Sistema de Clustering de Named Entities

Jesús Martínez Rodríguez

A mis padres y hermana, por su paciencia.

A Leticia por todo lo que significa para mi.

Sobre Esta Memoria

Sobre el Glosario de la Memoria

A lo largo del texto de la memoria, ocasionalmente aparecerán palabras con un asterisco a su lado*. Dichas palabras se encuentran definidas en el Glosario del Apéndice A. La mayoría pertenecen a términos de uso frecuente en el dominio de Procesamiento de Lenguaje Natural, pero que pueden resultar extraños para alguien que no esté acostumbrado. Se ha tomado esta decisión con la finalidad de que todo el mundo pueda comprenderla.

Sobre la Redacción de la Memoria

- Como editor de textos se ha utilizado Microsoft Office Word 2003[1].
- Para los dibujos, se ha utilizado la herramienta que proporciona Microsoft Office Word.
- Los gráficos incluidos en la memoria se han realizado con el programa de hojas de cálculo Microsoft Office Excel 2003[2].
- Los diagramas UML se han creado con Microsoft Office Visio 2003[3].
- La planificación y distribución de tareas del proyecto en el tiempo se han calculado con Microsoft Office Project 2003[4]
- Todo el proceso del proyecto se ha llevado a cabo en máquinas equipadas con Microsoft Windows XP[5].

[1] http://office.microsoft.com/es-es/word/default.aspx
[2] http://office.microsoft.com/es-es/excel/default.aspx
[3] http://office.microsoft.com/es-es/visio/default.aspx
[4] http://office.microsoft.com/es-hn/project/default.aspx
[5] http://www.microsoft.com/spain/windowsxp/default.mspx

Índice

1. Introducción 9
1.1 Contexto Histórico 9
1.1.1 La minería de textos 9
1.1.2 Las conferencias MUC. Nacimiento de las *Named Entities*. 11
1.2 Motivación 13
1.3 Reconocimiento y clasificación de Named Entities 14
1.4 Objetivo: Clustering de Named Entities 16
1.5 La aplicación SCNE 18
1.5.1 *Tokenizador* de textos 18
1.5.2 Reconocedor de *Named Entities* 19
1.5.3 *Clustering* de *Named Entities* 21
1.6 Organización de la memoria 23
2. Análisis de Requerimientos 25
2.1 ¿Qué hace el Sistema? 25
2.2 Requerimientos funcionales 27
2.3 Requerimientos no funcionales 31
3. Especificación 33
3.1 Modelo Conceptual 33
3.1.1 Diagrama de Clases 34
3.2 Casos de Uso 38
3.2.1 Diagrama de Casos de Uso 38
3.2.2 Descripción de los Casos de Uso 39
3.3 Modelo de Comportamiento 48
3.3.1 Diagramas de secuencia 49
3.3.2 Contratos de las Operaciones 49
4. Diseño 57
4.1 Arquitectura lógica del sistema 57

4.2 Capa de Presentación 58
4.2.1 Diseño 58
4.2.2 Pantallas de la aplicación 61
4.3 Capa de Dominio 69
4.4 Capa de Gestión de Datos 72
5. Implementación 77
5.1 Arquitectura del sistema 77
5.2 Herramientas utilizadas 84
6. Pruebas 85
6.1 Pruebas del sistema 85
6.2 Definición de parámetros 88
6.3 Evaluación 89
7. Planificación y valoración económica del proyecto 97
7.1 Análisis de tiempo de desarrollo 97
7.2 Valoración económica del proyecto 101
8. Conclusiones y trabajo futuro 103
8.1 Conclusiones 103
8.2 Trabajo futuro 104
9. Bibliografía 107
A. Glosario 109
B. Manual de Instalación SCNE 117
C. Manual de Usuario SCNE 121
Ejecución en modo gráfico 122
Ejecución en línea de comandos 132

Capítulo 1

1. Introducción

El proyecto presentado en esta memoria se centra en la minería de textos*, concretamente en la subtarea de reconocer, clasificar y hacer *clusters** de unidades lingüísticas llamadas *Named Entities*[6].

Antes de desarrollarlo considero interesante explicar qué es una *Named Entity**, sus orígenes y el proceso histórico que ha seguido hasta llegar al día de hoy.

1.1 Contexto Histórico

1.1.1 La minería de textos

La minería de textos tiene como objetivo tratar grandes colecciones documentales y extraer determinada información de ellas. Su origen se halla a principios de los años ochenta, tras constatar que:

- La mayor parte de la información se encuentra de manera textual,
- Dicha información reside en soportes digitales y,
- Debido al gran valor comercial que posee, interesa su extracción y clasificación.

[6] A lo largo del documento encontrarán palabras escritas en inglés que el autor considera que su traducción al castellano no existe o no es la más adecuada.

Teniendo en cuenta que extraer la información manualmente requería un gran esfuerzo humano tanto de tiempo como de recursos, la comunidad científica dedicada al Procesamiento de Lenguaje Natural* (PLN) mostró su interés por esta problemática y empezó a desarrollar técnicas para resolverla.

Inicialmente la investigación se centró en dos campos:

Recuperación de Información (*IR*[7]):

Sistemas que, a partir de una consulta formulada por el usuario, seleccionan documentos* de una cierta colección que, según ciertos criterios, pueden responder mejor a las necesidades de la consulta.

Ejemplo de estos sistemas son los buscadores de Internet (Google[8], Yahoo[9], MSN[10] etc.) que, a partir de una serie de palabras clave, operadores booleanos* y otros criterios como la lengua, la fecha de publicación, el país del dominio y demás, devuelven una lista de documentos ordenados según la relevancia que el sistema considera que pueden tener, en función de parámetros como el número de apariciones de las palabras clave o enlaces que apuntan al sitio Web.

El principal inconveniente de estos sistemas es el hecho de que sea el usuario el que debe ir abriendo documentos para encontrar la información que realmente le interesa.

Extracción de Información (*IE*[11]):

Sistemas que procesan colecciones de documentos de un dominio* dado para extraer la información relevante de manera estructurada. Son sistemas hechos a medida y normalmente reciben una plantilla definida que deben rellenar con información extraída del texto.

Un ejemplo de este tipo podría ser un sistema que, teniendo como entrada las noticias de un diario, las relacionase y obtuviese los actos de un político concreto entre unas fechas dadas. A partir de esta información, se podría

[7] *Information Retrieval*
[8] http://www.google.com
[9] http://www.yahoo.com
[10] http://www.msn.com
[11] *Information Extraction*

generar una base de datos estructurada (es lo que se denomina Integración de la Información[12]) para agilizar las posteriores consultas sobre los actos de la persona.

El principal inconveniente de estos sistemas es la poca portabilidad que demuestran ya que las plantillas que utilizan están diseñadas para un fin concreto y son difícilmente reutilizables.

Hoy día, se le presta cada vez más atención a la minería de textos multilingüe debido a la proliferación de información multilingüe en Internet y la conveniencia de no restringir el rango de la búsqueda a documentos en la misma lengua en que se formula la consulta.

1.1.2 Las conferencias MUC[13]. Nacimiento de las *Named Entities*.

Las conferencias MUC sirvieron de punto de encuentro, evaluación y comparación de los diferentes sistemas de extracción de información.

Fueron organizadas por el Grupo de Investigación y Desarrollo Naval (*NraD*[14]) del *NCCOSC*[15], entidad dependiente de la Marina de los Estados Unidos de América.

La primera MUC se celebró en el año 1987 y fue una primera toma de contacto en la que se estableció un dominio de documentos: mensajes de operaciones tácticas navales. No estableció ninguna tarea ni criterio de evaluación.

Fue en la segunda conferencia celebrada el año 1989 cuándo se creó la tarea de obtener información. Se daba una plantilla con el hecho de un avistamiento de barcos. Tenía 10 campos, entre los que figuraban datos como la hora, el lugar, el efecto…, y los sistemas tenían que rellenarla con cada uno de los avistamientos que apareciesen en los documentos aportados.

Hasta la MUC-5 se mantuvo esta tarea variando el dominio, ampliando las plantillas, introduciendo medidas de comparación de rendimientos entre sistemas y añadiendo otros idiomas.

[12] *Information Integration*
[13] *Message Understanding Conferences (http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html)*
[14] *Naval Research and Development Group*
[15] *Naval Command, Control and Ocean Surveillance Centre*

1. Introducción

El término *Named Entity* (*NE*) fue introducido en Noviembre de 1995 en la MUC-6. En esta conferencia, a parte de la evaluación de los sistemas habitual, se abarcó la importancia del reconocimiento de la información. Los sustantivos que se refieren a entidades individuales (nombres propios de persona, nombres de organizaciones y nombres locativos entre otros) junto con las expresiones numéricas (fechas, tiempo, porcentajes, dinero) destacaban en esta área. La extracción de dichos elementos (denominados *Named Entities*) fue entonces reconocida como una de las subtareas más importantes en la identificación de entidades de un texto.

A partir de las conclusiones obtenidas y dada su importancia, las *NE* fueron ampliadas con la finalidad de poder adaptarse a los diferentes ámbitos de trabajo (biología, política, medicina, tecnología) dando lugar a las *Extended Named Entities**.

De todas formas, la clasificación de *NE* más utilizada continúa siendo la establecida por *MUC*:

- Nombres de Persona.
- Organizaciones.
- Nombres locativos (políticos o físicos).
- Expresiones numéricas fecha-tiempo.
- Otras *NE* (medidas -porcentajes, monetarias, pesos-, direcciones de correo, direcciones Web, etc.).

1.2 Motivación

Problemas actuales respecto a las Named Entities

A partir del momento en el que la clasificación de *Named Entities* se da por buena se generan los tres problemas siguientes:

Ambigüedad de *Named Entities*:

Existen diferentes palabras que, aún escritas de la misma forma, deben ser clasificadas como *NE* de diferente clase. Como ejemplo tenemos la palabra Washington que, en diferentes contextos, puede hacer referencia tanto a una ciudad de EEUU como a un nombre propio de persona. De igual modo, el número 1945 puede ser un año o bien un cardinal.

Ambigüedad con palabras comunes:

Hay palabras que pueden ser una *NE* (Zapatero como apellido es una *NE* de persona) y a la vez ser una palabra común del vocabulario (zapatero como oficio).

Variación de *Named Entities* que hacen referencia a la misma entidad:

Las entidades del mundo no son identificables únicamente de una manera sino que podemos hacer referencia a ellas de diferentes formas. Pongamos como ejemplo la persona José Pérez González, esta entidad también responde al nombre de José Pérez, Pérez González, José etc.

Es este último punto el que se pretende abordar en éste proyecto mediante la agrupación o *clustering** de *NE.* Para ello, en primer lugar se necesita identificar en un documento las *NE* existentes con un reconocedor y, mejor aún con un reconocedor-clasificador de *NE*.

1.3 Reconocimiento y clasificación de *Named Entities*

Las *Named Entities* se reconocen y extraen de un documento usando los llamados Reconocedores (*NER*[16]). Una vez reconocidas, se pueden clasificar en las diferentes categorías (personas, organizaciones, lugares, etc.) procesando la entrada con un Clasificador (*NEC*[17]). Como observación cabe destacar que la mayoría de sistemas extractores de *NE* son a su vez reconocedores y clasificadores y se conocen como *NERC*[18].

Algunos sistemas *NER/NERC*

Existe una gran cantidad de sistemas reconocedores o bien reconocedores-clasificadores de *NE*, normalmente son servicios ofrecidos por paquetes software de tratamiento del lenguaje natural (NLP[19]) más amplios. Entre ellos destaco:

NLTK[20] (*Natural Language Toolkit*):

Proyecto open source. Se trata de un conjunto de *scripts* en Phyton muy potentes, capaces de procesar el lenguaje natural simbólica y estadísticamente.

NLTK es una librería básicamente dedicada a la docencia. Las fuentes de conocimiento incluidas para las tareas de *NER* y *NEC* son bastante limitadas.

openNLP[21]:

Paquete de código abierto que incluye elementos funcionales para el tratamiento de lenguaje natural. Entre las herramientas que ofrece se encuentran un detector de oraciones, *tokenizador*, *NER* y un analizador

[16] *Named Entitiy Recognition*
[17] *Named Entity Classification*
[18] *Named Entity Recognition and Classification*
[19] *Natural Language Processing* (Tratamiento del Lenguaje Natural)
[20] http://nltk.sourceforge.net/
[21] http://opennlp.sourceforge.net/

morfológico. Está especialmente dirigido a aquellos desarrolladores interesados en crear interfícies humano-maquina.

YamCha[22]:

Paquete open source que proporciona al usuario herramientas de proceso de lenguaje natural, entre ellas la destinada a la tarea de reconocimiento de *Named Entities* y clasificación semántica.

FreeLing[23]:

Paquete software analizador de idiomas de código libre desarrollado por TALP*[24] de la UPC[25]. Este sistema es capaz de reconocer y clasificar las *NE* de lengua castellana gracias a las reglas integradas que posee del idioma. En inglés y catalán, FreeLing incluye *NER* pero no *NEC*.

Balie[26]:

Software open source desarrollado por la universidad de Ottawa en Canadá y presentado en Abril de 2004. Es un reconocedor y clasificador de *NE* que, a su vez, es capaz de identificar la lengua en la que el texto está escrito.

YooName[27]:

YooName es un software que se presenta como la evolución de Balie en el año 2007. Como su predecesor, es un software de reconocimiento y clasificación de *NE* basado en un aprendizaje supervisado*. Identifica 9 categorías de *NE* (divididas en más de 100 subcategorías), entre las que destacan personas, organizaciones, locativas, de producto, de eventos, de objetos naturales, de medida y de miscelánea.

Como características destacables dado su aprendizaje, la aplicación es extensible (puede reconocer nuevas *NE*), y un mantenimiento automático gracias a sus actualizaciones de bases de datos mediante conexiones a la Web.

[22] http://chasen.org/~taku/software/yamcha/
[23] http://www.lsi.upc.es/~nlp/freeling/
[24] Centre de Tecnologies i Aplicacions del Llenguatge y la Parla
[25] Universitat Politècnica de Catalunya
[26] http://balie.sourceforge.net/
[27] http://www.yooname.com/

1.4 Objetivo: *Clustering* de *Named Entities*

En lenguaje natural existen diferentes maneras de referirse a una misma entidad de la realidad y este hecho se refleja de igual modo en el texto escrito (podemos encontrar palabras como UPC y expresiones como Universitat Politècnica de Catalunya, ambas referidas a la misma entidad real).
Se pretende agrupar en un único *cluster** todas las expresiones que hacen referencia a las diferentes entidades. Dentro de esta tarea, se prestará más atención a la creación de *clusters* de *NE* de personas y de organizaciones así como las generales (sin clasificación previa). Se ha prescindido de un tratamiento específico de la tercera gran familia de NE, los topónimos (*locations*), ya que existen en la red numerosos *gazetteers** de topónimos que proporcionan básicamente la misma funcionalidad.

Se ilustra a continuación el objetivo con un ejemplo:
Jordi Pujol es el nombre un expresidente de la Generalitat de Catalunya, pero no se le hace referencia siempre de la misma manera. Podemos encontrar su persona escrita como Pujol, Jordi Pujol i Solei o Jordi Pujol.
Asimismo, la Facultad de Informática de Barcelona se conoce también como FIB y se puede encontrar en los escritos como FI Barcelona. Las tres expresiones hacen referencia a la misma entidad de organización.
Con esta premisa, queremos un sistema que, teniendo como entrada una colección de *NE*, genere como salida el *clustering* de éstas. Así, en los ejemplos anteriores tendremos:

Cluster Jordi Pujol: {Pujol, Jordi Pujol i Solei, Jordi Pujol}
Cluster FIB: {Facultad de Informática de Barcelona, FI Barcelona, FIB}.

El *clustering* es el propósito final y la parte más importante del presente proyecto. Por las características de la aplicación, el clustering debe ser aglomerativo. Para decidir si dos *tokens* deben ir en el mismo *cluster*, deben existir ciertas reglas que así lo indiquen. En nuestro caso estas reglas son medidas de distancia entre las palabras candidatas. Así, el primer paso debe ser la definición de las distancias entre elementos a agrupar, esto es, la

diferencia máxima aceptada para tener como resultado que dos *Named Entities* se consideran iguales (pertenecen al mismo *cluster*).

Podemos encontrar muchos tipos de distancias métricas textuales* entre unidades:

- Distribucionales (dos unidades, como dos palabras, son próximas si los contextos en los que aparecen son parecidos. Ej: comer-beber).
- Semánticas (la medida de proximidad se hace sobre espacio semántico. Ej: hiperonímia/hiponímia del WordNet).
- De Edición (cambios que se deben realizar en una palabra –inserciones, supresiones, modificaciones de letras- para hacerla igual a otra).
- De n-grama* (letras coincidentes, pares de letras, tríos de letras etc.)
- Entre longitud del máximo *substring** coincidente en ambas palabras. (otorgando diferentes pesos según sea vocales o consonantes o prefijos/infijos/sufijos etc.)

El problema base que nos encontramos es: dadas un conjunto de *NE*, decidir en función de las distancias entre ellas, si las juntamos en un *cluster* o no lo hacemos.

1.5 La aplicación SCNE[28]

SCNE es la aplicación desarrollada en este PFC y su objetivo es hacer *clustering* de *Named Entities*.

Su programación se ha planteado de forma modular*, dividiéndose el proceso general en subprocesos totalmente independientes que pertenecen, cada uno de ellos, a alguna de las diferentes fases que hay que recorrer para obtener un resultado.

Los distintos módulos que forman parte de la aplicación son:

- *Tokenizador*
- Reconocedor de *NE*.
- Clusterizador de *NE*.

1.5.1 *Tokenizador* de textos

SCNE es un sistema preparado para recibir como entrada:

- Un fichero de texto plano (.txt).
- Un fichero html/htm.
- Un conjunto de ficheros de texto plano (.txt).
- Un conjunto de ficheros html/htm
- Una URL de Web finalizada en htm/html.

Una vez seleccionada la entrada deseada, la aplicación debe segmentar el texto dado en unidades más pequeñas o *tokens*. Para ello se han previsto dos métodos de *tokenización*, la *tokenización* básica y la *tokenización FreeLing*.

Tokenizador Básico

El usuario introduce la ubicación de la entrada y escoge el tipo de unidad en el que quiere que se divida el texto entre palabras, frases o párrafos. Al finalizar el proceso, se devuelve el conjunto de *tokens* que se ha pedido.

[28] Sistema de *Clustering* de *Named Entities*

Tokenizador FreeLing

Dado que *FreeLing* se ha integrado con la aplicación SCNE, se ha optado por poder elegir segmentar el texto mediante *FreeLing*. Para ello se utiliza su propio *tokenizador*. La división del texto en este caso se hace únicamente en palabras existiendo la opción de reconocer palabras compuestas.

Como se puede observar, los dos métodos de hacer *tokens* son perfectamente compatibles y complementarios puesto que, donde no llega uno lo hace el otro. Así se cubre prácticamente toda la casuística de entradas.
Cabe destacar que el *tokenizador* básico es notablemente más rápido que el *tokenizador* FreeLing, entre otros motivos porque no se debe llamar a ningún *script* que ejecute externamente la aplicación.

1.5.2 Reconocedor de *Named Entities*

Una vez el *tokenizador* elegido ha hecho su función con el texto de la entrada, las *NE* existentes deben reconocerse y, opcionalmente (si queremos separar los tipos de *NE*) clasificarse para realizar el *clustering* final deseado.
Para reconocer *NE*, la aplicación igual que sucede con la *tokenización*, puede utilizar en esta ocasión tres módulos de reconocedores: el reconocedor general básico propio, el reconocedor general *FreeLing* o bien el reconocedor y clasificador de *NE* de *FreeLing*, éstos últimos como reconocedores externos.

Reconocedor Básico

El reconocimiento básico de *NE* se ha implementado de forma que, con unos simples heurísticos, por ejemplo si la primera letra de una palabra es mayúscula, se eligen las *NE* y se guardan en un fichero temporal para el siguiente paso. Este reconocimiento se puede utilizar para escoger las *NE* generales ya que no se hace ninguna clasificación y, posteriormente se puede hacer un *clustering* de *NE* general, sin especificar el tipo.

Reconocedor *FreeLing*

Aprovechando que tenemos la posibilidad de utilizar *FreeLing* podemos hacer un reconocimiento general de la misma manera que la *tokenización*. Igual que en el proceso anterior, al tener un *script* externo que ejecuta *FreeLing*, es algo más costoso que el reconocimiento básico pero, a su vez, mucho más acertado ya que *FreeLing* no utiliza heurísticos simples sino que se basa en el contexto del candidato y otras variables para la detección. Este reconocimiento permite la posibilidad de hacer un *clustering* de *NE* General debido a que no existe clasificación de las mismas

Reconocedor y Clasificador *FreeLing*

Con esta opción de SCNE se tiene como resultado las *NE* clasificadas en los distintos tipos. Gracias a ello tenemos la opción de *clusterizar* posteriormente las *NE* de personas, *NE* de organizaciones y de tipo Otras *NE*. Es el proceso que más tiempo consume puesto que también es el que más información y más detallada proporciona.

Igual que ocurre con la *tokenización*, los tres tipos de reconocimiento se complementan. Si se desea reconocer rápidamente *NE* genéricas sin necesitar más que un 60-70% de acierto de un texto muy largo o una colección extensa de documentos, es ideal utilizar el reconocimiento básico. Si, por el contrario se necesita que el acierto sea máximo sin importar el tiempo de ejecución, pensar en el reconocimiento general *FreeLing* es lo más acertado. Finalmente, si se desea un reconocimiento y clasificación de *NE* para trabajar sobre las de persona u organización, la elección del usuario debe ser el reconocimiento y clasificación *FreeLing*.

1.5.3 *Clustering* de *Named Entities*

A continuación se presentan los diferentes métodos de *clustering* usados para cada una de las distintas *NE*.

Clustering de *Named Entities* Generales

Se han visto en el apartado anterior los tipos de distancias que se pueden llegar a utilizar para decidir si dos *NE* entran en un mismo *cluster* o no lo hacen. En este PFC, la aplicación deja libre la incorporación de nuevos sistemas de distancia aunque para hacer *clustering* de *NE* generales se ha optado por hacerlo en base a la distancia de edición entre los *tokens* candidatos. Para tal fin se normalizan* las medidas entre ellos teniendo así valores entre 1 –*tokens* totalmente distintos- y 0 –*tokens* idénticos. Partiendo de esa idea, se define un umbral* a partir del cual se deben agrupar o no los candidatos. El umbral será elegido por el usuario así como también el número máximo de *clusters* permitidos.

Clustering de *Named Entities* de Personas

Las personas normalmente vienen indicadas por el patrón: nombre + primer apellido + segundo apellido y, en ocasiones, se le antepone el título que éstas poseen (Dr., Sr. etc.). Con esta estructura se cubre la mayor parte de los casos, existiendo otros realmente imposibles de distinguir como si dos personas se llaman igual: aún siendo personas físicas diferentes se agruparán en el mismo *cluster*.

El representante del *cluster* en este caso de *NE* se forma con la estructura <Nombre , Apellido1, Apellido2>.

Clustering de *Named Entities* de Organizaciones

Debido a que en ocasiones las Organizaciones vienen indicadas no por su nombre, sino por un acrónimo que las representa, éste junto con el nombre de la organización pasan a ser los que forman el representante del *cluster* con la estructura <Acrónimo, Nombre>.

NOTA RESPECTO A LOS CLUSTERS:

En esta aplicación el número de *clusters* resultante puede ser muy elevado, es por eso que se debe tomar una decisión con respecto a ellos en cuanto a la comparación de las *NE* nuevas de la entrada y las ya existentes. Si la comparación se hace con absolutamente todas las *NE* existentes, el programa puede llegar a desbordarse porque se hace una cantidad enorme de comparaciones. Para evitarlo, no se miden las distancias con toda la información que se tiene sino con un elemento de cada *cluster* que equivale a su representante, el elemento con más información de todos los existentes en el *cluster*.

De esa manera, cada *cluster* esta identificado: por el número de *cluster* que posee y por su representante.

El *clustering* se ha centrado principalmente en el idioma castellano, aunque también se han hecho pruebas con los idiomas inglés y catalán.

1.6 Organización de la memoria

La memoria está estructurada siguiendo los pasos correspondientes a la ingeniería del software, que por otra parte son los que se han seguido en el desarrollo del proyecto.

En el capítulo *1.Introducción* se describen, a parte de la organización de la memoria, las motivaciones que han llevado a la decisión de realizar este proyecto, los objetivos a lograr en el desarrollo del mismo, un poco de historia sobre la minería de textos y una comparativa de aplicaciones reconocedoras de NE actuales así como una breve descripción de la aplicación creada.

En el capítulo *2.Análisis de requerimientos* se realiza un análisis detallado y una descripción informal de todos los requerimientos que debe cumplir el sistema a desarrollar, tanto funcionales como no funcionales y se especifican cuáles son las necesidades del usuario a cubrir.

En el capítulo *3.Especificación* se realiza una descripción formal y una especificación funcional en UML del sistema que se quiere construir. Se incluye el diagrama de clases (modelo conceptual), el modelo de casos de uso, los diagramas de secuencia y los contratos de las operaciones.

En el capítulo *4.Diseño* se describe la arquitectura del sistema y el diseño del sistema de ficheros.

En el capítulo *5.Implementación* se indican aspectos generales de la implementación y las herramientas utilizadas.

En el capítulo *6.Pruebas* se describen las pruebas realizadas para la validación del sistema. También se hace una evaluación de la aplicación en base a los resultados obtenidos y un estudio de los parámetros que se manejan.

En el capítulo *7.Planificación y Valoración Económica* se realiza un análisis y una estimación de los costes de desarrollo del proyecto.

En el capítulo *8.Conclusiones* se hace una reflexión de aspectos que han intervenido en el desarrollo del proyecto y posibles modificaciones a realizar en el futuro.

En el capítulo *9.Bibliografía* se muestra una relación de libros, manuales, páginas Web y otras fuentes consultadas.

Finalmente se encuentran los Anexos, el primero de ellos es el *Glosario* de la memoria, el segundo pretende ser un *Manual de instalación* de la aplicación sobre plataformas Windows y el tercero un *Manual de Usuario* de la aplicación creada.

Capítulo 2

2. Análisis de Requerimientos

El desarrollo del proyecto se inicia con el estudio previo y el análisis de los requerimientos del nuevo sistema para diseñar una arquitectura que los cumpla y satisfaga. El objetivo de esta fase es identificar y documentar lo que se necesita realmente que haga la aplicación.

2.1 ¿Qué hace el Sistema?

