Sistemas Informáticos

Curso 2011-2012

DESARROLLO DE UNA APLICACIÓN DE AYUDA PARA LA REDACCIÓN DE TEXTOS SIMPLIFICADOS

Realizado por:

Marta Rodríguez García

Teresa de Salas Garrido

Ana Belén Vargas Celino

Dirigido por:

Pablo Gervás Gómez-Navarro

Raquel Hervás Ballesteros

Dpto. Ingeniería del Software e Inteligencia Artificial

Facultad de Informática

Universidad Complutense de Madrid

Marta Rodríguez García, Teresa de Salas Garrido y Ana Belén Vargas Celino autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado.

Fdo. Marta Rodríguez García	Fdo. Teresa de Salas Garrido	Fdo. Ana Belén Vargas Celino

Resumen

Hoy en día en la sociedad en la que vivimos es imposible relacionarse si no se dispone de una buena comprensión lectora, ya que la posibilidad de leer aporta a las personas una gran confianza para expresar sus opiniones. Por desgracia, existen muchas personas que tienen problemas para leer, escribir o entender. Además las distintas organizaciones y personas como editores, escritores, profesores y traductores que elaboran textos basados en criterios de Lectura Fácil disponen de pocas facilidades en el momento de redactar un texto para su fácil lectura.

El presente proyecto de Sistemas Informáticos se ha desarrollado para facilitar el trabajo a los expertos que se dedican a la elaboración y simplificación de textos en inglés para su fácil lectura. Para ello integra herramientas de Procesamiento de Lenguaje Natural sobre un editor de texto multiplataforma y de código libre ya existente como es jEdit.

Se han desarrollado dos herramientas que ayudan en el proceso de simplificación y que se integran en forma de plugins sobre jEdit ya que éste permite ampliar su funcionalidad mediante el uso de los mismos.

La primera de las herramientas desarrolladas es la de simplificación léxica. Esta herramienta nos proporciona sinónimos para una palabra, ya que una de las formas más comunes de simplificación consiste en la sustitución de palabras complejas por palabras más sencillas.

La segunda herramienta desarrollada es la de simplificación sintáctica. Esta herramienta permite la separación de frases largas que contienen la conjunción coordinada “and” en frases más cortas sin estas conjunciones, cumpliendo con el objetivo, recomendado por distintas asociaciones que abordan la tarea de la simplificación de texto, de expresar una única idea por frase.

Ambas herramientas cuentan con interfaces similares y fáciles de manejar para personas con pocos conocimientos de informática. Además ambas herramientas cuentan con funcionalidades comunes tales como permitir guardar un registro de los cambios realizados en el documento y la de indicar visualmente en cada momento de qué tipo de simplificación se trata.

Finalmente, por todo lo anteriormente descrito, hemos querido que este proyecto sea software libre y multiplataforma, y una gran herramienta de apoyo en este campo de tal forma que facilite el trabajo de simplificación a los expertos de la Lectura Fácil.

Abstract

Nowadays the society in which we live does not allow people to interact if they do not have good reading comprehension, since the ability to read gives people a great confidence in order to express their points of view. Unfortunately, there are many people who have trouble when reading, writing or understanding. Moreover, many organizations and people as editors, writers, teachers and translators that produce texts based on Easy to Read criteria have few facilities when writing a text for easy reading.

This project has been developed to provide help for experts engaged in the development and simplification of English texts for easy reading. This project includes tools for Natural Language Processing on a multiplatform and open source text editor as jEdit.

Two tools have been developed which have been integrated as plugins on jEdit as its functionality can be easily extended in this way.

The first tool developed is a lexical simplification tool. This tool provides synonyms for a word, since one of the most common simplifications is the replacement of complex words by simpler ones.

The second tool developed is a syntactic simplification tool. This tool makes possible the separation of long sentences containing the coordinated conjunction “and” into shorter sentences without this conjunction, since it is usually recommended to express one idea per sentence.

Both tools have similar and user friendly interfaces for people that do not have a big knowledge about computer science. In addition, both tools have shared features such as a log of changes made on the document and visual indications about what type of simplification is being applied.

Finally, we have decided this to be an open source and multiplatform project, and a support tool which can make easier the work of text simplification for Easy Reading.

Palabras clave

- jEdit
- Lectura Fácil
- Log
- Plugin
- Procesamiento del Lenguaje Natural
- Simplificación Léxica
- Simplificación Sintáctica

Keywords

- jEdit
- Easy-to-read
- Log
- Plugin
- Natural Language Processing
- Lexical Simplification
- Syntactic Simplification

Índice

Índice de ilustraciones

Capítulo 1
Introducción

Capítulo 1 – Introducción

En el presente capítulo comentaremos la motivación y objetivos del proyecto, así como también la forma en la que se ha desarrollado el mismo.

1.1 Motivación

La posibilidad de leer aporta a las personas una enorme confianza, de tal forma que les permite expandir sus opiniones y ejercer un control sobre sus propias vidas. Sin embargo hay muchas personas que no poseen la suficiente capacidad para leer con fluidez, escribir o entender, por lo que la forma en la que se redacta un texto o se presenta la información puede excluir a estas personas de participar en la vida socioeconómica de la sociedad en la que viven. En esta nueva sociedad de la información es importante el uso y entendimiento de los sistemas de información que se encuentran actualmente en desarrollo, pero es esta misma sociedad la que muchas veces restringe el acceso a dicha información. Es por ello que existen diversas asociaciones que trabajan con entidades, editoriales e instituciones públicas asesorando en la elaboración de textos basados en los criterios de lectura fácil. Aun así son pocos los esfuerzos sistemáticos que se han hecho para documentar este tema, siendo el trabajo realizado en este sentido escaso de forma que las organizaciones y personas como editores, escritores, profesores y traductores pocas veces disponen de facilidades en el momento de adaptar un texto para su lectura fácil.

El hecho de que un texto sea fácil de leer y entender depende en gran parte de las capacidades y experiencia del lector, por lo cual el concepto de lectura fácil no puede ser universal y no se puede elaborar un texto que se adapte a las capacidades de todas las personas con problemas de lectura y comprensión. Es por esto que facilitar la tarea a los expertos que adaptan los textos de manera personalizada a cada colectivo necesitado es tan importante.

Por lo anteriormente descrito este proyecto va dirigido fundamentalmente a las personas que abordan la tarea de redactar un texto fácil de leer mediante la simplificación de otro ya existente cuya comprensión resulta más complicada. En un principio está dirigido a personas de habla inglesa ya que las herramientas utilizadas para el desarrollo del mismo están más extendidas en este idioma. El problema que se encuentra con estas herramientas es que no disponen de una interfaz amigable al uso del público experto en simplificación de textos, ya que requieren conocimientos de informática y programación. Nuestro proyecto trata de resolver estos problemas mediante el desarrollo de un software libre que ofrece la posibilidad de poder gestionar cambios a nivel léxico y sintáctico sobre un texto base.

Dado que uno de los aspectos que se contempla en el momento de realizar la simplificación de un texto es el hecho de utilizar un lenguaje sencillo, la herramienta desarrollada facilita esta tarea. Para ello hace uso de una de las formas más sencillas que permite expresar en un lenguaje más simple una palabra que se considera compleja y esto es mediante el uso de sinónimos. La herramienta léxica desarrollada ofrece al usuario la posibilidad de buscar una

serie de sinónimos para una determinada palabra. Además el usuario tiene la opción de poder editar la palabra compleja en caso de no estar conforme con lo que le ofrece la aplicación.

Otro de los aspectos que se contempla durante el proceso de simplificación es el hecho de expresar una única idea por frase, es por ello que la herramienta sintáctica desarrollada facilita esta labor. La herramienta sintáctica permite dividir frases largas cuya estructura sintáctica incluya la conjunción coordinada "*and*", consiguiendo de esta forma un texto mucho más claro y preciso.

Tanto la herramienta de simplificación léxica como la sintáctica se han desarrollado como plugins para un editor de texto ya existente.

Los cambios que se realicen utilizando ambas herramientas quedan almacenados de manera permanente mediante un *log* de cambios, que servirá para estudiar y automatizar las simplificaciones en un futuro.

Esperamos con todo lo anteriormente descrito que las personas que escriben textos y facilitan información dirigida a personas cuyas capacidades de lectura y entendimiento están disminuidas, encuentren en este proyecto una valiosa herramienta que les permita generar sus textos y que sirva como estímulo para la generación de documentos de fácil lectura en diversas lenguas.

1.2 Objetivos

En vista a las necesidades de apoyo lingüístico para la elaboración de textos para su fácil lectura se plantean los siguientes objetivos:

- Estudio de las tecnologías de Procesamiento de Lenguaje Natural que pueden resultar útiles para la simplificación semi-automática de textos.
- Desarrollo de una herramienta de simplificación léxica con una interfaz fácil de manejar e intuitiva que permita al editor realizar su trabajo de simplificación más cómodamente. Para ello se hará uso de herramientas de Procesamiento de Lenguaje Natural ya existentes y de un diccionario.
- Desarrollo de una herramienta de simplificación sintáctica con una interfaz fácil de manejar e intuitiva que permita al editor realizar su trabajo de simplificación sintáctica más cómodamente. Para ello se hará uso de herramientas de Procesamiento de Lenguaje Natural ya existentes.
- Diseño e implementación de una arquitectura general que permita integrar las herramientas mencionadas en un conocido editor de textos. La arquitectura permitirá también la fácil integración de nuevas herramientas de simplificación que se implementen en el futuro. Además esta arquitectura incluirá funcionalidades para almacenar los cambios realizados y su futuro análisis para conseguir automatizar el proceso.
- Publicación como software libre multiplataforma de las herramientas implementadas.

1.3 Estructura del documento

Este documento sigue la misma estructura que los objetivos que nos fijamos al inicio de este proyecto.

Al principio era necesario estudiar y analizar el contexto sobre el que íbamos a trabajar, es por ello que la primera parte de este documento (capítulo 2) se centra en el análisis del contexto de la lectura fácil, las diferentes asociaciones existentes en este ámbito y los proyectos parecidos al que queríamos desarrollar. Además en esta primera parte también analizamos las herramientas de Procesamiento del Lenguaje Natural que íbamos a necesitar para la realización del proyecto, así como los distintos editores de texto para seleccionar el más adecuado.

A continuación (capítulo 3) se explica con detalle todo lo relacionado con el editor de texto escogido, lo necesario para la instalación, creación y desarrollo de los plugins de este editor.

Una vez puestos en contexto comenzamos con el diseño y desarrollo de la arquitectura general (capítulo 4) para el plugin léxico y el plugin sintáctico. Para ello se realizó un análisis de distintos textos comparando el texto original con el texto simplificado. Tras el estudio de este análisis se decidió desarrollar dos plugins, uno encargado de la parte léxica y el otro de la parte sintáctica. Además se comentan las estructuras comunes de ambos plugins.

Tanto en el capítulo 5 como en el capítulo 6 se explica la funcionalidad, las herramientas usadas y la implementación de los dos plugins desarrollados.

Finalizamos el documento con las conclusiones y el trabajo futuro que se podría desarrollar al término de este proyecto.

Adjuntamos tres anexos que ayudan al correcto entendimiento de distintas partes del proyecto.

1.4 Gestión del proyecto

Para la organización y desarrollo del proyecto se han usado varias herramientas de apoyo. Para la implementación del proyecto hemos usado el lenguaje Java y el entorno de programación Eclipse, de código abierto y multiplataforma. Además este entorno integra Ant que permite la construcción de los plugins. Para alojar el código del proyecto se ha usado Google Project Hosting, repositorio Subversion gratuito que permite alojar proyectos open source como es nuestro caso. Para la gestión y control sobre los cambios efectuados en el código del proyecto se ha usado TortoiseSVN cliente gratuito de código abierto para el sistema de control de versiones Subversion. Sin embargo, los avances en la documentación del proyecto se han realizado mediante un servicio de alojamiento de archivos multiplataforma en la nube, como es el caso de Dropbox.

Para poder llevar a cabo el desarrollo del proyecto y teniendo en cuenta los objetivos anteriormente descritos, éste se ha dividido en tres iteraciones.

1.4.1 Iteración 1

Objetivos

- ✓ Entrar en contexto con la lectura fácil
- ✓ Comenzar a escribir la memoria
- ✓ Selección de un editor de texto base
- ✓ Resaltado de las palabras en el texto
- ✓ Desarrollo de la herramienta léxica
- ✓ Análisis de textos

Desarrollo

En esta primera iteración se empezó buscando información relacionada con la lectura fácil y las distintas asociaciones que apoyan a aquellas personas que se dedican a la simplificación de textos y ejemplos de las distintas herramientas que hiciesen una función parecida a lo que queríamos hacer en nuestro proyecto.

Gracias a las simplificaciones que realizaron personas de habla inglesa realizamos un análisis de textos simplificados, para así estudiar las simplificaciones que realizaron y poder concretar qué herramientas íbamos a necesitar desarrollar. Con este estudio concluimos que era necesario una herramienta de simplificación léxica que nos ofreciese los sinónimos de las palabras que se consideren complicadas y otra herramienta de simplificación sintáctica que pudiese partir en dos las frases largas y así expresar un único concepto por frase.

Se comenzó a redactar la memoria con toda la información obtenida.

También realizamos la búsqueda del editor que más se ajustase a nuestras necesidades. Una vez seleccionado, aprendimos a crear un plugin y a interactuar con el entorno escogido. Incluimos en el plugin las herramientas de Procesamiento del Lenguaje Natural, para poder realizar las simplificaciones léxicas.

Una labor complicada fue el resaltado de las palabras en el editor. En esta primera iteración se implementó una primera versión de esta funcionalidad, pero no lo suficientemente potente.

Grado de consecución

Comenzamos la redacción de la memoria con la información obtenida de la lectura fácil y con las conclusiones obtenidas de los análisis de textos. Especificamos las herramientas que íbamos a desarrollar.

Esta primera iteración la finalizamos con un prototipo de la herramienta léxica la cuál obtenía los sinónimos de una palabra seleccionada y resaltaba dicha palabra.

1.4.2 Iteración 2

Objetivos

- ✓ Ampliación de la memoria
- ✓ Depuración de la herramienta léxica

- ✓ Resaltado de las palabras en el texto
- ✓ Gestor de cambios en el documento (log)

Desarrollo

Para poder realizar el plugin e incluirlo en el entorno de jEdit, fue necesario el desarrollo de diversos archivos como el archivo de ayuda, el archivo que interactúa con la interfaz o el archivo de construcción.

Comenzamos a guardar los cambios realizados sobre el documento en un log, dándole una primera estructura y viendo que era lo que realmente queríamos almacenar, para poder semi-automatizar en algún futuro estas simplificaciones.

En el resaltado de palabras se decidió utilizar una modificación de un plugin de jEdit ya existente. Con las modificaciones en el plugin, conseguimos que el resaltado fuese de todas las apariciones de la palabra seleccionada.

En la memoria incluimos como realizar los archivos necesarios para crear un plugin, la herramienta de resaltado utilizada y el primer diseño del log.

Grado de consecución

Esta segunda iteración la finalizamos con la terminación del plugin léxico, que cuenta con una interfaz intuitiva y fácil de usar. Incluimos todos los archivos necesarios para su correcto funcionamiento. Modificamos el plugin del resaltado para que cumpliese las funciones necesarias. Por último guardábamos un log de las palabras sobre las que se habían aplicado las simplificaciones léxicas. Además continuamos ampliando la memoria.

1.4.3 Iteración 3

Objetivos

- ✓ Finalización de la memoria
- ✓ Desarrollo de la herramienta sintáctica
- ✓ Gestor de cambios (log)
- ✓ Publicación de los plugins

Desarrollo

En esta tercera iteración se incluyó un segundo plugin sintáctico que se encargase de la realización de las simplificaciones sintácticas, ya que en el análisis previo de textos era una simplificación que se utilizaba a menudo. Incluye las herramientas de Procesamiento de Lenguaje Natural, el resaltado de palabras y un log para registrar también las simplificaciones sintácticas realizadas.

Adaptamos el log para que se almacenasen tanto las simplificaciones sintácticas como las simplificaciones léxicas de un documento en un mismo log y no en dos logs independientes, para facilitar el posterior análisis de las simplificaciones al intentar automatizarlas.

Comenzamos el proceso para poder publicar los plugins en la página oficial del editor de texto, ya que este editor dispone de una central de plugins en donde se pueden publicar los nuevos plugins. A la vez se inició el proceso de publicación del código en SourceForge para su libre uso junto con los dos archivos que son los plugins que se deben añadir al editor de texto. De esta forma el proyecto ha sido publicado como software libre y multiplataforma.

Incluimos en la memoria todo lo añadido al proyecto en esta última iteración.

Grado de consecución

Desarrollamos el plugin sintáctico con una interfaz sencilla, con los archivos necesarios para la construcción y funcionamiento del plugin.

Realizamos una estructura final del log que contemplase todas las simplificaciones que se podían realizar.

Terminamos con la edición de la memoria incluyendo la estructura final del log, la redacción de éste segundo plugin y las conclusiones a las que hemos llegado al finalizar el proyecto.

Todo el proyecto está publicado en SourceForge y se puede encontrar en las páginas https://sourceforge.net/projects/synonymsp/ y https://sourceforge.net/projects/splitp/ .

A la hora del término de esta memoria la publicación de los plugins en la página oficial de jEdit se encontraba en proceso.

Capítulo 2

Estado del Arte

Capítulo 2 - Estado del Arte

En este capítulo vamos a introducir algunas nociones referentes al campo sobre el que vamos a aplicar el proyecto, las herramientas existentes en este ámbito así como también las herramientas que nos servirán de ayuda para el desarrollo del presente trabajo.

2.1 Lectura Fácil

2.1.1 ¿Qué es la Lectura Fácil?

La lectura fácil es una corriente que busca crear materiales redactados en un lenguaje claro, y que permitan una comunicación eficaz, para que puedan ser leídos y entendidos por personas con dificultades para la compresión lectora.

Un documento de fácil lectura según la definición que aparece en las Directrices Europeas para Facilitar la Lectura (Freyhoff, y otros 1998) es aquel que contiene sólo la información más importante, expresada y presentada de la forma más directa, de modo que su contenido pueda ser comprendido por el mayor número de personas posible.

Los textos de lectura fácil se dirigen a personas que tienen capacidades inferiores a la capacidad lectora que se le supone a un adulto hablante nativo. El colectivo destinatario de este tipo de documentos forma el 30% de la población. Podemos dividirlo en dos grandes grupos, las personas con dificultades lectoras transitorias (inmigrantes que no conocen bien el idioma del país de acogida, personas con información cultural limitada, niños de educación primaria que empiezan a leer, etc.) y las personas con dificultades lectoras permanentes (discapacidades físicas, tales como sordos, ciegos, etc., o discapacidades psíquicas, como disléxicos, autistas, afásicos, etc.). En la ilustración 1 se puede observar los distintos grupos de personas que tienen problemas con la compresión lectora (círculos), así como la necesidad de materiales de lectura fácil por estos grupos (cuadrado).

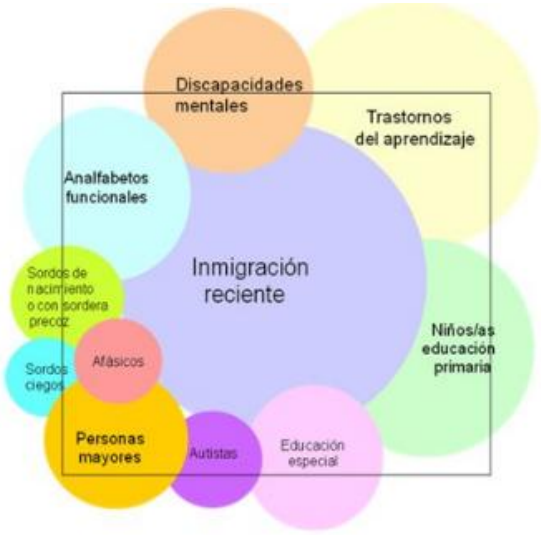

ILUSTRACIÓN 1: Gráfico Obtenido de la Asociación de Lectura Fácil de Cataluña

2.1.2 Directrices para la Lectura Fácil

Existen unas Directrices Europeas para Facilitar la Lectura (Freyhoff, y otros 1998) que ayudan a crear los textos de lectura fácil, y orientan al creador en cuanto al contenido, el lenguaje y la forma del texto. A pesar de ser unas directrices generales y dado que el concepto de lectura fácil no es un concepto universal, estas directrices se adaptan mejor a las necesidades de las personas con discapacidades que a las personas con dificultades transitorias. Hay cuatro características generales:

1. Usar un lenguaje simple y directo.
2. Expresar una sola idea por frase.
3. Evitar tecnicismos, abreviaturas e iniciales.
4. Estructurar el texto de manera clara y coherente.

El modo de estructurar el documento es importante, tiene que seguir un orden claro y coherente. Las ideas, vocablos, oraciones o frases innecesarias hay que evitarlas o quitarlas. Las fotografías, gráficos o símbolos servirán de apoyo al texto. La finalidad es conseguir una publicación fácil de entender y de leer por el mayor número posible de personas.

A la hora de redactar un documento de fácil lectura se pueden dar dos situaciones: que se disponga ya de un texto base que se quiera hacer accesible, o que se quiera generar un texto completamente nuevo. Para facilitar este trabajo en ambos casos se recomienda seguir los siguientes pasos.

