall**
	39.2 %	59.2 %	78.07	70.6 %

Tabla 6.11: Comparativa de rendimiento del sistema desarrollado frente a LingPipe para los mismos modelos.

Como se puede observar en la Tabla 6.11 los resultados obtenidos han sido, a rasgos generales, satisfactorios, ya que se ha cumplido el objetivo inicial del proyecto, que no es otro que mejorar el rendimiento de LingPipe para un mismo modelo. Como se puede comprobar, a tenor de los resultados expuestos, la diferencia de rendimiento entre los distintos idiomas se ha visto reducida de manera significativa si se compara con el rendimiento obtenido inicialmente por LingPipe. De esto se deduce que la mejora aportada por las heurísticas añadidas a LingPipe es mayor cuanto peor sea el resultado de partida de dicha herramienta. De esta manera puede verificarse que, gracias a las heurísticas incluidas, el sistema puede proporcionar unos resultados satisfactorios independientemente del modelo empleado.

Capítulo 7

Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones extraídas de la realización de este proyecto y los posibles trabajos futuros que pueden servir para continuar en esta línea de investigación.

7.1. Conclusiones generales

Tras finalizar este proyecto se ha comprobado que se ha cumplido de manera satisfactoria el objetivo principal que se había propuesto al inicio del mismo que, como ya se ha comentado, no es otro que el desarrollo de un sistema basado en LingPipe para el reconocimiento y la clasificación de EN en noticias para cuatro idiomas. No obstante, para la consecución de este PFC se han propuesto otros objetivos, como son el desarrollo de un sistema de traducción automática de EN basado en Wikipedia o el desarrollo de una aplicación web que facilite el manejo de la herramienta.

Por tanto, a tenor de las aportaciones del proyecto se han extraído una serie de conclusiones que se detallan a continuación:

- Queda verificado que el rendimiento de la herramienta LingPipe depende directamente del modelo empleado.
- Una combinación acertada de heurísticas y *gazetteers* sobre un modelo pequeño puede igualar o mejorar el rendimiento obtenido por un modelo superior.
- Queda patente que la inclusión de *gazetteers* en el sistema aporta más rendimiento que otras técnicas utilizadas.
- Las heurísticas incluidas presentan cierta independencia de los idiomas utilizados, por lo que son extensibles a otros idiomas.

- Las consultas sobre Wikipedia para la traducción automática de EN requieren que éstas sean estrictamente igual que su entrada en Wikipedia.

7.2. Conclusiones personales

Personalmente me ha parecido muy interesante todo lo relacionado con el PLN en general y la detección y clasificación de EN en particular, ya que anteriormente no había tenido la oportunidad de profundizar en este ámbito de la informática y me ha servido para cerciorarme de la complejidad que conlleva para una máquina algo tan sencillo, a priori, como es la identificación y clasificación de EN para una persona.

Un aspecto muy importante son los objetivos formativos marcados en un principio, entre los que se encuentran el aprendizaje de Java (haciendo uso del paradigma de la Programación Orientada a Objetos), el desarrollo de una aplicación web (mediante el aprendizaje de JSP), o tareas más costosas como son el manejo de una API o afrontar el reto de extender una herramienta previamente desarrollada (como es LingPipe), debido a la dificultad que entraña la comprensión y manejo de código no desarrollado por uno mismo.

Por último, otra conclusión importante que se extrae del proyecto es el proceso que conlleva la planificación y el desarrollo de un sistema informático, o la creación de una memoria técnica con un lenguaje específico, para lo que se ha utilizado la herramienta para la creación de documentos LaTeX.

7.3. Trabajos futuros

De los resultados obtenidos se abre un amplio conjunto de posibles trabajos que podrían servir de base a futuros proyectos, o que podrían ser de interés para la mejora del sistema implementado. A continuación se citan algunos de ellas:

- Incluir nuevas heurísticas al sistema que mejoren los resultados obtenidos o mejorar las ya existentes.
- Extender la herramienta a nuevos idiomas.
- Generar nuevos modelos para los idiomas trabajados.
- Crear nuevos *gazetteers* para la clasificación de ENs o traducir automáticamente los ya existentes a otras lenguas.