Para realizar el análisis, en primer lugar se ha consultado al usuario final cuáles son los requerimientos de la aplicación y, una vez reunidos, se ha diseñado un esquema con el resultado:

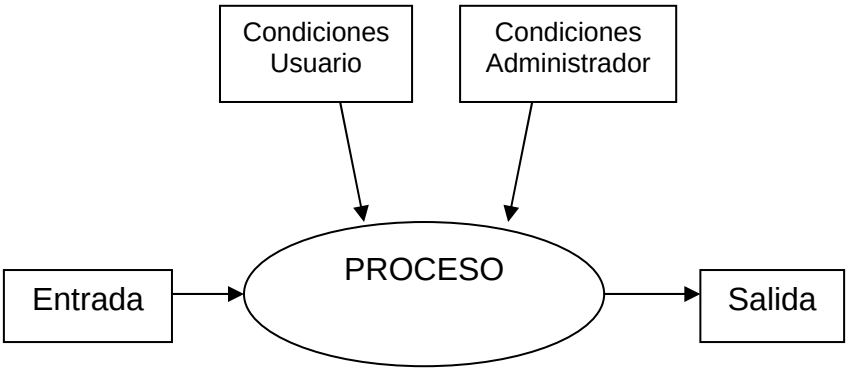


Figura 2.1: Esquema de funcionamiento del sistema

La aplicación debe aceptar como entradas documentos, colecciones de documentos o eventualmente, ficheros de *Named Entities* y, fijadas unas condiciones de funcionamiento por el administrador (configuración de la aplicación) junto con las condiciones requeridas por el usuario (tipo de proceso a realizar), se debe generar como salida los *clusters* correspondientes a dichas entradas.

Internamente el proceso del sistema se descompone en varios subprocesos individuales detallados a continuación.

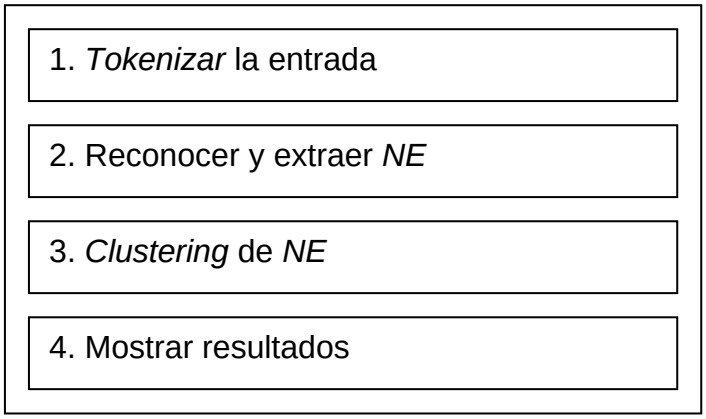


Figura 2.2: Subprocesos internos del sistema

1. *Tokenizar* la entrada: Dada una colección de documentos o un documento, en primer lugar se debe realizar la tarea de *tokenizar* el texto dividiéndolo en unidades simples. Debe existir la posibilidad de escoger que el proceso lo haga la aplicación externa *FreeLing* o bien mediante proceso básico.
2. Reconocer[29] y extraer *NE*: El reconocimiento y extracción de *NE* se realizará sobre la salida aportada por el proceso 1. De la misma manera que aquel, esta tarea se puede llevar a cabo mediante la aplicación *FreeLing* o con el proceso básico. En el caso de elección de

[29] El reconocimiento, según elija el usuario puede ser un reconocimiento General o bien un reconocimiento y clasificación de *NE*.

reconocimiento y clasificación de *NE*, lo realizará *FreeLing* con el módulo *NERC* que incorpora.

3. *Clustering* de *NE*: Proceso que se realizará sobre la salida del reconocimiento y clasificación en el caso de colecciones o documentos, o bien directamente sobre la entrada si esta es un fichero de *NE*. Se encargará de hacer aglomerativamente los *clusters* resultantes que se presentarán al usuario.
4. Resultados: La aplicación permitirá al usuario realizar consultas sobre los resultados obtenidos y los presentará por pantalla. En el caso de ejecución por línea de comandos, este punto se omite dado que no existirá interfaz gráfica.

Perfiles de usuarios

En el sistema existirán tres actores, dos corresponden a perfiles de usuario diferenciados: el Usuario Final y el usuario Administrador.

Por otro lado existirá un tercer actor, *FreeLing,* que representa la aplicación externa utilizada y que actuará siempre que el usuario requiera alguno de los procesos que ofrece.

2.2 Requerimientos funcionales

Los requerimientos funcionales especifican qué tiene que hacer el sistema.

Funcionalidades de la aplicación:

Las funcionalidades que debe ofrecer la aplicación al usuario Final son:

- Consultar el resultado de la *tokenización* del fichero de texto inicial.
- Consultar cuántas y qué *NE* se han reconocido del texto inicial. Se deben poder consultar las generales y, en el caso que se hayan clasificado, la consulta debe extraer las *NE* por el tipo que el usuario desee.
- Listar todos los *clusters* existentes.
- Dada una *NE*, listar los *clusters* en los que se encuentra.

- Dado un representante de *cluster*, listar las *NE* que lo forman.
- Dado un número de *cluster*, listar todas las *NE* que lo forman.
- Creación de nuevos *clusters* o bien realimentar los *clusters* existentes.
- Introducir como entrada una URL, un fichero o bien un directorio junto con la extensión de los ficheros que se desea procesar.

El usuario Administrador, además de cubrir las acciones del usuario final debe también poder:

- Iniciar sesión en el área de configuración.
- Configurar las rutas de datos de la aplicación
- Configurar los caracteres que harán la función de separadores en el *tokenizador* básico.
- Configurar los parámetros de proceso de *FreeLing*.
- Cambiar la contraseña de entrada al área de configuración.
- Realizar un mantenimiento de *clusters*, consultando, añadiendo, modificando o eliminando tanto los *clusters* como las propias *NE* existentes en cada uno de ellos.

Requerimientos funcionales de datos:

- Entrada: El sistema de *clustering* de *NE* trabajará con textos planos* o bien con textos codificados* en HTML[30] (tal y como se representan los ficheros de páginas Web). El texto plano será aquel formado por cadenas de caracteres ASCII[31] del cual nos interesa extraer e identificar las *Named Entities* deseadas. El texto codificado, por su parte, será aquel que se ha extraído de una URL[32] o bien se ha guardado previamente en el disco después de consultar la Web. Este tipo de ficheros necesita en primer lugar convertirse en textos planos. Para tal fin, se hará un pre-proceso eliminando las etiquetas existentes.

[30] *Hyper-Text Markup Language*
[31] *American Standard Code for Information Interchange*
[32] *Uniform Resource Locator*

- Nivel de procesamiento: Para el proceso de *clustering*, y dependiendo de la lengua del texto de entrada, se proporcionan tres procesadores: uno genérico aplicable a cualquier *NE* y dos específicos aplicables a personas y organizaciones.

- Salida: El resultado de las diferentes ejecuciones del sistema debe guardarse en ficheros para posteriores consultas o actualizaciones. Para tal fin se diseñará un sistema de ficheros dónde, en el directorio destino, se creará un fichero índice de *clusters* que se complementará con los n ficheros pertenecientes a los *clusters* y que contendrán las *NE* agrupadas. Cada *cluster* podrá ser identificado por el número o por su representante.

- La aplicación tendrá un fichero de configuración: En dicho fichero están predefinidos los diferentes parámetros que utiliza: rutas de datos en las que se trabaja y los parámetros genéricos de los procesos. El fichero de configuración se puede adaptar a las necesidades de los usuarios en la medida que éstos pueden elegir en la pantalla principal los parámetros y, en el caso de entrar como administradores, pueden predefinir tanto esos parámetros como las rutas de la aplicación para futuras ejecuciones.

Requerimientos funcionales de proceso:

- *Tokenización* : La *tokenización* espera en la entrada un texto plano o codificado en HTML y dará como resultado un conjunto de *tokens* en otro fichero. En ambos casos se podrá utilizar el tokenizador básico o bien el tokenizador que proporciona *FreeLing*.

- Reconocimiento : El reconocimiento de las *NE* sigue la misma pauta que la *tokenización* dejando al usuario la elección de que la extracción de las *NE* las haga la herramienta *FreeLing* o bien el sistema de reconocimiento básico.

- Clasificación de *NE*: Se ha escogido *FreeLing* como aplicación *NERC* externa a SCNE. Con ello, se podrán localizar *NE* de tipos y se podrán hacer *clusters* tanto *NE* de Personas como *NE* de Organizaciones.

- *Clustering*: Los *clusters* se generaran de formas diferentes dependiendo del tipo que deban agrupar. Se harán en base a diferentes parámetros que el usuario introducirá en la pantalla principal o bien por línea de comandos*.

Comunicación con programas externos: *FreeLing*

En el presente PFC se ha utilizado la herramienta de reconocimiento de *NE FreeLing* así como también la posibilidad que tiene ésa aplicación para clasificar las diferentes *NE* que encuentra en el texto. Con ello se podrán localizar *NE* tanto generales como por tipos y se podrá hacer *clustering* de una manera más exacta cuando de *NE* de personas o *NE* de organizaciones se trata. La clasificación de *NE* es válida únicamente para el idioma castellano debido a que hoy por hoy la herramienta no las clasifica en otros idiomas. De todas formas, en inglés y en catalán es posible realizar un reconocimiento de *NE* Generales junto con su *clustering* sin clasificación de éstas.
Debido a este hecho, y para evitar que la aplicación se limite a agrupar por tipos las *NE* de persona, organizaciones y otros únicamente en castellano, se ha habilitado la opción de introducir un listado de *NE* ya clasificadas (proceso que se puede realizar con otros programas *NERC*) en ficheros de texto directamente al clasificador.
Tanto el modo de utilización de *FreeLing* como el proceso de los datos se describen en los siguientes capítulos de Diseño e Implementación respectivamente.

2.3 Requerimientos no funcionales

El nuevo sistema deberá ser:

- **Sistema reutilizable**

  Los componentes desarrollados en el sistema deberán poder ser aprovechados por otros sistemas de PLN (procesado de lenguaje natural) haciendo que las clases sean fácilmente integrables y sea posible llamar a la aplicación mediante *scripts*.

- **Sistema flexible**

  El sistema debe permitir al usuario la parametrización de diferentes opciones como la elección de fichero, elección de umbral máximo, elección de número máximo de clusters etc., así como la configuración y adaptación de los componentes de forma fácil y sencilla. Igualmente, las rutas de datos de la aplicación deben tener la posibilidad de ser escogidas por el usuario administrador.

- **Sistema extensible**

  El sistema debe ofrecer la posibilidad de poder introducir nuevas clases en él para el cálculo de distancias, nuevas reglas para el reconocimiento de acrónimos, aceptar entradas de más tipos de ficheros etc.

- **Sistema eficiente**

  El sistema debe realizar las tareas de una manera correcta y en el menor tiempo posible. Se debe evitar hacer tareas redundantes o innecesarias.

- **Sistema portable**

  El sistema debe poderse ejecutar en cualquier máquina ya sea sobre Windows, Linux etc. por lo que se opta por trabajar con el lenguaje de programación Java ya que con una instalación de la Máquina Virtual Java (*Java Virtual Machine*) se podrá ejecutar en cualquiera de dichos sistemas operativos.

- **Sistema *user-friendly****

La aplicación debe ser intuitiva y de fácil manejo. Es importante que el usuario tenga la sensación de que está frente a una aplicación que domina y que le guía en los procesos que desea hacer.

- **Sistema basado en metodologías de Ingeniería de Software***

El sistema de *clustering* se realizará siguiendo los pasos que dicta la ingeniería de software, los cuales podemos definir como:

- Análisis de requisitos
- Especificación
- Diseño y definición de la Arquitectura.
- Implementación.
- Pruebas y experimentación.
- Documentación

Esta metodología tiene como principales cualidades la reducción de costes y el aumento de la calidad del producto obtenido. Permite hacer la aplicación entendible para futuros desarrolladores tanto su estructura como su uso y ofrece soluciones estándar que se basan en patrones.

La puesta a punto del sistema incluye pruebas de test (*software testing*) y pruebas de validación (*software validation*). Las pruebas que se hacen permiten reducir errores y cubren en su totalidad las funcionalidades del software.

Las pruebas de validación permiten además una medida objetiva de la calidad del sistema.

Capítulo 3

3. Especificación

En este capítulo se describirán formalmente, utilizando metodología *UML*[33]*, el modelo conceptual, los casos de uso y los contratos de las operaciones correspondientes a las funcionalidades de la aplicación.

La especificación determina una serie de pasos que ayudan a describir el comportamiento del sistema. Es una de las partes fundamentales para la realización de un proyecto informático ya que, a partir de su resultado, se dará un enfoque u otro a la fase del diseño.

3.1 Modelo Conceptual

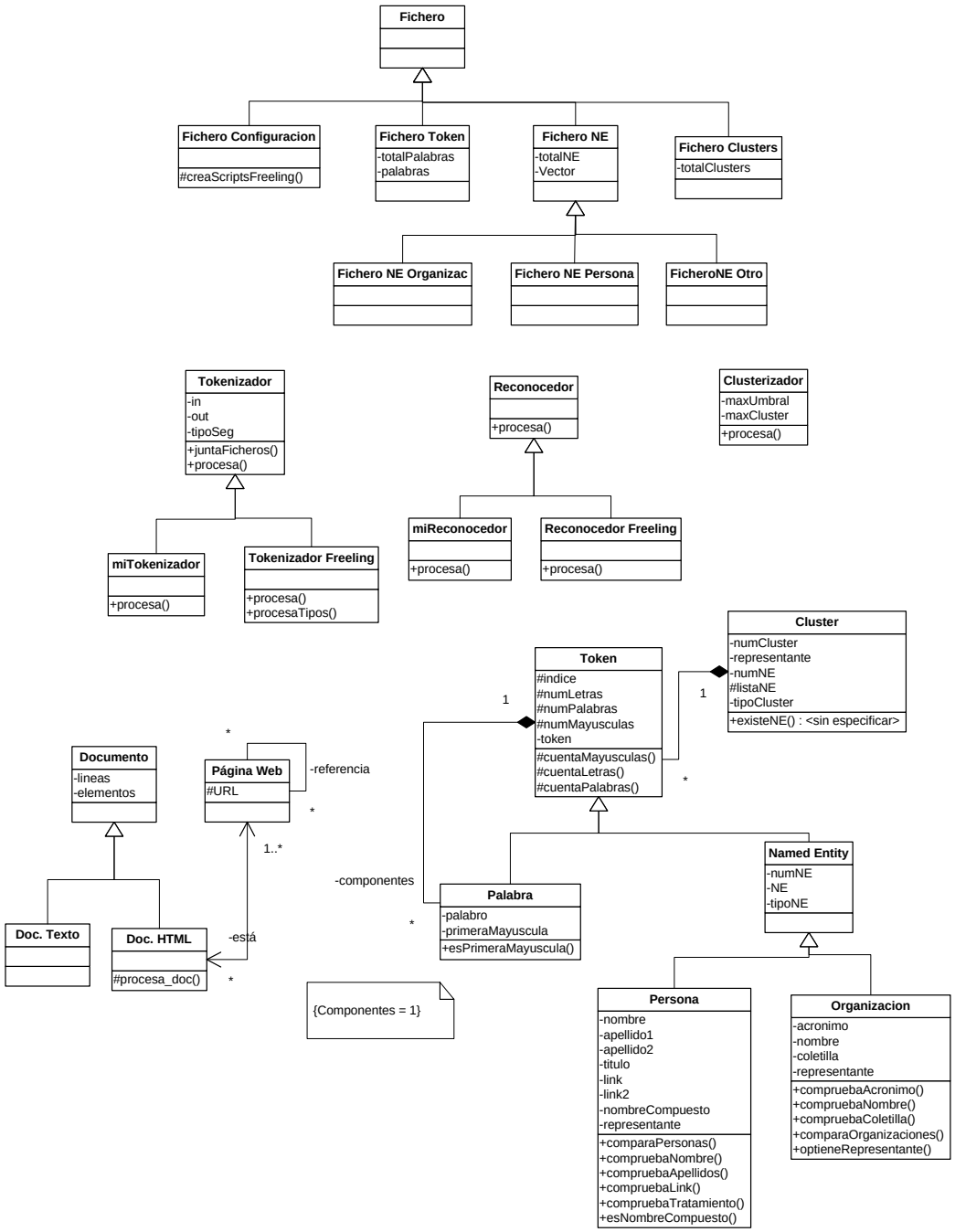
El modelo conceptual es la representación de los conceptos (objetos) de la realidad que son significativos en el dominio del sistema software que se desea desarrollar.

Este modelo consta de un diagrama de clases formado por clases de objetos existentes en el sistema, sus atributos y las relaciones entre estos (asociaciones) y también un conjunto de restricciones de integridad textuales que no se pueden expresar de manera gráfica con UML.

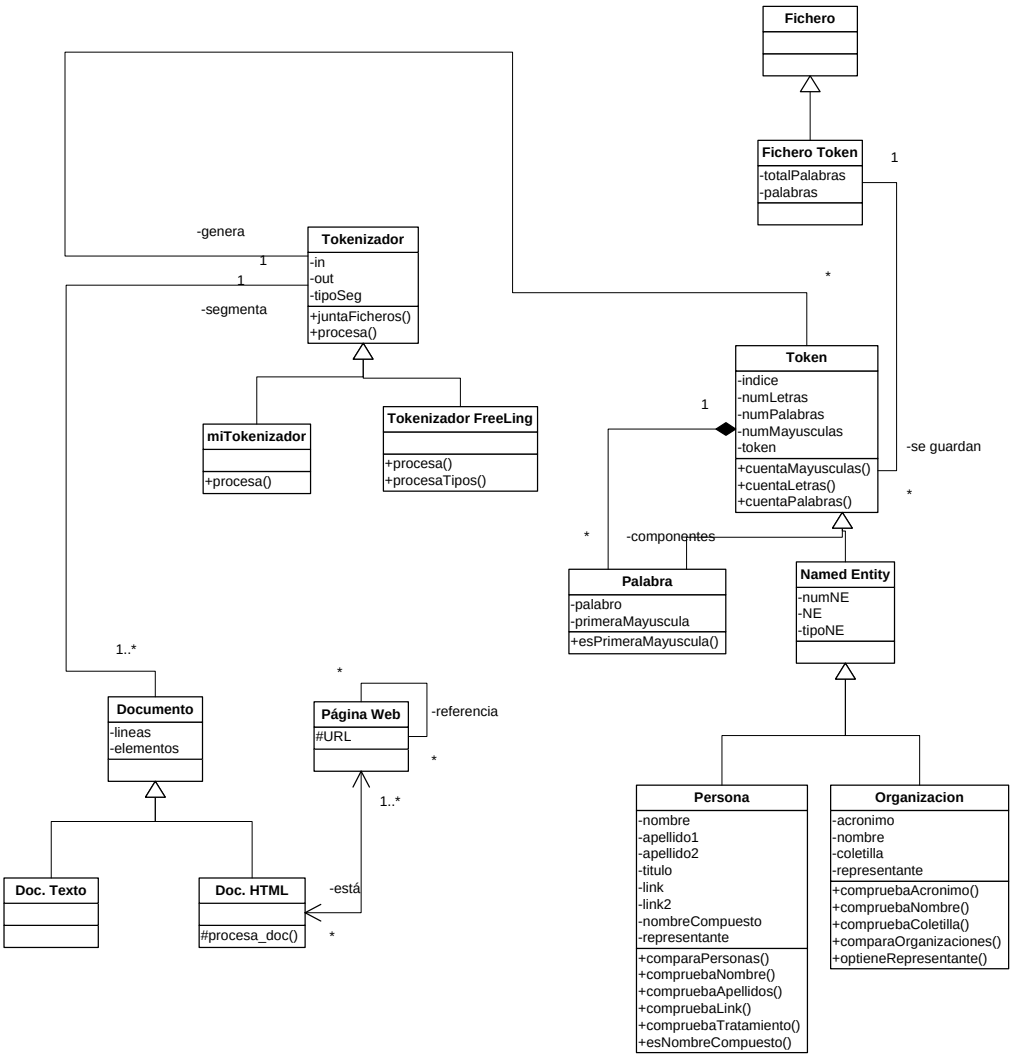
[33] *Unified Modeling Language* Lenguaje Unificado de Modelado

3.1.1 Diagrama de Clases

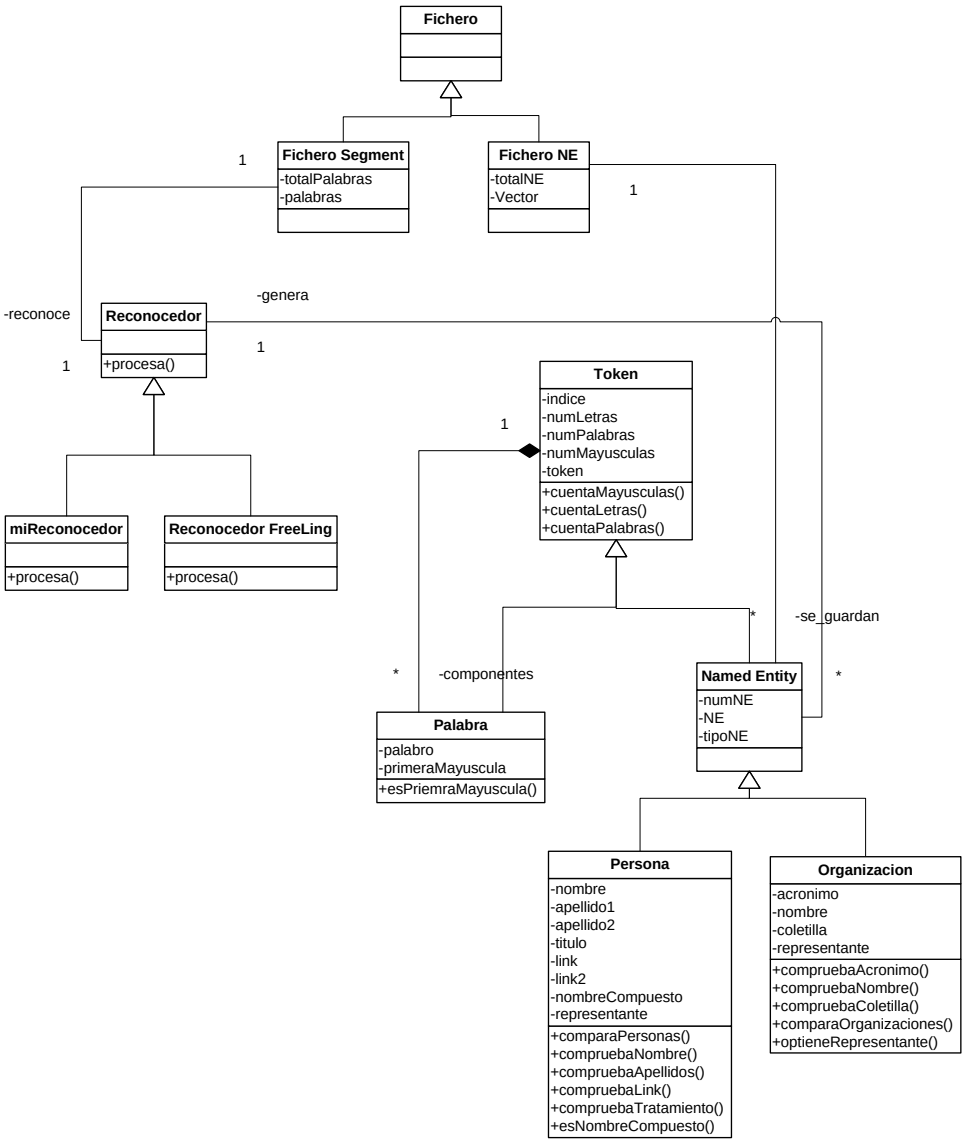
En la figura siguiente se pueden observar las clases del modelo sin ninguna asociación entre ellas. En las páginas sucesivas aparecen dichas relaciones en función de los procesos del sistema. Por otro lado, se indican los atributos y servicios más relevantes de las clases.



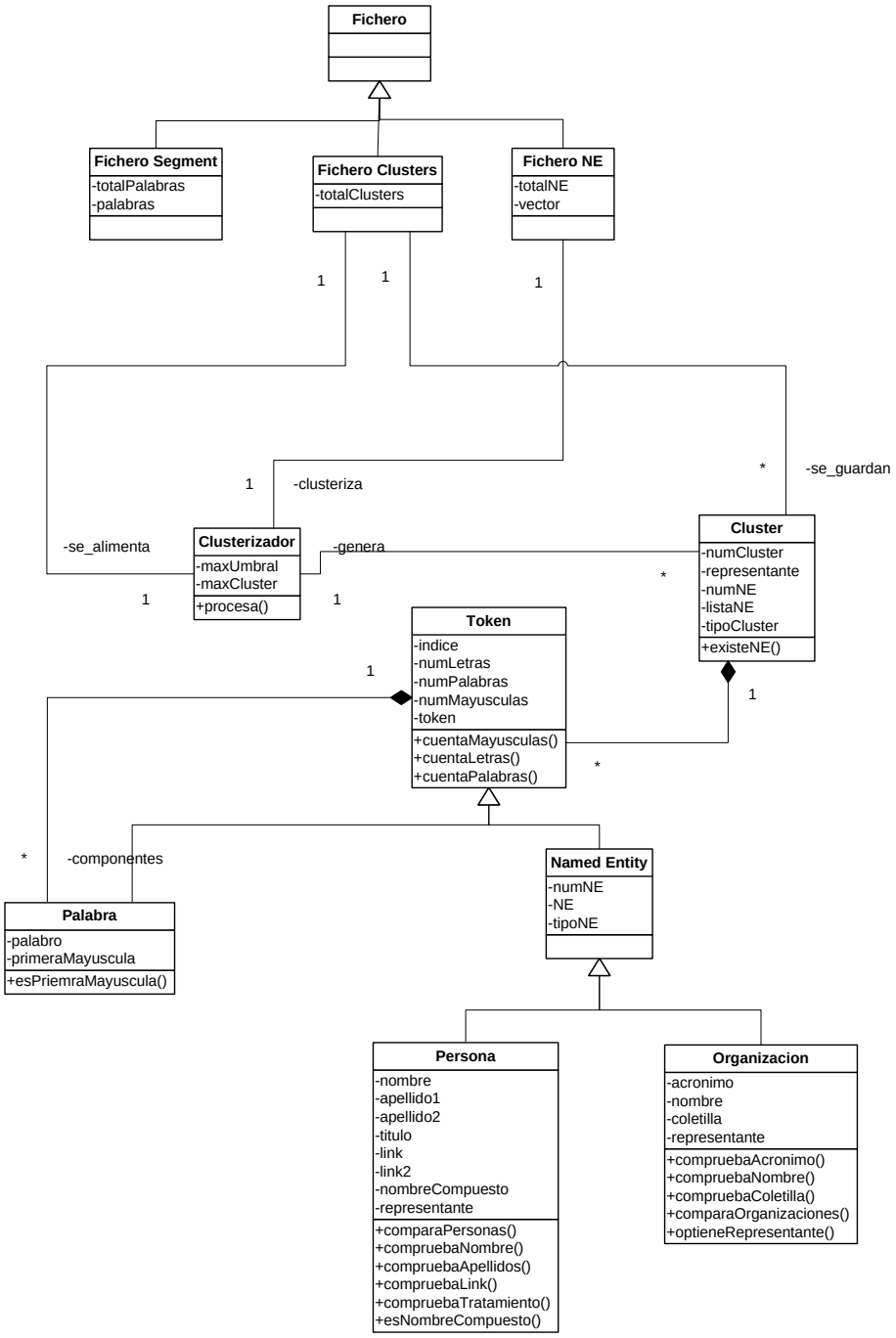
Tokenizador: Diagrama de clases que muestra las relaciones existentes entre los objetos que intervienen en el proceso de *tokenización*.