PASO 1: decidir la finalidad de la publicación.

¿Qué es lo que queremos decir y por qué es importante para las personas con dificultades? Con esto hallamos el principal objetivo: decidimos qué detalles incluir o evitar y si incluir dibujos o ilustraciones.

PASO 2: abordar el tema del contenido.

Elaborar una lista con los aspectos clave de la publicación.

- Seleccionar las secciones más importantes (se pueden omitir introducciones, comentarios…).
- Resumir los párrafos en una o dos oraciones. Cuanto más corto sea mejor será el documento.
- Comprobar si los resúmenes siguen una estructura lógica.
- Verificar que los resúmenes incluyen únicamente los aspectos clave.

Plasmar claramente su contenido y seguir un lógico (en el caso de estar ya escrito). Si es nuevo texto, estructurar el contenido de forma clara y coherente.

PASO 3: elaborar el borrador del texto.

Redactar el texto basándose en la lista de aspectos clave. Para dirigirse al mayor número posible de personas es preferible:

- Usar un lenguaje sencillo y directo.

- Evitar los conceptos abstractos.
- Emplear vocablos cortos relativos al lenguaje cotidiano hablado (escribiendo de acuerdo a la edad dirigida).
- Personificar el texto tanto como sea posible.
- Hacer uso de ejemplos prácticos.
- Dirigirse a los lectores de manera respetuosa.
- Utilizar oraciones cortas en su mayoría.
- Incluir una sola idea principal en cada oración.
- Utilizar un lenguaje positivo.
- Emplear preferentemente la voz activa frente a la pasiva.
- No dar por asumidos conocimientos previos sobre el tema en cuestión.
- Ser sistemático al utilizar las palabras (utilice la misma palabra para nombrar una misma cosa).
- Elegir signos de puntuación sencillos. Es recomendable no usar el punto y coma y el guión.
- No emplear el subjuntivo.
- Tener cuidado con el lenguaje figurativo o metafórico si son vocablos de uso poco común.
- Cuidado con el uso de números. Es mejor usar “muy alto” que el número concreto para referirse a números grandes, y evitar usar un porcentaje usando palabras que lo represente (la mayoría, algunos, etc.). Para hacer referencias a años lejanos en el tiempo es recomendable usar expresiones como “hace mucho tiempo”.
- No emplear palabras de otro idioma.
- Evitar el uso de referencias internas al documento.
- Mencionar una dirección de contacto para obtener mayor información, cuando sea posible.
- Evitar el uso de jergas, abreviaturas e iniciales.

Si se generan documentos largos, es mejor dividir el documento en varias secciones cortas para facilitar la lectura.

PASO 4: comprobar si las personas destinatarias del documento entienden el borrador que se ha elaborado y retocarlo para satisfacer las necesidades del grupo destinatario.

PASO 5: realizar una nueva comprobación.

Una vez realizados los cambios vuelva a ponerse en contacto con el grupo que leyó el borrador. Si no queda del todo claro, volver a realizar el mismo procedimiento.

Los dibujos o fotografías pueden facilitar la comprensión, por lo que no son únicamente un elemento decorativo, sino un medio de transmitir información. Los símbolos son una forma de comunicación más general y abstracta, suelen ser dibujos lineales que representan objetos, acciones o ideas, y pueden ser una buena opción para un grupo reducido y con un problema de comprensión lectora similar.

El diseño de un documento es importante para facilitar la lectura. No es recomendable usar, por ejemplo, letras blancas sobre fondo de color, o distintos tipos de letras en el mismo documento. Algunas pautas para el diseño son:

- No hacer uso de dibujos como fondo del texto.
- Una sola línea por oración.
- Papel mate y de buena calidad.
- No incluir demasiada información en la página.
- Como máximo dos tipos de letras: una para texto y otra para títulos.
- Letras claras (Arial, Helvética o Times New Roman).
- Letra grande (mínimo tamaño 14).
- No usar mayúsculas ni cursivas (usar negrita o subrayado).
- Ilustraciones nítidas.
- Nunca texto claro sobre fondo oscuro.
- Utilizar colores para los dibujos, los recuadros, etc. siempre que sea posible.
- No alinear el texto a la derecha.
- No usar guiones para separar palabras largas, mantener juntas las palabras.
- Para los números usar el carácter numérico, no la palabra equivalente. Para las fechas mejor usar el formato completo, por ejemplo, “Sábado, 26 de Septiembre de 1998”), y los números telefónicos separados (por ejemplo, 034-22.33.44).
- Incluir encabezamientos para facilitar la navegación.
- Garantizar amplia distribución (A4 o A3 doblado).
- Incluir la fecha de la publicación.

Las publicaciones de fácil lectura deberán ser claramente etiquetadas en la página del título de modo que los usuarios puedan identificarlas con facilidad.

También es recomendable el uso de otros formatos como complemento para entender mejor el texto. Por ejemplo:

- Cintas magnetofónicas: grabar el texto en una cinta es una buena opción para facilitar la lectura. Se debe leer de manera acompasada haciendo breves pausas entre frases. Si es un texto largo es recomendable usar más de una voz. Se puede alternar con música o efectos de sonido.
- Video: la combinación verbal y visual es una gran herramienta. Se debe disponer de un guión claro y coherente y es conveniente que ni el texto ni las ilustraciones cambien con demasiada rapidez.
- Medios interactivos: puesto que el uso de ordenadores es una costumbre en alza, esto puede facilitar la comprensión por parte de las personas necesitadas.

Las directrices ayudan a dar un formato al texto y características básicas del mismo, pero es necesario conocer al colectivo destinatario de la publicación para ajustar el documento a sus necesidades.

2.1.3 Asociaciones e iniciativas. ¿Quién promueve la Lectura Fácil?

La lectura fácil es una corriente en auge. Existen numerosas asociaciones que trabajan en este sector.

La **IFLA** (International Federation of Library Associations and Institutions) (IFLA 1971) es una organización mundial fundada en 1927 y registrada en 1971 para proporcionar a los bibliotecarios de todo el mundo una forma de intercambiar ideas, promoviendo la investigación y el desarrollo en todos los campos relacionados con la actividad bibliotecaria y la bibliotecología. Se trata de una organización no gubernamental sin ánimo de lucro cuyos objetivos son representar la profesión de bibliotecario, promover su formación permanente, y desarrollar, mantener y promover directrices para los servicios bibliotecarios. Esta asociación cuenta con más de 1700 miembros en 150 países. Una de sus publicaciones más importantes es el documento mencionado anteriormente: Directrices Europeas para Facilitar la Lectura (Freyhoff, y otros 1998).

En Iberoamérica encontramos **SIDAR** (Fundación Sidar 1997), asociación que cuenta con un grupo de trabajo sobre lectura fácil. Sus principales objetivos son estimular el diseño accesible en la web, el intercambio de información sobre directrices, herramientas y normas de accesibilidad en internet y promover el acceso a la red por parte de personas con discapacidad.

A nivel europeo podemos encontrar la ILSMH (International League of Societies for Persons with Mental Handicap) que forma parte de Inclusion International (Inclusion International s.f.) asociación que trabaja a nivel mundial defendiendo los derechos de las personas con discapacidades intelectuales y que fue constituida en 1988 y posteriormente renombrada **Inclusion Europe** (Inclusion Europe aisbl. 1988). Esta asociación es una organización sin ánimo de lucro que defiende los derechos e intereses de las personas con discapacidad intelectual. Coordina actividades en varios países europeos y organiza conferencias, grupos de trabajo y reuniones de intercambio. Partiendo de las directrices de la IFLA, esta asociación está desarrollando unas directrices más amplias para el desarrollo de texto de fácil lectura y traduciéndolas a todas las lenguas de la UE.

A nivel nacional, la asociación pionera es la **Asociación de Lectura Fácil** (Associació Lectura Fàcil 2002) (ALF) con sede en Barcelona y fundada en el 2003. Tiene servicios editoriales, de formación y de asesoramiento para la creación de textos de fácil lectura. A cada una de las publicaciones que se ajustan a las directivas de la IFLA les otorga el logotipo de Lectura Fácil (LF).

En Extremadura, el proyecto **Vive la Fácil Lectura** (Vive la fácil lectura s.f.) se pone en marcha con la colaboración de la Consejería de Cultura de la Junta de Extremadura y la Asociación Regional de Universidades Populares de Extremadura (AUPEX). Se trata de un proyecto destinado a los colectivos con más problemas de comprensión lectora que cuenta con numerosos textos de Lectura Fácil, desde guías naturales a texto explicativos de carácter tecnológico.

A título individual, Jaume Serra llevó a cabo un proyecto sobre la importancia de la lectura fácil para la compresión lectora del alumnado de Educación Secundaria inmigrante. Como resultado de este proyecto se consiguió la traducción simplificada al catalán del libro de Robert Louis Stevenson "La Flecha Negra" (Serra Milà, La fletxa negra 2007), además de algunos textos de aplicación en el ámbito escolar. También se elaboró una guía mencionando los pasos seguidos durante el proceso (Serra Milà, La lectura fàcil: una necessitat per a la inclusió de l'alumnat nouvingut d'ESO 2007).

2.1.4 PorSimples

El proyecto portugués PorSimples (Aluísio 2007) propone el desarrollo de una tecnología que facilite el acceso a la información a las personas con discapacidades cognitivas. Este proyecto propone dos sistemas para diferentes públicos. Por una parte se propone un sistema de creación que ayude a los autores a producir textos simplificados que serán validados por ellos mismos. Por otro lado se propone un sistema que ayude a las personas con discapacidades a leer determinados contenidos de la web, tales como textos producidos por el gobierno o periódicos de circulación general de forma que se promueva la inclusión digital y la accesibilidad. Esta tarea incluye la simplificación mediante un resumen textual y sintáctico del texto original. El idioma de trabajo de este proyecto es el portugués. Se trata del proyecto de simplificación de textos que está más avanzado.

Herramienta SIMPLIFICA

La herramienta Simplifica (PorSimples 2009) pertenece al proyecto web PorSimples. Es una herramienta que permite a los escritores adaptar textos originales a textos simplificados. Dentro de la aplicación el autor juega un papel activo en la generación de textos simplificados naturales o fuertemente simplificados aceptando o rechazando las simplificaciones propuestas por el sistema sobre una sentencia base y pudiendo editarla si es necesario.

A pesar de poder tomar decisiones sobre el nivel de las sentencias del texto, no resulta sencillo para el autor juzgar el nivel de complejidad del texto en función a un determinado tipo de lector, siendo ésta la principal motivación para el desarrollo de dicha herramienta.

Esta herramienta de evaluación de lectura detecta automáticamente el nivel de complejidad de un texto en cualquier momento del proceso de creación y además guía al autor a producir una simplificación léxica y sintáctica en un nivel adecuado de acuerdo con el tipo de lector. Simplifica nos permite clasificar un texto en uno de los siguientes tres niveles: rudimentario, básico o avanzado.

Es posible que algunas sugerencias de simplificación automática sean inadecuadas. En esos casos el autor puede elegir no usarlas y editar manualmente el texto original. A través de la función de legibilidad de esta herramienta el autor puede obtener información sobre el nivel del texto.

Los textos simplificados son generalmente más largos que los originales, debido a la distribución de las sentencias y a la repetición de las mismas para conectarlas entre sí. En esta herramienta la reducción del texto es una responsabilidad del autor y se centra más que nada en la estructura lingüística.

Funcionamiento Simplifica:

La interfaz (ilustración 2) es un simple editor de texto con herramientas que permiten dar formato al mismo como cortar, copiar, pegar e insertar datos entre otras.

No existe ningún orden para aplicar la simplificación léxica y sintáctica, los botones que permiten estas dos simplificaciones se encuentran por encima del área de texto de la interfaz principal, así como el botón que analiza el nivel del texto se encuentra en la parte inferior del editor (marcados en rojo).

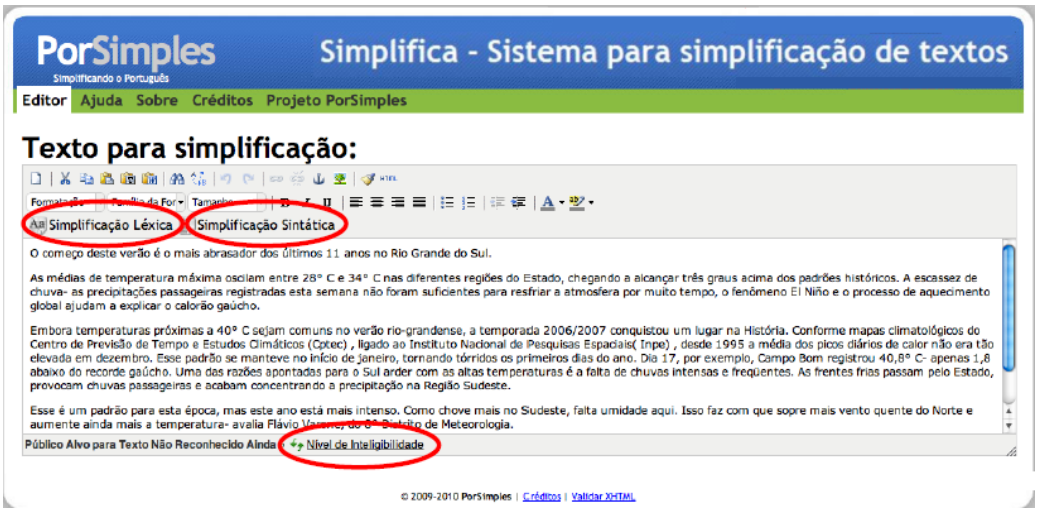

ILUSTRACIÓN 2: Interfaz de Simplifica

Para la **comprobación del nivel** de lectura del texto se dispone de un botón situado por debajo del área de texto, además se puede observar un área reservada para mostrar la respuesta de dicho cálculo (nivel avanzado, nivel básico y nivel rudimentario). El enlace "Mais Informações" en el círculo rojo muestra una breve explicación del significado del resultado del nivel de lectura, tal como se muestra en la ilustración 3.

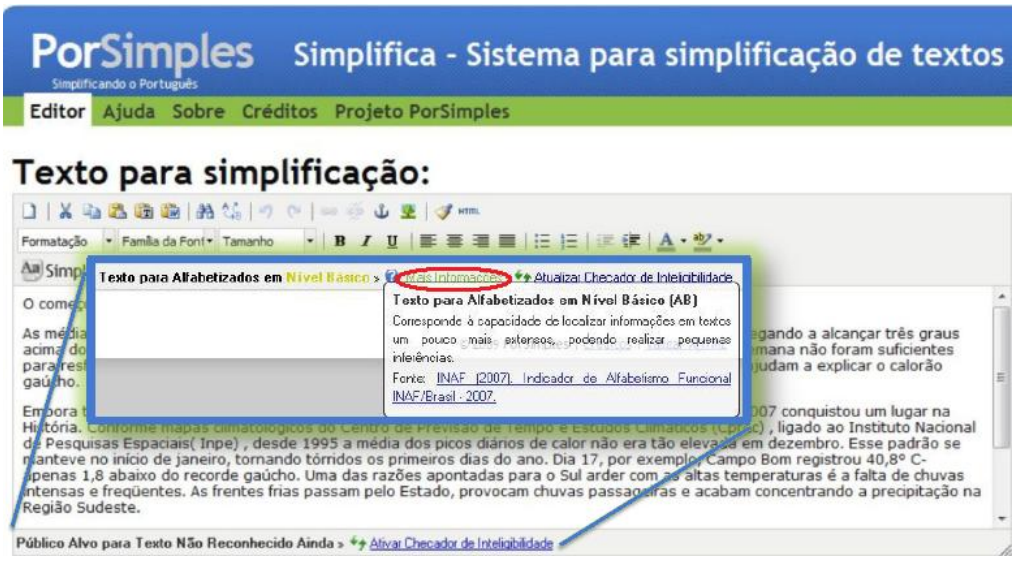

ILUSTRACIÓN 3: Botones para detección del nivel de lectura del texto

La **simplificación léxica** busca en el texto original las palabras complejas y las destaca. En una primera fase se le ofrece al usuario la opción de configurar dos diccionarios para buscar sinónimos como se muestra en la ilustración 4.

ILUSTRACIÓN 4: Elección del diccionario

Una vez realizado el proceso de análisis léxico (segunda fase), se marcan en el texto las palabras que fueron consideradas complejas (ilustración 5) para las que la herramienta busque sinónimos en los diccionarios seleccionados.

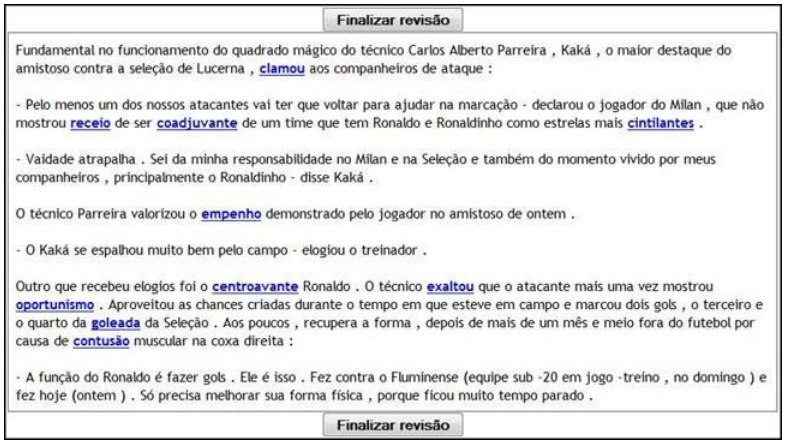

ILUSTRACIÓN 5: Marcado de palabras complejas

A continuación se puede seleccionar cualquiera de las palabras resaltadas y la herramienta nos ofrece sinónimos para la elegida como se muestra en la ilustración 6.

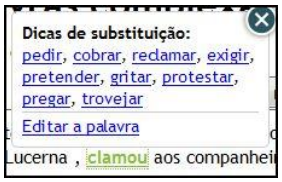

ILUSTRACIÓN 6: Sinónimos para la palabra elegida

Existen tres posibles casos de sustitución que nos permiten elegir, validar o editar la palabra seleccionada.

- Puede ocurrir que cuando seleccionamos el sinónimo, nos demos cuenta de que el resultado no es el esperado por cuestiones gramaticales. En este caso el usuario deberá editar la palabra para corregirlo. Para ello se hará clic sobre la palabra y luego sobre la opción “editar palavra” que nos abre una nueva ventana de edición en donde se puede escribir la palabra correcta tal y como se observa en la ilustración 7.

ILUSTRACIÓN 7: Modificación del sinónimo elegido para que exista concordancia gramatical

- Podría ocurrir que el sinónimo seleccionado encaje perfectamente en el texto, para ello el usuario sólo tiene que seleccionarlo y automáticamente se reemplazará como se ve en la ilustración 8.

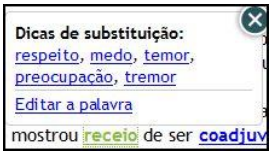

ILUSTRACIÓN 8: Selección y sustitución del sinónimo elegido

- Finalmente podría ocurrir que no se encuentre el sinónimo, para lo cual la herramienta muestra un mensaje como el de la ilustración 9, y es en estos casos cuando se le pide al usuario que edite la palabra manualmente.

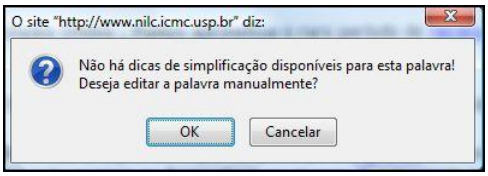

ILUSTRACIÓN 9: Introducción del nuevo sinónimo al no ofrecer opciones la herramienta

Este proceso se repite para el resto de palabras resaltadas y una vez que el usuario termine la simplificación léxica debe hacer clic en el botón “Finalizar revisão” (finalizar revisión) como se muestra en la ilustración 10 y volveremos a la interfaz principal.

ILUSTRACIÓN 10: Finalizar revisión

La **simplificación sintáctica** consiste en aplicar un conjunto de operaciones a las oraciones del texto original de acuerdo con la presencia de determinados fenómenos sintácticos. Los fenómenos complejos considerados son la yuxtaposición, subordinación, oraciones relativas, coordinación, voz pasiva, las oraciones en cualquier otro orden de sujeto y verbo y las frases adverbiales. Las operaciones de simplificación para hacer frente a estos fenómenos son

sentencia dividida, cambio de frases ambiguas por unas más simples, transformación de voz pasiva a activa u ordenar el sujeto y el verbo.

Es posible configurar el tipo de simplificación sintáctica deseada mediante un menú como el mostrado en la ilustración 11.

1. Personalizada: pudiendo seleccionar los fenómenos sintácticos que se quieren tener en cuenta.
2. Fuerte: en la que todos los fenómenos sintácticos considerados por la herramienta serán simplificados.
3. Natural: en la que el sistema de aprendizaje de la herramienta selecciona que sentencias deben simplificarse. Actualmente esta opción no está disponible.