- Ampliar el alcance de las ENs a otras categorías.
- Mejorar las heurísticas de identificación de ENs para las categorías de lugar y organización.

Anexo 1

Manual de instalación de la aplicación web

El sistema desarrollado esta formado por una herramienta implementada en Java y JSP. La herramienta, por lo tanto, puede ser ejecutada en cualquier equipo y con cualquier sistema operativo.

Requisitos del sistema

Los requisitos tecnológicos para ejecutar la aplicación son los siguientes:

- Maquina Virtual de Java.
- Eclipse Java EE IDE para Web Developers (Opcional).
- Apache Tomcat 6.0 Servlet/JSP Container.

Instalación

Es posible realizar la instalación de la aplicación de dos formas:

Sin utilizar Eclipse

1. Instalación del servidor Apache Tomcat 6.0.
2. Insertar el fichero "NER.war" en la carpeta "webapps" del servidor Tomcat. Acceder a la dirección http://localhost:8080/NER/index.jsp.

Utilizando Eclipse

1. Instalación de Eclipse Java EE IDE para Web Developers.
2. Instalación del servidor Apache Tomcat 6.0. No es necesario instalarlo antes de importar la aplicación, pero es más sencillo porque cuando se importa ya se elige el servidor que se utilizará cuando se ejecute.

3. Una vez instalada la aplicación de Eclipse se procede a importar el proyecto a partir del fichero "NER.war". Para ello hay que pulsar File, Import, Web, WAR File (Ver Figura 7.1). El "Web project" es el nombre del proyecto que se va a crear eligiendo para este caso "NER". Si todavía no se tiene instalado el servidor Tomcat, en el campo "Target runtime" no aparecerá ninguna aplicación.

4. Importar las librerías correspondientes activando el "check" de las propias librerías y pulsando "Finish"(Ver Figura 7.2).

5. Ejecutar la aplicación pulsando "Open Web Browser". La dirección de la página principal es http://localhost:8080/NER/index.jsp. Si al proyecto se le ha llamado de otra forma se tiene que sustituir la cadena "NER" por el nombre elegido.

Generación de fichero WAR

Un fichero WAR (Web Archive) no es más que un fichero comprimido que contiene todos los archivos necesarios para la aplicación web. Este fichero puede generarse de la siguiente manera utilizando Eclipse: Botón derecho sobre el proyecto sobre el que se quiere generar el archivo, Export, WAR file (Ver Figura 7.3).

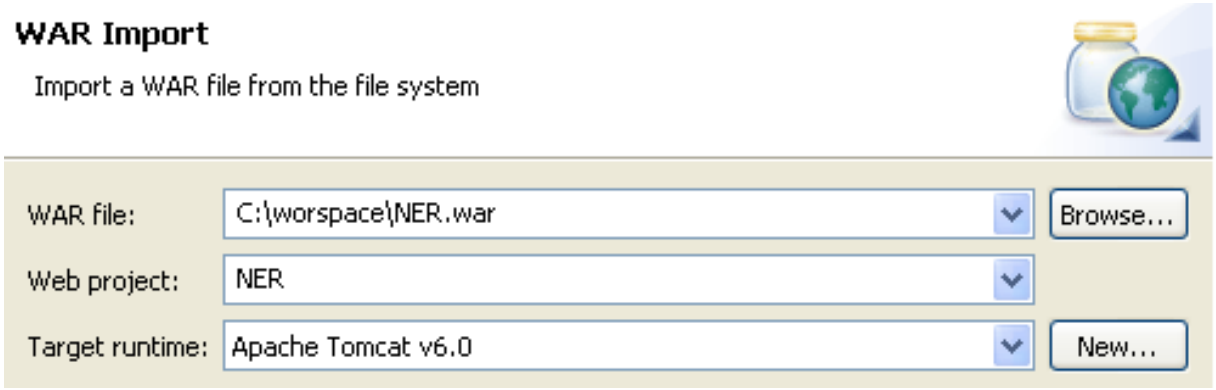

Figura 7.1: Importar el fichero NER.war en Eclipse.