Reconocimiento: Diagrama que muestra las relaciones entre los objetos que intervienen a la hora de hacer el reconocimiento de *NE*.



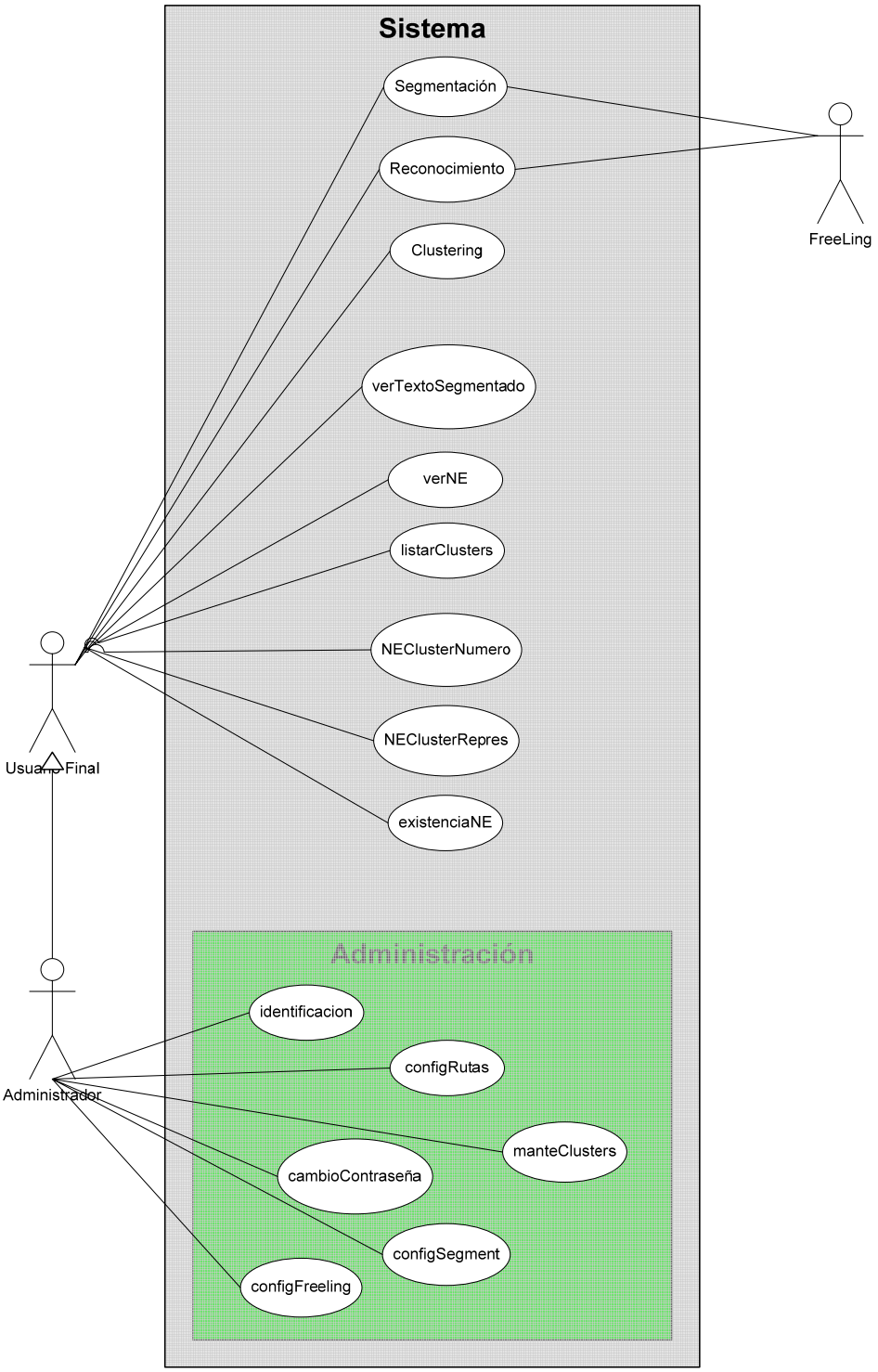
Clustering: Diagrama que muestra las relaciones entre los objetos que intervienen en el proceso de *clustering* de *NE*



3.2 Casos de Uso

3.2.1 Diagrama de Casos de Uso

El diagrama de casos de uso incluye todas las funcionalidades definidas en el análisis de requerimientos funcionales, junto con los actores que intervendrán.



3.2.2 Descripción de los Casos de Uso

La descripción de los casos de uso nos permite describir la secuencia de eventos que realiza un actor del sistema para llevar a cabo un proceso de su interés.

A continuación se detallan los casos de uso especificados en el diagrama anterior:

Casos de uso del Usuario Final

Caso de uso: *Tokenización*

Actores: Usuario Final.

Propósito: Dar como resultado la división en *tokens* de una entrada elegida por el usuario.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El usuario introduce manualmente o mediante una búsqueda en carpetas el origen y tipo de de datos deseados. 2.El usuario selecciona en las acciones a realizar *tokenización* pudiendo elegir entre *tokenización* de Básica o *tokenización FreeLing*. 3.El usuario pulsa Siguiente.	
	4.El sistema procesa la entrada. Al finalizar, deja los resultados en el destino predefinido en el fichero de configuración.

Cursos alternativos:

línea 1.Si no selecciona origen, el sistema envía un aviso en el punto 3.

línea 2.Si no selecciona ninguna acción a realizar se envía un aviso en el punto 3.

Caso de uso: Reconocimiento

Actores: Usuario Final.

Propósito: Dar como resultado el conjunto de *NE* que existen en el total de *tokens* extraídos de un texto.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El usuario selecciona en las acciones a realizar *tokenización* de un texto y reconocimiento NE.	
2.El usuario elige realizar el reconocimiento con *FreeLing* o bien con un reconocedor implementado para el sistema. El fichero de origen para el reconocimiento está predefinido en el programa y será el resultado de la *tokenización* por defecto.	
3.El usuario pulsa Siguiente.	
	4.El sistema procesa la entrada y deja los resultados en el destino predefinido en el fichero de configuración.

Cursos alternativos:

línea 1.Si no selecciona origen, el sistema envía un aviso en el punto 3.
línea 1.Si no selecciona ninguna acción a realizar se envía un aviso en el punto 3.
línea 1.Si no selecciona *tokenización* se envía un aviso en el punto 3.

Caso de uso: Clustering

Actores: Usuario Final.

Propósito: Dar como resultado un conjunto de *clusters* de *NE*.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1. El usuario selecciona en las acciones a realizar *tokenización* de un texto, reconocimiento de *NE* y *clustering*.	

2.El usuario elige el umbral a partir del cuál se consideraran los *clusters* y/o el número máximo de *clusters* permitido. Por defecto aparecen los valores predefinidos en el fichero de configuración 3.El usuario pulsa Siguiente.	
	4.El sistema procesa la entrada y deja los resultados en el destino predefinido en el fichero de configuración.

Cursos alternativos:

línea 1.Si no selecciona origen, el sistema envía un aviso en el punto 3.
línea 1.Si no selecciona ninguna acción a realizar se envía un aviso en el punto 3.
línea 1.Si la selección es diferente a tipo fichero *NE* y no se tienen marcados los procesos *tokenización* y reconocimiento de *NE*, el sistema envía un aviso en el punto 3.

Caso de uso: verTextoSegmentado

Actores: Usuario Final.

Propósito: Mostrar el texto resultante después de la *tokenización*.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario marca la opción de ver *tokens*.	
	2.El sistema presenta en una nueva ventana Java el resultado de la consulta.
3.El usuario puede consultar el resultado de la *tokenización* del texto.	

Cursos alternativos:

No existen.

Caso de uso: verNE

Actores: Usuario Final.

Propósito: Mostrar la *Named Entities* resultantes después del reconocimiento.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario final marca la opción de ver *NE* reconocidas.	
	2.El sistema presenta en una nueva ventana Java el resultado de las *NE* que se han reconocido.
3.El usuario puede consultar el resultado del reconocimiento.	

Cursos alternativos:

No existen.

Caso de uso: listarClusters

Actores: Usuario Final.

Propósito: Mostrar la lista de *clusters* totales.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario final marca la opción de Listar *Clusters* Existentes.	
	2.El sistema presenta en una nueva ventana Java la lista de clusters existentes.
3.El usuario puede consultar el resultado.	

Cursos alternativos:

línea 1.Si no se marca el tipo de cluster a consultar, el sistema envía un aviso en el punto 2.

Caso de uso: listarClusterRepres

Actores: Usuario Final.

Propósito: Mostrar la lista de *clusters* y *NE* de los mismos, que tienen en común el representante elegido.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario final marca la opción de listar *clusters* por su representante.	
	2.El sistema presenta en una nueva ventana Java la lista de *clusters* existentes junto con sus *NE* que coinciden con los parámetros introducidos.
3.El usuario puede consultar el resultado.	

Cursos alternativos:

línea 1.Si no se marca el tipo de cluster a consultar, el sistema envía un aviso en el punto 2.
línea 2.Si el representante no existe, el sistema lo informa mediante un mensaje.

Caso de uso: listarClusterNumero

Actores: Usuario Final.

Propósito: Mostrar la lista de *NE* que tiene un *cluster* elegido por su número.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario final marca la opción de listar *clusters* por su número.	
	2.El sistema presenta en una nueva ventana Java la lista de *clusters* existentes junto con sus *NE* que coinciden con la consulta elegida.
3.El usuario puede consultar el resultado.	

Cursos alternativos:

línea 1.Si no se marca el tipo de cluster a consultar, el sistema envía un aviso en el punto 2.
línea 2.Si el número de *cluster* no existe en sistema lo indica mediante un mensaje.

Caso de uso: existenciaNE

Actores: Usuario Final.

Propósito: Mostrar el número de *cluster* en el que existe la *NE* deseada.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario final marca la opción de comprobar existencia *NE*.	
	2.El sistema presenta en una nueva ventana Java la lista de *clusters* en los que existe la *NE* elegida y su número.
3.El usuario puede consultar el resultado.	

Cursos alternativos:

línea 1.Si no se marca el tipo de cluster a consultar, el sistema envía un aviso en el punto 2.
línea 2.Si la *NE* no existe como tal en ningún *cluster* el sistema lo informa por mensaje.

Casos de uso del Administrador

Caso de uso: identificación

Actores: Administrador.

Propósito: Entrar al área de configuración de la aplicación.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario administrador pulsa el botón de Configuración.	
	2.El sistema muestra una pantalla donde se pide usuario y contraseña de administrador.
3.El administrador introduce usuario y contraseña y pulsa aceptar.	
	4.El sistema valida la entrada y, si es correcta, muestra pantalla de configuración.

Cursos alternativos:

línea 4.Si usuario o contraseña no son correctos, el sistema lo informa mediante un mensaje.

Caso de uso: configRutas

Actores: Administrador.

Propósito: Cambiar la configuración existente de inicio en el fichero de configuración.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario administrador entra en el área privada de configuración.	
2.El administrador cambia las rutas y los parámetros tal y como desea editando	

los cambios en el fichero de configuración.
3.El administrador graba los cambios pulsando Aplicar en la pantalla de configuración.

6.El sistema guarda los cambios.

Cursos alternativos:

línea 3.Si quedan parámetros obligatorios en blanco, se envía un mensaje por pantalla.
línea 3.Si se desea salir sin guardar cambios, se cierra la ventana por el aspa superior derecha.

Caso de uso: configSegment

Actores: Administrador.

Propósito: Cambiar la configuración existente de separadores para el *tokenizador* básico.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario administrador entra en el área privada de configuración, en el apartado configurar *tokenizador* básico.	
2.El administrador realiza cambios agregando o eliminando caracteres separadores de la lista presentada tal y como desee.	
3.El administrador pulsa Guardar cambios.	
	4.El sistema guarda los cambios.

Cursos alternativos:

línea 3.Si se desea salir sin guardar cambios, se cierra la ventana con el aspa superior derecha

Caso de uso: configFreeling

Actores: Administrador.

Propósito: Cambiar la configuración existente de ejecución de la aplicación *FreeLing*.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario administrador entra en el área privada de configuración, en el apartado configurar *FreeLing*.	
2.El administrador realiza cambios agregando o eliminando opciones o cambiando los datos mediante desplegables tal y como desee.	
3.El administrador pulsa Guardar cambios.	
	4.El sistema guarda los cambios.

Cursos alternativos:

línea 3.Si se desea salir sin guardar cambios, se cierra la ventana por el aspa superior derecha.

Caso de uso: cambiarContraseña

Actores: Administrador.

Propósito: Cambiar la contraseña de entrada al área de configuración.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario administrador entra en el área privada de configuración, en el apartado cambio de contraseña.	
2.El administrador introduce su contraseña actual y la nueva deseada.	
3.El administrador pulsa cambiar.	
	4.El sistema guarda los cambios.

Cursos alternativos:

Línea 2.Si no coinciden las nuevas contraseñas o la antigua contraseña no es correcta, se informa al usuario.

Caso de uso: mantenClusters

Actores: Administrador.

Propósito: Agregar, eliminar o modificar tanto los *clusters* como las *NE* de cada uno de ellos.

Curso típico de sucesos:

Acción actores	Respuesta del sistema
1.El caso de uso empieza cuando un usuario administrador entra en el área privada de configuración, en el apartado mantenimiento de *clusters* existentes.	
2.El administrador realiza los cambios deseados identificando los *clusters* por su número y las *NE* por su nombre.	
3.El administrador pulsa agregar/eliminar.	
	4.El sistema realiza la acción deseada.

Cursos alternativos:

línea 3.Si se introduce algún dato no válido o no existente, el sistema lo informa por pantalla.

3.3 Modelo de Comportamiento

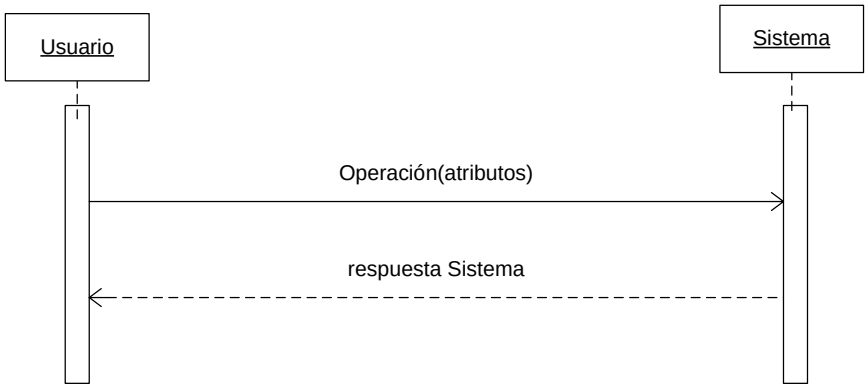
El modelo de comportamiento indica cómo tienen que actuar los actores con el sistema de una forma más concreta.

Este modelo consiste en:

- Diagramas de secuencia, muestran la secuencia de eventos entre los actores y el sistema
- Contratos de las operaciones, describen el efecto que tienen las operaciones del sistema.

3.3.1 Diagramas de secuencia

Los diagramas de secuencia del presente sistema siguen todos un mismo esquema debido a que la interacción entre usuario y sistema es directa y se realiza siempre de la misma manera. No se describirán detalladamente aunque a continuación se presenta un diagrama de secuencia genérico con el que se corresponden las operaciones del sistema.



3.3.2 Contratos de las Operaciones

Un contrato de operación es la especificación concreta de su comportamiento y efecto. UML define el comportamiento usando los conceptos siguientes:

Caso de uso	Caso de uso de la operación
Operación	Nombre de la operación del contrato en cuestión
Clase de retorno	Nombre y tipo de salida de la operación
Responsabilidades	Breve descripción del propósito de la operación
Precondiciones	Condiciones que deben cumplirse a priori para una correcta ejecución de la operación
Postcondiciones	Describe los efectos producidos por la ejecución de la operación
Salida	Descripción de la salida de la operación

Los contratos de las operaciones del sistema SCNE son los que se muestran a continuación:

Caso de uso	Tokenización
Operación	Procesa()
Clase de retorno	-
Responsabilidades	Se realiza la *tokenización* de un texto y, se escribe ésta en un fichero de forma estructurada.
Precondiciones	La entrada debe ser: -Un fichero con extensión .TXT o .HTML/.htm. -Una URL acabada en .html / .htm -Un directorio que contenga ficheros .html o .txt. Los ficheros con otra extensión no serán tratados.
Postcondiciones	Se genera el fichero resultante de la *tokenización* en el directorio indicado en la configuración. Se recargan los vectores de la clase ficheroToken.
Salida	Se muestra la pantalla de resultados al usuario para su posible consulta.

Caso de uso	Reconocimiento
Operación	Procesa()
Clase de retorno	-
Responsabilidades	Se realiza un reconocimiento de las *NE* de un texto genéricas o por tipo según se indique.
Precondiciones	-Se debe haber *tokenizado* previamente el texto. -Debe existir el vector de ficheroToken. -Actualmente el clasificador de *NE* es *FreeLing* y sólo realiza el proceso con textos en castellano.
Postcondiciones	Se genera el fichero de *NE* en el directorio indicado en el fichero de configuración. Se rellenan los vectores de *NE* de la clase ficheroNE.
Salida	Se muestra la pantalla de resultados al usuario para su posible consulta.

Caso de uso	Clustering
Operación	Procesa()
Clase de retorno	-
Responsabilidades	Realiza un *clustering* de las *NE* leídas del ficheroNE.
Precondiciones	-Debe indicarse el tipo de *NE* a clusterizar. -Debe indicarse si la entrada es un documento o bien un conjunto de *NE* y el idioma en el que está escrito. -Deben existir los vectores de *NE* recargados en ficheroNE.
Postcondiciones	Se generan tantos ficheros como *clusters* así como un fichero índice de estos en las carpetas indicadas según el tipo de *cluster* requerido.
Salida	Se muestra la pantalla de resultados al usuario para su posible consulta.

Caso de uso	verTextoSegmentado
Operación	escribeResultado()
Clase de retorno	-
Responsabilidades	Enseña en una nueva ventana Java el resultado de la *tokenización* de una entrada.
Precondiciones	El fichero debe existir y debe contener *tokens*.
Postcondiciones	-
Salida	Ventana nueva de Java

Caso de uso	verNE
Operación	escribeResultado()
Clase de retorno	-
Responsabilidades	Enseña en una nueva ventana de Java el resultado del reconocimiento de *NE* de una entrada.
Precondiciones	El fichero debe existir y las *NE* haber sido reconocidas
Postcondiciones	-
Salida	Ventana nueva de Java

Caso de uso	listarClusters
Operación	escribeResultado()
Clase de retorno	-
Responsabilidades	Enseña en una nueva ventana de Java la lista de *Clusters* totales
Precondiciones	Debe existir el fichero listaclusters.txt en el directorio destino de clusters.
Postcondiciones	-
Salida	Ventana nueva de Java

Caso de uso	listarClusterNumero
Operación	escribeConsutaPorNumero()
Clase de retorno	-
Responsabilidades	Enseña en una nueva ventana de Java la lista de *NE* del *cluster* deseado.
Precondiciones	Debe existir el *cluster* identificado con el número elegido.
Postcondiciones	-
Salida	Ventana nueva de Java

Caso de uso	listarClusterRepres
Operación	escribeConsultaPorNE()
Clase de retorno	-
Responsabilidades	Enseña en una nueva ventana de Java la lista de *NE* del *cluster* deseado.
Precondiciones	Debe existir el *cluster* identificado mediante el representante elegido.
Postcondiciones	-
Salida	Ventana nueva de Java

Caso de uso	existenciaNE
Operación	consultaNEClusters()
Clase de retorno	-
Responsabilidades	Enseña en una nueva ventana de Java la lista de *clusters* en los que existe la NE deseada.
Precondiciones	Debe existir la *NE* deseada.
Postcondiciones	-
Salida	Ventana nueva de Java

Caso de uso	configRutas
Operación	configurarRutasAplicacion()
Clase de retorno	-
Responsabilidades	Modificar las nuevas rutas de datos de la aplicación.
Precondiciones	Debe existir en la ruta c:\SCNE\ un fichero llamado configuración.txt
Postcondiciones	-
Salida	Se modifican las rutas del fichero de configuración tal y como lo indica el administrador.

Caso de uso	configSegment
Operación	configSegment()
Clase de retorno	-
Responsabilidades	Modificar los separadores del *tokenizador* básico
Precondiciones	Debe existir en la ruta c:\SCNE\ un fichero llamado configuración.txt
Postcondiciones	-
Salida	Se modifican los separadores tal y como lo indica el administrador.

Caso de uso	configFreeling
Operación	configuracionFreeling()
Clase de retorno	-
Responsabilidades	Modificar la configuración *FreeLing* tal y como se indica.
Precondiciones	Debe existir en la ruta c:\SCNE\ un fichero llamado configuración.txt
Postcondiciones	-
Salida	Se modifican los scripts de llamada a *FreeLing* tal y como lo indica el administrador.

Caso de uso	cambiarContraseña
Operación	cambiaPass()
Clase de retorno	-
Responsabilidades	Cambiar la contraseña del usuario administrador del sistema.
Precondiciones	Debe existir en la ruta c:\SCNE\ un fichero llamado configuración.txt
Postcondiciones	-
Salida	Se cambia la contraseña existente en el fichero de configuración.

Caso de uso	mantenClusters
Operación	MantenimientoClusters()
Clase de retorno	-
Responsabilidades	Cambiar los *clusters* existentes y las *NE* de cada uno de ellos siempre que el usuario lo requiera.
Precondiciones	Deben existir tanto los *clusters* como el fichero índice listacluster.txt en la carpeta de salida seleccionada
Postcondiciones	Los cambios realizados se reflejan en el sistema de ficheros.
Salida	Inserción o borrado en el sistema de ficheros.

Caso de uso	Identificación
Operación	Identifica()
Clase de retorno	-
Responsabilidades	Debe permitir o denegar el acceso al área de configuración de la aplicación como administrador.
Precondiciones	Debe existir en la ruta c:\SCNE\ un fichero llamado configuración.txt
Postcondiciones	-
Salida	Acceso a la pantalla de configuración de la aplicación.

Capítulo 4

4. Diseño

Una vez finalizada la especificación, donde se ha visto una descripción del comportamiento externo que debe tener el sistema desde el punto de vista del usuario o el entorno, la fase de diseño define la arquitectura que se utilizará para la construcción del sistema software y sobre que tipo de plataforma funcionará. Al empezar esta fase se debe tener elegida la tecnología con la que se desarrollará y de esta manera poder adaptar las especificaciones.

4.1 Arquitectura lógica del sistema

La arquitectura lógica del sistema será la resultante de aplicar el patrón arquitectónico de “Arquitectura en 3 capas”:

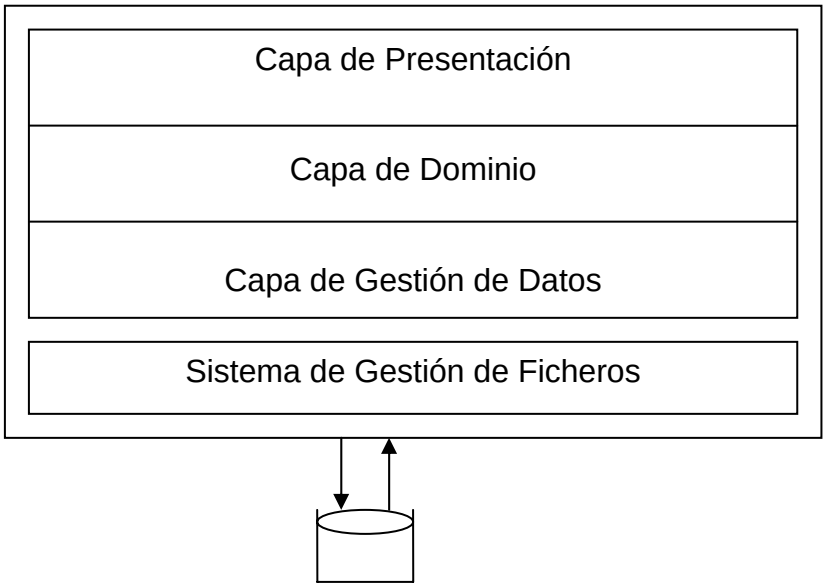


Figura 4.1: Arquitectura en 3 capas

Con esta arquitectura se tiene como principales beneficios el cumplir las características de sistema modificable, reusable, portable y verificable. En contrapartida, con este tipo de arquitectura es fácil perder eficiencia y hacer tareas redundantes. Estos inconvenientes se tendrán en cuenta y se diseñará la aplicación de manera que no suceda.

4.2 Capa de Presentación

4.2.1 Diseño

La capa de presentación tiene dos fases de diseño diferenciadas:

- Diseño externo: Define la interacción del usuario con el sistema software. Se diseñan los elementos que el usuario podrá ver y con los que podrá interactuar.
- Diseño interno: Se diseñan los mecanismos que recogen procesan y dan respuesta a las peticiones de los usuarios. Estos mecanismos permiten a la presentación comunicarse con la capa de dominio y transmitir las peticiones de los usuarios y los datos que este introduzca así como devolver los resultados.

Diseño externo

SCNE permite dos métodos de ejecución: mediante línea de comandos o utilizando la interfície gráfica.

- Línea de comandos

  La aplicación únicamente da mensajes de error al usuario si la entrada introducida no es correcta, informándole en todo momento de la causa del error. Una vez acabada la ejecución, el usuario debe ir a consultar los resultados a la carpeta correspondiente del sistema indicada en el fichero de configuración. Esta ejecución se hace secuencialmente y el usuario no interactúa con la aplicación.

Para facilitar el uso de este tipo de ejecución, se ha creado una ayuda que se puede consultar por pantalla.