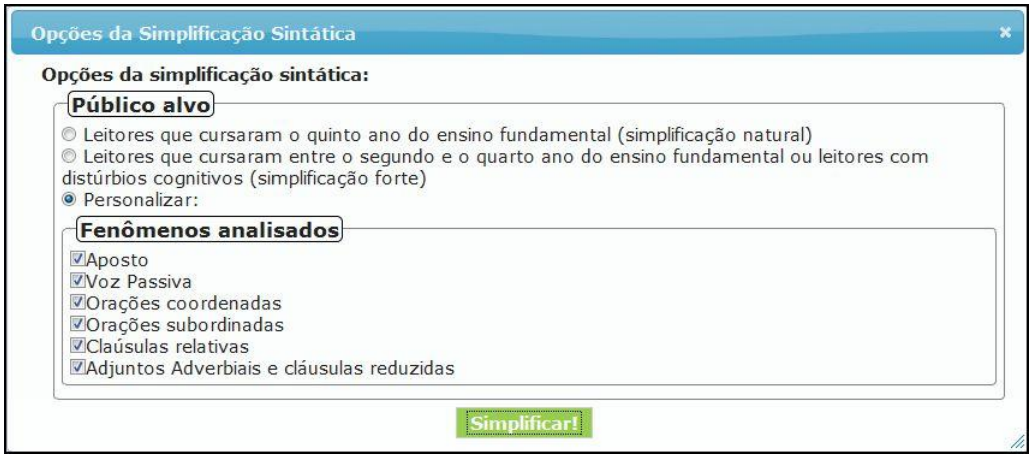

ILUSTRACIÓN 11: Tipos de simplificación y parámetros para la simplificación

Cuando se han terminado de hacer los ajustes para éste tipo de simplificación el usuario puede hacer clic en "Simplificar!" y la herramienta a continuación analizará todas las frases que pueden simplificarse de acuerdo con los parámetros especificados produciendo un resultado como el mostrado en la ilustración 12.

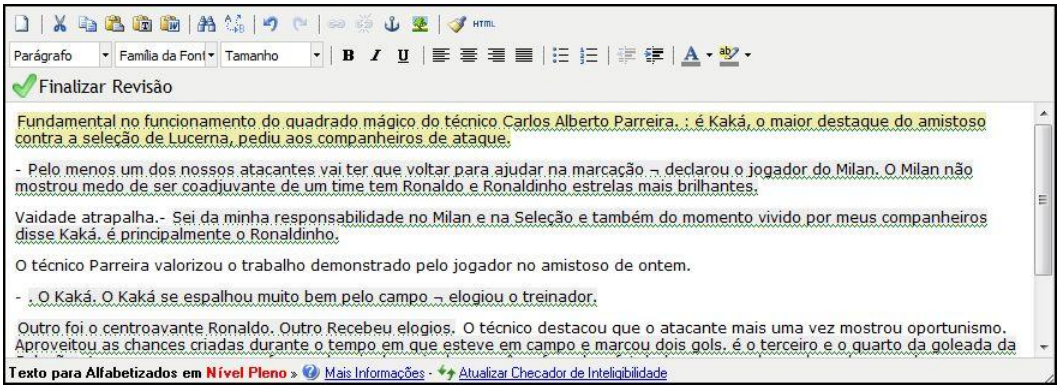

ILUSTRACIÓN 12: Resaltado de frases para las que existe simplificación

Para iniciar la simplificación de una frase el usuario debe seleccionarla y a continuación se abrirá una ventana en la que aparecen las versiones simplificadas de la frase como en la ilustración 13.

El usuario puede no estar de acuerdo con la simplificación propuesta y puede editar manualmente el texto siguiendo las sugerencias ofrecidas por la herramienta.

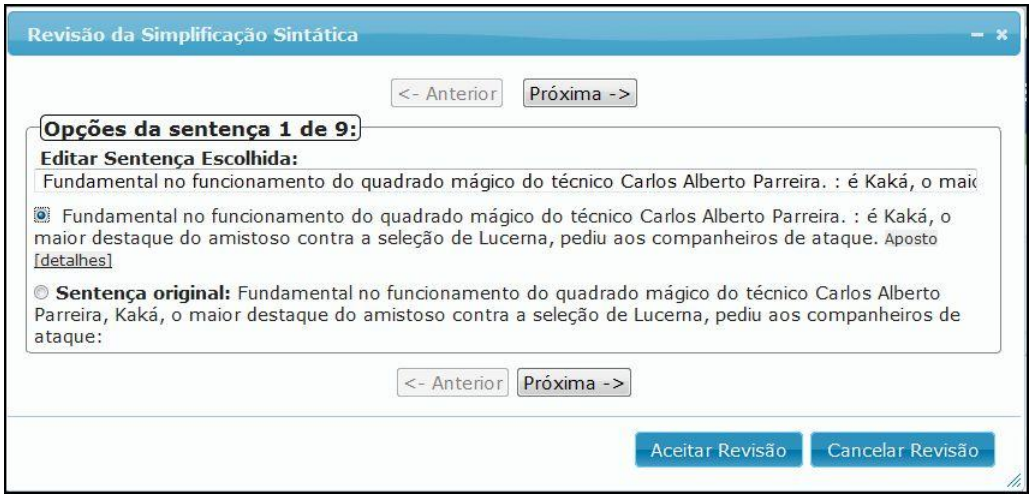

ILUSTRACIÓN 13: Elección de la nueva frase

Se pueden ver los detalles para la simplificación de la oración actual haciendo clic en "detalhes". Para desplazarse entre las distintas oraciones que requieren simplificación se deberán usar los botones "Próxima" y "Anterior".

Herramienta Educational FACILITA

Educational FACILITA (Educational Facilita s.f.) es una aplicación que permite a los usuarios acceder a un texto disponible en la web y simplificarlo para su comprensión. Es parte del proyecto PorSimples.

Esta aplicación adapta los contenidos de las páginas web, de manera que ayuda a los usuarios a comprender las palabras más inusuales que puedan encontrarse en ellas, ofreciendo sinónimos o definiciones cortas de aquellas palabras poco comunes o difíciles de entender.

Esta herramienta usa tres módulos distintos de procesamiento de texto:

1. Facilidad de lectura: mediante relaciones concretas entre palabras pertenecientes a una frase. Se selecciona dentro de la página el elemento que tiene la idea principal y mediante esta idea principal es más fácil identificar las frases.
2. Reconocimiento de entidades: es lo siguiente que se necesita para identificar y presentar a los usuarios información relativa a nombres de entidades en un texto, es importante saber que el método de reconocimiento de entidades identifica el tipo de las entidades mencionadas en el texto. Haciendo esto se puede extender en un futuro el prototipo Educational Facilita de manera que busque referencias acerca de la entidad en cuestión en otras fuentes disponibles de la web.
3. Elaboración léxica: este último módulo se encarga de evaluar la complejidad de cada palabra y de acuerdo con esta evaluación se mostrará al usuario los sinónimos correspondientes de manera que sea más fácil de comprender.

Primero se aplica el módulo de lectura en la página web de forma que se pueda identificar el texto principal, luego se aplica el módulo de reconocimiento de entidades y finalmente el módulo de elaboración léxica.

Funcionamiento Educational Facilita:

La instalación de la herramienta dependerá del tipo de navegador que el usuario utilice. Una vez realizada la instalación, se debe acceder a una página web cuyo contenido en portugués esté disponible para la fácil lectura. Con el ratón debemos seleccionar el contenido del texto que se desea simplificar, a continuación hacemos clic en el enlace Facilita agregado al explorador durante la instalación. La simplificación del texto necesita unos segundos para completarse dependiendo del tamaño del mismo. Finalmente el contenido de la simplificación se presentará como se puede ver en la ilustración 14.

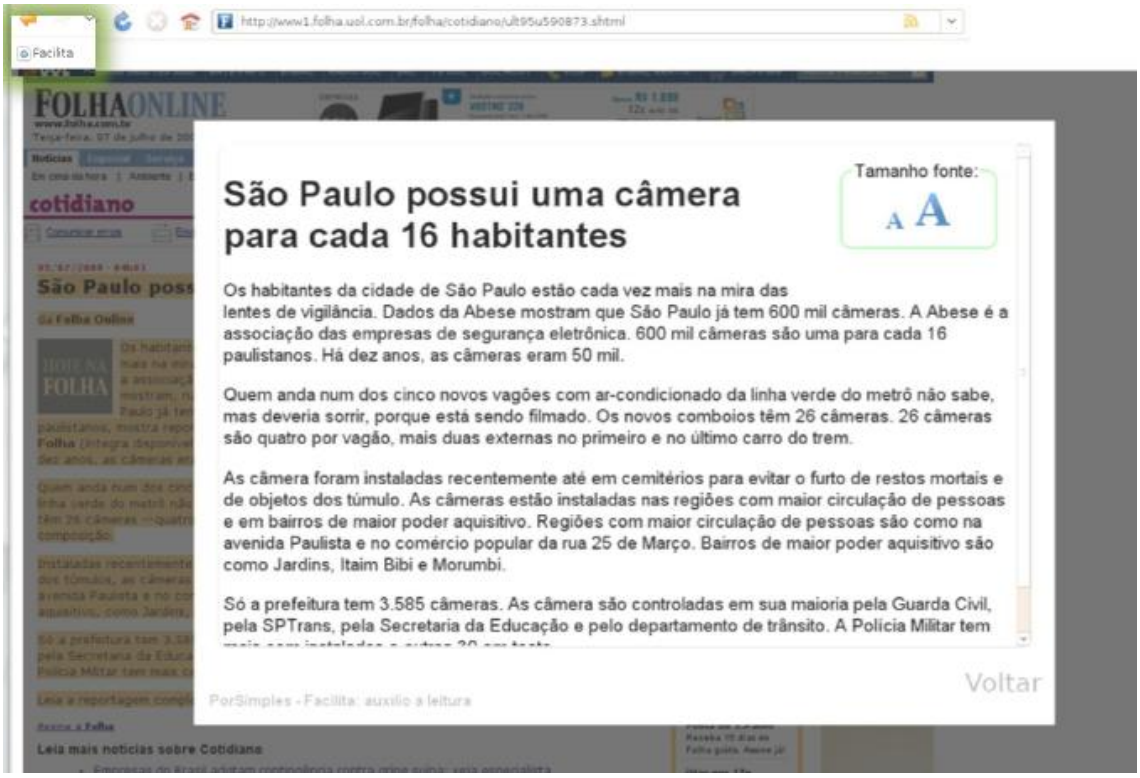

ILUSTRACIÓN 14: Aspecto final de la simplificación

2.1.5 Conclusiones

Dado que el 30% de la población tiene disminuidas sus capacidades para la compresión lectora, las directrices de la IFLA son muy útiles para redactar textos que contengan sólo la información más importante, expresada y presentada de la forma más directa, de forma que puedan ser comprendidos por el mayor número de personas posible. Siguiendo sus características generales, se puede llegar a un patrón común de simplificación y facilitar el trabajo a las múltiples asociaciones que se ocupan de este asunto.

Nuestro proyecto, al igual que el proyecto PorSimples, el más avanzado en esta materia, también ayuda a facilitar este trabajo, en concreto, la simplificación léxica y la simplificación sintáctica. La principal diferencia está en que el proyecto PorSimples está desarrollado en

portugués mientras que el nuestro está desarrollado en inglés, siendo éste un idioma más extendido.

2.2 Herramientas de Procesamiento del Lenguaje Natural

Existen diversas herramientas de Procesamiento de Lenguaje Natural (PLN) que son útiles en el proceso de simplificación. A continuación se describen brevemente cuales de ellas incluiremos en nuestro proyecto.

2.2.1 OpenNLP

OpenNLP (OpenNLP Community s.f.) es un conjunto de herramientas basado en técnicas de aprendizaje automático para el Procesamiento de Lenguaje Natural (*Natural Language Processing*). Da soporte a las tareas de PLN más comunes como son la detección de frases y la tokenización, que consiste en la división de una frase en palabras. También da soporte al marcado de part-of-speech (POS) que devuelve el etiquetado gramatical de cada una de las palabras obtenidas durante la tokenización. Esta herramienta está desarrollada para diferentes lenguas, y, aunque vamos a usar la versión 1.5.1 que es la más avanzada para el idioma inglés, también existe para el castellano.

Su licencia es Apache 2.0, lo que nos permite usar, copiar, modificar y distribuir este código. Se puede consultar en el siguiente enlace: http://www.apache.org/licenses/LICENSE-2.0.

Esta herramienta resulta adecuada para poder analizar un texto, obtener cada una de sus frases, dividirlas en tokens y obtener su etiqueta gramatical. En el anexo A, se muestra un ejemplo de las diferentes funcionalidades de OpenNLP que usaremos en el proyecto.

2.2.2 WordNet

WordNet (Miller 1995) es una gran base datos léxica en inglés. Sustantivos, verbos, adjetivos y adverbios aparecen agrupados en conjuntos de sinónimos que se denominan *synsets*. Cada *synset* expresa un significado distinto de la palabra buscada. De esta forma, distintos significados de una misma palabra aparecerán en distintos *synsets*.

WordNet es una red de palabras y conceptos relacionados por su significado por la que podemos navegar y así jugar con los posibles sinónimos de una palabra.

La licencia de WordNet también nos permite usar, copiar, modificar y distribuir el código. Se puede consultar en el siguiente enlace: http://wordnet.princeton.edu/wordnet/license/

Existen varias interfaces para acceder a WordNet, pero todas tienen el mismo problema: el usuario tiene que instalar en su ordenador WordNet para que funcionen. Por ello, usaremos LightWeight WordNet (Seco 2005), una interfaz para WordNet que incluye el diccionario completo en su distribución.

2.2.3 MorphAdorner

MorphAdorner (Northwestern University 2007) es una aplicación Java basada en línea de comandos creada por la Northwestern University y dividida en aplicaciones más pequeñas e independientes que pueden usarse de manera conjunta.

La licencia de MorphAdorner es NCSA, una licencia de la Universidad de Illinois. Se trata de una licencia compatible con GNU, así que también nos permite usar, copiar, modificar y distribuir el código.

La aplicación concreta que utilizamos de MorphAdorner es el pluralizador. Usamos esta aplicación de MorphAdorner para trabajar con las formas plurales y singulares de algunas palabras que necesitamos buscar en el diccionario, ya que WordNet no trabaja con plurales. De esta forma, dada una palabra en plural obtenemos su singular, la buscamos en WordNet para obtener sus sinónimos y de nuevo hacemos la transformación para obtener el plural de cada uno de los sinónimos.

Esto permite facilitar más al usuario su trabajo de simplificación, ya que evitamos que el usuario tenga que hacer manualmente la concordancia del texto. Esta funcionalidad no se encuentra por ejemplo en el proyecto PorSimples, que lo que permite para evitar los problemas de concordancia es la edición (manual) de las palabras sustitutas en el texto.

2.3 Editores de texto

Actualmente muchos de los usuarios hacia los cuales va dirigida nuestra aplicación hacen uso de editores de texto para llevar a cabo el proceso de simplificación. Es por eso que para poder facilitar este proceso y que nuestra herramienta pueda ser usada de forma sencilla hemos decidido integrarla en un editor de texto existente. A continuación, se mencionan todos los editores de texto que hemos tenido en cuenta en el momento de realizar nuestra elección.

2.3.1 AbiWord

AbiWord (AbiSource Community 2002) es el procesador de textos oficial del entorno gráfico GNOME, formando parte de la suite ofimática GNOME Office. Incluye soporte multiplataforma para Windows, Linux, QNX, FreeBSD y Solaris. Ha sido diseñado para integrarse con el sistema operativo sobre el cual se instale, ya que hace uso de ciertas funcionalidades del mismo tales como la carga de imágenes y la impresión.

La interfaz de AbiWord para Windows se muestra en la ilustración 15.

Trabaja bajo la licencia GLP (GNU General Public License) y está orientada a proteger la libre distribución, modificación y uso de software.

Un aspecto a destacar son los distintos lenguajes de programación usados en su desarrollo, siendo el lenguaje más usado C++, seguido de ANSI C, Shell y en una menor proporción lenguaje ensamblador, Perl, Yacc, Pascal, Sed y Awk.

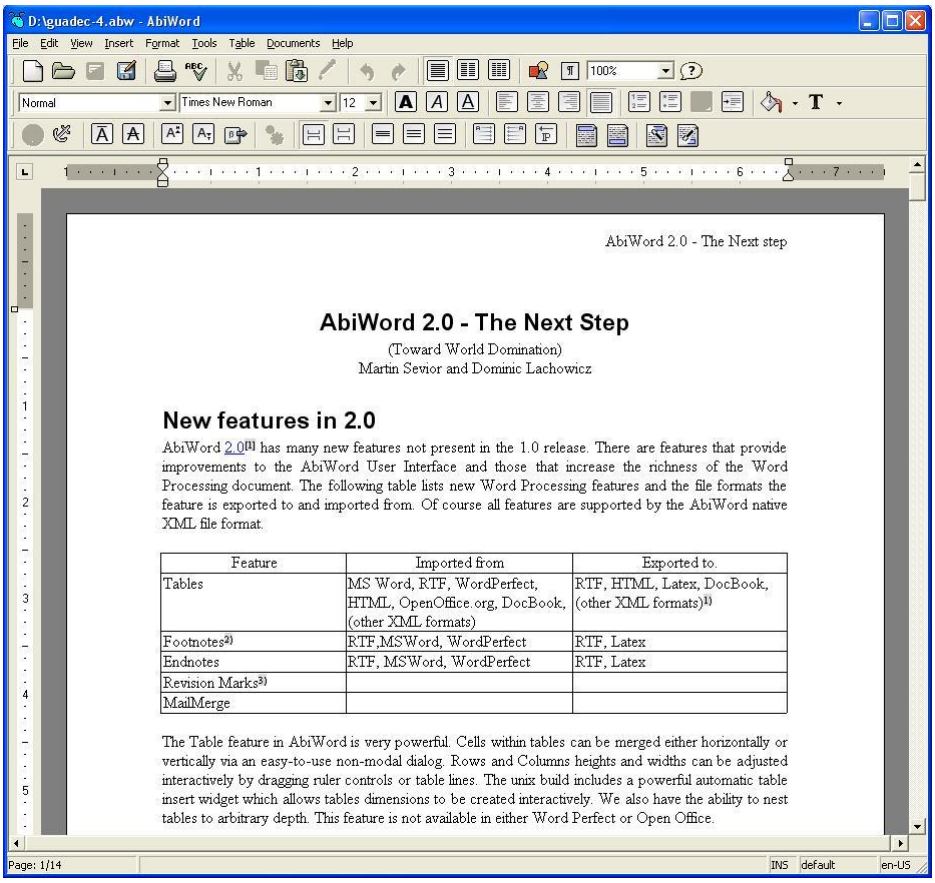

ILUSTRACIÓN 15: Interfaz de AbiWord

Es capaz de leer y escribir todo tipo de documentos estándar tales como XML, RTF, Microsoft Word, WordPerfect, LaTex, páginas web en HTML, OpenDocument y muchos más.

Se pueden encontrar las utilidades más básicas para el procesamiento y edición de un texto tales como las funciones de copiar y pegar, la función de resaltar una palabra así como también herramientas de formato común. También es posible realizar la comprobación ortográfica en varios idiomas. La última versión se integra con el servicio web AbiCollab que permite compartir documentos.

Para facilitar el uso de AbiWord en un entorno de servidor, se dispone de una interfaz de línea de comandos que permite generar formularios de cartas, documentos impresos o convertir documentos a cualquier formato de archivo.

Se puede extender AbiWord de distintas maneras. Para ello existe AbiWord Plugin Matrix pero hay que decir que estos plugins no están disponibles para todas las plataformas que soportan AbiWord.

Una gran variedad de plugins han sido diseñados para dar a este procesador toda la funcionalidad que se podría esperar. Dentro de los más importantes podríamos destacar los plugins para importar imágenes, para resumir textos y para buscar la referencia sobre cualquier palabra en la wikipedia.

2.3.2 KWord

KWord (KOffice Community s.f.) es el procesador de textos de la suite ofimática KOffice. Incluye soporte multiplataforma diseñado solo para Unix en un principio, aunque desde la versión 2.0 se puede ejecutar en Mac OS X y Windows.

La interfaz de KWord se muestra en la ilustración 16.