Aplicación Java

La aplicación Java desarrollada se compone de dos módulos principales: El módulo *Main*, que se encarga de identificar y clasificar EN, y el módulo *ConsultaWiki*, que se encarga de la traducción automática de EN. La clase principal para el primero es la clase *RunChunker*, que para su correcta ejecución requiere, como primer parámetro, el idioma sobre el que se quiere analizar la noticia (mediante los valores "1" para Castellano, "2" para Inglés, "3" para Italiano y "4" para Francés), y como segundo parámetro el texto

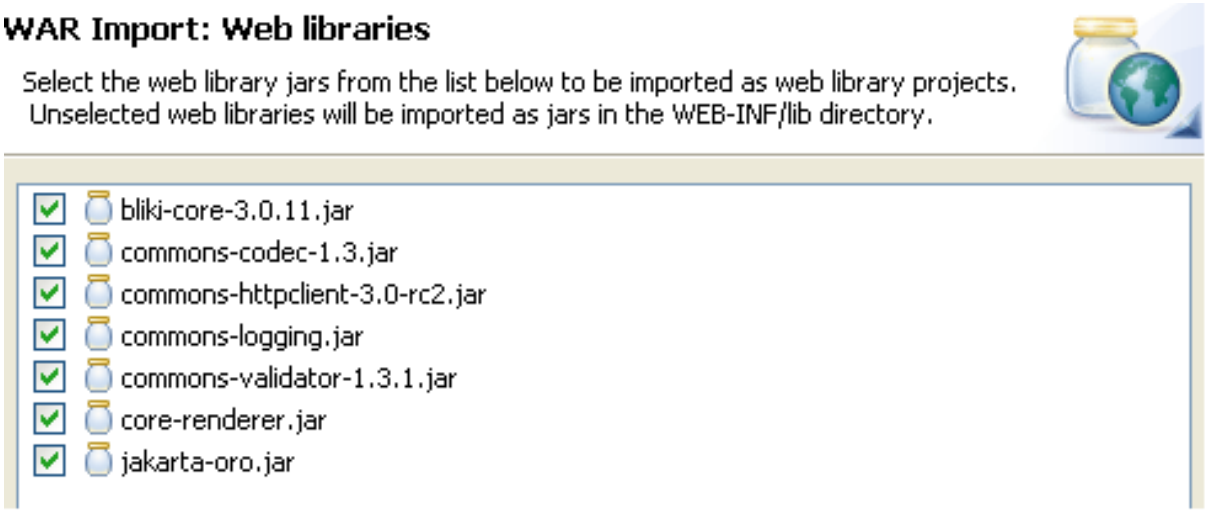

Figura 7.2: Importar las librerías.