- Modo gráfico

Los resultados de la ejecución se muestran gráficamente y la interacción con el usuario se hace mediante ventanas, botones, menú y diálogos. La interfície se procura que sea lo más usable posible ya que es la parte de la que dispone el usuario para interactuar con el sistema. Los mecanismos de presentación de la información deben ser claros y entendibles. Para ello se hace el proceso lo más guiado posible, reduciendo al máximo las acciones que el usuario debe conocer e informando al usuario con mensajes claros y explicativos.

Diseño interno

Para el diseño interno de la capa de presentación se ha utilizado Java Swing[34], paquete Java que ofrece soluciones tanto para la construcción de la interfície como para gestionarla.

Los componentes visuales más significativos que se han utilizado para la construcción de la aplicación se recogen en el siguiente esquema:

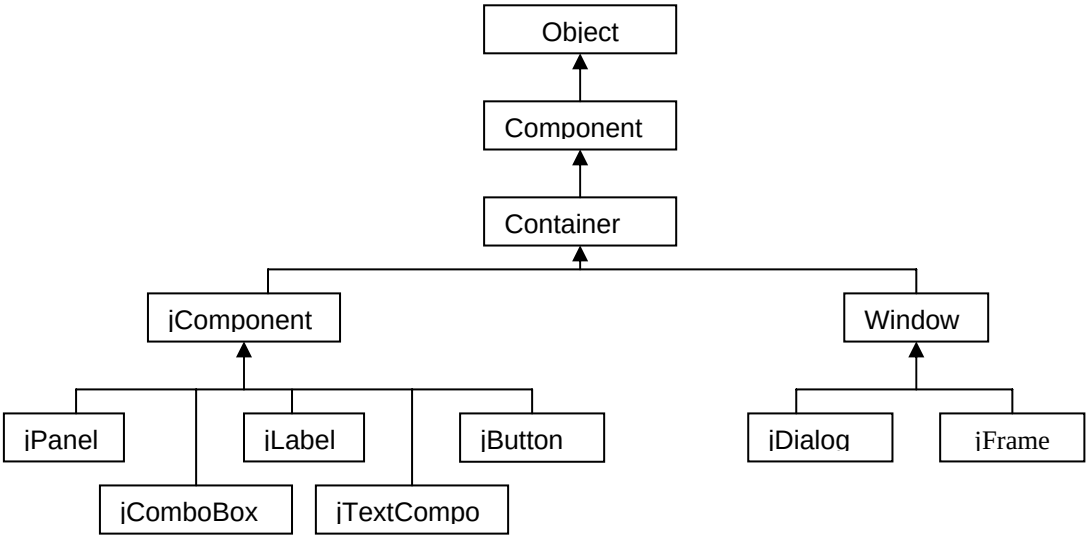


Figura 4.2: Esquema de componentes Java SWING

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

En cuanto a la gestión de la interfície, se sigue el modelo presentado en el siguiente esquema:

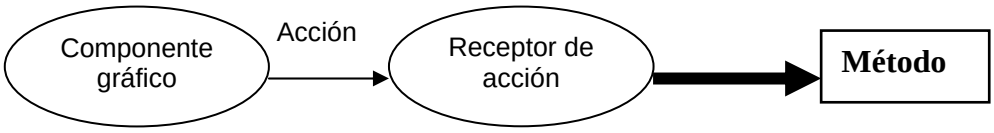


Figura 4.3: Esquema de gestión de la interfície

Componente gráfico: Objeto que cuando se produce un cambio sobre el comportamiento gráfico, lanza una acción. Ej: jButton, jFrame
Acción: Comunica la ocurrencia del cambio sucedido a todos los receptores. Ej: mouseEvent.
Receptor de acción: Es una interfície java con métodos definidos.
Método: Acción a realizar después de recibir la notificación de un cambio.

4.2.2 Pantallas de la aplicación

Con la metodología presentada se han creado las 8 pantallas de las que consta el sistema. A continuación se hace una breve descripción de cada una de ellas.

Pantalla principal

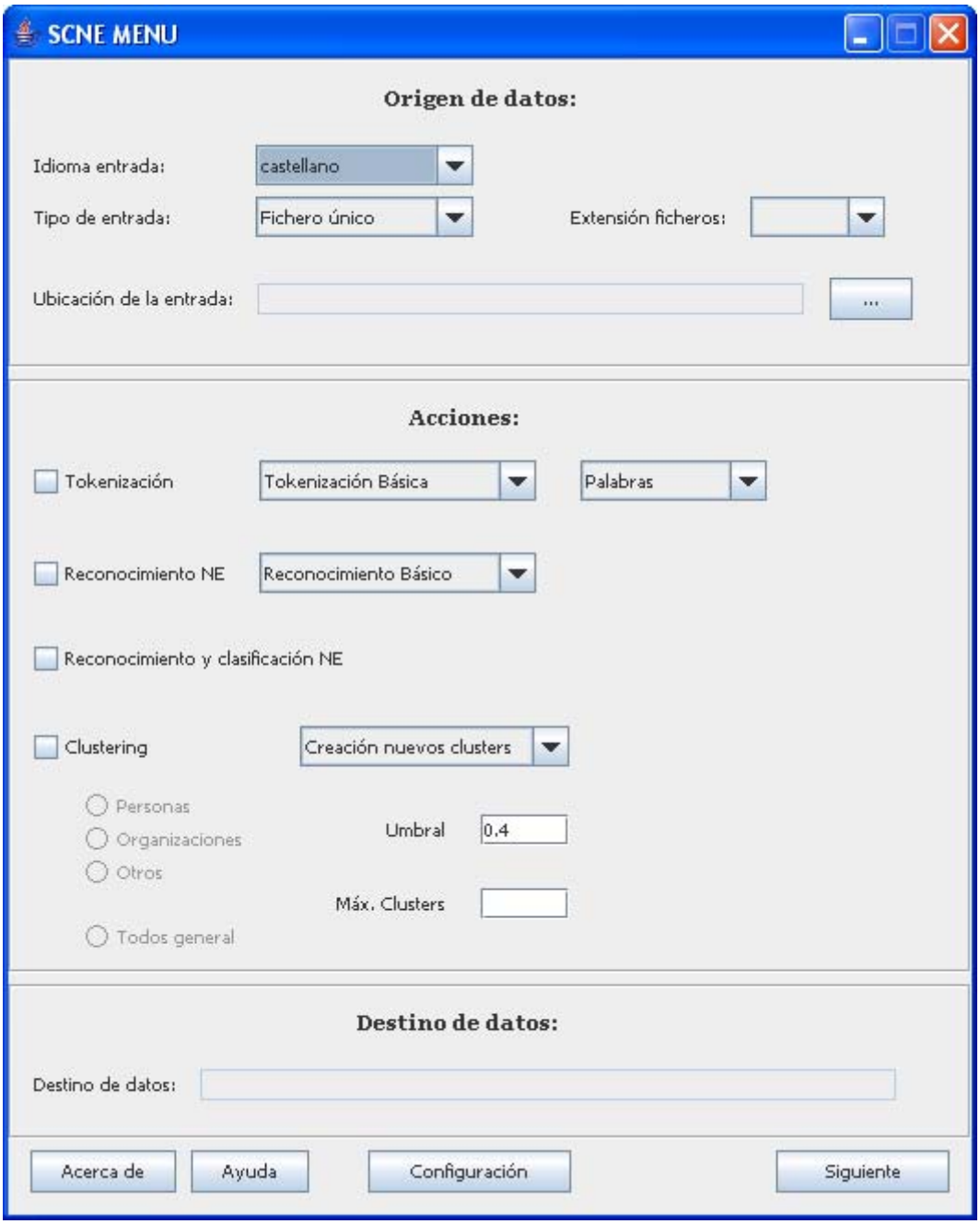


Figura 4.4: Pantalla principal de la aplicación

Es la pantalla más importante de la aplicación. Desde ella se inician tanto los procesos de datos de la aplicación como la configuración de SCNE según indique el usuario.

Es una pantalla intuitiva y sigue un esquema guiado de ayuda al usuario. Se da opción a escoger el idioma en el que se quiere que la aplicación trabaje, el tipo de entrada que se va a utilizar y la selección de la misma mediante un diálogo de ficheros accesible por el botón de búsqueda.

Se deben indicar también los procesos a ejecutar, sus opciones y los parámetros que se deben utilizar.

El destino de datos se indicará automáticamente en el caso de elección de 1 tipo de *NE*. Si los tipos son diferentes se mantiene el destino vacío aunque deja los resultados en los directorios indicados en la configuración.

La ejecución se inicia con el botón “Siguiente”.

Para acceder a la configuración de la aplicación por parte del administrador se debe pulsar el botón “Configuración”.

Mediante el botón “Ayuda” de la parte inferior de la ventana se accede al manual de usuario de la aplicación.

El botón “Acerca de” da información sobre la versión de SCNE así como del autor del mismo.

Pantalla de identificación

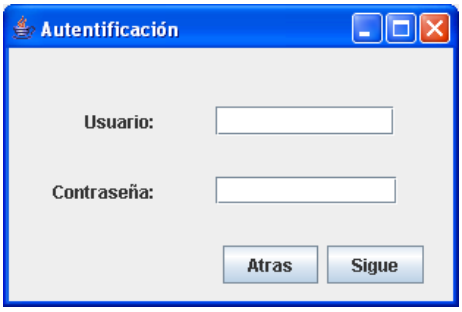


Figura 4.5: Pantalla de identificación

Esta pantalla a la que se accede pulsando el botón “Configurar” de la ventana principal, controla el acceso a la configuración de la aplicación mediante el usuario y la contraseña del usuario Administrador.

Una vez introducidos los datos se debe pulsar el botón “Sigue”. En caso que éstos no sean los correctos, se devuelve un mensaje al usuario indicando el motivo.
Si se desea abandonar el acceso a la configuración se debe indicar pulsando el botón “Atrás”.

Pantalla de configuración de la aplicación

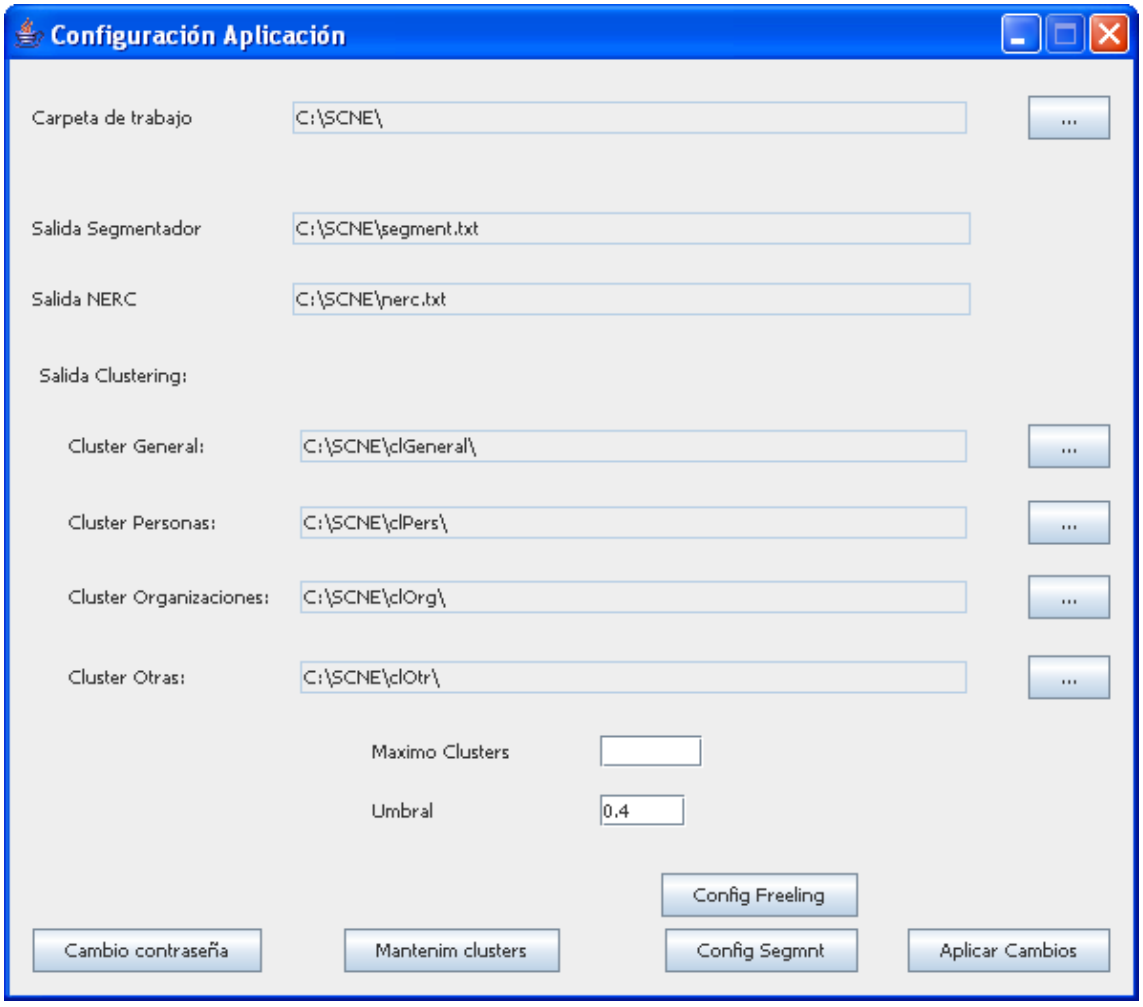


Figura 4.6: Pantalla de configuración de la aplicación

Una vez identificado el usuario correctamente como administrador después de pulsar el botón “Configuración” de la pantalla principal, aparece la pantalla menú de configuración de la aplicación. Desde aquí se pueden modificar las rutas de la aplicación que son controladas mediante una búsqueda guiada de ficheros o carpetas con la finalidad de evitar errores a la hora de definirlas.

Esta pantalla permite también el acceso a cambios de contraseña del administrador (botón “Cambiar contraseña”), hacer cambios en la configuración de las opciones *FreeLing* a la hora de detectar *NE* (botón “Config FreeLing”) , configurar los delimitadores de los tokens en el caso del *Tokenizador* básico (“Config Segmnt”) o bien a hacer un mantenimiento de *clusters* manual (“Mantenim Clusters”).
Una vez realizador los cambios deseados, y para que éstos tengan efecto, se debe pulsar el botón “Aplicar Cambios”. En el caso de no querer guardar lo realizado, se puede salir de la ventana mediante el aspa superior derecha.

Pantalla de cambio de contraseña

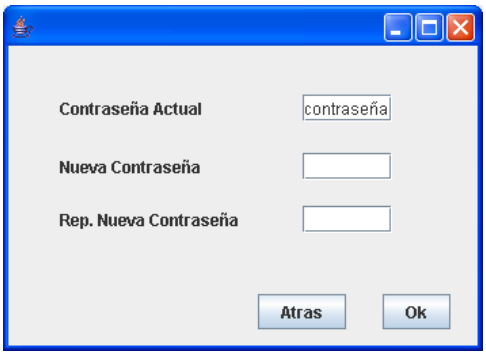


Figura 4.7: Pantalla de cambio de contraseña

Desde esta ventana el usuario administrador puede cambiar la contraseña de acceso a la configuración de SCNE. En el caso de error en las contraseñas introducidas, la aplicación indica al usuario el motivo. Haciendo clic en “OK” se hace el cambio de contraseña, si no se desea realizar ninguna modificación se debe pulsar el botón “Atrás”.

Pantalla de configuración FreeLing

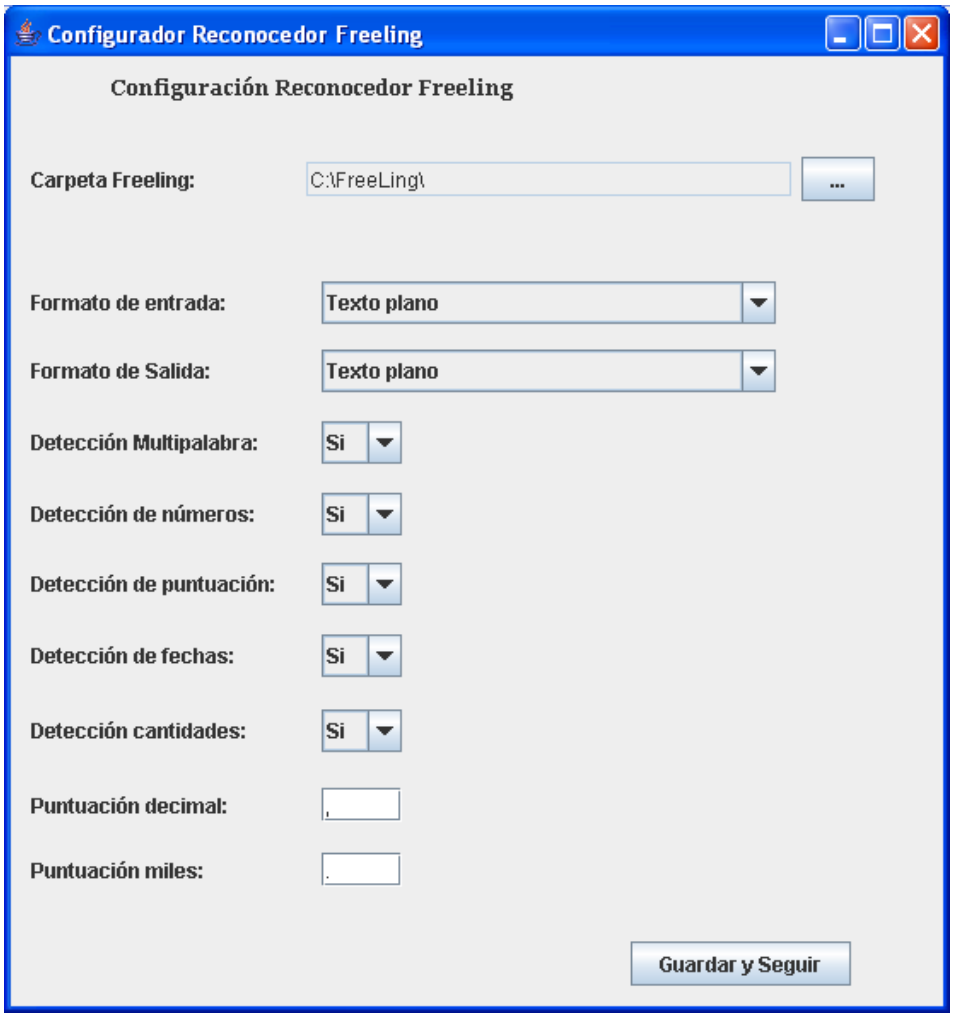


Figura 4.8: Pantalla de configuración FreeLing

Esta pantalla controla los diferentes parámetros usados por *FreeLing* en la tarea de reconocimiento de *NE*. Es una pantalla intuitiva y posee todas las opciones de *FreeLing* guardadas en menús desplegables con la finalidad de ser lo más sencilla posible.

Para que los cambios tengan efecto se debe pulsar “Guardar y Seguir”. Si se desea salir sin cambios se debe cerrar la ventana con el aspa superior derecha.

Pantalla de configuración Tokenizador Básico

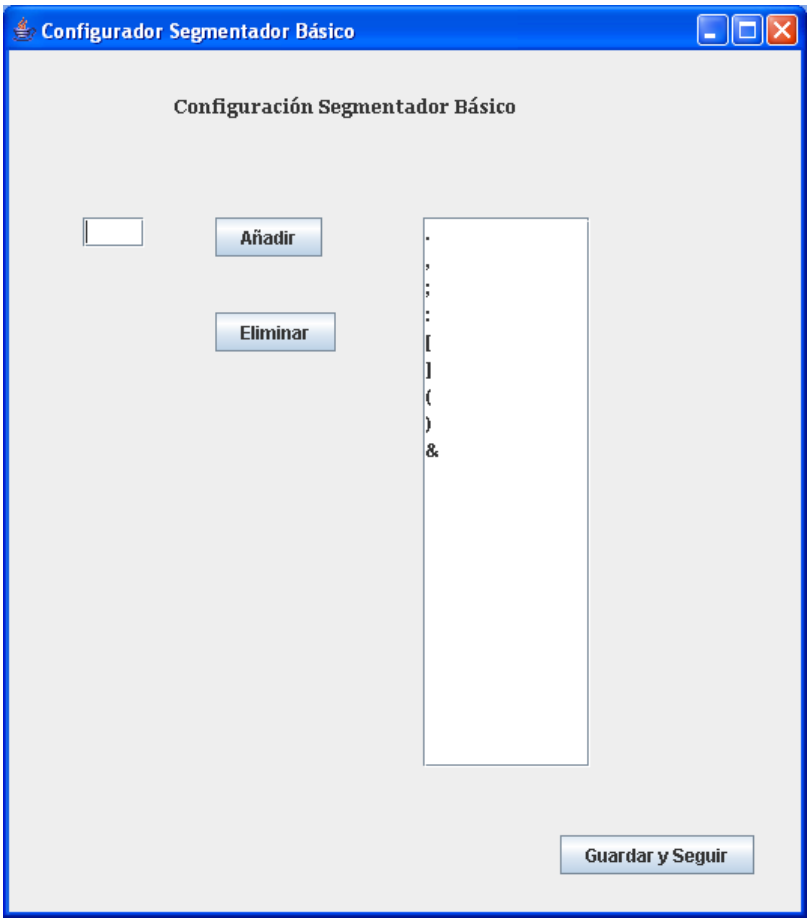


Figura 4.9: Pantalla de configuración del tokenizador Básico

Desde esta ventana se seleccionan los caracteres separadores que el tokenizador básico utiliza para realizar el proceso.

Permite la opción de indicar en primer lugar el carácter deseado e introducirlo posteriormente a la lista de caracteres mediante el botón “Añadir”. Si se desea borrar de la lista alguno de los delimitadores, se debe seleccionar y pulsar el botón “Eliminar”.

Igual que en el resto de las pantallas, se puede salir guardando los cambios marcando “Guardar y Seguir” o bien sin guardarlos cerrando la ventana.

Pantalla de mantenimiento de clusters

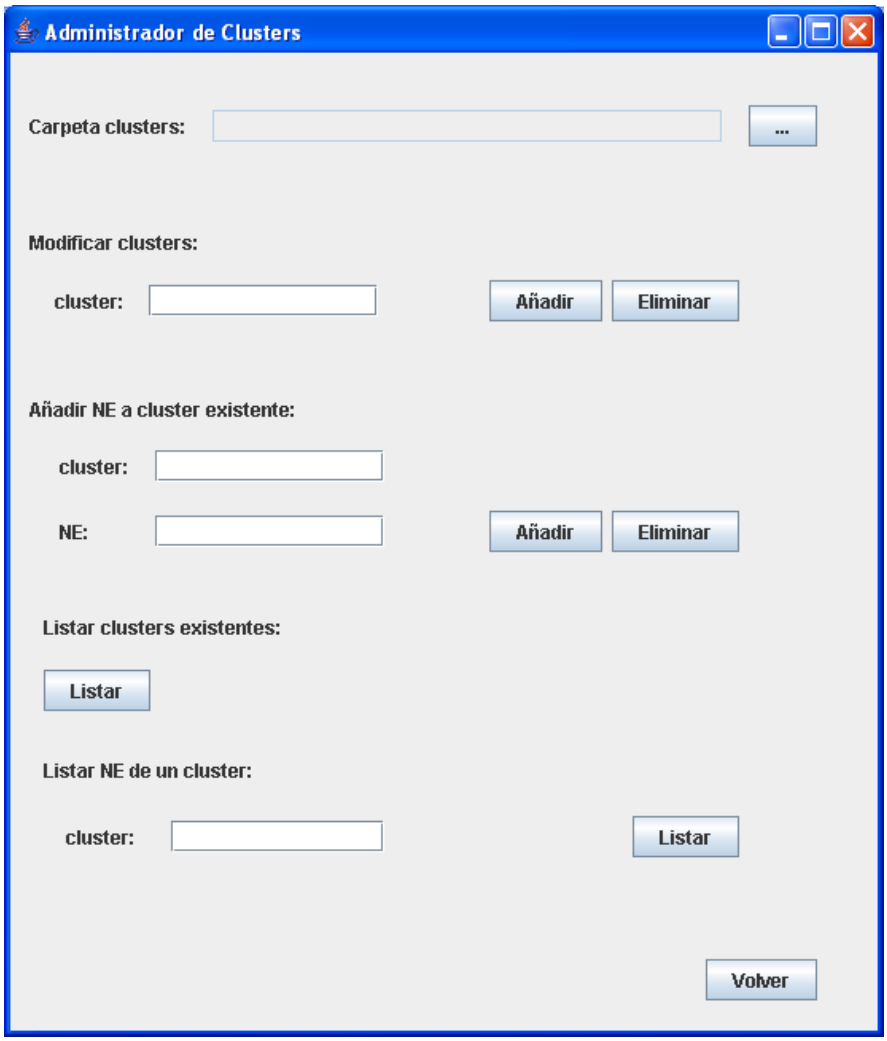


Figura 4.10: Pantalla de mantenimiento de *clusters*

Esta pantalla está diseñada para realizar un mantenimiento manual de los *clusters* creados por la aplicación SCNE. Con ello se pueden eliminar o añadir así como agregar o borrar *NE* de los *clusters* existentes.

Debe seleccionarse la carpeta contenedora de clusters y seguidamente indicar las acciones necesarias. En el caso de que la acción no sea permitida o no sea posible, se le informa al usuario junto con la causa.

Una vez hechos los cambios, se vuelve a la pantalla de configuración mediante el botón “Volver”.