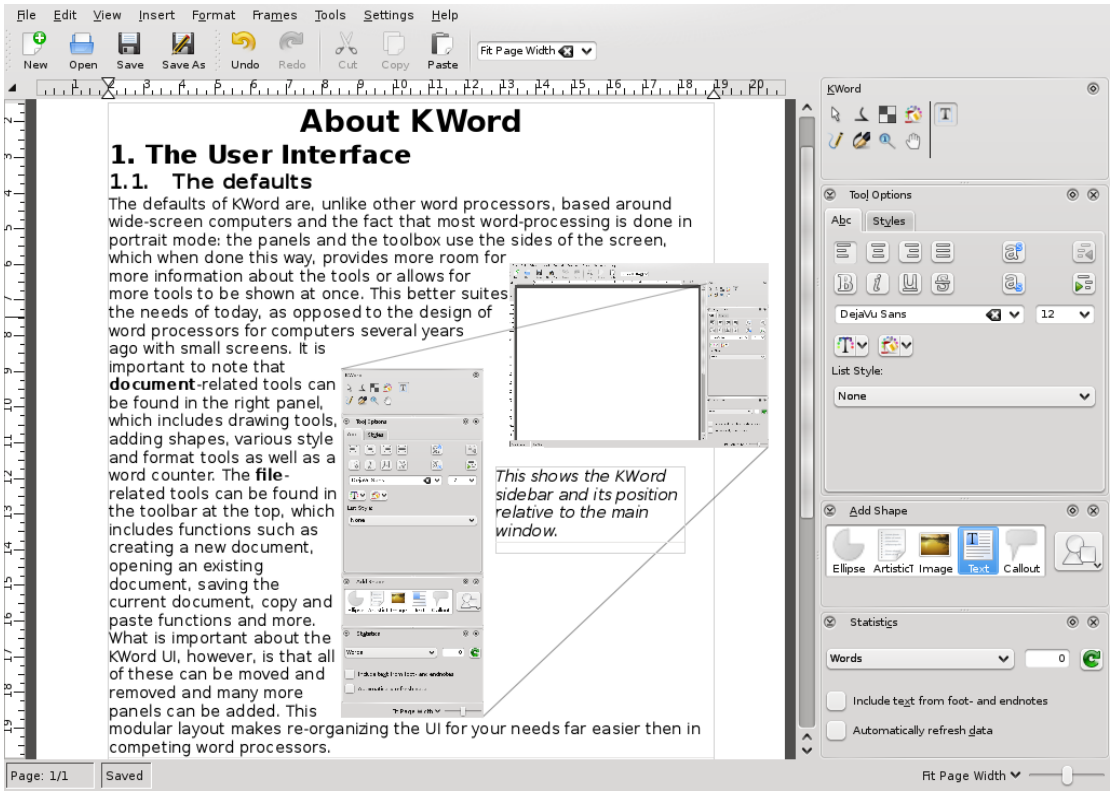

ILUSTRACIÓN 16: Interfaz de KWord

Está distribuido bajo la licencia GPL y LGPL y su código fuente está escrito en C++.

Los documentos usan la codificación Unicode, permitiendo escribir textos en lenguas que se escriben de derecha a izquierda (árabe o hebreo) pero todavía no permite las que escriben de arriba a abajo.

La disposición de los textos se basa en marcos. Éstos pueden contener texto, gráficos u objetos incrustados. Necesita pocos recursos pero no está optimizado para documentos de gran tamaño.

Este procesador puede trabajar como un procesador de textos tradicional o como una aplicación WYSIWYG.

Usa y soporta el formato ODF que es un formato de ficheros para aplicaciones de oficina que hace que la distribución de documentos sea más fácil e independiente del sistema operativo. Permite la creación de ficheros PDF sin tener que usar algún programa externo. Además permite importar y exportar documentos de otros procesadores debido a su gran lista de

filtros. Contiene soporte para poder trabajar con hipervínculos en internet y para direcciones e-mail.

Se ha desarrollado una plataforma para facilitar el desarrollo de plugins que permiten mejorar su funcionalidad.

2.3.3 LibreOffice Writer

LibreOffice Writer (The Document Foundation 2000) es un procesador de texto multiplataforma que forma parte de la suite ofimática OpenOffice.org (gratuita y de código abierto), y que funciona con los sistemas operativos Windows, Macintosh y GNU/Linux. Podemos crear desde una simple carta hasta un documento complejo con capítulos, anexos y referencias bibliográficas.

Esta suite surgió a raíz de que algunos miembros del proyecto OpenOffice formaron un nuevo grupo llamado The Document Foundation, que publicó una bifurcación del mismo al que denominaron LibreOffice. La primera versión de LibreOffice fue una réplica de OpenOffice, con lo cual se podría considerar que es una continuación del mismo.

La interfaz de LibreOffice Writer se puede ver en la ilustración 17.

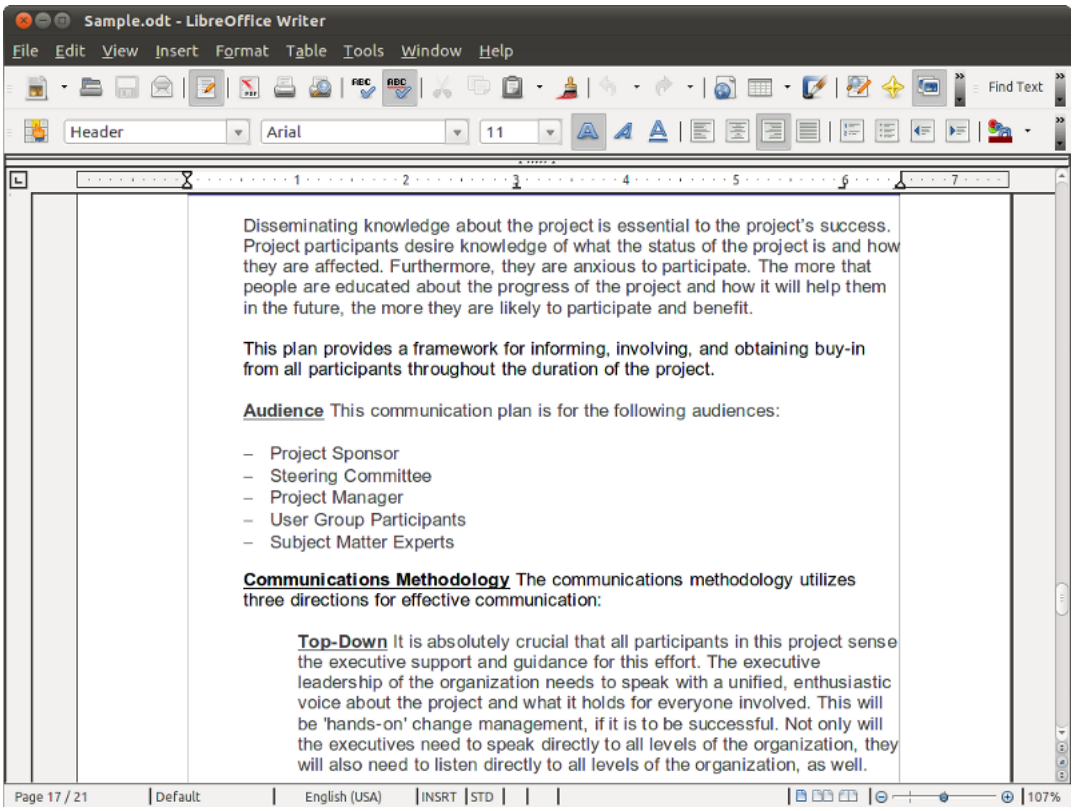

ILUSTRACIÓN 17: Interfaz de LibreOffice Writer

Trabaja bajo una licencia LGPL (GNU Lesser General Public License) y está escrito en C++ y Java.

LibreOffice Writer puede importar y exportar documentos desde varios formatos incluyendo DOC y DOCX, PDF, HTML, ODF y otros no tan usados como WordPerfect, Microsoft Works, Lotus Word Pro y las versiones anteriores de OpenOffice y StarOffice.

Puede ser usado como un editor WYSIWYG de forma que se pueden crear y editar páginas web.

Presenta funcionalidades de revisión ortográfica, diccionario de sinónimos, autocorrección y separador de sílabas. Tiene numerosas plantillas por defecto, y además nos permite crear nuestras propias plantillas mediante el asistente correspondiente. Contiene numerosas herramientas de maquetación y dibujo aptas para la creación de documentos profesionales. Permite la configuración de la interfaz del programa de acuerdo a las preferencias del usuario.

Las extensiones y plugins de OpenOffice también sirven para LibreOffice. Sin embargo LibreOffice dispone de un centro de extensión que permite la descarga de las mismas.

2.3.4 Microsoft Word

Microsoft Word (Microsoft Corporation s.f.) es un potente procesador de texto que se utiliza para la creación de distintos tipos de documentos. Fue creado por la empresa Microsoft y pertenece a la suite ofimática Microsoft Office.

La interfaz de Microsoft Word se muestra en la ilustración 18.

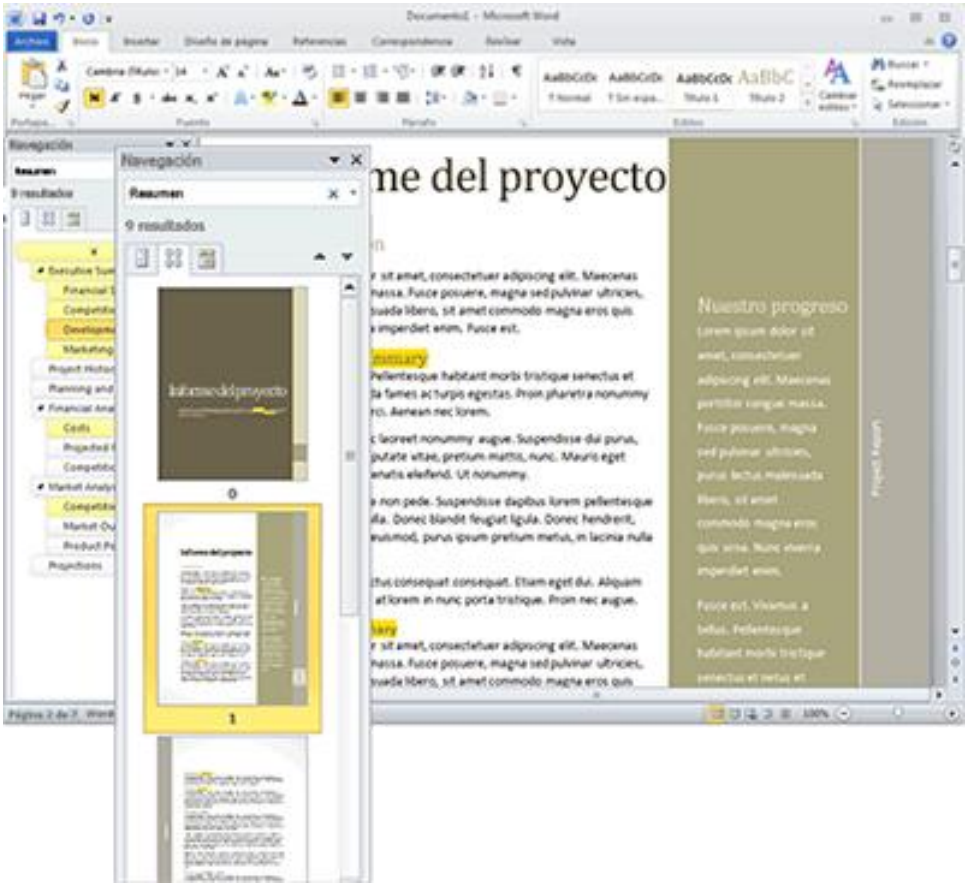

ILUSTRACIÓN 18: Interfaz de Microsoft Word

No es un software de código abierto ni gratuito. Funciona con los sistemas operativos Windows, Macintosh y para sistemas UNIX 5.1.

Usa un formato nativo cerrado y comúnmente denominado DOC que tiene la desventaja que los archivos con esta extensión poseen un gran tamaño en comparación con otros. En las nuevas versiones se usa el formato DOCX que comprime más el documento.

Se puede usar para crear textos que incluyan fotografías o imágenes y proporciona distintas características que ayudan en la creación y edición de un texto.

2.3.5 Notepad++

Notepad++ (Ho s.f.) es un editor de texto plano, incluyendo soporte multiplataforma para Microsoft Windows y en GNU/Linux mediante Wine.

La interfaz de Notepad++ se puede observar en la ilustración 19.

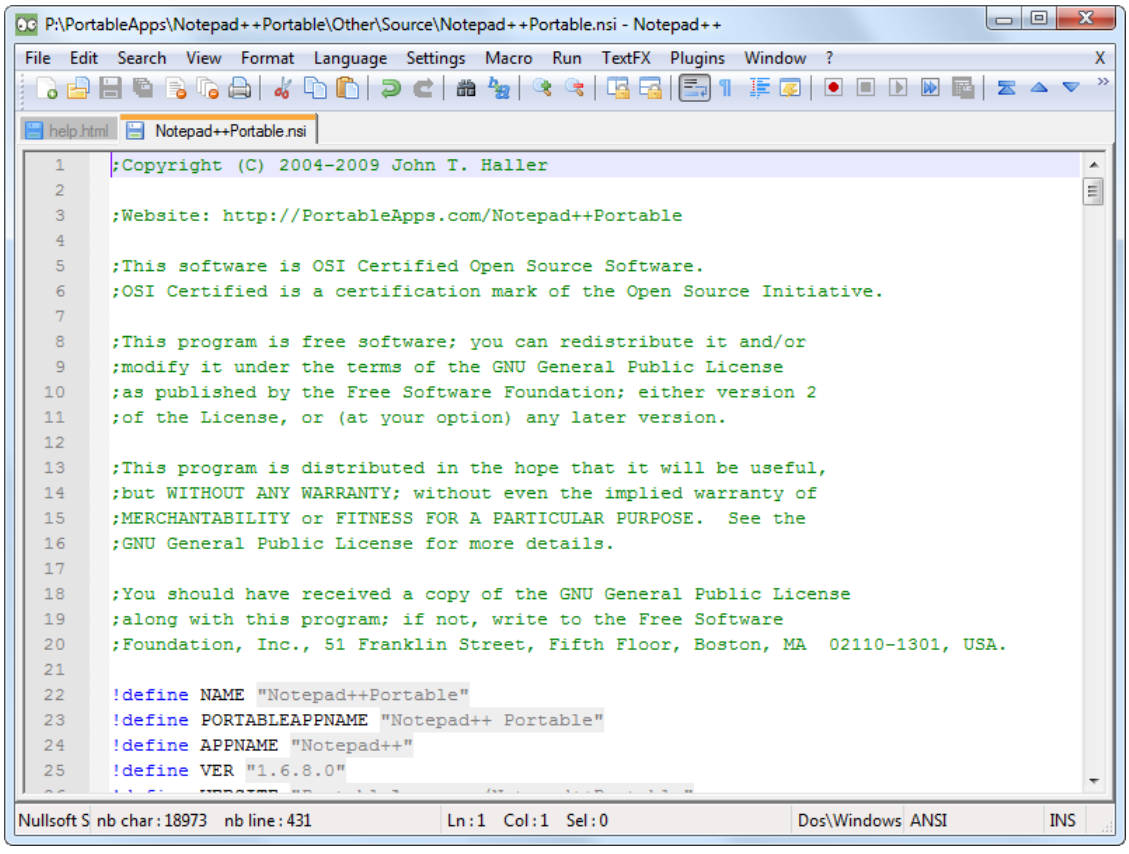

ILUSTRACIÓN 19: Interfaz de Notepad++

Se distribuye bajo la licencia GLP. Está escrito en C++ y basado en la componente de edición Scintilla (es un componente libre de edición de código fuente, escrito en C++). Es rápido en ejecución y no ocupa un gran tamaño debido a que utiliza directamente la API de Windows y STL.

Soporta varios lenguajes de programación (Ada, C, C#, C++, COBOL, Fortran, Haskell, HTML, Java, JavaScript, Lisp, Makefile, MATLAB, NSIS, Pascal, PHP, Python, Ruby, Shell, SQL, TeX, VHDL, XML, etc.). Si se escribe en algún lenguaje de programación marca las expresiones

propias de la sintaxis de dicho lenguaje y permite definir al usuario su propio lenguaje. Nos permite la opción de abrir varios documentos y organizarlos por pestañas.

Es rápido en ejecución y no ocupa un gran tamaño debido a que utiliza directamente la API de Windows y STL. Para aumentar su funcionalidad soporta la opción plugins, incluyendo una gran lista de plugins por defecto.

2.3.6 OpenOffice Writer

OpenOffice Writer (The Apache Software Foundation 2000) es un procesador de texto multiplataforma que forma parte de la suite ofimática OpenOffice.org y es una suite de distribución gratuita y de código abierto, que funciona con los sistemas operativos Microsoft Windows, GNU/Linux, BSD, Solaris y Mac OS X.

En la ilustración 20 podemos ver la interfaz de OpenOffice Writer.

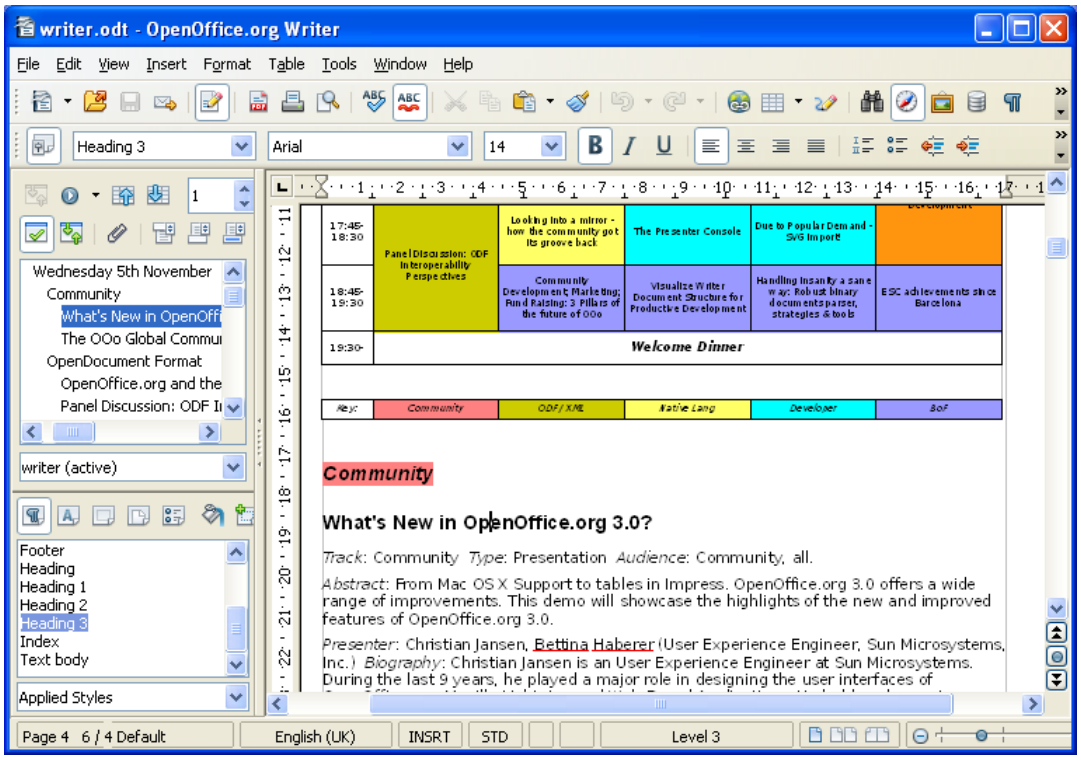

ILUSTRACIÓN 20: Interfaz de OpenOffice Writer

Trabaja bajo licencia LGPL (GNU Lesser General Public License). Esto significa que puede ser usado para propósitos domésticos, comerciales, administrativos y educativos.

Soporta numerosos formatos de archivo, como ODF (OpenDocument) y DOC (Microsoft Word), y también soporta más de 110 idiomas. Cabe destacar que puede usarse como editor HTML.

Permite la exportación a ficheros XML y PDF sin usar programas intermedios. También permite proteger documentos con contraseña, guardar versiones del mismo e insertar imágenes y objetos OLE que permiten a un editor encargar a otro la elaboración de parte de un documento y posteriormente volverlo a importar. Además admite firmas digitales, símbolos, fórmulas, tablas de cálculo, gráficos, hiperenlaces, marcadores, formularios, etc.

Su código fuente está escrito en C++ y ofrece una funcionalidad independiente del idioma y secuencias de comandos, incluyendo APIs de Java.

OpenOffice Writer permite el desarrollo de extensiones por parte de terceros, pudiendo desarrollarse desde librerías Basic, paquetes con marcos Java/JavaScript o Phyton, hasta extensiones más sofisticadas en la forma de componentes UNO (Universal Network Objects) implementadas en Java, C++ o Phyton.

2.3.7 jEdit

jEdit (Pestov s.f.) es un editor de texto plano cuyo autor es Slava Pestov. Incluye soporte multiplataforma para los sistemas que dispongan de máquina virtual de Java.

Su interfaz gráfica se puede ver en la ilustración 21.