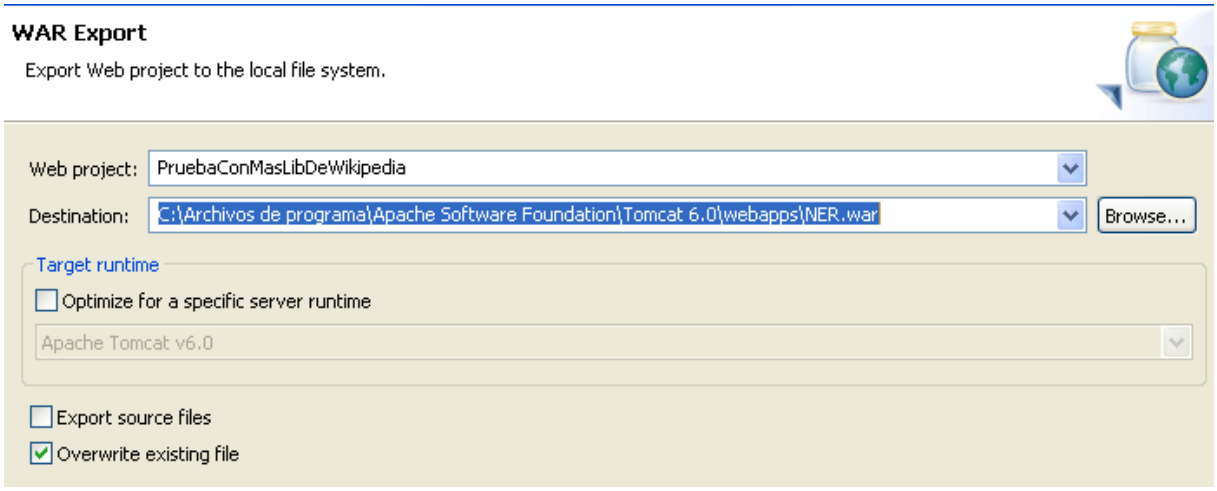

Figura 7.3: Generar fichero WAR mediante Eclipse.

a analizar. En cuanto al módulo encargado de la traducción automática de EN, su clase principal es la clase *Consulta*, y requiere de tres parámetros: El primero corresponde al idioma de origen de la EN a traducir, el segundo corresponde a una o un listado de EN separadas por un salto de línea, y el tercero atañe al idioma de destino al que se quiere traducir dichas EN. Las correspondencias numéricas para el primer y el tercer parámetro son las siguientes: “1” para Castellano, “2” para Inglés, “3” para Italiano, “4” para Francés, “5” para Alemán, “6” para Holandés y “7” para Portugués.

Anexo 2

Contenidos del CD

A continuación se citan los elementos incluidos en el soporte digital que acompaña a la memoria:

- Código de la aplicación web.
- Librerías necesarias.
- Noticias originales en formato XML.
- Memoria del PFC en formato PDF.
- Fichero en formato excel con los resultados experimentales obtenidos.
- Noticias resultantes en formato XML.
- Modelos utilizados para crear los distintos modelos estadísticos.
- Ficheros de entrenamiento utilizados para generar los modelos.
- Entorno de ejecución Java JRE (*Java Runtime Environment*)

Anexo 3

Listado de palabras gatillo

A continuación, en la Tabla 7.1 y en la Tabla 7.2 se exponen las palabras gatillo incluídas para los idiomas Inglés, Castellano, Italiano y Francés:

Inglés		**Castellano**	
Ambassador	Mademoiselle	Abogado	Primer Ministro
Architect	Major	Arquitecto	Princesa
Baron	Marchioness	Barón	Profesor
Baroness	Marquess	Brigadier	Príncipe
Begum	Master	Canciller	Reina
Brigadier	Minister	Capitán	Rey
Captain	Miss	Comandante	Senador
Chancellor	Mister	Conde	Señor
Colonel	Mistress	Condesa	Señora
Commander	Monsieur	Coronel	Señorita
Consul	Mr	Cónsul	Teniente
Count	Mrs	Diputado	Vicepresidente
Countess	Ms	Doctor	Vizconde
Dame	President	Don	
Deputy	Prince	Doña	
Doctor	Princess	Duque	
Duchess	Professor	El Honorable	
Duke	Queen	Embajador	
Earl	Reverend	General	
Engineer	Senator	Ingeniero	
Judge	Sheikh	Jeque	
King	Sir	Juez	
Lady	The Honourable	Maestro	
Lawyer	Vicepresident	Marqués	
Lieutenant	Viscount	Mayor	
Lord	Viscountess	Ministro	
Madam	Wing Commander	Presidente	

Tabla 7.1: Listado de palabras gatillo para Inglés y Castellano.