Pantalla de resultados

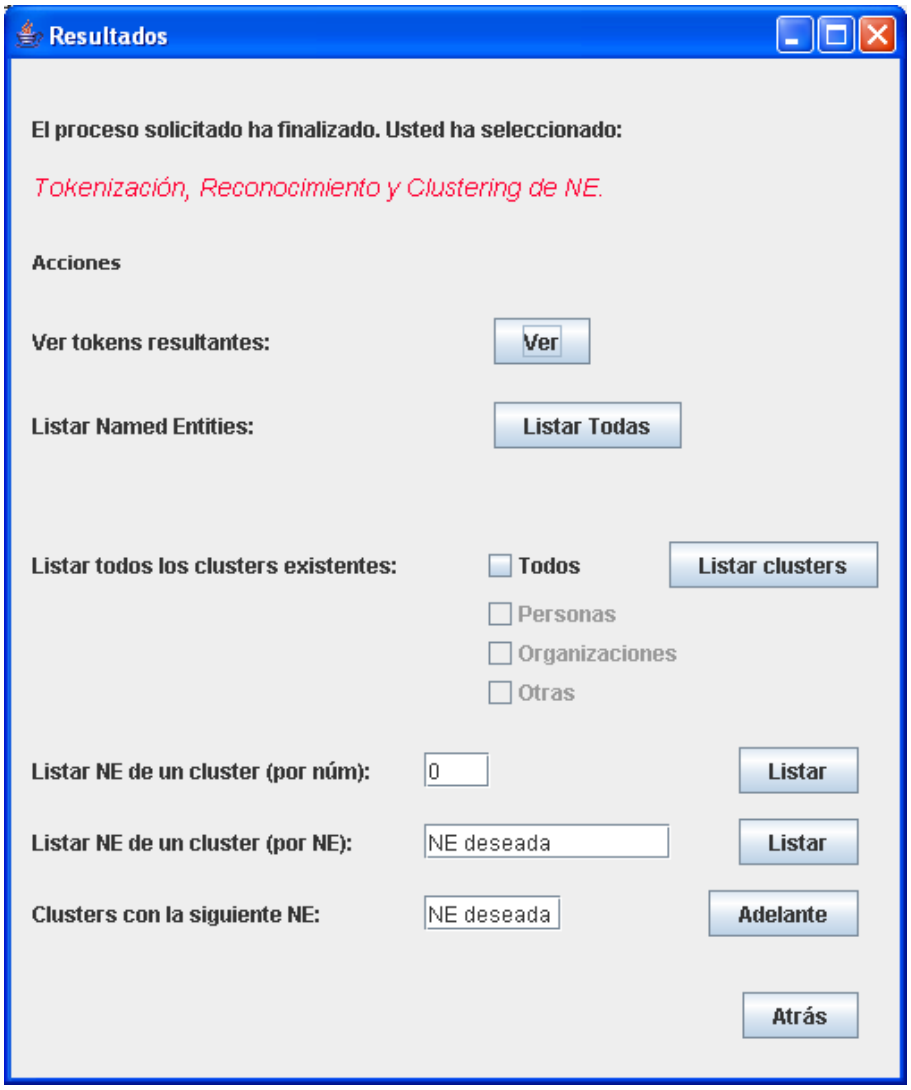


Figura 4.11: Pantalla de consulta de resultados

Es la ventana mostrada al finalizar los procesos de la aplicación. Según las acciones escogidas por el usuario en la pantalla principal se pueden realizar diferentes consultas sobre los resultados obtenidos: ver el texto resultante de la *tokenización*, ver las *NE* reconocidas, listar todos los *clusters* creados o listar *NE* de los *clusters* que cumplen los requisitos que el usuario indica. Si la petición del usuario no es correcta, se devuelve un mensaje indicando el motivo.

Para cerrar esta pantalla y volver al menú principal se debe pulsar el botón “Atrás”.

4.3 Capa de Dominio

La capa de dominio tiene la responsabilidad de la implementación de las funcionalidades del sistema. Debe encargarse de realizar las comprobaciones y acciones necesarias para cada una de las operaciones iniciadas desde la capa de presentación.

Las clases de la capa de dominio son las descritas en el modelo conceptual de la especificación.

Para el diseño de la capa de dominio, se ha utilizado el patrón Controlador-Fachada, concretamente Fachada-Sistema, clusterUI. Todas las acciones externas son controladas por él de forma que inicia todas las operaciones del sistema o bien informa al usuario del error existente.

El papel de controlador lo asume la clase de la interfaz gráfica que se muestra a continuación.

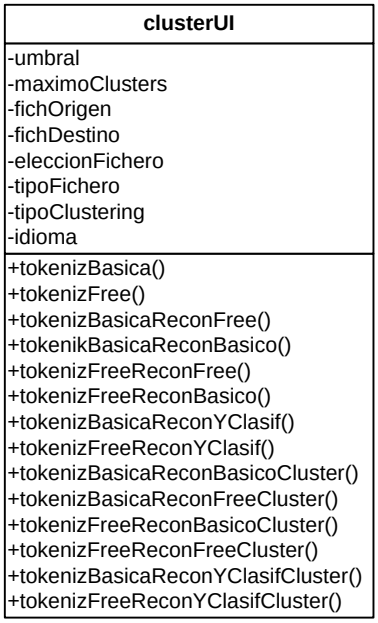


Figura 4.12: Controlador Fachada-Sistema de SCNE

El controlador, al recibir un evento nuevo se encarga de crear nuevos objetos y les asigna responsabilidades.

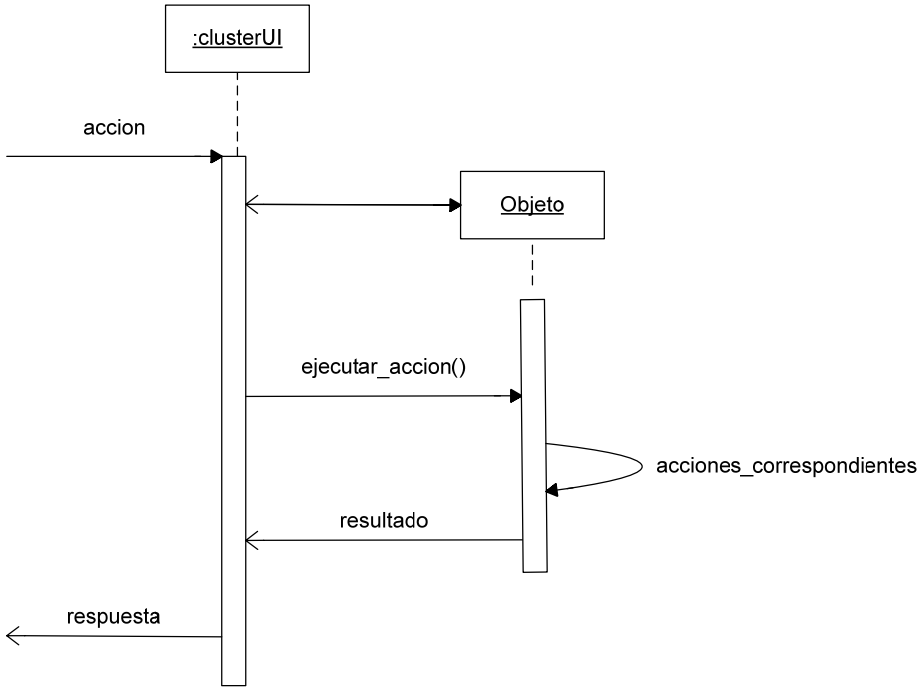


Figura 4.13: Diagrama de secuencia general

Tal y como se observa en la figura, los objetos reciben la responsabilidad del controlador, realizan la acción y, finalmente, devuelven el control a clusterUI que se encarga de dar la respuesta al usuario.

Comunicación con *FreeLing*

FreeLing es un paquete integrado de procesado de lenguaje natural (PLN) que, entre otros muchos, ofrece el servicio de *NERC*. Este sistema es capaz de reconocer y clasificar las *NE* de lengua castellana gracias a las reglas integradas que posee del idioma. En inglés y catalán *FreeLing* incluye *NER* pero no *NEC*.

Los servicios que ofrece son:

- *Tokenizadores* de texto.
- Separadores de frases.
- Análisis morfológico*.
- Detección de *Named Entities*.
- Reconocimiento de fechas/números/medidas/magnitudes.
- Clasificación en categorías gramaticales.
- Analizador de dependencias gramaticales.
- Significado de las unidades basado en Word Net (castellano y catalán con EuroWordNet, inglés con Word Net 1.6).
- Clasificación de *Named Entities*.
- Lenguas: castellano, catalán, gallego, inglés, italiano.

FreeLing está diseñado para ser utilizado como una librería externa en cualquier aplicación que requiera el tipo de servicios de *NER*, *NERC* o *tokenizadores*. Igualmente, puede ejecutarse directamente como una aplicación para analizar textos desde la línea de comandos.

Para establecer la comunicación entre SCNE y *FreeLing*, se ha escogido el método de la llamada de *script*: la aplicación SCNE escribe, dependiendo de las necesidades del usuario, *scripts* que guarda en la carpeta de la aplicación. En el momento de la interacción con *FreeLing*, se ejecuta el *script* pertinente y se espera la finalización del proceso. Finalmente se recogen los resultados procesándolos adecuadamente para conseguir dar la respuesta correcta al usuario.

En el apartado implementación se detalla la interacción entre las aplicaciones así como los ficheros intermedios.

4.4 Capa de Gestión de Datos

La capa de gestión de datos interrelaciona la capa de dominio con el sistema de ficheros. Esta capa conoce dónde se encuentran los datos con los que trabajar y de qué manera están estructurados.

La aplicación desarrollada utiliza ficheros de texto planos en todo momento. Mediante consultas a ellos conoce dónde se ubican los datos y cuáles son éstos. Los ficheros principales son:

Ficheros de la aplicación

- Fichero de configuración.

El fichero de configuración de la aplicación se consulta al iniciarse y permite obtener información en cuanto a:

- Carpeta de trabajo de la aplicación.
- Carpeta destino de *clusters* Generales.
- Carpeta destino de *clusters* de Persona
- Carpeta destino de *clusters* de Organización
- Carpeta destino de Otros *clusters*
- Fichero de Salida de *Tokenización*
- Fichero de Salida del reconocimiento.
- Máximo número de clusters predefinido
- Umbral predefinido
- Caracteres que limitan la *tokenización* básica
- Carpeta de instalación de *FreeLing*
- Contraseña del administrador.

- Ficheros resultado del *Clustering*

La aplicación debe dar los resultados de forma entendible y de fácil acceso al usuario. Para ello se ha diseñado un sistema en el que, para cada proceso, dada la carpeta destino se generan 2 tipos de fichero:

- Fichero de Lista de *clusters.*

Es el fichero índice del resultado de aplicar *clustering* a un documento. Este fichero hace las funciones de índice de los *clusters* generados. Para ello se presenta como una lista de la forma: número + *string*.

El número hace referencia al número de *cluster* asignado, el *string* por su parte es el representante del *cluster*.

- Fichero de *clusters*.

Cada fichero de *cluster* contiene las *NE* que se han agrupado después del proceso. El nombre del fichero coincide con el índice que aparece escrito en el fichero de lista de *clusters*. En su interior está la lista de *NE* resultantes del proceso de *clustering*.

Con este sistema, el usuario puede guardar los resultados en la carpeta deseada. En posteriores consultas debe mirar en primer lugar el fichero índice y, una vez encontrado el *cluster* que le interesa, abrir el fichero correspondiente a ese *cluster* para consultar las *NE* agrupadas en él.
A continuación se presenta un ejemplo:

Si se desea consultar un *cluster* cuyo representante sea Martínez_Rodríguez, se abrirá el fichero lista de *clusters* y se buscará el índice con el que se corresponde. Acto seguido, para consultar las *NE* que posee se abrirá el fichero con ese índice.

Carpeta del sistema contenedora de los *clusters*:

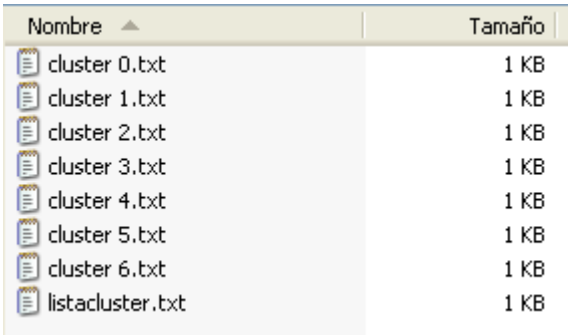


Figura 4.14: Ficheros resultantes en la carpeta del sistema

Contenido de listacluster.txt:

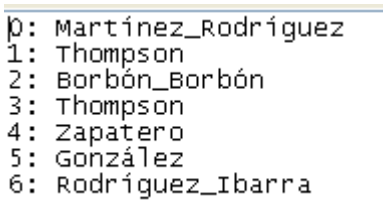


Figura 4.15: Contenido del fichero listaclusters.txt

Contenido de cluster 0.txt:

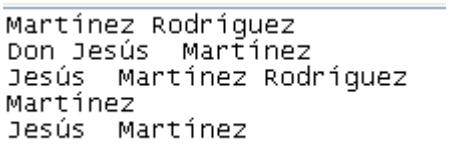


Figura 4.16: Contenido del fichero cluster0.txt

- Ficheros auxiliares para el proceso de *clustering*

Los diferentes tipos de *NE* que pueden ser clasificadas por el sistema son *NE* de persona y *NE* de organización (también es posible realizar el *clustering* con *NE* genéricas, es decir, sin clasificar). Para realizar el *clustering* de esos tipos, las *NE* se deben dividir en las diferentes partes que definen tanto a una persona como a una organización.

De las *NE* de persona se debe extraer el título que ostenta la persona, su nombre, sus apellidos y los links[35].

En el caso de las organizaciones, se debe reconocer tanto su acrónimo como su nombre y su coletilla[36].

Se han creado ficheros de ayuda para esta identificación cuyo contenido definirá cada una de las partes de las *NE*. Estos ficheros son:

- Ficheros de nombres de personas. (nombres masc.txt y nombres fem.txt)
- Fichero de Links. (links.txt)
- Fichero de Tratamientos. (tratamientos.txt)
- Fichero de Coletillas de empresas (coletilla.txt)

[35] Partículas que se pueden encontrar entre el nombre y el primer apellido o bien entre los dos apellidos de una persona. Ej: de, la, y…

[36] Ejemplos: SL, SA, Corp.

La capa de gestión de ficheros se ha diseñado con el propósito de llevar un control total de los ficheros de la aplicación.
Para ello se han creado las clases Fichero_NE, Fichero_Token, Fichero_Cluster y Fichero_Configuración.

Estas clases conocen en cada momento dónde ir a buscar los datos de entrada de cada uno de los procesos así como dónde escribir los resultados y de qué manera hacerlo.
A continuación se presenta un esquema de ellas:

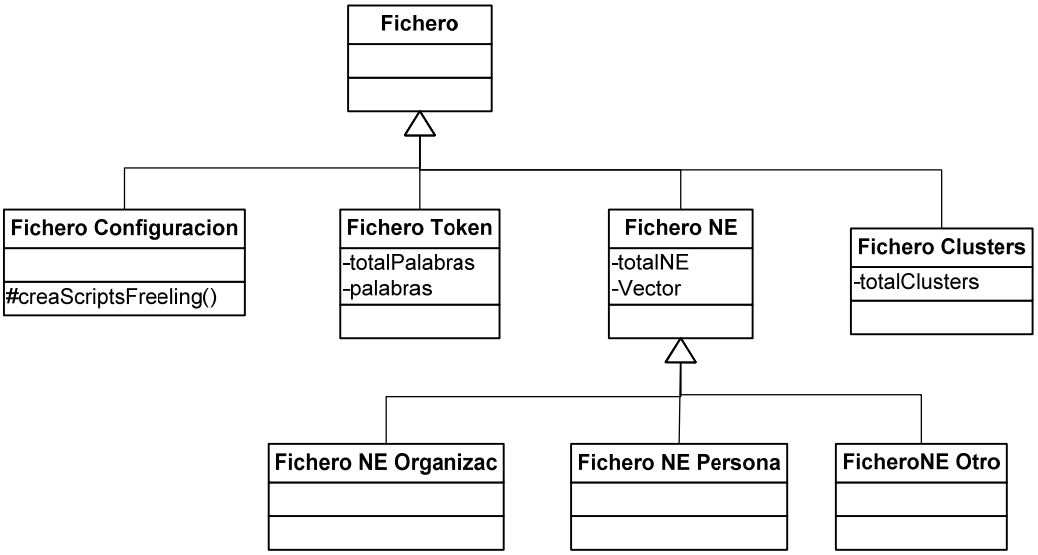


Figura 4.17: Diagrama conceptual de la clase Fichero

A parte del trabajo con los ficheros, las clases poseen vectores donde se almacena la información para la consulta del usuario. De esta manera se evita:
- Tener que leer de nuevo los ficheros que se acaban de escribir con lo que la velocidad aumenta considerablemente.
- Errores en el sistema de lectura de ficheros.

Capítulo 5

5. Implementación

Después de la especificación y el diseño llega el momento de codificar toda la información según lo planeado en los apartados anteriores. El resultado de este proceso de implementación tiene que ser un sistema que cumpla los requisitos y objetivos definidos al principio del desarrollo

5.1 Arquitectura del sistema

El funcionamiento de la aplicación es el siguiente:

El sistema debe realizar los procesos indicados por el usuario: *tokenización*, reconocimiento o reconocimiento/clasificación y *clustering*. Si el usuario decide que *FreeLing* debe intervenir en alguno de los procesos, la aplicación crea los *scripts* necesarios para que, tras ejecutarlos, satisfagan su petición.
Una vez iniciada la ejecución, el sistema realiza los procesos secuencialmente. Si se debe usar *FreeLing*, se ejecutan los *scripts* que llaman al programa externo y éste se ejecuta con las órdenes indicadas en ellos. Al acabar su ejecución, la aplicación lee y procesa los resultados dejados por *FreeLing* para continuar con la ejecución.
En el caso de que no se utilice *FreeLing*, es la aplicación la que genera y trabaja con los ficheros temporales del proceso y, finalmente guarda los resultados haciendo los cambios pertinentes en el sistema de ficheros del

sistema (creando y guardando los ficheros resultantes de la petición inicial del usuario).

A continuación se presenta un esquema que pretende ilustrar el funcionamiento previamente descrito:

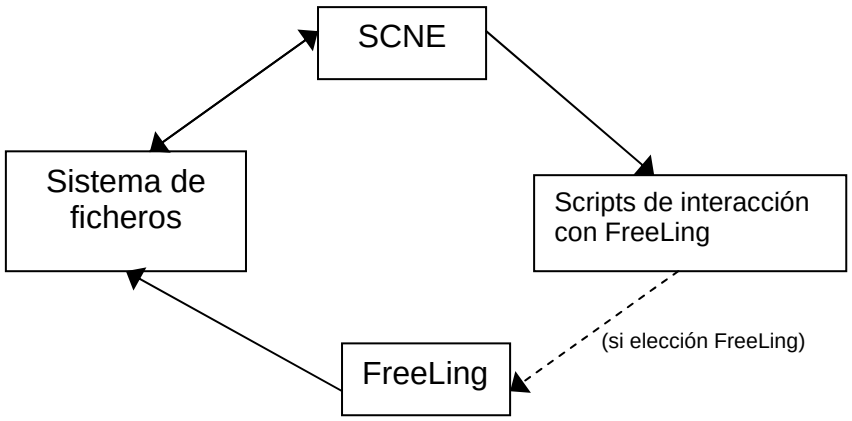


Figura 5.1: Esquema de funcionamiento de la aplicación

- **Implementación del Dominio**

La *tokenización* y el reconocimiento de *NE* se pueden realizar de dos formas distintas: utilizando la herramienta externa *FreeLing* o no con lo que serán procesos que denominamos básicos.

Procesos Básicos.

En el caso que se decida tanto segmentar cómo reconocer *NE* prescindiendo de *FreeLing*, la capa de dominio procesa las peticiones indicando a la capa de gestión de datos qué ficheros se necesitan leer y cuáles se escriben. Los procesos se ejecutan directamente sin la necesidad de ninguna comunicación externa de la aplicación y se presenta un resultado al usuario.

Tokenización Básica.

El sistema procesa la entrada dividiéndola en *tokens* definidos según las preferencias del usuario. En el caso de elegir división en palabras, la aplicación devuelve una lista con las diferentes palabras del texto inicial dado que el principal separador en este caso es el espacio en blanco. La selección de frases hace lo propio utilizando como separador principal los puntos y si la elección es la *tokenización* en párrafos se usa como separador el salto de línea.

Reconocimiento Básico de *NE* Generales.

Mediante el reconocimiento básico de *NE* implementado se sigue la regla de aceptar como *NE* toda palabra que empieza con letra mayúscula. La segunda regla es juntar en una misma *NE* todas aquellas palabras que empiezan en mayúscula y se encuentran seguidas en el texto.

El reconocimiento tiene como entrada la lista de palabras resultante de la *tokenización* hecha previamente.

Ejemplo: Me llamo José González y vivo en Barcelona.

Resultado: NE1: Me
NE2: José González
NE3: Barcelona

Este sistema de reconocimiento no es demasiado preciso pero se ha elegido hacerlo así porque se asegura que la selección de *NE* resultante contiene todas las *NE* del documento (aunque seleccionará como *NE* palabras que realmente no lo son, no se dejará ninguna verdadera *NE* sin seleccionar).

Procesos *FreeLing.*

Si la elección del usuario es segmentar con *FreeLing*, reconocer *NE* con *FreeLing* o bien reconocer y clasificar *NE*, la capa de dominio es la encargada en éste caso de recibir la petición y generar un *script FreeLing* que satisfaga las necesidades del usuario. En el momento de llegar al proceso, la capa de

dominio ejecuta el *script* y, cuando éste finaliza, recoge los datos que ha dejado en un fichero para poder procesarlos adecuadamente.

Tokenización FreeLing.

El *script* generado para la *tokenización FreeLing* tiene la siguiente estructura:

analyzer.exe [fichero configuración FreeLing] –token < [entrada] > [salida].

Esta orden es la llamada a *FreeLing* indicando que se desea segmentar en *tokens* el fichero indicado en la entrada y dejar el resultado en el fichero de salida. También se especifica el lenguaje en el que está la entrada en el fichero de configuración. *FreeLing* se ejecuta y da como resultado una lista de *tokens* en el fichero de salida.

Ejemplo: Me llamo José González y vivo en Barcelona.

Resultado:

Me
llamo
José
González
y
vivo
en
Barcelona

Reconocimiento Básico de *NE* Generales.

El *script* básico creado para el reconocimiento de *NE* Generales que genera *FreeLing* se tiene el siguiente aspecto:

analyzer.exe [fichero configuración FreeLing] –morfo < [entrada] > [salida].

Esta orden llama a *FreeLing* indicando que sobre la entrada especificada se debe hacer un análisis morfológico de las palabras que existen y debe dejarse el resultado en el fichero de salida.
Si el administrador hace cambios sobre la configuración *FreeLing* en la aplicación, dichos cambios se agregan al *script* con lo que la llamada a *FreeLing* indica directamente las preferencias del proceso.

Ejemplo: Me llamo José González y vivo en Barcelona.

Salida temporal *FreeLing*:

Me	P010S000
llamo	VMIP1S0
José_González	NP00000
y	CC
vivo	VMIP1S0
en	SPS00
Barcelona	NP00000

Resultado: NE1: José González
NE2: Barcelona

El fichero de salida generado, posee información que indica la categoría o función que realiza la palabra en el texto.
De esa manera, SCNE recoge el fichero resultante de *FreeLing* y hace un post-proceso en el que comprueba si una palabra es una *NE* o no. Una *NE* en el fichero resultante se identifica por la clasificación NPxxxxx.
En caso de encontrar una *NE*, la palabra se agrega al fichero de *NE*, si no es así se omite y se pasa a mirar la siguiente de la lista. De esa manera, al finalizar el proceso se tiene en un fichero todas las *NE* reconocidas por *FreeLing*. Con ellas se puede empezar a hacer *clustering* o bien mostrarlas al usuario si así lo requiere.

Reconocimiento y Clasificación de *NE*.

El *script* que ejecuta la acción de reconocer y clasificar *NE* por *FreeLing* tiene la siguiente estructura:

analyzer.exe [fichero configuración FreeLing] –tagged < [entrada] > [salida].

Esta orden llama a *FreeLing* indicando que se debe hacer un análisis morfológico y clasificación de las palabras que existen en la entrada especificada, dejando el resultado en el fichero de salida. En este fichero de salida se guarda información de cada palabra indicando la categoría o función que realiza en el texto.

Ejemplo: Me llamo José González y vivo en Barcelona.

Salida temporal *FreeLing*:

Me	PP1CS000
llamo	VMIP1S0
José_González	NP00SP0
y	CC
vivo	AQ0MS0
en	SPS00
Barcelona	NP00G00

Resultado: NE persona1: José González
NE otros1: Barcelona

SCNE recoge el fichero resultante de *FreeLing* y realiza un post-proceso de la misma manera que al reconocer *NE* Generales con la diferencia de que los identificadores serán:
NE de persona si la clasificación es del estilo: NP00SP0
NE de organización si la clasificación es del estilo: NP00O00
NE de tipo otras si se reconoce como NE (empieza por NP) y no es de persona ni de organización.

Al finalizar el proceso se almacenan en un fichero las *NE* reconocidas por *FreeLing* y su clasificación. Con ellas se puede empezar a hacer *clustering* o bien mostrarlas al usuario.

***Clustering*.**

Se han implementado 3 tipos de *clustering* diferentes en la aplicación:

- *Clustering* General

Dada una *NE*, se recorren los *clusters* existentes calculándose la distancia de edición con el representante del *cluster*. Si, después del cálculo y la normalización, el resultado es mayor que el umbral que se ha determinado, la *NE* no se añade al *cluster* actual y se procede a la comprobación con el siguiente. Si por el contrario es menor, se agrega y se continúa mirando los demás *clusters*. Al acabar los *clusters* existentes, se crea uno nuevo que contiene la *NE* si no existía previamente.

- *Clustering* de personas

Dada una *NE* de personas, para cada *cluster* existente se comprueba si los apellidos del representante son iguales a los del candidato. En caso afirmativo, se comprueba que las *NE* del cluster no sean iguales a la actual y sean compatibles con ella (mismo nombre). En el caso que se cumpla, la *NE* se añade al *cluster* y no genera uno nuevo. En caso negativo, recorrerá así todos los *clusters* y, si no se añade en ninguno, se creará uno nuevo teniendo a la actual *NE* como representante.

- *Clustering* de organizaciones

Para realizar *clustering* de organizaciones se procede igual que en el *clustering* de personas, teniendo en cuenta el representante acrónimo de cada *NE* y, en caso que sea igual, se hace la comprobación del nombre.

5.2 Herramientas utilizadas

Para la implementación del sistema se han utilizado las herramientas detalladas a continuación:

Netbeans IDE 5.5 con java SDK 1.5

NetBeans es un entorno integrado de desarrollo (IDE[37]) de código abierto cuya principal misión es hacer fácil el desarrollo de todo tipo de aplicaciones en la plataforma Java (JVM). Con NetBeans es posible desarrollar tanto aplicaciones de escritorio, como aplicaciones empresariales en varias capas, o programas para todo tipo de dispositivos móviles.

Para el diseño e implementación de las pantallas de ha utilizado la biblioteca gráfica de java *Swing*. Sigue una arquitectura de modelo-Vista-Controlador y tiene las siguientes características:

-Independiente de plataforma.

-Extensible.

-Customizable: permite representar diferentes 'look and feel' de la aplicación.

FreeLing (versión para Windows 1.5 de Junio '2007[38])

FreeLing es la herramienta que utiliza la aplicación para una mejor selección de *NE* dentro de los documentos, una búsqueda de *NE* dependiente de la lengua de los documentos y, finalmente para reconocer y clasificar las *NE* que se pueden encontrar.

Su uso se limita a dar respuesta a los *scripts* que lo llaman.