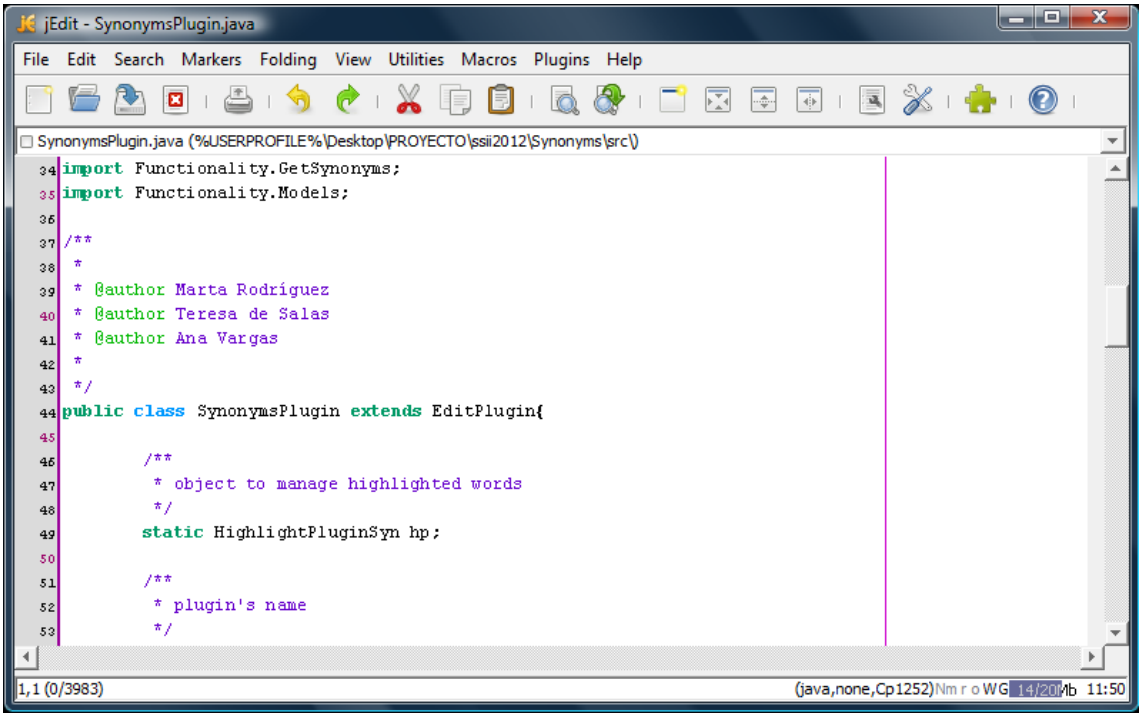

ILUSTRACIÓN 21: Interfaz de jEdit

Está distribuido bajo la licencia pública general GNU y su código fuente está escrito en Java. Los requisitos necesarios para usarlo son tener instalado Java 1.1 con Swing o Java 2.

Es altamente configurable y personalizable. Dispone de muchos plugins para personalizar la aplicación, y en la mayoría de los casos no necesitan ser descargados e instalados manualmente, sino que se puede hacer directamente mediante el administrador de complementos incluido en el editor. También se puede personalizar y extender con macros.

Es una aplicación monousuario que trabaja de manera local, aunque cuenta con algunos servicios para ser usados sobre una conexión en red como la descarga de plugins. Su arquitectura está formada por: BeanShell (sólo usado por las macros), las clases generales de

jEdit (para desarrollar macros y plugins) y las clases de EditBus (para el manejo del sistema de mensajes EditBus).

JEdit cuenta con atajos de teclado y ayuda en línea. Además, nos ofrece la posibilidad de deshacer o rehacer los cambios de forma ilimitada, así como también se puede tener almacenado un número ilimitado de clipboards. También podemos insertar marcadores para recordar posiciones en los archivos y poder volver a la parte en que lo dejamos.

Otra de las características más significativas es que podemos abrir varias ventanas del editor. Cada ventana se puede dividir en áreas y en cada una de éstas ver un archivo diferente. Tiene selección rectangular y selección múltiple para manipular varias partes del texto a la vez.

jEdit también nos permite hacer una búsqueda en el documento hacia atrás y hacia delante pudiendo reemplazar el texto buscado por el texto que queramos. Además incluye resaltado de sintaxis para más de 130 lenguajes de programación y soporta gran variedad de codificaciones incluyendo UTF8 y Unicode.

Para ampliar su funcionalidad jEdit incluye una gran variedad de plugins, que se pueden descargar desde la página oficial o mediante una opción del menú. Además existe un proyecto en SourceForge llamado *jEdit Plugin Central* que permite a cualquier usuario crear y publicar sus propios plugins.

2.3.8 Conclusiones

Analizando todos los editores, hemos decidido usar jEdit para el desarrollo de nuestro proyecto. Señalando sobre todo que se trata de un proyecto open source, escrito en Java, lo que nos facilita la conexión con las herramientas de Procesamiento del Lenguaje Natural, es por lo que hemos escogido este editor para nuestro proyecto. Además cuenta con una gran comunidad de apoyo al desarrollo de plugins de cualquier tipo, facilitando así el inicio del trabajo con la implementación de jEdit.

Capítulo 3

jEdit

Capítulo 3 – jEdit

Como se ha comentado anteriormente, jEdit es un editor de texto plano. Incluye soporte multiplataforma diseñado para Windows, GNU/Linux, MAC OS X y otros sistemas operativos que dispongan de máquina virtual de Java. Es el editor que usaremos para el desarrollo de nuestro proyecto, ya que podemos ampliar su funcionalidad mediante el desarrollo de plugins.

Nuestro proyecto está desarrollado en forma de una serie de plugins para jEdit por lo que en este capítulo explicamos cómo instalarlos y qué se debe hacer para crearlos.

3.1 Instalación de un plugin

La instalación de un plugin en jEdit es muy sencilla. Simplemente hay que copiar en la carpeta *[directorio de instalación de jEdit]/jars* el plugin que queremos instalar. Estos plugins son archivos .jar que no hace falta descomprimir.

Una vez copiado el jar en la carpeta, podemos añadir y quitar el plugin desde el *Plugin Manager* de jEdit, tal como se muestra en la ilustración 22.

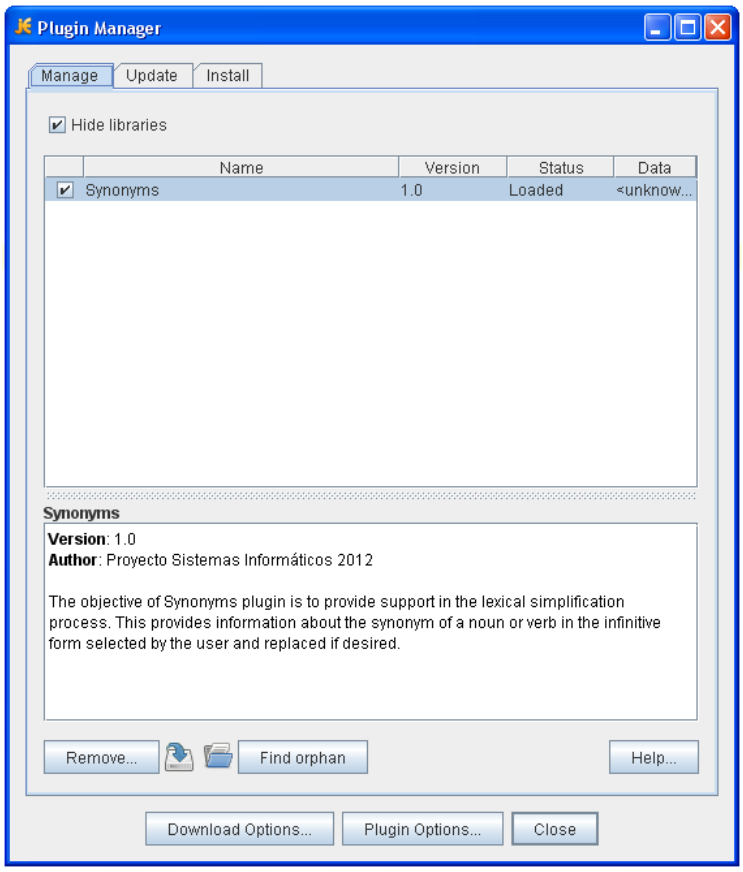

ILUSTRACIÓN 22: Plugin Manager

3.2 Carga de un plugin

jEdit crea un objeto de la clase PluginJAR por cada plugin encontrado. A continuación, carga las propiedades definidas en cualquier archivo que encuentre dentro de ese jar con la extensión *.props*, para después leer el archivo *actions.xml* en el que se definen las definiciones de las acciones que realizará el plugin (los métodos a los que llamará cada una de las opciones de los menús del plugin). Después se carga y se parsea el archivo *dockables.xml*, que contiene el código para crear las ventanas gráficas asociadas al plugin. A continuación se busca un archivo en el jar que acabe en *Plugin.class*. Esta es la clase principal del plugin y debe extender la clase *EditPlugin* de jEdit.

Finalmente, después de que todos los recursos y clases del jar se han cargado, se carga el plugin.

3.3 Creación de un plugin

3.3.1 La clase principal del plugin

En la clase principal del plugin es donde se llevan a cabo la mayoría de las características del plugin. Es necesario que el nombre de la clase principal acabe en “Plugin”, puesto que es este nombre el que se busca cuando se va a cargar el plugin en jEdit.

Tomemos como ejemplo el plugin Synonyms que se ha desarrollado para la gestión de sinónimos comentado más detalladamente en el capítulo 5. En la clase principal SynonymsPlugin se gestiona el panel de sinónimos, su creación y la modificación de palabras por el sinónimo elegido.

3.3.2 El archivo de propiedades

Este archivo se usa en la carga del plugin para almacenar sus propiedades: las propiedades propias del plugin como el nombre del plugin, los autores, la versión o la descripción del plugin, y las propiedades propias del usuario, como los nombres de los paneles visibles o las pestañas del menú.

Se trata de un archivo con una sintaxis muy simple. El archivo de propiedades del plugin Synonyms es el siguiente:

```
# The presence of this property also tells jEdit the plugin is using
the new API
plugin.SynonymsPlugin.activate=defer
plugin.SynonymsPlugin.depend.0=jdk 1.6
plugin.SynonymsPlugin.depend.1=jedit 04.05.99.00

# These two properties are required for all plugins
plugin.SynonymsPlugin.name=Synonyms
plugin.SynonymsPlugin.author=Proyecto Sistemas Informáticos 2012

# Version number
```

```
plugin.SynonymsPlugin.version=1.0

# Online help
plugin.SynonymsPlugin.docs=index.html

plugin.SynonymsPlugin.description=The objective of Synonyms plugin
is to provide support in the lexical simplification process. This
provides information about the synonym of a noun or verb in the
infinitive form selected by the user and replaced if desired.

# Plugin option pane
plugin.SynonymsPlugin.option-pane=synonyms
plugin.SynonymsPlugin.defaultColor=#FF8C00
plugin.SynonymsPlugin.defaultColor.text=Choose the color for
selected word:

# Option pane activation BeanShell snippet
options.synonyms.code=new SynonymsOptionPane();

# Option pane labels
options.synonyms.label=Synonyms

# Menu item label
Synonyms.menu.label=Synonyms
plugin.SynonymsPlugin.menu = Synonyms.synonyms

Synonyms.synonyms.label = Get synonyms

# Synonym window
Synonyms.title=Synonyms
Synonyms.caption=Synonyms
```

La opción *activate* de las propiedades del plugin es necesaria para que el plugin se pueda cargar dinámicamente. La opción *depend* sirve para indicar las versiones requeridas de otros programas que necesite el plugin. Las demás propiedades del plugin son las comentadas antes: el nombre del plugin, los autores, la versión, la ayuda del plugin y una breve descripción de la funcionalidad.

También se gestiona en este archivo el panel de las opciones del plugin. Este panel lo tienen todos los plugins de jEdit y se puede acceder desde *Plugins < Plugins Options*.

Además también se da nombre en este archivo a la pestaña correspondiente al plugin dentro de la opción de jEdit *Plugins < Synonyms* y la pestaña interna de éste, *Plugins < Synonyms < Get Synonyms*.

Por último, se pone la etiqueta de la ventana que gestiona los sinónimos.

3.3.3 El archivo de acciones

En este archivo se definen los procedimientos necesarios para activar cada una de las opciones del menú, de un botón de la barra de herramientas o un atajo de teclado.

Estas acciones pueden ser secuencias de comandos cortos, o bien delegar en algún método de alguna de las clases del plugin, de forma que se encapsule la acción.

En el plugin Synonyms sólo existe un menú y su acción se define en el método *haveSynonyms(view)* de la clase SynonymsPlugin. A continuación, se ve el código del fichero de acciones del plugin Synonyms.

```
<?xml version="1.0"?>
<!DOCTYPE ACTIONS SYSTEM "actions.dtd">
<ACTIONS>
   <ACTION NAME="Synonyms.synonyms">
      <CODE>
     SynonymsPlugin plugin=jEdit.getPlugin("SynonymsPlugin");
     try {
        SynonymsPlugin.haveSynonyms(jEdit.getActiveView());
     } catch (NullPointerException e1) {
        JOptionPane.showMessageDialog(null, "You need to add
        the folder \"wn_index\" to the jEdit's jars folder. \n
        You can find it on Synonyms/lib/wn_index");
     } catch (Exception e2){
        e2.printStackTrace();
     }
      </CODE>
   </ACTION>
</ACTIONS>
```

Las acciones se definen mediante la creación de un archivo xml. El nombre de este archivo debe ser necesariamente actions.xml, puesto que es el archivo que se busca en la carga del plugin.

3.3.4 El archivo dockables.xml

Este archivo se utiliza para crear la interfaz visible de un plugin. Se trata de un archivo xml y normalmente muy corto. En él, sólo se suele crear la parte más grande de la interfaz del plugin y no los paneles más pequeños o mensajes al usuario.

En los plugins que hemos creado no usamos este archivo, sino que creamos la interfaz desde la API de Java.

3.3.5 El archivo services.xml

Este archivo sirve para añadir opciones al menú contextual de jEdit que aparece cuando se hace clic derecho. Se trata de un archivo con formato xml que crea un objeto de una clase que extiende la clase *DynamicContextMenuService* propia de jEdit y que debe implementarse en el código del plugin.

```
<?xml version="1.0"?>
<!DOCTYPE SERVICES SYSTEM "services.dtd">
<SERVICES>
```

```
    <SERVICE
    CLASS="org.gjt.sp.jedit.gui.DynamicContextMenuService"
    NAME="SynonymsPlugin">
         new ContextMenuService();
    </SERVICE>
  </SERVICES>
```

3.4 Compilación de un plugin

Los plugins de código libre incluyen entre sus ficheros fuentes un archivo makefile de tipo xml para Ant. Ant es una librería de Java y una aplicación de línea de comandos para compilar proyectos Java. Se usa para compilar el código junto con el código de jEdit y todos los archivos anteriormente descritos para poder obtener el archivo Java (jar) necesario para ejecutarse como plugin.

Build.xml

Lo primero que hay que hacer para crear un archivo de este tipo es definir sus tres propiedades básicas:

```
<project name="Synonyms" default="remove" basedir=".">
```

En *name* se debe poner el nombre por defecto; en *default* el objetivo que se debe ejecutar siempre y que tiene que estar definido más adelante; y en *basedir* la ruta base a la que se enlazarán todas las rutas que necesite este archivo.

A continuación se definen etiquetas (name) para poder llamar de una forma abreviada a los valores (value) que representan:

```
    <property name = "jar.name" value = "synonyms.jar"/>
    <property name = "temporal.dir" location = "temporal"/>
    <property name = "jarFolder.dir" location = "jarFolder"/>
    <property name = "docs" location = "${temporal.dir}/docs"/>
```

La forma de usar estas etiquetas es `${name}` como se puede ver en los ejemplos que aparecen más abajo.

Después se definen los objetivos o *targets*, cuya finalidad es dividir todo el proceso de compilación en distintas etapas. En el caso del plugin Synonyms existen cuatro *targets*: newFolder, prepare, makeJar y remove. Cuando uno de los objetivos depende de otro, es necesario expresarlo mediante *depends*, como en el caso del objetivo makeJar, que depende del newFolder y del prepare.

La primera etapa es la etapa *newFolder*, en donde se crea una carpeta para almacenar temporalmente todos los archivos que se van a añadir al jar del plugin y una carpeta para almacenar el jar.

```
<target name = "newFolder">
  <mkdir dir = "${temporal.dir}"/>
  <mkdir dir = "${jarFolder.dir}"/>
```

```
</target>
```

La segunda etapa es *prepare*. En ella, se añaden a la carpeta temporal creada en el *target* anterior todos los archivos necesarios para compilar el jar. En la raíz de jar se copian todos los archivos necesarios de OpenNLP para poder ejecutar el proyecto, los archivos *.class* del código de nuestro plugin que se encuentran dentro de la carpeta *bin* y todos los archivos mencionados en el apartado anterior (el archivo de propiedades, el de acciones, etc.). Además se añaden las imágenes que se utilizan en el plugin, así como todas las librerías utilizadas en el proyecto descomprimidas. Es por esto que para las librerías se usa la opción *zipfileset*, para copiarlo en la carpeta temporal descomprimido, y no *fileset*, que simplemente copia ese archivo al temporal.

```
<target name = "prepare">
  <copy todir = "${temporal.dir}">
    <fileset dir = "OpenNlpFiles"/>
    <fileset dir = "bin"/>
    <fileset dir = "${basedir}">
      <include name = "*.xml"/>
      <include name = "*.props"/>
      <include name = "*.html"/>
    </fileset>
    <fileset dir = "${basedir}/images">
      <include name = "*.png"/>
    </fileset>
    <zipfileset includes="**/*" src="lib/opennlp-maxent-3.0.1-
      incubating.jar"/>
    <zipfileset includes="**/*" src="lib/opennlp-tools-1.5.1-
      incubating.jar"/>
    <zipfileset includes="**/*" src="lib/basicHighlighterSyn.jar"/>
    <zipfileset includes="**/*" src="lib/lucene-1.4.3.jar"/>
    <zipfileset includes="**/*" src="lib/lwwn.jar"/>
    <zipfileset includes="**/*" src="lib/jdom-1.1.2.jar"/>
    <zipfileset includes="**/*" src="lib/morphadorner.jar"/>
  </copy>
</target>
```

La tercera etapa *makeJar* es la etapa donde se añaden todos los archivos de la carpeta temporal al jar de nuestro plugin. Es necesario que se hayan realizado antes las dos etapas anteriores y por eso se pone explícitamente con *depends*.

```
<target name = "makeJar"
  depends = "newFolder,prepare"
  description = "make the project jar">
  <jar destfile = "${jarFolder.dir}/${jar.name}">
    <fileset dir = "${temporal.dir}">
      <include name = "**/*"/>
    </fileset>
  </jar>
</target>
```

La última etapa *remove*, que también es el *target* por defecto como se ha definido al principio, es la etapa en la que se elimina la carpeta temporal que se ha creado.

```
<target name = "remove"
   depends = "makeJar">
   <delete dir = "${temporal.dir}"/>
</target>
```

Creando un archivo con esta estructura, podemos generar, a través de Ant, el jar de nuestro plugin. Es necesario que el jar generado se copie en la carpeta *[directorio de instalación de jEdit]/jars/* para poder ejecutarlo.

Capítulo 4

Especificación y Arquitectura

Capítulo 4 – Especificación y arquitectura

Este proyecto de Sistemas Informáticos consta de un sistema de apoyo a la simplificación de textos en inglés que se ha decidido implementar de una forma flexible sobre un editor existente, jEdit, y más específicamente implementando una serie de plugins.

Para el desarrollo de estos plugins, hemos realizado un estudio previo de las simplificaciones más útiles a la hora de simplificar un texto para su fácil lectura.

4.1 Motivación: análisis de textos simplificados

Para llegar a saber qué tipo de simplificaciones son más útiles, hemos realizado un estudio con cuatro personas nativas de habla inglesa. Tres de ellas son americanas y una de nacionalidad inglesa, todas ellas entre los 20 y 30 años. Los textos completos, comparando el original con la versión simplificada, se pueden leer en el anexo B.

De ese estudio hemos sacado varias conclusiones, tanto a nivel léxico como a nivel sintáctico.

A nivel léxico, hemos observado que se sustituían muchas palabras o expresiones que los nativos han considerado complicadas por sinónimos o frases más sencillas. Por ejemplo, la oración "*who was spotted by Coast Guard officials"* se ha sustituido por "*who was seen by boat workers"*, o la oración *"the chaotic and terrifyin evacuation was underway"* se ha simplificado como *"during the crazy and scary rescue"*. También vemos como la expresión *"for suspected manslaughter"* se ha cambiado por *"for possibly killing people"*, o la palabra *"prosecutor"* se ha explicado y definido como *"the lawyer against him"*.

En la oración *"the captain abandoned the stricken liner before all the passengers had escaped"* se han cambiado varias de las palabras por sinónimos más sencillos resultando la frase *"the captain left the damaged ship before all the people on the ship had left"*. También la oración *"a captain who abandons a ship in danger can face up to 12 years in prison"* ha sufrido varias modificaciones de este tipo, resultando la simplificación *"a captain who leaves a ship in a problem can go to jail for as many as 12 years*".

En otro de los textos, también observamos este tipo de simplificación. Por ejemplo, se ha sustituido el verbo *"turned up"* por *"appeared"*, o *"got in touch"* por *"called him"*.

En el tercer texto analizado, también es común la simplificación léxica. Uno de los verbos que se sustituyen por otro más sencillo es *"have picked up"* por *"have averaged"*.

En el último de los textos analizados observamos también simplificaciones léxicas de sustantivos como "*board*" sustituido por "*ship*". También se pueden encontrar simplificaciones de verbos como "*emerged*" que se ha cambiado por el verbo "*went up*", o el verbo "*forced upon*" que se ha sustituido por "*came up*".

Gracias a este análisis, hemos comprobado que la simplificación léxica más utilizada es la sustitución de palabras por frases o expresiones equivalentes, y sobre todo por sinónimos. Es por esto que hemos desarrollado un plugin léxico que nos ofrece sinónimos para una palabra seleccionada.

A nivel sintáctico, el análisis resulta más complicado. Los nativos que realizaron las simplificaciones no tienden hacia una simplificación sintáctica que siga unos determinados patrones.

La mayoría de las simplificaciones que hemos encontrado de este tipo hacen una reestructuración del texto, intentando redactarlo de una manera más clara. Por ejemplo, en uno de los textos originales encontramos la oración: "*According to the Italian navigation code, a captain who abandons a ship in danger can face up to twelve years in prison."*. Esta oración ha sido sustituida por: "*The rules of the Italian boat workers says that a captain who leaves a ship in a problem can go to jail for as many as twelve years."*.