Italiano		**Francés**	
Ambasciatore	Maestra	Ambassadeur	Prince
Architetto	Maestro	Architecte	Professeur
Avvocato	Maggiore	Avocat	Président
Barone	Marchesa	Brigadier	Reine
Baronessa	Marchese	Bégum	Vice-président
Cancelliere	Ministro	Capitaine	Vicomte
Capitano di fregata	Nobildonna	Capitaine de frégate	
Cavalierato	Nobiluomo	Chancelier	
Cavaliere	Onorevole	Cheikh	
Colonnello	Patrizia	Colonel	
Console	Patrizio	Comte	
Conte	Perito	Dame	
Coscritta	Presidente	Docteur	
Coscritto	Principe	Duc	
Dama	Principessa	Député	
Deputato	Professore	Général	
Dottore	Ragioniere	Géomètre-expert	
Dottoressa	Re	Ingénieur	
Duca	Regina	Juge	
Duchessa	Sceicco	L'honorable	
Generale	Senatore	Lieutenant	
Geometra	Tenente	Major	
Giudice	Visconte	Marquis	
Ingegnere	Viscontessa	Maîtresse	

Tabla 7.2: Listado de palabras gatillo para Francés e Italiano.

Bibliografía

[1] Procesamiento del Lenguaje Natural en la Inteligencia Artificial http://www.monografias.com/trabajos17/lenguaje-natural/lenguaje-natural.shtml

[2] Asociación Mexicana para el procesamiento del Lenguaje Natural http://www.cicling.org/ampln/

[3] Procesamiento del Lenguaje Natural http://es.wikipedia.org/wiki/Procesamiento_de_lenguaje_natural

[4] Balbontín Gutiérrez y Jose Javier Sánchez Martín. SPNER, Reconocedor de entidades nombradas para el español.

[5] FreeLing: http://garraf.epsevg.upc.es/freeling/

[6] Stanford Named Entity Recognizer: http://nlp.stanford.edu/software/CRF-NER.shtml

[7] Rembrandt: http://xldb.di.fc.ul.pt/Rembrandt/

[8] Balie: http://balie.sourceforge.net/

[9] YooName: http://www.infoglutton.com/named_entity_extraction.html

[10] POO: http://www.desarrolloweb.com/articulos/499.php

[11] Eclipse: http://www.eclipse.org/

[12] Cohen, William W., MinorThird: Methods for Identifying Names and Ontological Relations in Text using Heuristics for Inducing Regularities from Data. 2004

[13] OpenNLP: http://opennlp.sourceforge.net/

[14] LingPipe: http://alias-i.com/lingpipe/

[15] Piattini Velthuis M.G., Análisis Y Diseño De Aplicaciones Informáticas De Gestión. Una Perspectiva De Ingeniería Del Software, Editorial Ra-ma, 2003

[16] Jacobson I., Booch G., Rumbaugh J., El Proceso Unificado De Desarrollo De Software, Pearson Addisson-Wesley, 2000

[17] Cockburn, Alistair, Agile Software Development, Highsmith Series

[18] Beck, Kent, Extreme Programming Explained: Embrace Change, Addison-Wesley Professional, 1999

[19] Aho, Alfred V; Margaret J. Corasick, Efficient string matching: An aid to bibliographic search, Communications of the ACM. 1975

[20] James Gosling, Bill Joy, Guy Steele, y Gilad Bracha, The Java language specification, tercera edición. Addison-Wesley, 2005.

[21] De Castro Korgi, Rodrigo. El universo LaTeX, 2da edición, Universidad Nacional de Colombia, Facultad de Ciencias. Departamento de Matemáticas, Bogotá, 2003.

[22] FreeLing `http://www.lsi.upc.edu/~nlp/freeling/`

[23] Named Entity Tagger `http://l2r.cs.uiuc.edu/~cogcomp/asoftware.php?skey=NE`

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