[37] *Integrated Development Environment*

[38] https://lafarga.cpl.upc.edu/frs/?group_id=32

Capítulo 6

6. Pruebas

Una vez finalizada la fase de implementación, el último paso antes de poner un sistema informático en producción son las pruebas. Se trata de una de las fases más importantes del ciclo de vida de un sistema software ya que representa una revisión final de las fases de análisis, diseño y codificación. Este punto acostumbra a recibir el 50% de los recursos totales destinados al sistema.

Las pruebas tienen como objetivo la detección de errores y, por lo tanto, una prueba tiene éxito si descubre un nuevo error. Así se reduce de forma considerable el coste del software pues de lo contrario las posteriores revisiones lo elevarían sustancialmente.

6.1 Pruebas del sistema

Las pruebas realizadas para la validación del sistema han sido tanto pruebas de bajo nivel (verifican cada segmento de código) como de alto nivel (pruebas que validan los requerimientos funcionales de la aplicación).
Están destinadas a comprobar que se obtiene un resultado correcto a una petición concreta.
Los resultados, a su vez, deben reflejar el correcto funcionamiento de cada uno de los procesos ejecutados.

Procesos de datos

- Lengua castellana:

Para realizar estas pruebas se han escogido 7 ficheros de texto plano del total del corpus parcial LEXESP[39] y se han ejecutado todas las combinaciones posibles de los procesos del sistema. Además han sido probados los procesos de crear nuevos *clusters* o realimentar los *clusters* existentes.

Se ha verificado que los resultados son los esperados y, variando los diferentes parámetros de la aplicación, se ha comprobado que cambian dichos resultados de forma lógica. Además, se ha podido verificar que la aplicación SCNE trabaja con *FreeLing* sin problemas.

El paso siguiente ha sido el probar la ejecución para ficheros .HTML o .htm comprobando que la salida es la esperada.

Finalmente se ha verificado la correcta recuperación de ficheros mediante su URL y el reconocimiento y *clustering*.

- Lengua catalana:

Las pruebas de lengua catalana han sido hechas con diferentes ficheros extraídos desde páginas web. Se ha podido comprobar con ellos que los resultados de procesar .txt (archivos creados copiando y pegando el texto en ficheros de texto plano) son idénticos a los resultados de la descarga directa de la URL y a los de procesar ficheros .html (guardando directamente las páginas web).

Se ha comprobado que el reconocimiento de *NE* sea el correcto tanto por el reconocedor básico como por el reconocedor de *FreeLing*, así como la realización del *clustering* general.

Además, se han procesado ficheros de *NE* de personas y organizaciones comprobando que el *clustering* de las mismas es correcto.

[39] Corpus de lengua castellana referenciado en la bibliografía

- Lengua inglesa:

Las pruebas de lengua inglesa han sido realizadas igual que las de catalán con idéntico resultado.

Inicio de sesión de administrador

Se han efectuado pruebas de entrada con errores y el sistema ha respondido a ellos con mensajes de ayuda al usuario.

Cambio de contraseña de entrada

El cambio de contraseña del usuario administrador se ha probado con resultado satisfactorio.

Rutas de la aplicación

Se ha comprobado que el cambio de rutas de la aplicación no afecta al funcionamiento del sistema. Los nuevos parámetros son aceptados y funcionan correctamente.

Mantenimiento de clusters

Se ha probado:

- Que el listado de *clusters* es correcto
- Que el listado de *NE* es correcto.
- Que la modificación (inserción o borrado) de *clusters* es correcta.
- Que la modificación (inserción o borrado) de *NE* es correcta.
- Si el directorio introducido no contiene *clusters*, se indica al usuario.
- Si el *cluster* indicado no existe, se indica al usuario.
- Si la *NE* especificada no existe en el *cluster*, se indica al usuario.

Resultados

Se ha comprobado que, si la entrada no es correcta en algún campo de listado o bien no se selecciona la acción que se desea, el sistema responde indicando al usuario cómo debe proceder para realizar la consulta.

6.2 Definición de parámetros

Este apartado tiene como objetivo definir buenos parámetros para realizar el *clustering* de *NE* generales o de tipo otras.
Pretende ser un pequeño estudio sobre el número máximo de *clusters* y umbral aceptable a partir del cuál se puede hacer *clustering* correctamente.

Número máximo de *clusters*

Es aconsejable que el número máximo de *clusters* de la aplicación SCNE sea elevado ya que normalmente se pretende abarcar todas las *NE* del texto realizando de esa manera tantos *clusters* como sean necesarios. Aún así cabe la posibilidad de limitarlo. De cara a posibles ampliaciones, se podría obligar a que el proceso se limite con este parámetro y hacer que el umbral se ajuste automáticamente dependiendo del valor indicado.
Es posible no limitar el número máximo de *clusters* y así es como se aconseja trabajar.

Umbral

El umbral es el valor más importante de la aplicación para la agrupación de NE Generales o bien de *NE* de tipo Otras porque define si 2 *NE* forman parte del mismo *cluster* o no.

Para una misma entrada, se puede afinar al máximo con valores comprendidos entre 0.1-0.3 (las entidades, para estar en el mismo *cluster*, deben tener una distancia de edición entre el 10% y el 30%).
Se ha comprobado que un umbral con estos valores es útil para corregir errores ortográficos:
Ej: Jesús Martínez – Jesu Martínez – Jesús Martine

De igual forma, un umbral amplio clusteriza muchas más entidades con lo que las agrupaciones, serán menos precisas. El ruido en este caso será enorme ya que normalmente muchas *NE* cumplen el requisito del umbral:

Ej: Jesús Martínez – Jesús – Jesús López – Jesus – Jerónimo Flores – Jesús Martínez Rodríguez

Después de probar diferentes valores tanto de umbral como de número máximo de *clusters*, se ha comprobado que los valores que dan mejor resultado son:

Número de *clusters*: ilimitado
Umbral: 0.4-0.5

6.3 Evaluación

Las pruebas de evaluación de la aplicación sirven para comprobar que los resultados son correctos para cada una de las lenguas, poder hacer comparaciones de tiempos de los procesos y, finalmente, extraer unas conclusiones del sistema en cuanto a si se ha logrado un buen *clustering* o no.

Evaluación de procesos

La evaluación de los procesos del sistema se ha realizado con un fichero de entrada creado a partir de varios de los documentos del *corpus* LEXESP seleccionados aleatoriamente. Su contenido de la entrada reúne tanto noticias como novelas y diálogos. Dado que los diferentes textos están relacionados entre si, tenemos una entrada válida para la evaluación. El documento obtenido finalmente contiene aproximadamente 50.000 palabras.

Se han utilizado los parámetros de máximo de *clusters* ilimitados y 0.4 en el umbral obedeciendo a las conclusiones obtenidas durante las pruebas de parámetros.

- Evaluación *clustering NE* General.

Se han seleccionado las opciones de *Tokenización* Básica, Reconocimiento *FreeLing* y *Clustering* de *NE* Generales.

De las 2.323 *NE* obtenidas, se han creado 1.108 *clusters*. La revisión que se ha hecho ha sido manual y se ha observado que el *clustering* de *NE* Generales tiene un resultado del **83%** de acierto.

Se han analizado los fallos más frecuentes aparecidos en este método de *clustering* Los más frecuentes son

- El sistema falla al hacer *clustering* de personas con el proceso *NE* General (no con el de personas) si éstas tienen un apellido corto y un nombre largo. Ejemplos vistos en el resultado: Karl Otto Poehl no forma *cluster* con Poehl o Manuel Fraga no lo hace con Fraga.
- El sistema falla igualmente con las *NE* en el proceso General si contienen palabras comunes largas. Ejemplos: José María Maravall, José María Calviño, José María García, José María Aznar y José María Treviño se encuentran en el mismo *cluster* o Partido Andalucista, Partido Socialista, Partido Comunista, Partido Reformista

Observación: La mayor parte de errores se refieren a *NE* de personas u organizaciones. Estos se pueden evitar si se utiliza el sistema de *clustering* propio diseñado para esas *NE* en vez del *clustering* general
Este método puede ser útil para listar *NE* que tengan nombres en común como personas que se llamen José María o bien encontrar los partidos políticos en un texto.

- Evaluación *clustering NE* de Personas

Para la evaluación del proceso y con la intención de tener en la entrada únicamente *NE* de personas, en primer lugar se ha pasado con SCNE el proceso de reconocimiento y clasificación de *NE* de Persona. Acto seguido, se ha revisado manualmente el fichero creado con las personas eliminando las que no lo eran (así evitamos el error acumulativo del reconocimiento de *FreeLing*). Finalmente, obtenido un fichero formado únicamente por *NE* de personas, se le ha pasado el proceso de *clustering* desde Fichero de *NE*.

Las 819 *NE* de persona han generado 265 *clusters* resultantes que se han chequeado manualmente, observando que este tipo de proceso tiene un acierto del **90%**.

Los errores más comunes detectados de este tipo de *clustering* son:

- Clasificación de diferentes nombres no existentes en los ficheros de nombres de la aplicación como apellidos. Nombres como Rosaura, Dorita, Encarna o nombres escritos en mayúscula se tratan como apellidos cuando en realidad no lo son.
- Clasificación de diferentes títulos no existentes en el fichero de tratamientos de la aplicación. Se ha observado que Padre o Santo/Santa se han tratado como apellidos cuando en realidad no lo son.
- Nombres indicados con una inicial se han tratado como título. S. Andreski se ha tratado como Sr. Andreski en vez de Stanislav Andreski.
- Si una persona tiene un apellido que también puede ser nombre, la aplicación trata apellido + nombre como si de un nombre compuesto se tratase.
- No distinción de género masculino o femenino en los tratamientos con lo que se puede dar el caso de encontrar *NE* como Don Martínez en un *cluster* con representante Laura Martínez.

- Evaluación *clustering NE* de Organización

Como en el caso de *NE* de personas, para iniciar el proceso se ha pasado el reconocimiento y clasificación de *NE* de Organizaciones. Se ha limpiado el fichero manualmente eliminando las *NE* que no son organizaciones y se le ha pasado el proceso de *Clustering* desde Fichero de *NE* de organizaciones.

De las 284 organizaciones iniciales, se han creado 129 *clusters*. Después de la revisión manual que se ha hecho de los resultados, se ha observado que la aplicación, en el caso de *clustering* de Organizaciones, tiene un acierto del **94%**.

Los errores más comunes detectados de este tipo de *clustering* son:

- No hacer *clustering* de organizaciones en el caso de que su acrónimo sea especial. En el texto tenemos uno de estos acrónimos: Enaco que se refiere a Empresa Nacional de la Coca pero no hacen *cluster* ya que la aplicación espera ENC como acrónimo. Otros ejemplos serian Renfe o CEPSA.
- En ocasiones, a las organizaciones se las aplica un diminutivo más corto que el nombre original. En estos casos la aplicación falla. En el texto no se ha hecho *cluster* de Partido Socialista con PSOE ya que se espera Partido socialista Obrero Español.
- En el caso de que los acrónimos de las organizaciones no se escriban de manera adecuada no se hace *cluster*. Ejemplo del texto lo tenemos en Convergència i Unió que aparece como CIU y no como CiU, por esto la aplicación no tiene un resultado correcto.

- Evaluación *clustering NE* de Otros tipos

  Igual que en los dos apartados anteriores, se ha procesado con SCNE el reconocimiento y clasificación de *NE* de Otros. Se ha revisado el fichero creado dejando únicamente las *NE* que respondían a esa clasificación y se ha pasado el *clustering* de fichero *NE* Otros.

  Se han obtenido 245 *clusters* de las 710 *NE* existentes. Una vez analizados manualmente los *clusters*, se ha visto que la aplicación ha tenido un acierto del **89%**.

  Dado que el sistema utilizado es el mismo que en el *clustering* de *NE* Generales, los errores que se han encontrado son los mismos. No obstante, se eliminan casos debido a la no existencia de *NE* de personas y organizaciones.

Se ha verificado que el proceso de *clustering* para las diferentes lenguas tiene un resultado más que aceptable para *clusters* de *NE* de personas y de organizaciones.

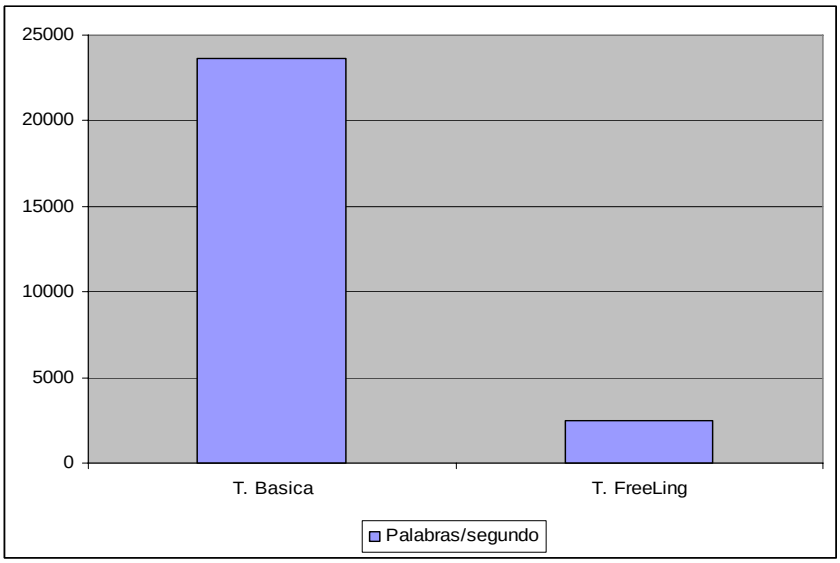
Aún así, si dos personas se llaman igual (nombre y dos apellido o nombre y un apellido en inglés), la aplicación las junta en el mismo *cluster* aunque son *NE* referidas a personas distintas.

Tanto los *clusters* de *NE* Otras como los de *NE* generales, tienen menor acierto puesto que son los que juegan con el umbral y las distancias de edición.

Evaluación de rendimiento

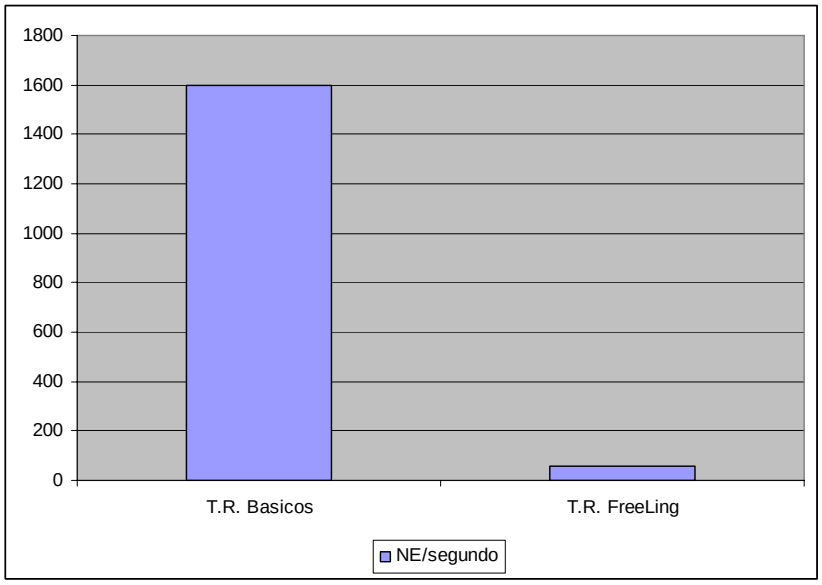
Los tiempos obtenidos de pasar procesos sobre un mismo fichero (d1.txt) son:

Tokenización Básica:
2,5 segundos, 59113 palabras = 23.645 palabras/seg.
Tokenización FreeLing:
25 segundos, 62898 palabras = 2.515 palabras/seg



Tokenización y Reconocimiento Básicos: 3 segundos, 4788 *NE*=1596 *NE*/seg

Tokenización y Reconocimiento *FreeLing*: 56 segundos,3333 *NE*=59,52*NE*/seg

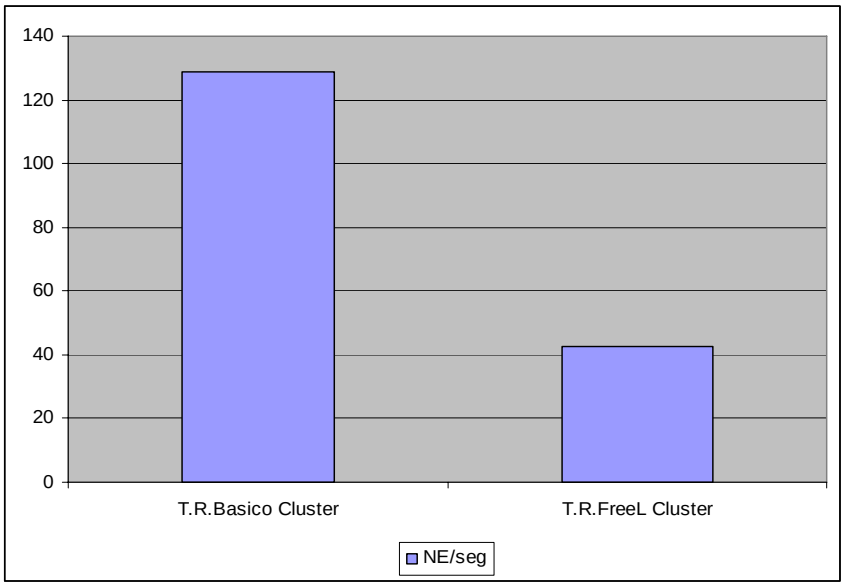


Tokenización y Reconocimiento Básicos y *Clustering* General:

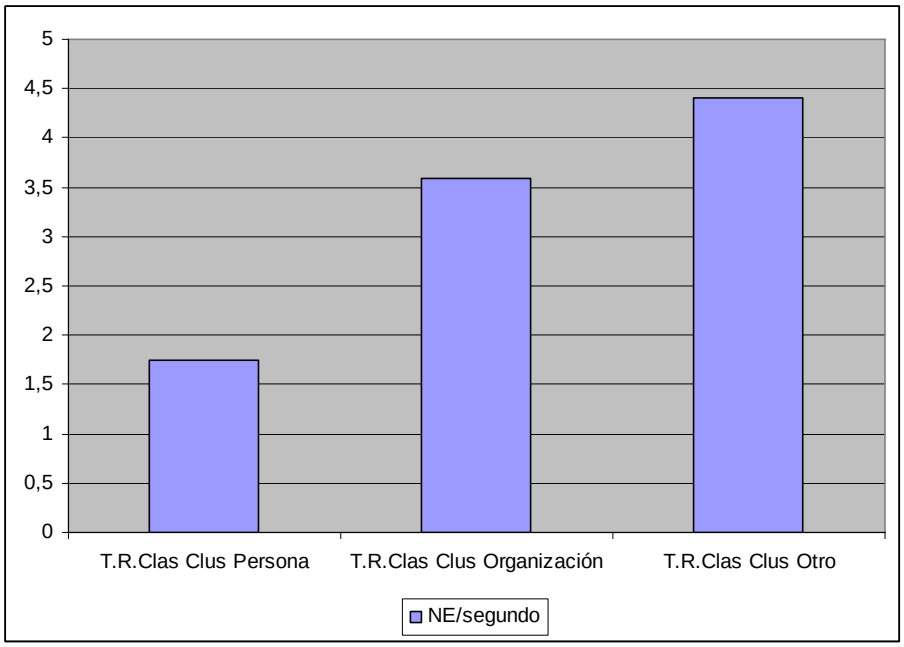
37 segundos, 1902 clusters = 129 *NE*/seg

Tokenización y Reconocimiento *FreeLing* y *Clustering* General:

1 minuto 18 segundos, 1368 clusters = 42,73 *NE*/seg



Tokenización, Reconocimiento y Clasificación, *Clustering* de tipo persona:
12 minutos 26 segundos, 1308 personas, 486 clusters = 1,75 *NE*/seg
Tokenización, Reconocimiento y Clasificación, *Clustering* de tipo organización:
3 minutos 46 segundos 812 organizaciones, 231 clusters = 3,59 *NE*/seg
Tokenización, Reconocimiento y Clasificación, *Clustering* de tipo otros:
3 minutos 50 segundos, 1016 NE, 525 clusters = 4,41 *NE*/seg



Capítulo 7

7. Planificación y valoración económica del proyecto

7.1 Análisis de tiempo de desarrollo

La realización del proyecto pasará por diferentes etapas: toma de contacto, análisis, diseño, prueba, implementación y documentación.

Toma de contacto

En esta etapa se realizará un estudio del dominio y una especificación de los requerimientos. Esto nos facilitará la comprensión de los objetivos de la aplicación y las necesidades del usuario. También nos ayudará a familiarizarnos con los elementos que debemos o podemos utilizar (y de qué manera podemos comunicarnos con ellos) y a definir un conjunto de sistemas que pueden satisfacer todas las necesidades u objetivos. De esa manera, podremos evaluar las diferentes posibilidades y escoger aquella que desarrolla el sistema más adecuado.

Análisis

Durante la etapa de análisis se realizará la especificación del sistema. Esta engloba las siguientes tareas:

- Modelo Conceptual
- Modelo de Casos de Uso
- Modelo de Comportamiento

Diseño

Esta etapa tiene como objetivo el definir las siguientes tareas:

- Diagrama de clases
- Diseño de pantallas

Implementación

A lo largo de esta etapa, la más larga del proyecto, se realizarán las siguientes tareas:

- Implementación de las clases del sistema (a partir del diagrama de clases).
- Implementación de las pantallas de la aplicación (a partir del diseño)
- Implementación del sistema de ficheros
- Implementación de la comunicación con los elementos externos al sistema.

Pruebas

Durante esta etapa se somete la aplicación desarrollada a pruebas con la finalidad de detectar posibles fallos de los resultados obtenidos. También se tomarán tiempos de respuesta de la aplicación y se comprobará la correcta interacción con las aplicaciones externas que pueda tener.

Documentación

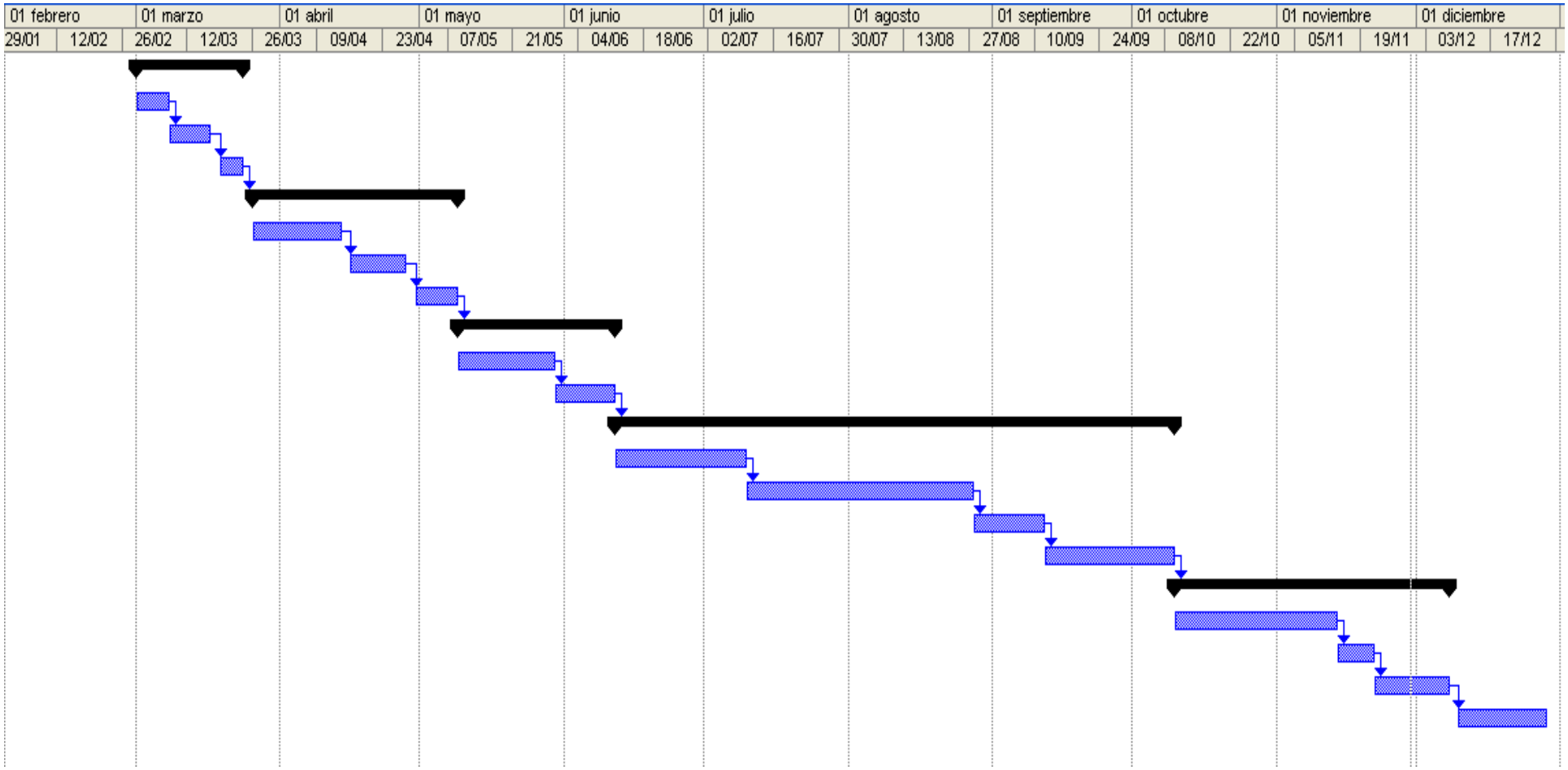
Esta etapa tiene como objetivo el completar la memoria del proyecto ya que a lo largo del mismo se debe ir documentando cada una de las partes desarrolladas.