En estas oraciones, además de las simplificaciones léxicas anteriormente descritas, observamos como se ha sustituido *"according"* por *"says that"*, resultando una frase reestructurada mucho más fácil de entender.

Otra de las simplificaciones comunes es la omisión de datos irrelevantes, como adjetivos o complementos circunstanciales que no aporten datos importantes al texto. Por ejemplo el sujeto *"the ship's Italian owner"* se ha sustituido por *"the owner of the ship"*, eliminando el adjetivo *"Italian"*.

Finalmente, la simplificación de la oración *"An Indonesian girl swept away in 2004 tsunami has been reunited with her parents seven years on, the family say."* permite estructurar el texto de una manera más clara y coherente, dividiéndolo tres oraciones más simples utilizando una sola idea por frase, siendo el resultado de esta simplificación: *"In 2004 a tsunami swept away an Indonesian girl. She was separated from her family. After seven years her family says that she was reunited with them."*. Este tipo de simplificación también se ha encontrado en otro de los textos en la oración "*The little mermaid who was the youngest and have a wonderful voice spent her time thinking on the surface, the world of the men.*" que se ha modificado y cambiado por "*The little mermaid was the youngest and had a wonderful voice. She spent her time thinking about the world of the men.*".

Podemos observar que las simplificaciones sintácticas son más variadas que las léxicas y es por esto que la elección del plugin sintáctico ha sido más complicada.

Hemos considerado que la simplificación más útil es la última explicada, por lo que el plugin consiste en la separación de frases largas que contienen conjunciones coordinadas (como "*and*") en oraciones más cortas sin estas conjunciones, de forma que se exprese una sola idea por frase.

4.2 Especificación de los plugins implementados

A partir del análisis de textos simplificados que se ha realizado previamente, se observa la necesidad de desarrollo de dos herramientas, una para la simplificación léxica y otra para la simplificación sintáctica, que se desarrollarán como plugins de jEdit.

El plugin léxico sustituye palabras que el usuario considera difíciles por sinónimos más sencillos, para ello se mostrará un serie de sinónimos y se permitirá la edición de dicha palabra en caso de no estar satisfecho con ninguno de los sinónimos ofrecidos. Por otro lado el plugin sintáctico separa oraciones largas en oraciones más simples. La oración original debe contener la conjunción coordinada "*and*", de tal forma que al separarse en oraciones más cortas se exprese una sola idea por frase.

Ambos plugins contarán con interfaces similares y sencillas y podremos acceder a su funcionalidad desde el menú propio de jEdit *Plugins* o desde el menú contextual que aparece al hacer clic derecho sobre el área de edición de jEdit. Además ambos plugins contienen funcionalidades comunes y utilizarán herramientas existentes.

4.3 Arquitectura general

Los plugins especificados previamente tienen partes comunes ya que ambos comparten funcionalidades. La primera de ellas es la necesidad de registrar los cambios realizados y el orden en el que han sido aplicados de forma que se pueda automatizar en un futuro la simplificación. Otra de las funcionalidades es la de identificar visualmente las palabras clave en cada tipo de simplificación, para ello se resaltan de un determinado color en la interfaz de jEdit.

4.3.1 Log

El log guarda cada uno de los cambios que se han realizado sobre el archivo original. Si sobre un mismo archivo se han realizado simplificaciones con ambos plugins, todas estas simplificaciones se almacenan en el log, ya sean simplificaciones léxicas o sintácticas. Además en el log aparece reflejado el orden en que se han realizado las simplificaciones.

Cada archivo que se simplifica tiene un log asociado, cuyo nombre es el mismo que el del archivo con el que se está trabajando, pero con la extensión *xml*. Este log se crea siempre y cuando el archivo sobre el que se trabaja tenga nombre y no se renombre, y puede encontrarse en la misma ruta que la del archivo con el que se trabaja.

El formato del log se ha elegido pensando en la utilidad futura del mismo, de forma que se almacene toda la información necesaria para deshacer los cambios y poder llegar a generar el archivo original. Así, se puede llegar a analizar la usabilidad de cada plugin. Por ejemplo, si uno es más utilizado que otro significaría que ese tipo de simplificación es más utilizada por los expertos. El orden en que se han realizado los cambios también es importante para así poder

analizar la forma en la que un experto simplifica un texto y poder automatizar este proceso en un futuro.

La elección de este formato presenta muchas ventajas de cara al futuro y se puede tomar como base para llegar a obtener una aplicación que realice la simplificación automática de textos.

Para desarrollar el log se ha usado JDom. JDom (JDOM Project 2000) es una API desarrollada específicamente para Java que da soporte al tratamiento de ficheros XML, permitiendo parsearlos, modificarlos, generarlos y serializarlos. Al ser un modelo desarrollado específicamente para Java, es más eficiente y más fácil de usar para el desarrollador Java.

JDom se autoriza y distribuye bajo la licencia Apache, sin la cláusula de *acknowledgment*. Esta licencia es una de las licencias disponibles menos restrictiva, que permite a los desarrolladores usar JDom en la creación de nuevos proyectos sin la necesidad de que estos sean lanzados como productos de código abierto.

Usamos la versión 1.1.2 para la creación y el mantenimiento del log que tienen todos los plugins, incluyendo jdom-1.1.2.jar en el proyecto. La forma que tiene JDom para parsear un documento XML es mediante la generación de un árbol de nodos.

JDom se utiliza en cada uno de los plugins. En el apartado correspondiente de cada plugin se da más información del formato concreto. Con el fin de ilustrar el uso del log y de cómo puede ayudarnos en el análisis de las simplificaciones aplicadas, se presenta un ejemplo sencillo del mismo en el anexo C.

4.3.2 BasicHighlighter

Para poder indicar al usuario o que el usuario señale las posibles simplificaciones, hay que resaltar trozos del texto y para ello se ha usado una simplificación del plugin Highlight.

Highlight es un plugin creado por Matthieu Casanova para jEdit con licencia GNU. Este plugin resalta palabras o expresiones regulares en color.

En este plugin existen distintas formas de resaltar una palabra. La primera de ellas consiste en seleccionar una palabra y resaltar todas las apariciones de la misma. La segunda forma es mediante un panel que nos permite crear una expresión regular y elegir el color de resaltado entre otras opciones. Se puede resaltar la última palabra buscada y también tener más de una palabra resaltada al mismo tiempo. Es posible desactivar y volver a activar más tarde el resaltado de una palabra. Además se dispone de un panel que permite trabajar con las palabras resaltadas.

Nos basamos en este plugin para crear una versión simplificada puesto que sólo necesitamos resaltar una o varias palabras, pero no necesitamos el resto de opciones que tiene el plugin Highlight. Una característica de Highlight que omitimos en nuestro BasicHighlighter es que permitía guardar las palabras resaltadas permanentemente, y sólo nos interesa que lo haga en el texto actual y durante la sesión actual. El plugin original además de palabras completas

permitía resaltar expresiones regulares, funcionalidad que no nos interesaba puesto que la unidad mínima que vamos a usar para las simplificaciones es la palabra.

El código de este plugin se simplifica, eliminando las características no deseadas, y se añade a cada uno de los plugins que hemos desarrollado para resaltar las palabras o frases que se van a simplificar.

4.5 Diagrama de arquitectura

La arquitectura interna del proyecto es la que se muestra en la ilustración 23 en la que podemos observar que existe una parte común a los dos plugins que vamos a desarrollar. Esta parte común incluye el editor de texto jEdit y varias herramientas como Highlighter que utilizaremos para el marcado de las palabras en el texto, OpenNLP que es un conjunto de herramientas basado en técnicas de aprendizaje automático para el Procesamiento de Lenguaje Natural y la parte dedicada a la gestión de cambios. También podemos observar que ambos plugins trabajaran sobre el mismo log físico.

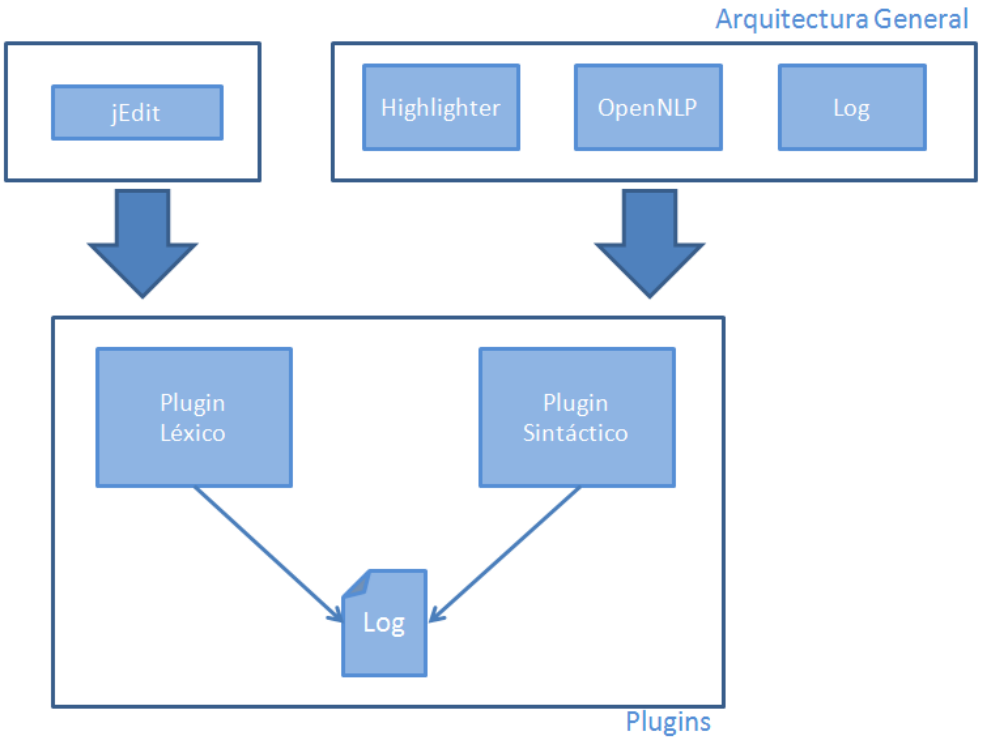

ILUSTRACIÓN 23: Diagrama de arquitectura general

Capítulo 5

Plugin Synonyms

Capítulo 5 – Plugin Synonyms

Synonyms es el primero de los plugins que se ha desarrollado. Su objetivo es proporcionar apoyo en el proceso de simplificación léxica en inglés ofreciendo sinónimos para una palabra que el usuario considere complicada.

5.1 ¿Qué hace? Funcionamiento

Synonyms permite consultar los sinónimos de un nombre o verbo en infinitivo seleccionado por el usuario y sustituirlo si se desea.

Cuando se quiere simplificar un texto, uno de los primeros pasos es buscar las palabras que se consideran más complicadas y sustituirlas por otras más sencillas.

Para trabajar con este plugin primero debe instalarse como se indica en el capítulo 3 y copiar una carpeta que se encuentra en el archivo comprimido del plugin, llamada wn_index, en la misma ruta donde se copia el archivo jar del plugin. En caso de no encontrar esta carpeta cuando se trabaja con Synonyms, aparecería un mensaje de error como el mostrado en la ilustración 24.

ILUSTRACIÓN 24: Error asociado a la carpeta wn_index

Para poder hacer uso del plugin debemos abrir con jEdit el archivo que queremos simplificar. Haciendo clic en *Plugins < Synonyms < Get Synonyms* (ilustración 25) podemos empezar a trabajar con el plugin.

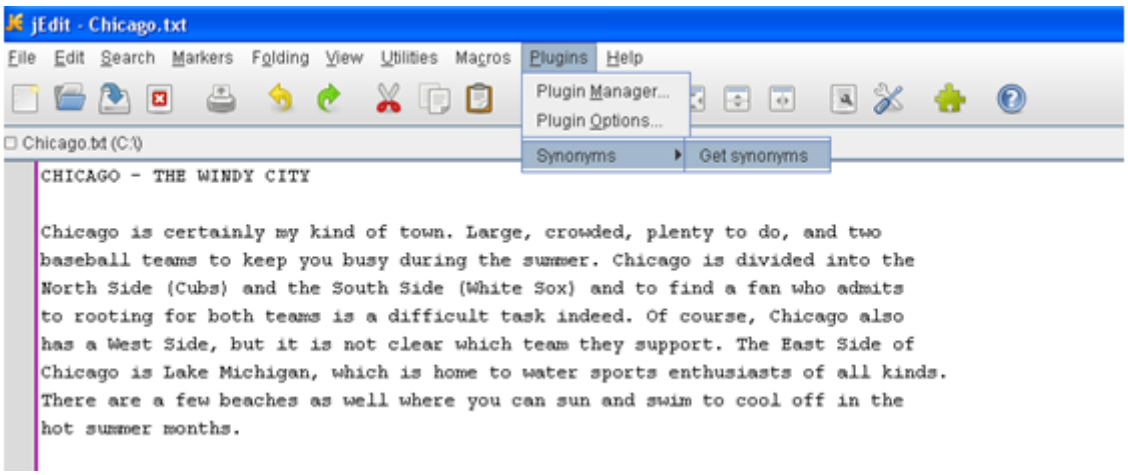

ILUSTRACIÓN 25: Menú del plugin Synonyms

También se puede empezar a trabajar con el plugin mediante el menú contextual, como se muestra en la ilustración 26, que aparece al hacer clic derecho sobre el área de texto.

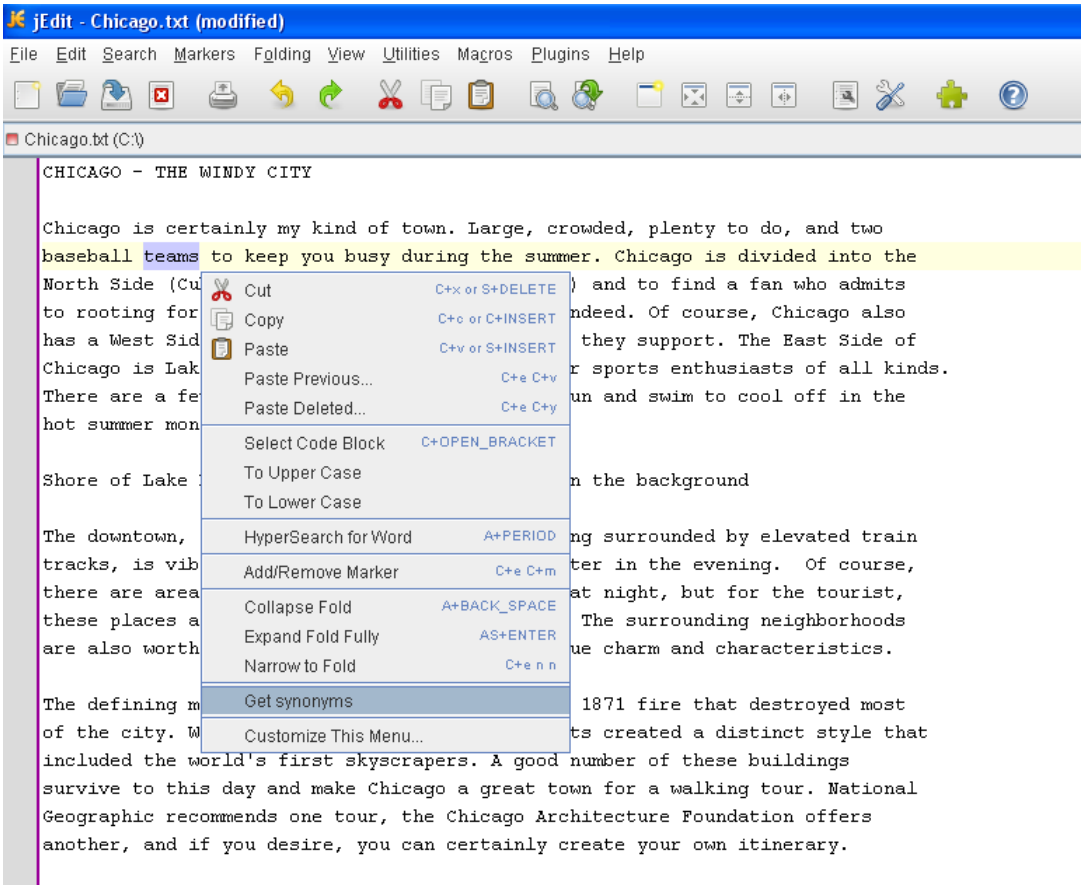

ILUSTRACIÓN 26: Menú contextual del plugin Synonyms

En caso de que no se haya seleccionado una palabra, obtenemos el mensaje de error de la ilustración 27.

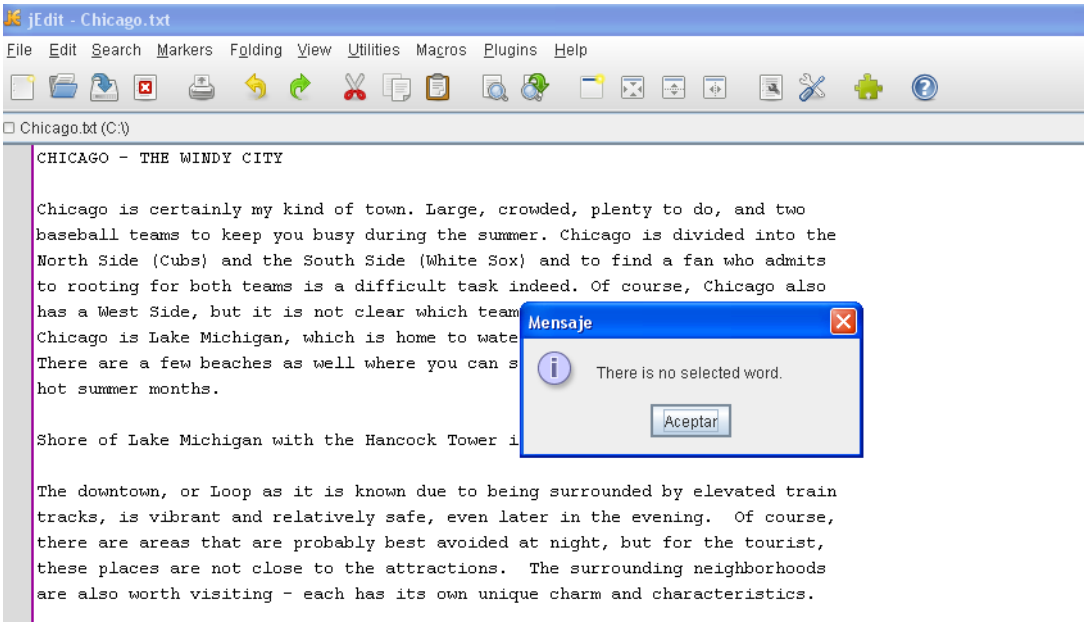

ILUSTRACIÓN 27: Mensaje de error - No hay ninguna palabra seleccionada

Si no existen sinónimos de la palabra seleccionada aparece una ventana como la de la ilustración 28 indicándonos la situación.

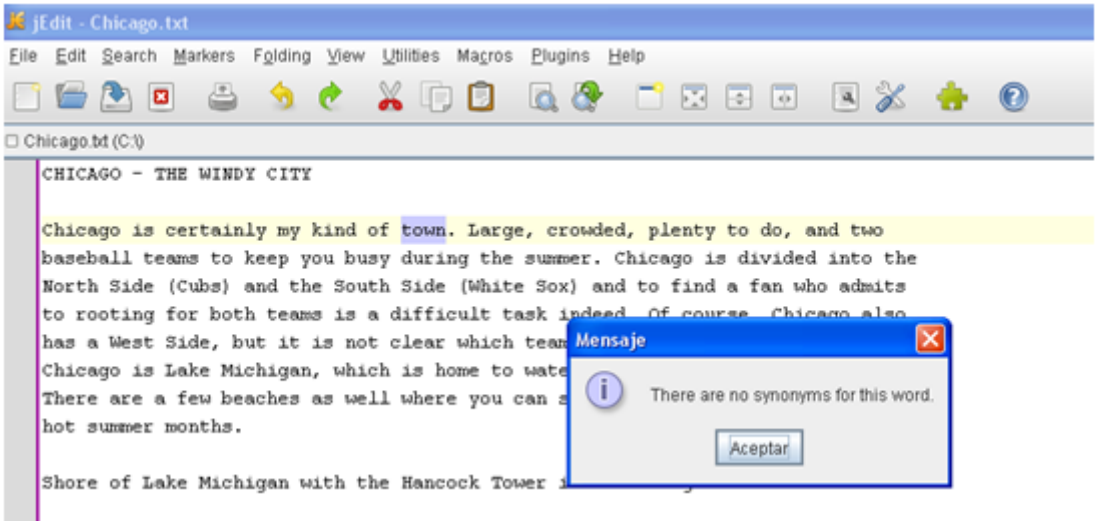

ILUSTRACIÓN 28: Mensaje de error - No hay sinónimos para la palabra

Si la palabra seleccionada tiene sinónimos, obtenemos el panel de la ilustración 29.

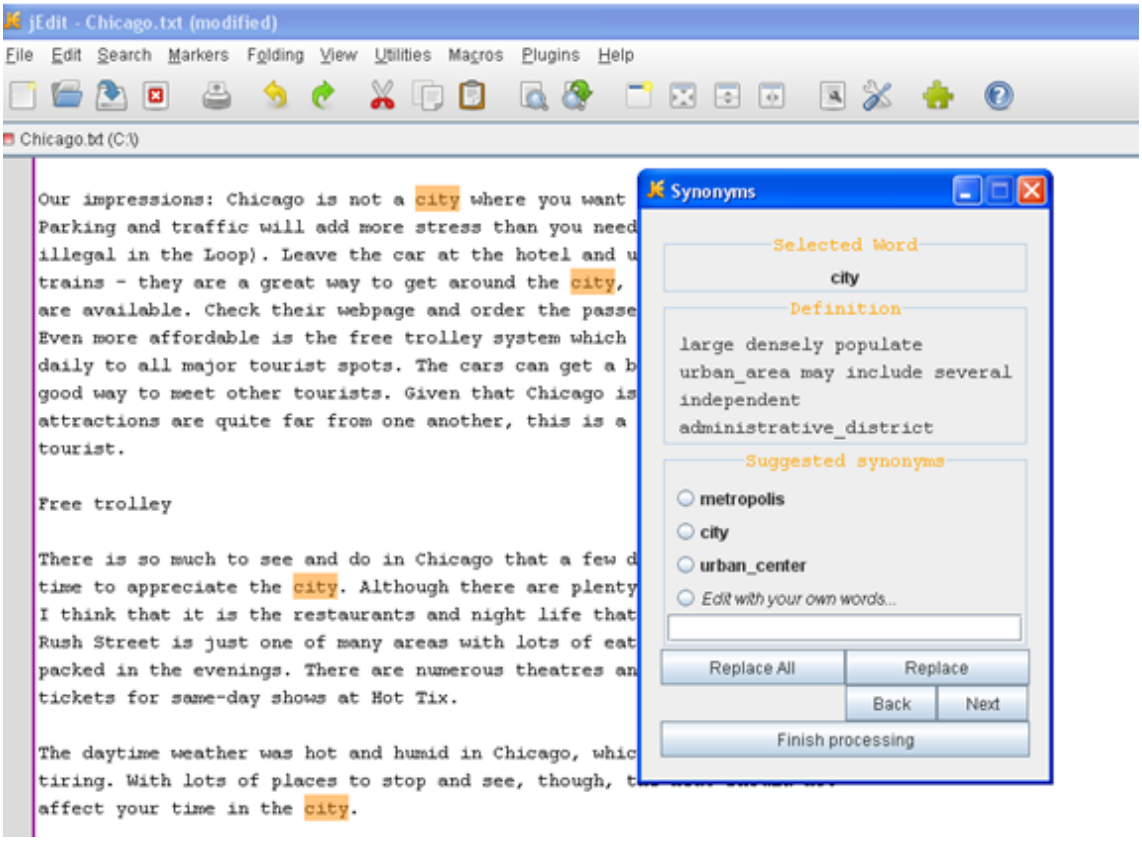

ILUSTRACIÓN 29: Panel del plugin Synonyms

En la ilustración 29 se puede observar que la palabra que se quiere simplificar es *city* y cómo se resaltan todas las apariciones de esta palabra en el texto.

A continuación se puede ver la definición de la palabra seleccionada. Wordnet, diccionario que se usa como base para obtener los sinónimos, puede tener varios significados para una palabra, y en función a ese significado, varios sinónimos. Para nuestro proyecto, accedemos a los sinónimos del primer significado (que en Wordnet es siempre el más frecuente) así que es esta definición la que se puede ver en el panel.

Después se pueden ver las posibles sustituciones de la palabra seleccionada. Para cada una de las apariciones, empezando por la que se ha seleccionado, se puede optar por realizar varias acciones:

- dejarla igual haciendo clic en *Next* o sustituyéndola por sí misma,
- cambiarla por uno de los sinónimos que aparecen en la lista, seleccionando el sinónimo deseado y haciendo clic en *Replace*,
- editarla nosotros mismos, seleccionando la opción *Edit with your own words*, y escribiendo en el área de texto nuestro sinónimo. Como en la opción anterior, para reemplazarla, se debe hacer clic en *Replace*.

La opción *Replace All* reemplaza todas las apariciones que están marcadas en el texto por la opción seleccionada.

Se puede navegar entre todas las apariciones de la palabra haciendo clic en *Next* o en *Back*, avanzando o retrocediendo por todo el texto hasta llegar al final o al principio, dónde se tiene la opción de volver a empezar con la búsqueda.

Además, se puede elegir dejar marcadas en el texto las palabras seleccionadas que no han sido sustituidas y de esta forma dejarlas para más adelante, o decidir que es mejor esa palabra que cualquiera de posibles sinónimos, y por tanto, finalizar la simplificación de esa palabra, haciendo clic en *Finish processing*.

Todos los plugins de jEdit tienen un panel de opciones para cambiar las opciones que admita el plugin. En este caso, la única opción disponible es cambiar el color con el que se subrayan todas las apariciones de la palabra seleccionada. Podemos acceder a esta opción mediante el menú *Plugins < Plugin Options* obteniendo un panel como el de la ilustración 30.

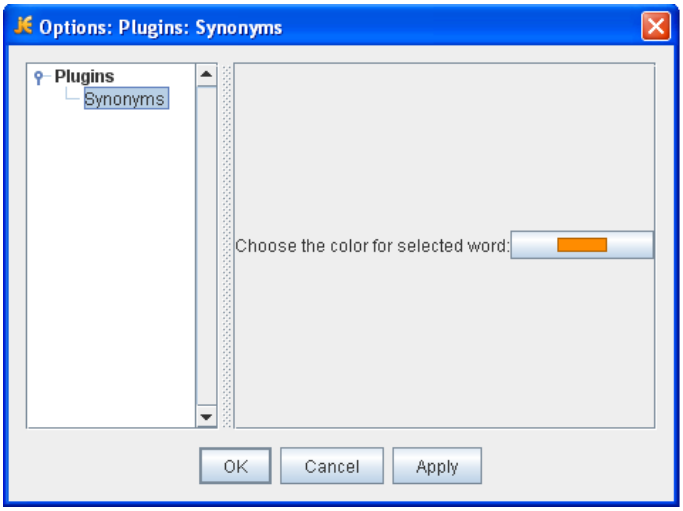

ILUSTRACIÓN 30: Opciones asociadas al plugin Synonyms

5.2 Herramientas usadas y cómo se usan

Para desarrollar este plugin hemos necesitado de herramientas externas que se describen a continuación.

5.2.1 OpenNLP

OpenNLP (mencionado en el capítulo 2.2.1) se usa para el marcado del part-of-speech (POS o categoría gramatical) de la palabra de la que queremos obtener sinónimos y saber así si se trata de un verbo o de un sustantivo para poder buscarlo en el diccionario. Dada una frase como por ejemplo:

A boy was riding a bike while he was eating an apple.

Si aplicamos el detector de tokens de OpenNLP obtenemos tanto los tokens que forman la frase como el etiquetado para cada token:

TOKENS	PART-OF-SPEECH DE CADA TOKEN
A	*A* (DT)
boy	*boy* (NN)
was	*was* (VBD)
riding	*riding* (VBG)
a	*a* (DT)
bike	*bike* (NN)
while	*while* (IN)
he	*he* (PRP)
was	*was* (VBD)
eating	*eating* (VBG)
an	*an* (DT)
apple	*apple* (NN)
.	. (.)

Una vez que el usuario ha seleccionado en el texto la palabra de la que quiere buscar sinónimos, lo primero que hacemos es comprobar que efectivamente se trata de una única palabra, un único token. Para ello usamos el detector de tokens de OpenNLP. Aplicándoselo a la frase del ejemplo, obtendríamos los tokens de la primera columna de la tabla.

Sabiendo que trabajamos con un único token, pasamos a conocer su etiquetado, su part-of-speech. De esta forma podemos saber si la palabra que hemos seleccionado es un sustantivo (NNx) o un verbo (VBx) y buscarla en el diccionario. El etiquetado de la frase del ejemplo, es el correspondiente a la segunda columna de la tabla.

Además también usamos el detector de frases de OpenNLP para obtener la frase original y la frase que resulta tras la sustitución de la palabra seleccionada por su sinónimo.

Por ejemplo, dado el siguiente texto:

A boy was riding a bike. He was also eating an apple. The road was empty.

Si aplicamos la detección de frases de OpenNLP resultan las siguientes frases con el offset de inicio y fin de cada una de ellas:

FRASES	OFFSET INICIO Y FIN DE CADA FRASE
A boy was riding a bike.	0..24
He was also eating an apple.	25..53
The road was empty.	54..73

La primera columna de la tabla contiene cada una de las frases que forman el texto. La segunda columna de la tabla contiene el offset de inicio y fin de cada frase, que nos indica la posición de la frase en el texto y se usa para conocer cuál es la frase original y la nueva frase resultante de sustituir la palabra por el sinónimo seleccionado. Esto se explica en detalle, en el capítulo 5.3, correspondiente al log del plugin Synonyms.

5.2.2 WordNet

Usamos esta herramienta y la interfaz LWWN (mencionada en el capítulo 2.2.2) para conseguir el significado y los sinónimos de la palabra que queremos simplificar.

A esta base de datos, le pasamos la etiqueta léxica obtenida con OpenNLP. Consultamos el primer significado que encontramos en la base de datos y devolvemos todos sus sinónimos para que el usuario que edita el texto elija el que crea más conveniente.

Por ejemplo, buscamos *table* en WordNet y el primer significado con los primeros sinónimos (el primer synset) es el siguiente:

```
1. (57) table, tabular array -- (a set of data arranged in rows and
   columns; "see table 1")
```

De aquí obtenemos los sinónimos *table* y *tabular array* y su significado, *a set of data arranged in rows and columns*.

Usando nuestro plugin Synonyms obtenemos los resultados de la ilustración 31.

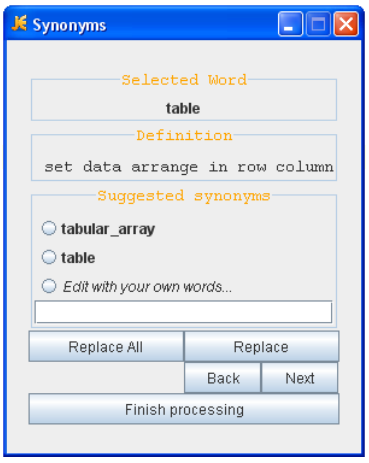

ILUSTRACIÓN 31: Sinónimos para *table*

5.2.3 MorphAdorner

La aplicación concreta que utilizamos de MorphAdorner (mencionado en el capítulo 2.2.3) es el pluralizador.

Cuando se unió OpenNLP y WordNet, nos encontramos con el problema de que WordNet no tiene en su base de datos sustantivos en plural pero OpenNLP sí que detecta sustantivos en plural. Por ello, se introdujo en el plugin el pluralizador. Cuando se selecciona una palabra y OpenNLP determina que es un nombre en plural, ésta se transforma a singular mediante el pluralizador para buscar su significado y sus sinónimos en WordNet. Una vez obtenidos los sinónimos, se pasan a su forma plural usando de nuevo el pluralizador.

Esto permite facilitar más al usuario su trabajo de simplificación, ya que evitamos que el usuario tenga que hacer manualmente la concordancia del texto. Esta funcionalidad no se encuentra por ejemplo en el proyecto PorSimples, que lo que permite para evitar los problemas de concordancia es la edición (manual) de las palabras sustitutas en el texto.

MorphAdorner tiene más aplicaciones útiles para este proyecto, como el conjugador de verbos, pero hemos dejado este trabajo para futuras mejoras.

5.3 Log

El log del plugin *Synonyms* guarda cada uno de los cambios léxicos que se han realizado sobre el archivo original. De la simplificación léxica es interesante almacenar la palabra que se ha sustituido y la palabra que la ha sustituido además del contexto en el que se encontraba, de esta forma se puede llegar a obtener las palabras que los usuarios consideran más sencillas puesto que son las que más aparecen en el log.

En el plugin *Synonyms* para cada palabra sustituida se genera un elemento XML de la siguiente forma:

```
<OriginalWord word="palabra a reemplazar">
  <Synonym word="sinónimo por el que se ha reemplazado la palabra">
    <Substitution>
      <OriginalPhrase>
        Frase original con la palabra a reemplazar.
      </OriginalPhrase>
      <NewPhrase>
        Nueva frase con la palabra reemplazada.
      </NewPhrase>
      <SuggestedSynonyms>
        Lista de sinónimos sugeridos por la aplicación.
      </SuggestedSynonyms>
      <Edited>
        Indica si el usuario ha editado la palabra a reemplazar:
        Yes / No.
      </Edited>
      <Order>
        Orden en la lista de sustituciones.
```

```
        </Order>
        <Action>
          Forma de reemplazo: Replace / Replace All.
        </Action>
      </Substitution>
    </Synonym>
  </OriginalWord>
```

Por cada elemento *OriginalWord* se almacena una palabra de las que han sido sustituidas, y para cada elemento de este tipo, se crea un elemento *Synonym* para cada uno de los sinónimos por los que ha sido sustituida. Cada elemento *Synonym* consta de un elemento *Substitution* que tiene seis campos: el campo *OriginalPhrase* en el que se almacena la frase original con la palabra que se quiere sustituir; el campo *NewPhrase* en el que se almacena la frase resultante de cambiar la palabra sustituida por el sinónimo sustituto; el campo *SuggestedSynonyms* en el que se almacena una lista con los sinónimos sugeridos por la aplicación; el campo *Edited* nos indica si el usuario ha decidido editar la palabra a reemplazar ("Yes") o si el usuario decide usar uno de los sinónimos ofrecidos por la aplicación ("No"); el campo *Order*, en el que se almacena el orden que ocupa esta sustitución en la lista de sustituciones; y el campo *Action* en el que se especifica por qué acción ha sido sustituida la palabra, por un *Replace* o por un *Replace All*.

Para obtener los campos *OriginalPhrase* y *NewPhrase* usamos el detector de frases de OpenNLP, explicado en el apartado 5.2.1. Cuando seleccionamos la palabra para la que buscamos los sinónimos, guardamos el offset de la palabra seleccionada. Este offset nos sirve para obtener la frase original dentro del texto y la nueva frase.

Por cada una de las palabras sustituidas se crea un elemento *OriginalWord* en el documento XML, cuya etiqueta raíz es *SynonymsLog*. El documento XML tiene el siguiente aspecto:

```
<SynonymsLog>
  <OriginalWord word="word1">
    <Synonym word="synonym11">
      <Substitution>
        <OriginalPhrase> … </OriginalPhrase>
        <NewPhrase> … </NewPhrase>
        <SuggestedSynonyms> … </SuggestedSynonyms
        <Edited> … </Edited>
        <Order> … </Order>
        <Action> … </Action>
      </Substitution>
      <Substitution>
        <OriginalPhrase> … </OriginalPhrase>
        <NewPhrase> … </NewPhrase>
        <SuggestedSynonyms> … </SuggestedSynonyms
        <Edited> … </Edited>
        <Order> … </Order>
        <Action> … </Action>
      </Substitution>
    </Synonym>

    <Synonym word="synonym12">
```

```
        <Substitution>
          <OriginalPhrase> … </OriginalPhrase>
          <NewPhrase> … </NewPhrase>
          <SuggestedSynonyms> … </SuggestedSynonyms
          <Edited> … </Edited>
          <Order> … </Order>
          <Action> … </Action>
        </Substitution>
      </Synonym>
    </OriginalWord>

    <OriginalWord word="word2">
      <Synonym word="synonym21">
        <Substitution>
          <OriginalPhrase> … </OriginalPhrase>
          <NewPhrase> … </NewPhrase>
          <SuggestedSynonyms> … </SuggestedSynonyms
          <Edited> … </Edited>
          <Order> … </Order>
          <Action> … </Action>
      </Synonym>
    </OriginalWord>
  </SynonymsLog>
```

En este ejemplo concreto, existirían dos palabras que han sido sustituidas, *word1* y *word2*. La primera de ellas habría sido sustituida por dos palabras diferentes, *synonyms11* y *synonyms12*, y el *synonyms11* habría sido usado como sustituto, en dos ocasiones diferentes (hay dos elementos *Substitution* para esa palabra).

5.4 Diagrama de arquitectura

La organización interna del plugin Synonyms es la que se muestra en la ilustración 32, en la que podemos observar que se dispone de una clase SynonymsPlugin que es necesaria para poder desarrollar un plugin para jEdit, tiene que heredar por requerimiento de jEdit de la clase EditPlugin, recordemos que el nombre de la clase principal tiene que terminar con el sufijo Plugin. También consta de una simplificación de la herramienta de resaltado como es el caso de HighlightPluginSynonyms.

La clase Synonyms es la encargada de gestionar la interfaz del plugin además de la gestión de la herramienta de Procesamiento de Lenguaje Natural y del diccionario (base de datos de sinónimos).

Finalmente se dispone de un gestor de cambios a nivel léxico, estos cambios se ven reflejados en el log físico.

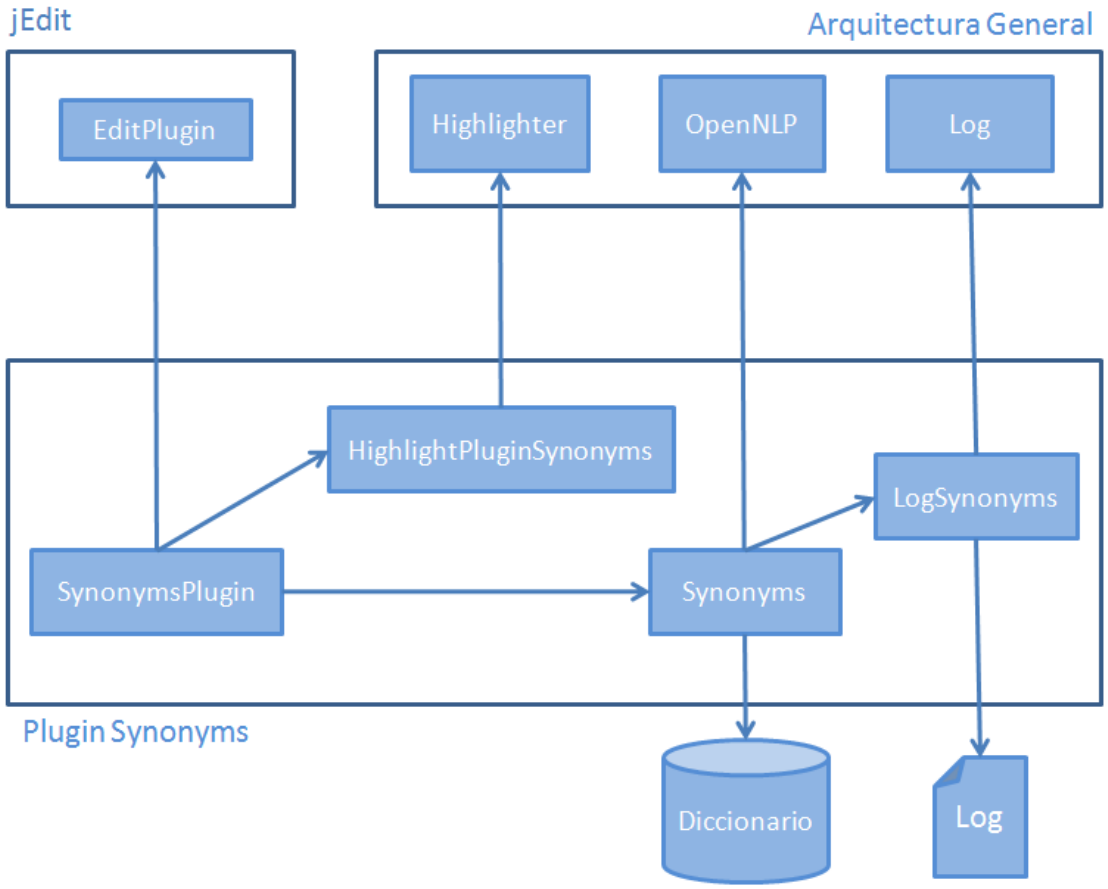

ILUSTRACIÓN 32: Diagrama de arquitectura del plugin Synonyms

Capítulo 6

Plugin Split

Capítulo 6 – Plugin Split

El objetivo de este plugin es proporcionar el apoyo en el proceso de simplificación sintáctica en inglés, ofreciendo la frase separada por la conjunción coordinada de una oración que el usuario seleccione.

6.1 ¿Qué hace? Funcionamiento

Una forma de simplificar textos consiste en separar las frases largas o con mucho contenido semántico por la conjunción coordinada “and” y así tener dos frases más cortas.

Para trabajar con este plugin primero debe instalarse como se indica en capítulos anteriores y a continuación abrir con jEdit el archivo que se quiere simplificar.

Para poder empezar a usar este plugin es necesario activarlo previamente haciendo clic en *Plugins < Split < Enable to Split* como se puede observar en la ilustración 33. Una vez activado se iluminarán todas las apariciones de la conjunción "and" de color verde (por defecto, es configurable) para facilitar la situación de las conjunciones.