Planificación de las tareas[40]:

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Toma de contacto	**17 días**	**jue 01/03/07**	**vie 23/03/07**	
Justificación y motiva	5 días	jue 01/03/07	mié 07/03/07	
Estudio del dominio	7 días	jue 08/03/07	vie 16/03/07	2
Especificación de los	5 días	lun 19/03/07	vie 23/03/07	3
Análisis	**32 días**	**lun 26/03/07**	**mar 08/05/07**	**4**
Modelo conceptual	15 días	lun 26/03/07	vie 13/04/07	
Modelo de casos de	10 días	lun 16/04/07	vie 27/04/07	6
Modelo de comportan	7 días	lun 30/04/07	mar 08/05/07	7
Diseño	**24 días**	**mié 09/05/07**	**lun 11/06/07**	**8**
Diagrama de clases	15 días	mié 09/05/07	mar 29/05/07	
Diseño de pantalla	9 días	mié 30/05/07	lun 11/06/07	10
Implementación	**86 días**	**mar 12/06/07**	**mar 09/10/07**	**11**
Clases del sistema	20 días	mar 12/06/07	lun 09/07/07	
Pantalla	35 días	mar 10/07/07	lun 27/08/07	13
Sistema de ficheros	11 días	mar 28/08/07	mar 11/09/07	14
Comunicación Freelin	20 días	mié 12/09/07	mar 09/10/07	15
Pruebas	**43 días**	**mié 10/10/07**	**vie 07/12/07**	**16**
Aplicación	25 días	mié 10/10/07	mar 13/11/07	
Def. Parámentros	6 días	mié 14/11/07	mié 21/11/07	18
Evaluación	12 días	jue 22/11/07	vie 07/12/07	19
Redacción documentación	15 días	lun 10/12/07	vie 28/12/07	20

[40] Estos tiempos no corresponden a planificación inicial sino a tiempos reales de desarrollo

Calendario:


01 febrero
01 marzo
01 abril
01 mayo
01 junio
01 julio
01 agosto
01 septiembre
01 octubre
01 noviembre
01 diciembre
29/01
12/02
26/02
12/03
26/03
09/04
23/04
07/05
21/05
04/06
18/06
02/07
16/07
30/07
13/08
27/08
10/09
24/09
08/10
22/10
05/11
19/11
03/12
17/12

7.2 Valoración económica del proyecto

Para el desarrollo del proyecto serán necesarios dos perfiles de trabajadores, los analistas y los programadores. El papel de los analistas es el de realizar la toma de contacto, el análisis y el diseño del sistema mientras que los programadores son los encargados de la implementación de la aplicación, realizar las pruebas y documentar el proyecto.

Los costes para cada tipo de perfil son:
€/hora de Analista: 60
€/hora de programador: 35

Con ello, y teniendo en cuenta las tareas que forman el proyecto tenemos la siguiente tabla de costes:

Tarea	Duración	€/h	Total
Toma de contacto	68	60	4080
Análisis	128	60	7680
Diseño	96	60	5760
Implementación	344	35	12040
Pruebas	172	35	6020
Documentación	60[41]	35	2100
Total	868		37680

El coste total del proyecto será de 37.680€ IVA no incluido.

[41] Las horas son las utilizadas en la redacción de la memoria, el tiempo dedicado a la documentación técnica está incluido en el apartado de Implementación.

7. Planificación y valoración económica del proyecto

Capítulo 8

8. Conclusiones y trabajo futuro

8.1 Conclusiones

Una vez finalizado el proyecto puedo afirmar que se han alcanzado satisfactoriamente los objetivos marcados en un principio.
De la realización del proyecto puedo sacar diferentes conclusiones, la mayoría son positivas pero también hay alguna negativa.

Lo más positivo es que las metas marcadas al empezar el proyecto, tanto de funcionalidades como de requerimientos, se han cumplido en su totalidad, ya que se ha creado un sistema informático que:

- Realiza un *clustering* de *Named Entities* Generales.
- Realiza *clustering* de *NE* de personas.
- Realiza *clustering* de *NE* de Organizaciones.
- Permite consultar los resultados de los procesos.
- Permite modificar los resultados de los procesos.
- Dado un texto *tokeniza* y reconoce *NE* (con o sin clasificación).
- Integra con éxito la herramienta *FreeLing*.
- Es un sistema modificable.

Otro aspecto positivo del proyecto ha sido que realmente he quedado muy satisfecho con el trabajo realizado, además, inicialmente me había marcado realizar el proyecto utilizando al máximo todo lo que había aprendido en la carrera, y las características del proyecto me han ayudado a poder utilizar el

conocimiento de un gran número de asignaturas realizadas. A pesar de eso, reconozco que en algunos momentos he echado de menos no haber realizado alguna otra asignatura optativa de la carrera ya que me hubiera evitado bastantes problemas que he tenido en algún punto del proceso.

Como en todo proyecto, también ha habido aspectos negativos. En primer lugar no ha sido posible la adaptación de la aplicación a sistemas LINUX debido a los cambios que era necesario hacer para conseguir un buen funcionamiento en cuanto a la comunicación con el sistema *FreeLing*. El segundo aspecto negativo ha sido el no poder realizar clasificación con otro idioma que no fuese castellano de un texto dado con *FreeLing* aunque se ha solucionado de forma satisfactoria mediante la incorporación de entradas previamente reconocidas.

8.2 Trabajo futuro

A continuación se presentan diferentes propuestas de ampliación de la aplicación SCNE. Entre otras, las posibles actualizaciones podrían ser:

- Añadir reconocimiento y clasificación *NE* ingles y catalán con otro reconocedor de *NE* existente directamente mediante llamadas con *scripts* tal y como se trabaja actualmente con *FreeLing*. Otra opción sería esperar a que *FreeLing* lo haga en futuras versiones, ya que la creación de *scripts* existente sería válida.

- Añadir nuevas distancias para definir un mejor *clustering* general y de *NE* de tipo otros. Con esas nuevas distancias que se crearían, la aplicación mejoraría estos tipos de *clustering* otorgando pesos diferentes a las medidas y se podrían elegir las que interesasen en cada momento.

- Ampliar la clasificación de *NE*. Ya que los *NERC* existentes, al hacer los procesos de clasificación clasifican las *NE* en diferentes tipos, se podría hacer un *clustering* específico para cada uno de ellos como se hace para personas y organizaciones. Con ello se podría hacer *clustering* finamente de *NE* locativas y *NE* temporales entre otras.

- Integración de *FreeLing* por medio de clase Java y no mediante *scripts*. La última versión *FreeLing* permite esta opción con lo que desde SCNE se tendría un mayor control de los parámetros de configuración así como de los procesos que realiza la aplicación externa.

- Aceptación de más tipos de documentos. Se podría adaptar la entrada a diferentes extensiones de ficheros previa transformación de éstos a texto plano. Así se aceptarían otros tipos de ficheros de texto.

- Alimentación de los ficheros auxiliares de nombres de personas, tratamientos, links y coletillas de organizaciones así como agregación de reglas de reconocimiento de acrónimos nuevas con la finalidad de afinar más en la creación de *clusters* de personas y organizaciones.

- Añadir un módulo de aprendizaje a la aplicación.

- Crear una base de datos bajo la aplicación que reúna los *clusters* creados a lo largo de las diferentes ejecuciones.

- Recompilar y hacer funcionar la aplicación sobre sistemas Linux.

8. Conclusiones y trabajo futuro

Capítulo 9

9. Bibliografía

Para la realización del proyecto se han consultado diferentes fuentes como libros, páginas Web y manuales.

Libros:

Dolors Costal, M.Ribera Sancho, Ernest Teniente. **Enginyeria del Software Especificació.** Edicions UPC, 2000

Cristina Gómez, Enric Mayol, Antoni Olivé, Ernest Teniente. **Enginyeria del Software Disseny I.** Edicions UPC, 2003

Diccionario de informática e Internet de Microsoft 2ª edición
McGraw-Hill/Interamericana de España SAU

Enlaces Web:

MUC: http://cs.nyu.edu/cs/faculty/grishman/muc6.html
Historia y glosario: http://www.wikipedia.org/
JAVA: http://es.sun.com/java/
Interfícies Java: http://www.programacion.net/java/tutorial/swing/
HAPNIS: http://www.cs.utah.edu/~hal/HAPNIS/
NAME:
http://www.mpi-inf.mpg.de/~suchanek/downloads/javatools/doc/javatools/Name.html

Programas NERC:

NLTK: http://nltk.sourceforge.net/

openNLP: http://opennlp.sourceforge.net/

YamCha: http://chasen.org/~taku/software/yamcha/

FreeLing: http://www.lsi.upc.es/~nlp/freeling/

Balie: http://balie.sourceforge.net/

YooName: http://www.yooname.com/

Manuales:

FREELING USER MANUAL June 2006

FREELING REFERENCE MANUAL August 2006

A parte de la bibliografía citada se han consultado **apuntes** de las asignaturas VPE, HDC, ES:D1, ES:D2, PESBD y ADA, impartidas en la FIB.

Se ha utilizado para validar la aplicación entradas del corpus* de lengua castellana LEXESP:

Sebastián, N., M.A. Martí, M.F. Carreiras y F. Cuetos, 2000. **LEXESP: Léxico Informatizado del Español.** Barcelona: Universitat de Barcelona.

Anexo 1

A. Glosario

Acrónimo

Es una palabra que resulta de la unión de las letras iniciales de una o más palabras. Suele usarse para referencias abreviadas.

Análisis Morfológico

Análisis de las palabras para extraer raíces, rasgos flexivos, unidades léxicas compuestas y otros fenómenos.

API (*Application Programming Interface)*

Una API (Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

Aprendizaje Automático

Forma de adquisición automática de conocimiento, normalmente a partir de ejemplos.

Aprendizaje Supervisado

Forma de aprendizaje automático en el que los ejemplos están previamente clasificados (supervisados).

Cluster

Agrupación o conjunto de objetos que comparten alguna propiedad. En nuestro caso los elementos de un *cluster* son denominaciones varias de la misma entidad.

Clustering

Acción de crear o formar *clusters*.

Corpus

Colección de textos seleccionados mediante un criterio que tienen por objetivo el de ser ejemplos de alguna lengua.

Distancia métrica textual

Al hablar de distancia entre cadenas de caracteres se esta hablando de medir las diferencias que hay entre estas. Algunas de las diferencias interesantes pueden ser encontradas en las longitudes de las cadenas que se están comparando, o en palabras de una misma longitud, los cambios de unas letras por otras.

Dominio

Ámbito real o imaginario de una actividad determinada..

Documento

Un documento es un escrito o archivo que contiene información.

Entidad Nombrada (*Named Entity*, NE)

Términos que corresponden a nombres propios que identifican una entidad del mundo real y que no se encuentran en los diccionarios como tales.

ENE, Entidad Nombrada Extendida (Extended Named Entity)

Clasificación más fina que la habitual en 8 clases de *Named Entities* que define la propuesta MUC. Son conocidas las clasificaciones de Sekine o la propuesta en el programa ACE (*Automatic Content Extraction*).

Fichero de Configuración

Archivo que contiene las especificaciones de operación o los valores por omisión que la máquina puede entender para un elemento de hardware o de software o que contiene información sobre otro archivo o sobre un usuario específico, como, por ejemplo, el identificador de inicio del usuario.

Gazetteer

Lista que contiene *Named Entities* de una determinada categoría, por ejemplo listas de organizaciones o de ciudades.

Interfaz gráfica

Entorno visual informático que representa en la pantalla los programas, archivos y opciones mediante imágenes gráficas como iconos, menús y cuadros de diálogo. El usuario puede seleccionar y activar estas opciones señalando y pulsando con el ratón o, a veces, con el teclado.

Lexicón

El lexicón es el "diccionario" en el que se registran las palabras que conoce un hablante. Este "diccionario" especifica los rasgos de las piezas léxicas (palabras). Normalmente se reserva el nombre de lexicón a los repositorios de piezas léxicas para uso computacional.

Línea de Comandos

Método para manipular con instrucciones escritas al programa que subyace debajo. Se interactúa con la información de la manera más simple posible, sin gráficas ni nada más que el texto crudo. Las órdenes se escriben como líneas de texto (de ahí el nombre), y, si los programas responden, generalmente lo hacen poniendo información en las líneas siguientes.

Minería de Textos

1-Conjunto de técnicas encaminadas a la extracción de "conocimiento" procesable implícito en las bases de datos de las empresas. Las bases de la minería de datos se encuentran en la Inteligencia Artificial y en el Análisis Estadístico. Mediante los modelos extraídos utilizando técnicas de minería

de datos se aborda la solución a problemas de predicción, clasificación y tokenización entre otros..

2-El proceso de identificar relaciones, problemas o patrones comercialmente útiles en una base de datos, en un servidor Web o en otro repositorio informatizado utilizando herramientas estadísticas avanzadas.

MUC

Las conferencias MUC sirvieron de punto de encuentro, evaluación y comparación de los diferentes sistemas de extracción de información.

Fueron organizadas por el Grupo de Investigación y Desarrollo Naval (*NraD*[42]) del *NCCOSC*[43], entidad dependiente de la Marina de los Estados Unidos de América entre los años 1987 y 1996.

N-grama

Subsecuencia de n elementos de una secuencia dada. Los n-gramas se emplean en varias áreas del procesamiento estadístico del lenguaje natural.

NER, Named Entity Recognition

Un NER es un reconocedor capaz de detectar entidades nombradas en fragmentos de texto.

NEC, Named Entity Classification

Un NEC es un clasificador de entidades nombradas en sus diferentes tipos existentes.

NERC, Named Entity Recognition & Classification

Un NERC es a la vez un reconocedor y clasificador de entidades nombradas. Agrupa NER y NEC.

Normalizar

Ajustar a un tipo, a un modelo, tipificar.

[42] *Naval Research and Development Group*
[43] *Naval Command, Control and Ocean Surveillance Centre*

Operadores booleanos

Los operadores booleanos (verdadero/falso) consideran el 0 como falso y el 1 como verdadero.

PLN, Procesado Lenguaje Natural

1-Abreviado PLN, o NLP del idioma inglés *Natural Language Processing*. Es una subdisciplina de la Inteligencia Artificial y la rama ingeniera de la lingüística computacional. 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.

2-Campo de la ciencia informática y lingüística que estudia los sistemas informáticos que puede reconocer y reaccionar al lenguaje humano, ya sea hablado o escrito.

Programación modular

Valiéndose de la técnica del diseño estructurado para el diseño de algoritmos consigue desarrollar programas a partir de un conjunto de módulos, cada uno de los cuales desempeña una tarea necesaria para el correcto funcionamiento del programa global.

Script

En informática, un script es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de comandos y usualmente son archivos de texto.

Substring

Subconjunto de los símbolos en una secuencia, donde el orden de los elementos se preserva.

Tag

Una etiqueta (o *tag*) es una marca con tipo que delimita una región en los lenguajes basados en HTML o XML.

TALP

El TALP es un centro de I+D que trabaja en el desarrollo de las tecnologías del lenguaje humano. Está formado por la unión de dos grupos de investigación de la UPC: el Grupo de Procesado de la Voz (VEU) y el Grupo de Procesamiento del Lenguaje Natural (GPLN)

Texto Codificado

Texto con formato, etiquetas, flags etc. que se utiliza para transferir documentos de texto entre aplicaciones incluso si éstas se ejecutan en diferentes plataformas.

Texto Plano

Texto sin formato, contiene sólo caracteres. Estos caracteres se pueden codificar de distintos modos dependiendo de la lengua usada. Algunos de los sistemas de codificación más usados son: ASCII, ISO-8859-1 o Latín-1, Unicode, etc.

Token

En programación, un elemento individual en un lenguaje de programación. Es un bloque de texto categorizado. Por ejemplo una marca de puntuación, un operador, un identificador, un número, etc. Por extensión, en PLN se consideran tokens las palabras o términos multipalabra.

Tokenizador

Programa que intenta encontrar los límites entre unidades léxicas consecutivas de un texto o un conjunto de datos.

Umbral

En nuestra aplicación, el umbral define el valor máximo de la distancia entre dos *Named Entities* para agruparlas en un *cluster*.

UML (Unified Modeling Language)

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es el lenguaje de modelado de sistemas de software más conocido y utilizado en

la actualidad; aún cuando todavía no es un estándar oficial, está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

***User-Friendly* (Amistoso)**

Característica del software que hace que el programa informático sea fácil de aprender y de utilizar.

Anexo 2

B. Manual de Instalación SCNE

El presente capítulo pretende ser una guía detallada de instalación de la aplicación. Se indicarán los pasos a realizar para la correcta instalación del software sobre Windows, plataforma sobre la cual se ha realizado el sistema.

Contenido del CD:

- java jdk 1.5.0
- carpeta SCNE
- carpeta FreeLing

Si en la máquina destino existe una versión JDK inferior a la entregada, se debe instalar la proporcionada en el CD ya que, de lo contrario, la aplicación no funcionará. Con versiones superiores debería ser compatible pero igualmente se recomienda que la versión sea la entregada.

Una vez asegurados de que la versión de Java es la correcta (en línea de comandos basta con ejecutar *java –version* y comprobarlo), la instalación de *FreeLing* y SCNE se realiza copiando directamente las carpetas existentes en el CD a la c:\ de la máquina destino. Deben quedar las aplicaciones con las rutas: c:\SCNE\ y c:\FreeLing\ con idéntico contenido al que tienen en el CD.

Los pasos a realizar son (siempre sobre plataforma Windows):

- comprobación de JDK existente e instalación del proporcionado si es necesario.

- Copiar la carpeta *FreeLing* y todo su contenido a c:\
- Copiar la carpeta SCNE y todo su contenido a c:\

Una vez hecho esto, la aplicación está lista para funcionar. Se ha probado la instalación en varias máquinas con Windows XP y ha sido correcta.

En caso de problemas, verificar lo siguiente:

- existe el fichero c:\SCNE\configuracion.txt
- existe el fichero c:\SCNE\nombres masc.txt
- existe el fichero c:\SCNE\nombres fem.txt
- existe el fichero c:\SCNE\tratamientos.txt
- existe el fichero c:\SCNE\links.txt
- existe el fichero c:\SCNE\coletillas.txt
- existe el fichero c:\SCNE\temphtml.txt
- existe el fichero c:\SCNE\defhtml.txt
- existe el fichero c:\SCNE\segment.txt
- existe el fichero c:\SCNE\nerc.txt
- existe la carpeta c:\SCNE\clGeneral\
- existe la carpeta c:\SCNE\clPers\
- existe la carpeta c:\SCNE\clOtr\
- existe la carpeta c:\SCNE\clOrg\
- existe el fichero c:\FreeLing\es.cfg
- existe el fichero c:\FreeLing\ca.cfg
- existe el fichero c:\FreeLing\en.cfg

Dado que desde un primer momento la aplicación ha sido desarrollada y pensada para funcionar sobre plataformas Windows, la instalación en sistemas Linux no se ha logrado con éxito, los dos motivos principales han sido:

- La versión de FreeLing para Linux es diferente a la que corre sobre Windows y los *scripts* de llamada son diferentes: No existe el ejecutable analyzer.exe.
- Las rutas del fichero de configuración están creadas con el path completo y no parcial. Debido a que el sistema de ficheros Linux no es compatible

con el sistema de ficheros Windows, las rutas de la aplicación no funcionan sobre Linux.

Se ha añadido en el apartado de trabajo futuro la adaptación de SCNE para el funcionamiento sobe sistemas Linux. Los pasos a realizar para lograrlo son:

1- Cambiar las rutas del fichero de configuración.
2- Instalar y configurar *FreeLing* para Linux.
3- Reimplementar de nuevo el contenido de los *scripts* de llamada a *FreeLing* así como las rutas internas de la aplicación.
4- Recompilar la aplicación con la versión de Java para Linux superior a la JDK 1.5.

Anexo 3

C. Manual de Usuario SCNE

El presente capítulo pretende ser una guía detallada de uso de la aplicación de *clustering* SCNE. Para ello se utilizarán figuras de las pantallas de la aplicación y la descripción de éstas.

Una vez instalada la aplicación y los componentes satélites que necesita[44], para iniciar la aplicación se hará desde la línea de comandos del sistema (en el proyecto se ha utilizado el MS-DOS de Windows) o bien desde la barra de tareas hacer: Inicio→Ejecutar→ java –jar c:\SCNE\dist\SCNE.jar.

Antes de empezar con la descripción de las pantallas y procesos, se cree conveniente dar unas definiciones técnicas y explicar QUÉ HACE SCNE.

Definiciones:

***NE (Named Entity)*:** Una *Named Entity* es un término o palabra que corresponde a un nombre propio que identifica una entidad del mundo real y que no se encuentra en los diccionarios.

***Cluster*:** Un *cluster* es una agrupación de objetos que comparten alguna propiedad, en nuestro caso los elementos de un *cluster* son denominaciones varias de la misma entidad.

***Token*:** Palabras o términos multipalabra existentes en un texto.

[44] Véase capítulo anterior

El sistema:

El sistema SCNE, dada una entrada elegida por el usuario (documento, colección de documentos, página web o fichero de *NE*), la *tokeniza* en palabras, reconoce las *NE* existentes en el fichero y hace *clusters* con ellas, con lo que se obtiene como resultado agrupaciones con las diferentes maneras de hacer referencia a la misma entidad.

Las entradas deben ser obligatoriamente ficheros con extensión .txt (planos) o .html/.htm (codificados con HTML), o bien una URL acabada en .html/.htm.

El proceso de *clustering* de *NE* se puede realizar sobre *NE* Generales sin clasificar o bien sobre *NE* de Personas, Organizaciones u Otras *NE* si se ha hecho previamente un reconocimiento y clasificación de *NE* (disponible únicamente para el idioma castellano).

Ejemplo: Si en el texto aparece Jordi Pujol, Jordi Pujol Solei y Pujol Solei, se crea un *cluster* identificado como Pujol_Solei que contiene las tres formas de referenciar a su persona.

Al final del manual se encuentra una relación de ficheros de ejemplo que se proporcionan con el programa por si el usuario desea realizar pruebas.

Una vez introducida la aplicación, se procede a explicar las diferentes pantallas y opciones que se pueden utilizar.

SCNE tiene dos métodos de ejecución distinguidos según los parámetros de entrada: ejecución gráfica o bien ejecución interna.

Ejecución en modo gráfico

Este método está orientado al usuario final que quiera tener un acceso rápido, simple y visual a todos los parámetros de la aplicación: qué se va a realizar y cómo se llevará a cabo. El usuario administrador deberá escoger este método de ejecución para realizar los pertinentes cambios que crea convenientes tanto en rutas de datos, como en configuraciónes del programa o hacer el mantenimiento de *clusters* existentes.

Para ejecutar la aplicación en modo gráfico se debe ejecutar en línea de comandos la siguiente instrucción:

java –jar [ruta fichero proyecto.jar]

Ej. java –jar c:\scne\dist\scne.jar

Pantalla principal de la aplicación, proceso

El menú principal de la aplicación es intuitivo y presenta las acciones a realizar tal y como se puede observar en la figura A3.1.

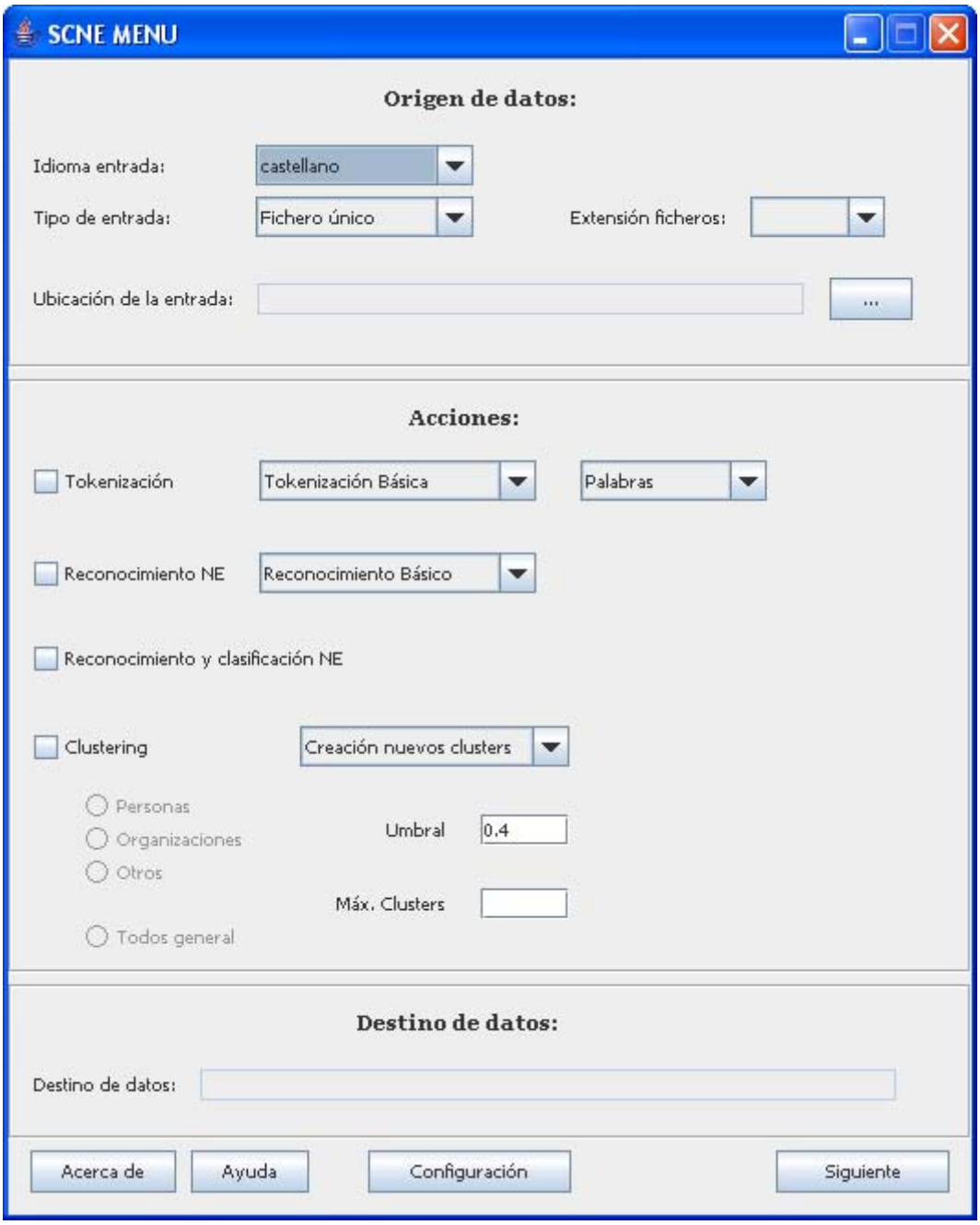


Figura A3.1: pantalla principal de la aplicación SCNE

En primer lugar el usuario debe escoger el idioma[45] de los ficheros que va a usar en la entrada de la aplicación.

Figura A3.2: opciones de idioma

Seguidamente deberá escoger el tipo de entrada entre las opciones disponibles:

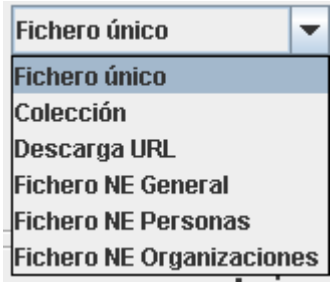


Figura A3.2: opciones de entrada

Si la opción escogida es una colección, obligatoriamente se deberá escoger la extensión de los ficheros entre las disponibles.

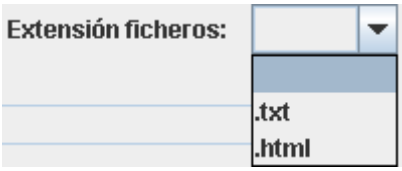


Figura A3.3: extensiones de ficheros

Debido a que la aplicación tiene como finalidad el que el usuario esté guiado en todo momento, los datos de entrada se presentan vacíos para que la persona los rellene mediante el botón de búsqueda (quedando éstos grabados al escogerse).

Figura A3.4: visualización de la entrada

[45] Los idiomas catalán e inglés, dado que la aplicación utiliza reconocimiento y clasificación de *NE* proveniente de *FreeLing*, y que esa opción en la versión de *FreeLing* utilizada (1.5) no está disponible, no pueden escoger esa opción. Si se desea, se puede pasar un reconocimiento y clasificación de *NE* externamente y, una vez se tienen las *NE* clasificadas, utilizar la opción de SCNE para hacer *clustering* ficheros de *NE* directamente.

A continuación la aplicación requiere que se definan los procesos a llevar a cabo (*tokenización*, *tokenización* y reconocimiento o bien *tokenización*, reconocimiento y *clustering* de *NE*) y con qué parámetros de deben procesar las tareas, número máximo de *clusters* (si se desea un número ilimitado, se debe dejar en blanco) y umbral de diferencia entre *NE* a partir del cuál se definirán los *clusters*.

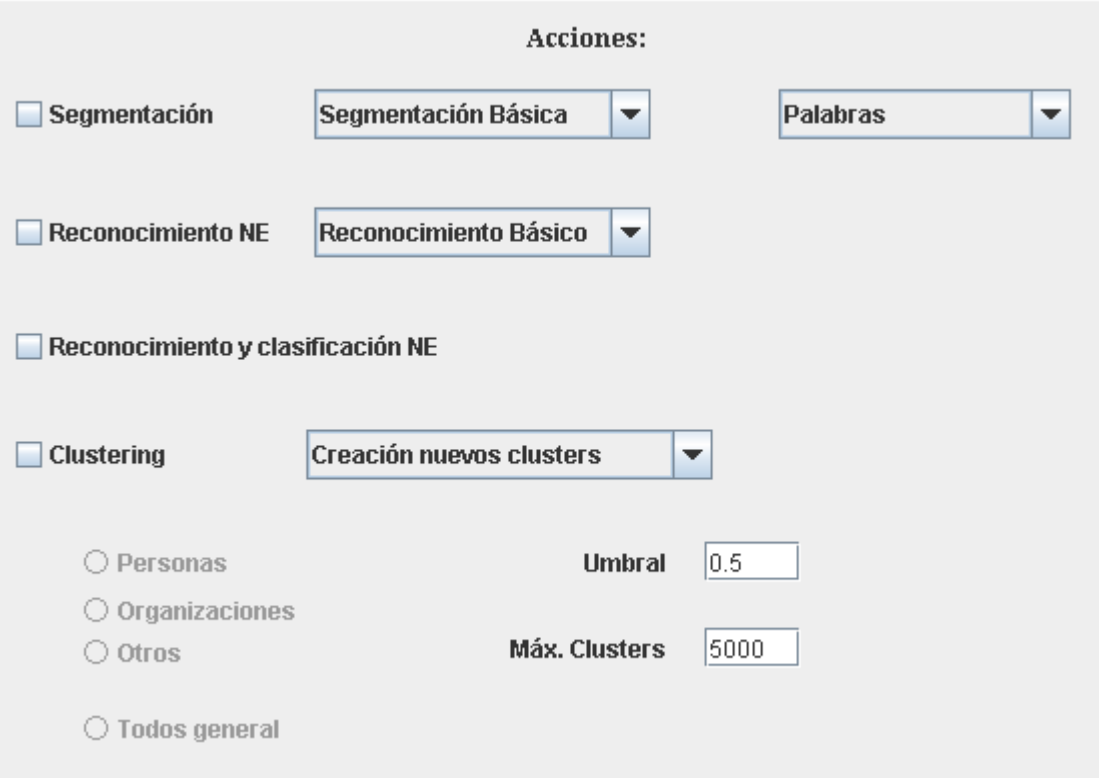


Figura A3.5: menú de procesos

En la realización de *clustering*, el usuario tiene la opción de crear nuevos *clusters* o bien alimentar los *clusters* existentes (deben en ese caso existir en la ruta de *clustering* definida).

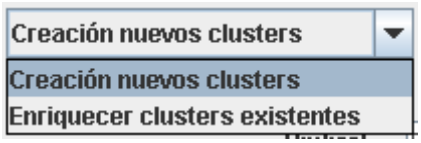


Figura A3.6: opciones de *clustering*

Finalmente, el destino de datos se rellenará automáticamente según la elección del usuario y la ubicación será aquélla que la aplicación tiene predefinida en el fichero de configuración.

Figura A3.7: destino de datos

Cuando el usuario haya definido sus preferencias, para iniciar el proceso debe pulsar el botón “Siguiente”.

Resultados de la aplicación

Una vez acabados los procesos del sistema, se presenta la pantalla de resultados del programa que podemos ver en la figura A3.8.

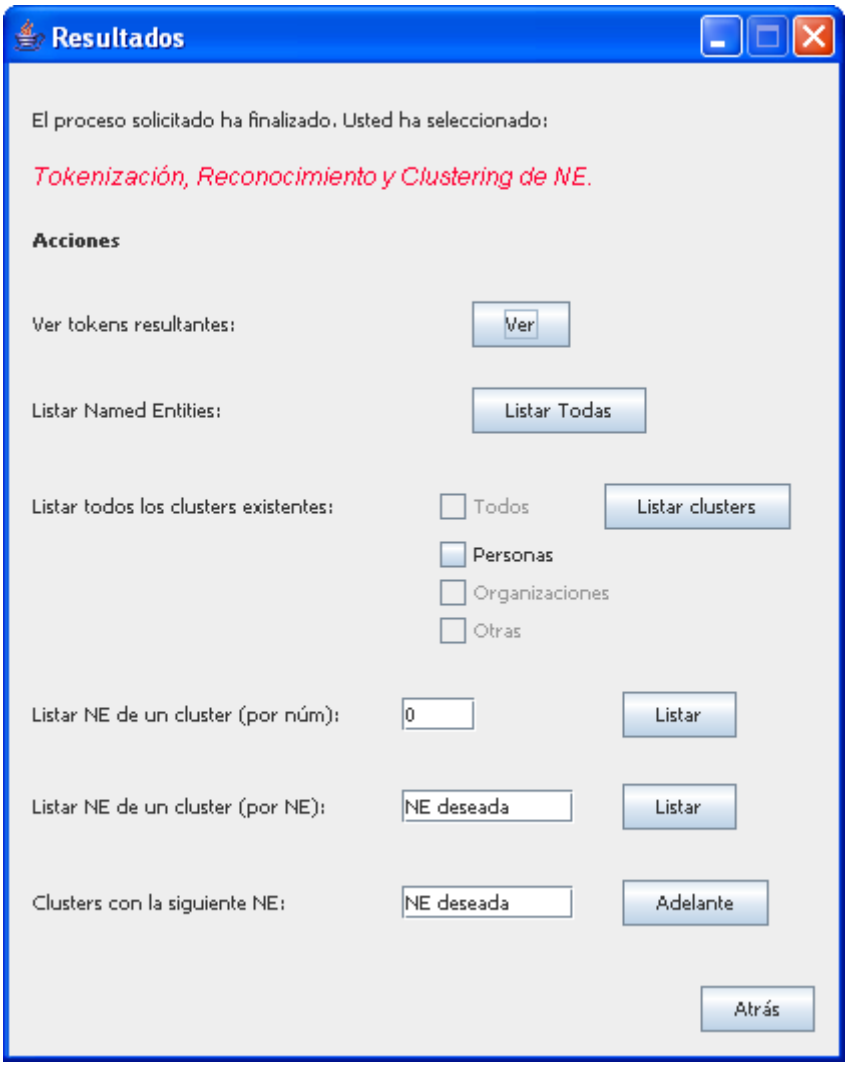


Figura A3.8: pantalla de resultados

Desde esta pantalla se pueden consultar los *clusters* creados así como las NE que poseen cada uno de ellos y, dichos resultados se presentarán al usuario mediante nuevas ventanas Java.

Dependiendo de cuál haya sido la selección del usuario en la pantalla principal, estarán activas unas u otras consultas en la pantalla de resultados.

Configuración de la aplicación

Se puede acceder a la configuración del sistema mediante usuario y contraseña (figura A3.9). Esta opción aparece al pulsar el botón “Configurar” de la pantalla inicial.

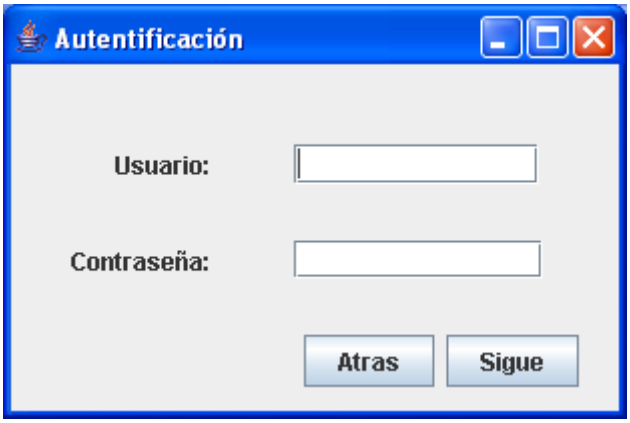


Figura A3.9: pantalla de autentificación de usuario administrador.

Una vez la autentificación es correcta, se accede a la pantalla de configuración de rutas de la aplicación (figura A3.10).

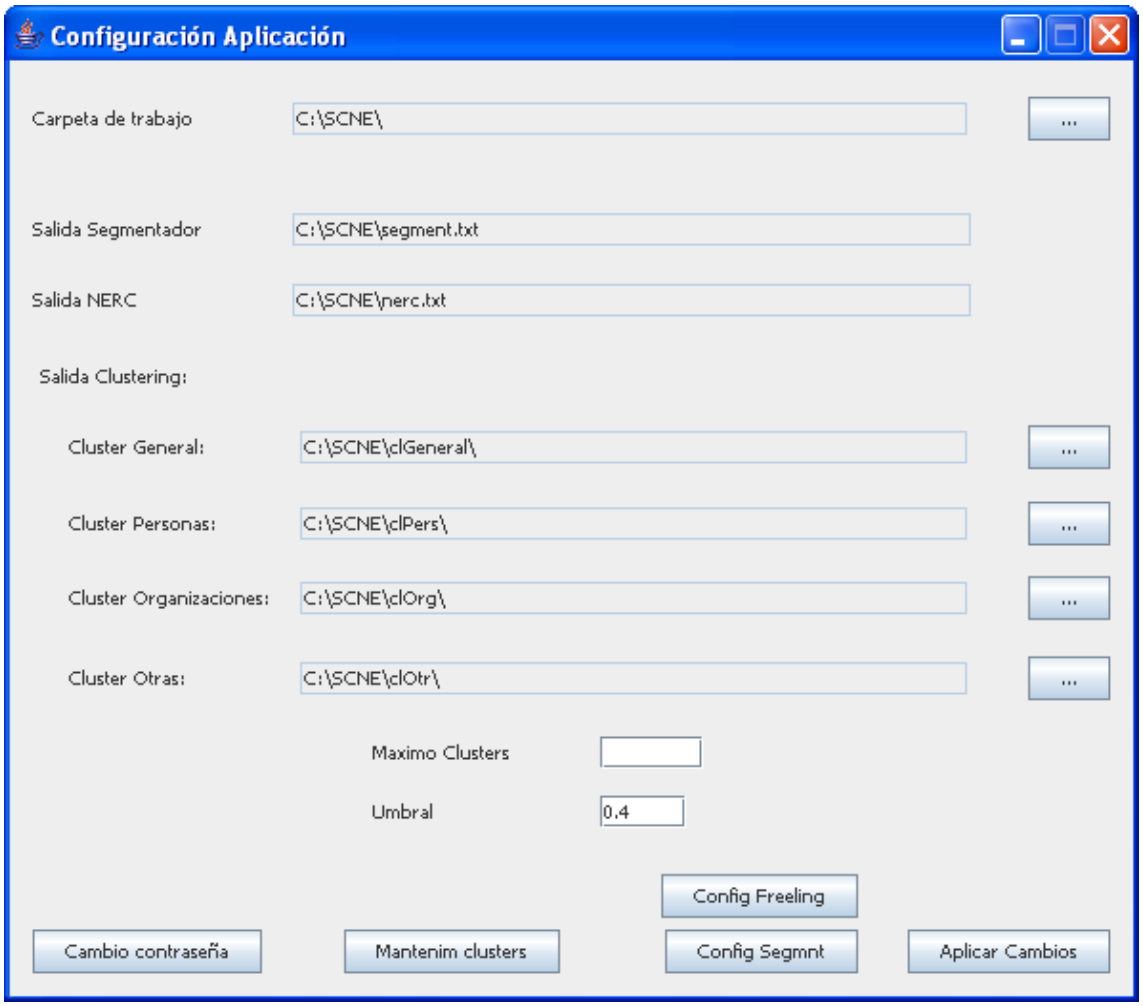


Figura A3.10: pantalla principal de configuración de rutas

Si se desea hacer un cambio de contraseña para el usuario administrador, debe pulsarse el botón “Cambio de contraseña” y aparecerá la ventana que permite realizar la acción (figura A3.11).

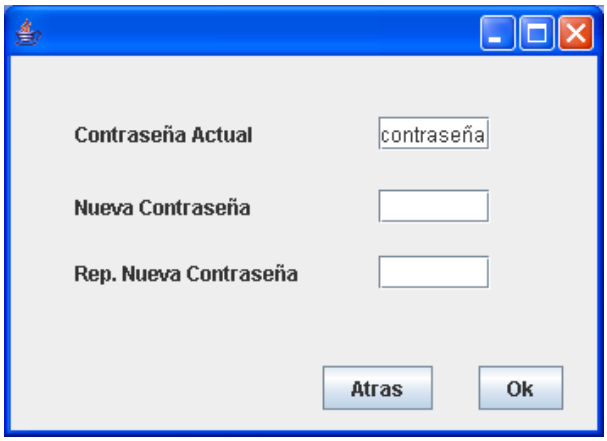


Figura A3.11: pantalla de cambio de contraseña

Si se desea cambiar los parámetros de configuración de FreeLing, se debe acceder mediante el botón “Config Freeling”. La pantalla se presenta en la figura A3.12.

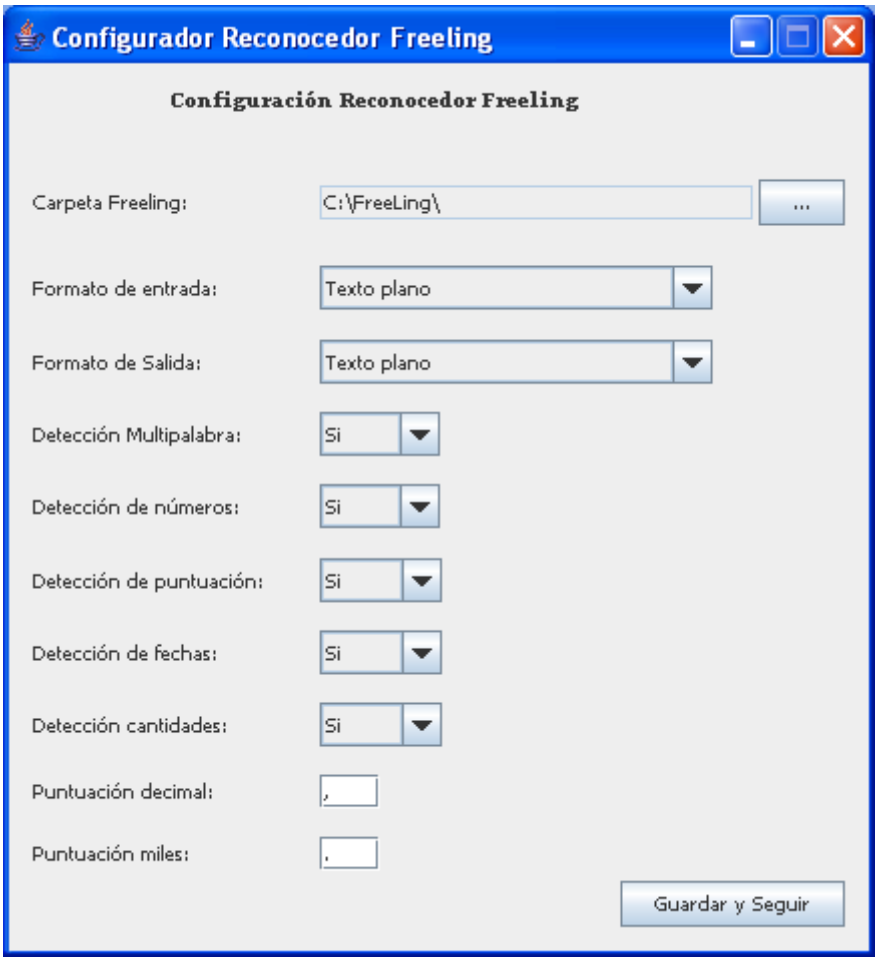


Figura A3.12: pantalla de configuración Freeling

Otra opción de configuración es la elección de los caracteres a utilizar para segmentar un texto con el *tokenizador* básico implementado en la aplicación, para ello se deberá pulsar en el botón “Config Segment” y aparecerá la pantalla siguiente (figura A3.13).

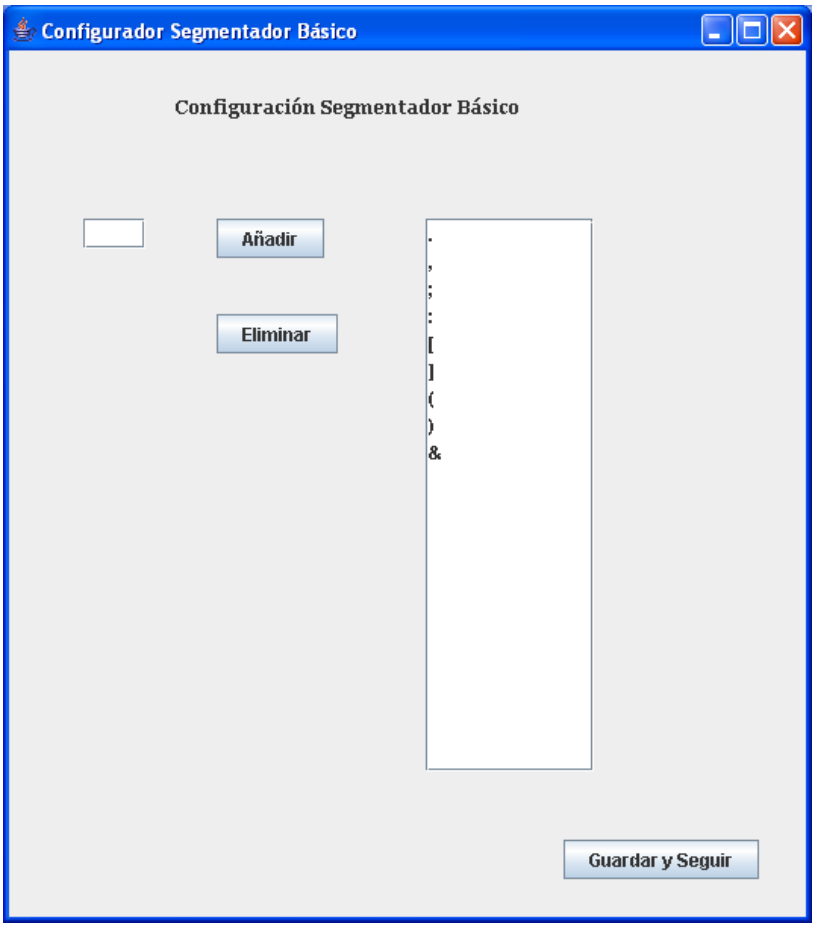


Figura A3.13: pantalla de configuración Tokenizador básico

Finalmente, desde la pantalla configuración se puede acceder al apartado de mantenimiento de los *clusters* existentes. Se pueden añadir, eliminar o modificar tanto estos cómo las listas de *NE* que existen dentro de cada uno de ellos. Para ambas acciones existen los botones de la parte inferior de la pantalla que llevan a esas nuevas opciones (figura A3.14).

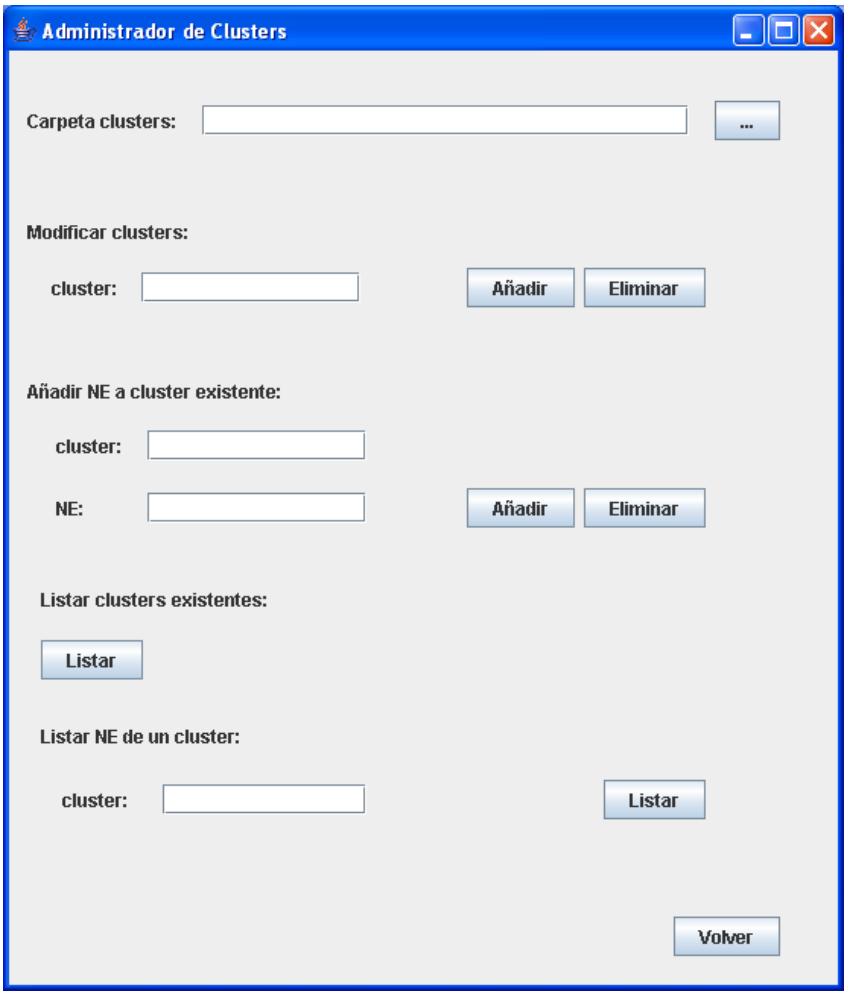


Figura A3.14: pantalla de mantenimiento de clusters

Si el usuario administrador no desea guardar los cambios realizados, en cualquier momento puede cerrar la pantalla y no tendrán efecto. Solamente lo tendrán al pulsar los botones creados para tal fin.

Ejecución en línea de comandos

El segundo método de ejecución de la aplicación es la llamada por línea de comandos. Se han cubierto todas las funcionalidades que se pueden encontrar en modo gráfico menos la descarga URL. La instrucción en este caso tiene la estructura que sigue:

java –jar [ruta fichero proyecto.jar] –e:[ruta entrada] –t:[tipo entrada] –r/-rc:aop –c:u:X.Xn:YYYY –[lengua]

Donde:
-e: indica la entrada que se desea para la aplicación. Puede ser un fichero o bien un directorio del sistema dónde exista la colección de ficheros que se desea hacer *clustering*. En el caso de directorio, SCNE aplicará *clustering* sobre los ficheros cuya extensión sea .HTML o .txt en este orden.

-t: indica el tipo de entrada que se escogerá. Las opciones son:
doc: documento de texto
col: colección de textos
neg: fichero de *Named Entities* generales definidas.
nep: fichero de *Named Entities* de personas definidas.
neo: fichero de *Named Entities* de organizaciones definidas.

-r/-rc: opción de reconocimiento o reconocimiento y clasificación de NE. En el caso que se desee reconocimiento y clasificación tenemos la opción de elegir qué tipos de NE queremos: a→ otras; o→ organizaciones; p→ personas y se pueden combinar de la forma ap, op, apo, opa, o etc. Si únicamente se desea reconocimiento general de NE se deberá escribir –r sin ningún otro parámetro.

-c:u:x.xn:yyyy: opción de *clustering*. En este caso x.x equivale al umbral deseado a partir del cuál *clusterizar*. Siempre será un valor entre 0.0 y 1.0 y se leerá de forma 0% de semejanza o 100% de semejanza. yyyy es el número máximo de *clusters* a realizar y es un valor definido en el intervalo 0000-9999.

-[lengua]: opción que indica la lengua con la que se tratan las entradas. Las definiciones de los idiomas:
es: castellano
ca: catalán
en: inglés

Ejemplos varios:
java –jar c:\SCNE\dist\scne.jar –e:c:\prueba.txt –t:col –r –c:u:0.6n:1000 -es
java –jar c:\SCNE\dist\scne.jar –e:c:\prueba.txt –t:doc –rc:op –c:u:0.2n:0500 -es
java –jar c:\SCNE\dist\scne.jar –e:c:\prueba.txt –t:nep –rc:ao –c:u:0.9n:7245 -ca
java –jar c:\SCNE\dist\scne.jar –e:c:\prueba.txt –t:neo –rc:ap –c:u:0.2n:1500 -en

Se ha pensado esta forma de ejecución debido a que la aplicación, de esta manera, podrá ser llamada desde terceros sistemas mediante *scripts* pudiendo aprovechar el *clustering* que realiza para otras finalidades.

Al finalizar el proceso deseado, este método de ejecución guarda los resultados sin presentarlos por pantalla en las rutas de salida predeterminados por el administrador. Para consultarlos se deben abrir por separado los ficheros.

Para finalizar el capítulo, SCNE posee una ayuda interna para la ejecución en línea de comandos que se puede consultar ejecutando:

java –jar [ruta fichero proyecto.jar] -h

Ej: java –jar c:\SCNE\dist\SCNE.jar -h

Esta instrucción presenta en línea de comandos un pequeño resumen de ayuda para guiar al usuario a su ejecución.

FICHEROS DE Ejemplo proporcionados con SCNE:

Fichero de NE de Personas: C:\SCNE\pruebas\personas.txt

Fichero de NE de Organizaciones: C:\SCNE\pruebas\organizaciones.txt

Pruebas con ficheros de evaluación: C:\SCNE\pruebas\

Pruebas para el castellano: C:\SCNE\pruebas\cast\

Pruebas para el catalán: C:\SCNE\pruebas\ca\

Pruebas para el inglés: C:\SCNE\pruebas\en\