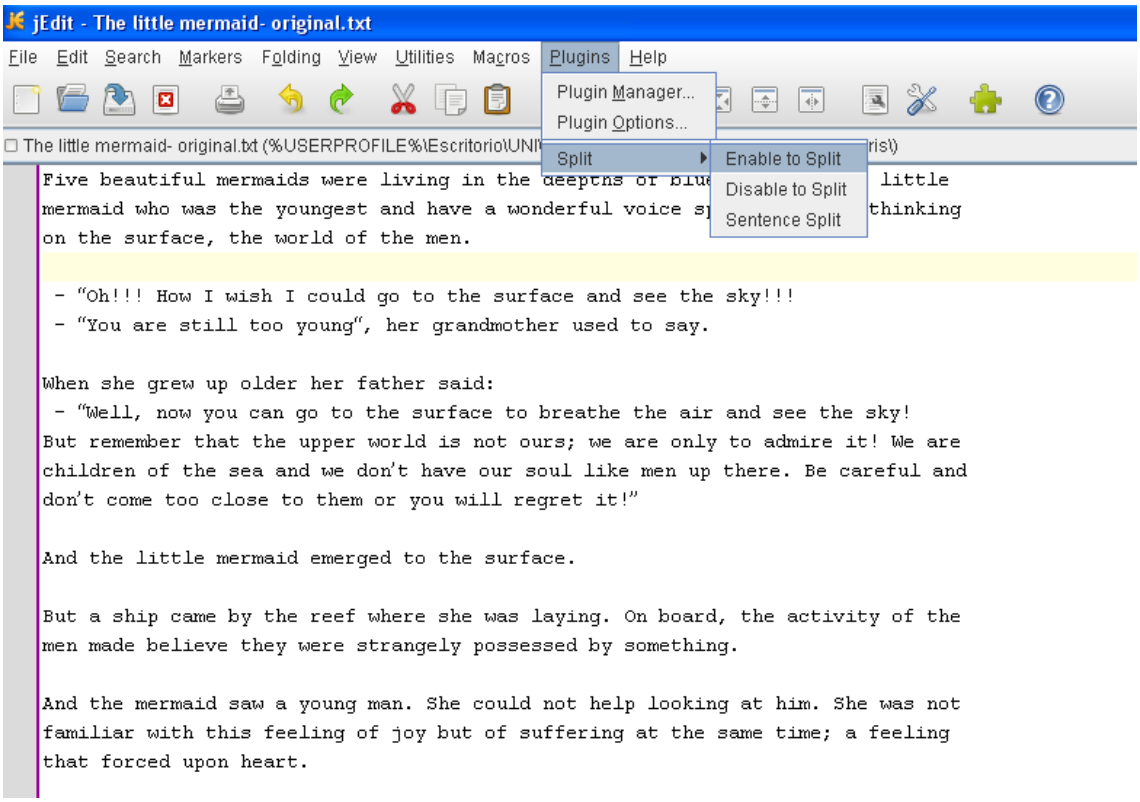

ILUSTRACIÓN 33: Activación del plugin Split

También se puede activar el plugin mediante el menú contextual al hacer clic derecho sobre el área de texto tal y como se ve en la ilustración 34.

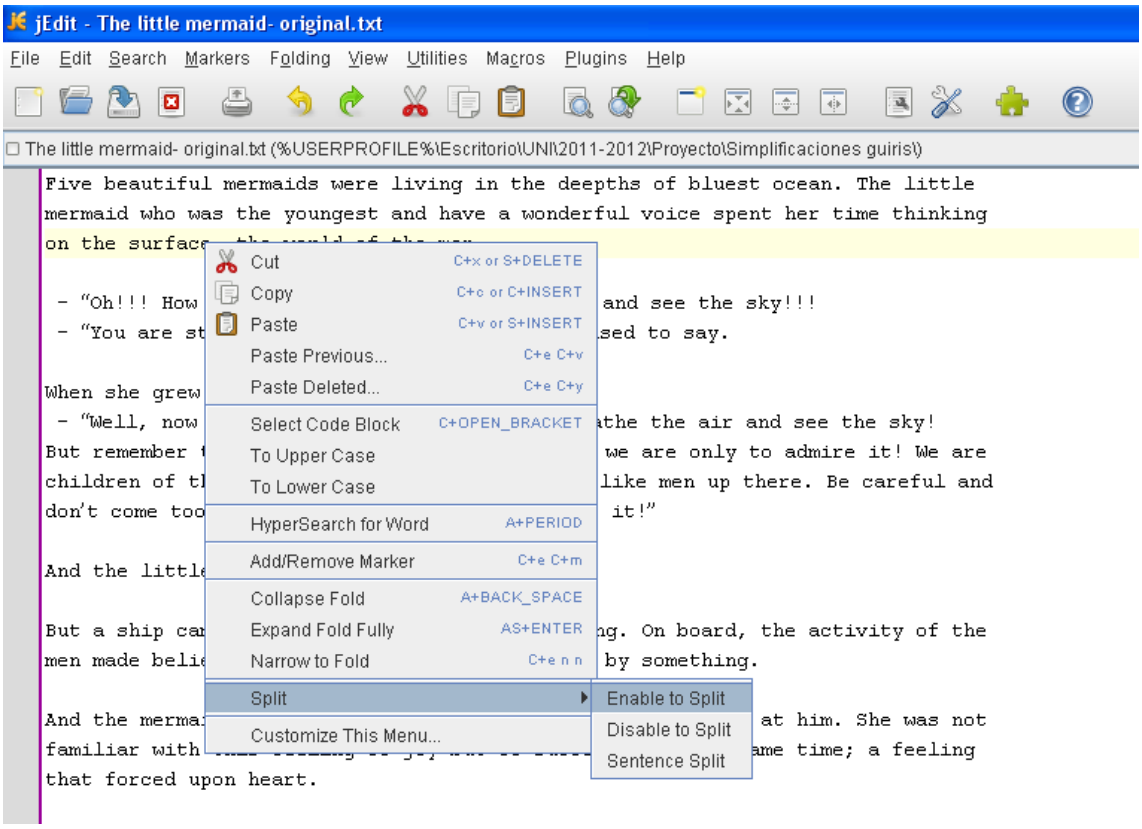

ILUSTRACIÓN 34: Activación mediante el menú contextual del plugin Split

En caso de no activar esta opción e intentar seleccionar la opción de *Sentence Split*, aparecerá el mensaje de error de la ilustración 35.

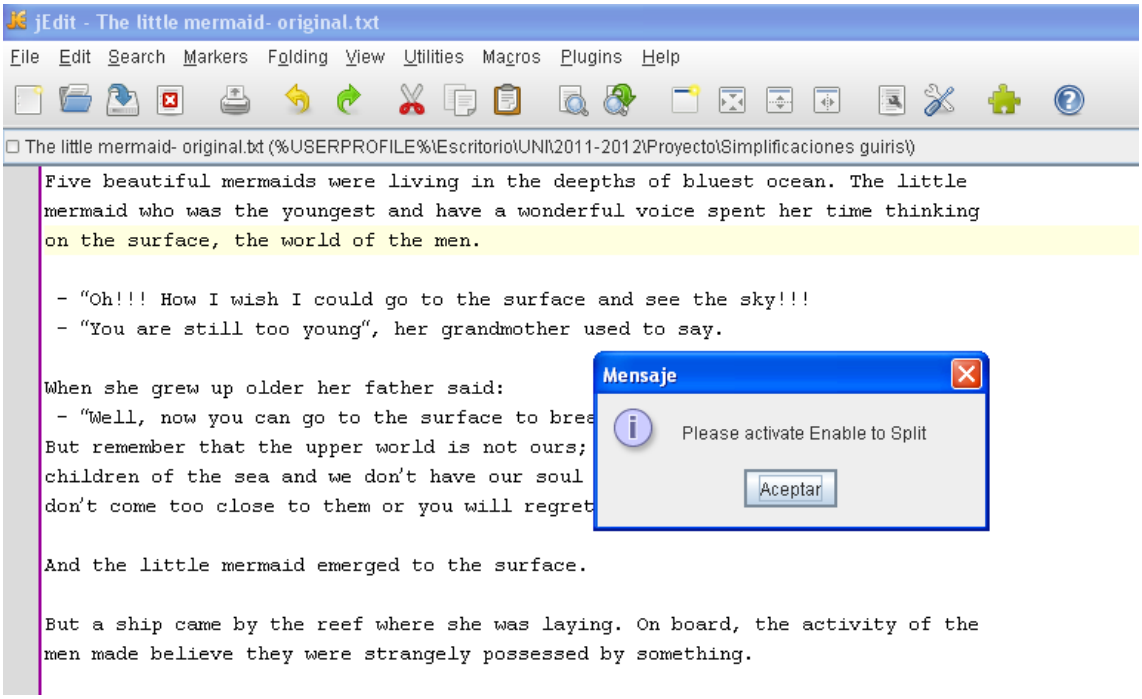

ILUSTRACIÓN 35: Mensaje de error - No está activado el plugin Split

En caso de haber activado la opción *Enable to Split* pero no se seleccione una de las conjunciones marcadas, aparecerá el mensaje de error mostrado en la ilustración 36.

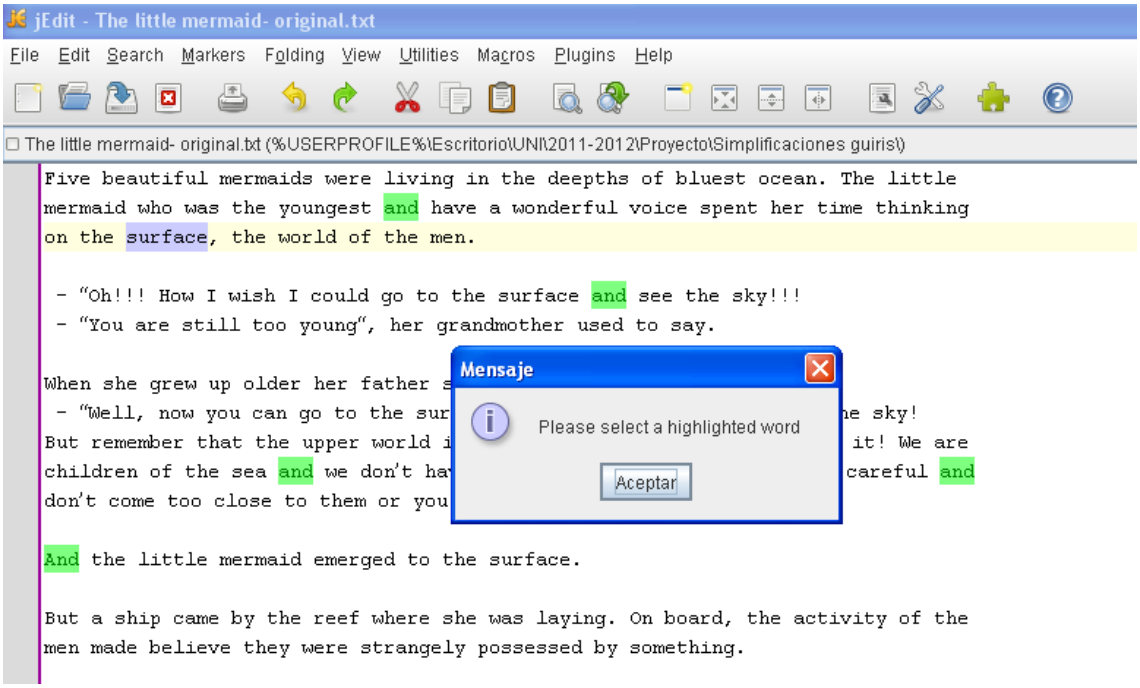

ILUSTRACIÓN 36: Mensaje de error - No hay ninguna conjunción seleccionada

Si esta activado y se selecciona una conjunción correcta aparecerá el panel de la ilustración 37.

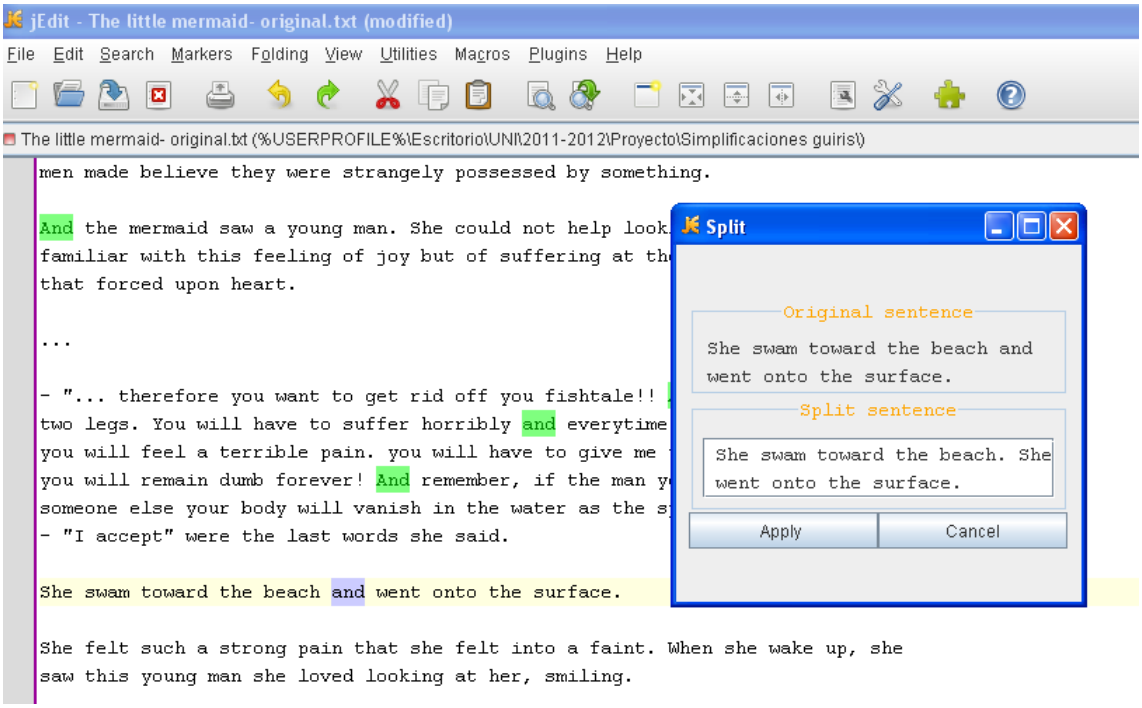

ILUSTRACIÓN 37: Panel del plugin Split

Se muestra la frase original en la parte superior y la frase simplificada en la parte inferior pudiéndose editar.

Para que el cambio tenga efecto sobre el texto se pulsa el botón *Apply*. Si no se quiere aplicar el cambio se pulsa el botón C*ancel*.

Cuando se quiere dejar de trabajar con el plugin para que las conjunciones no queden resaltadas hay que seleccionar la opción *Disable to Split* haciendo clic en *Plugins* < *Split* < *Disable to Split* como se muestra en la ilustración 38.

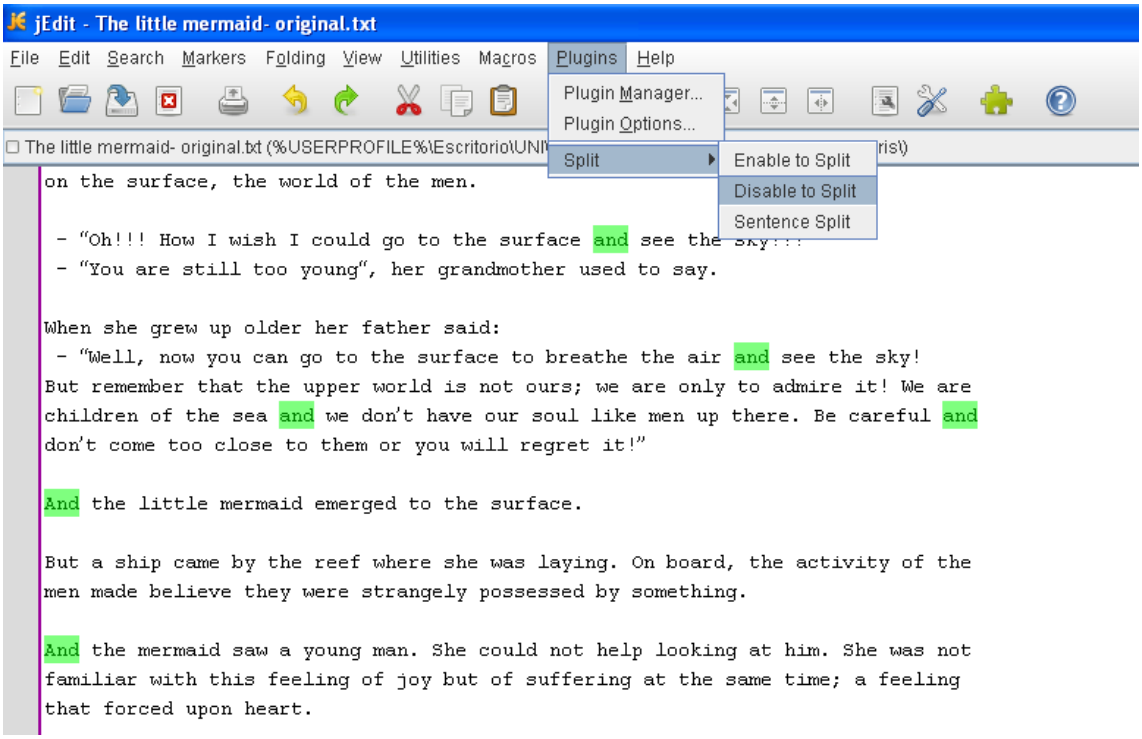

ILUSTRACIÓN 38: Desactivación del plugin Split

Todos los plugins de jEdit tienen un panel de opciones para cambiar las opciones que admita el plugin. En este caso, la única opción disponible es cambiar el color con el que se subrayan las conjunciones coordinadas "*and*". Podemos acceder a esta opción mediante el menú *Plugins* < *Plugin Options* como se puede observar en la ilustración 39.

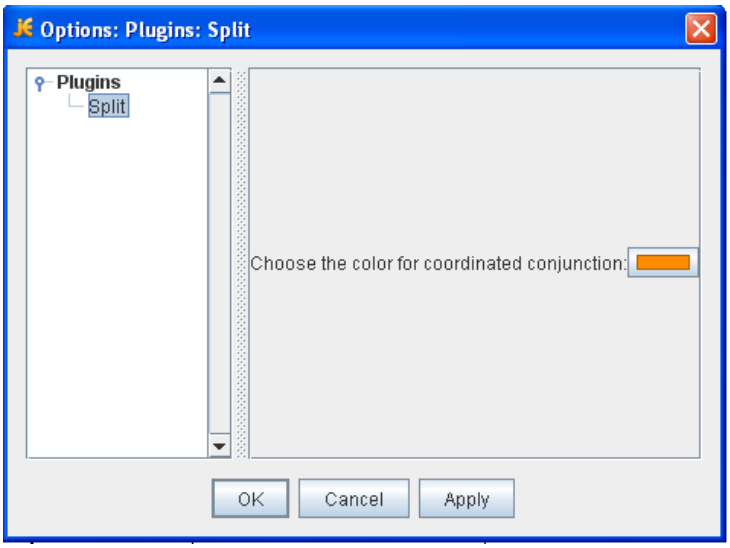

ILUSTRACIÓN 39: Opciones del plugin Split

6.2 Herramientas usadas e implementación

Para desarrollar este plugin hemos necesitado de herramientas externas que se describen a continuación.

6.2.1 OpenNLP

Al igual que para el plugin Synonyms hemos usado OpenNLP para el marcado de part-of-speech, el detector de tokens y el detector de frases, explicados en el capítulo 5.2.1.

Para este plugin además de estas herramientas hemos usado el parser, que recibe una frase (donde cada token tiene que estar separado por un espacio) y nos devuelve su parseo.

Si aplicamos el parser de OpenNLP a la siguiente frase "a *boy is running in the beach and a girl is eating an apple."*, obtenemos:

```
(TOP(S(S(NP(DT  A)(NN  boy))(VP(VBZ  is)(VP(VBG
running)(PP(IN in)(NP(DT the)(NN beach))))))(CC
and)(S(NP(DT  a)(NN  girl))(VP(VBZ  is)(VP(VBG
eating)(NP(DT an)(NN apple)))))(. .)))
```

Con esta cadena podemos generar un árbol sintáctico para la oración que usaremos para la implementación.

6.2.2 Implementación

Para facilitar la manipulación de la cadena devuelta por el parser se genera un árbol sintáctico. Las oraciones que reconocemos tienen estructuras claras que reconocemos gracias a este árbol sintáctico generado.

Todas las cadenas generadas por OpenNLP tienen una raíz que se etiqueta como *TOP* y normalmente otra elemento que le sigue, que se etiqueta con *S* y se denomina "sentencia"·. Después de estas etiquetas generadas es donde debemos reconocer la estructura de la oración que queremos reconocer.

Un primer tipo de oración reconocida es una oración cuyo parseo de OpenNLP contenga en sus niveles más altos (después del *TOP* y la sentencia) una sentencia seguida por una conjunción coordinada "*and*" y a continuación otra sentencia. En el ejemplo "*A boy is running in the beach and a girl is eating an apple."*, el árbol sintáctico generado después del parseo es el mostrado en la ilustración 40. En este árbol observamos la estructura de los niveles anteriores y la del nivel que nos interesa reconocer, marcado en color amarillo. Podemos observar también que la oración reconocida es la formada por las hojas del árbol sintáctico.

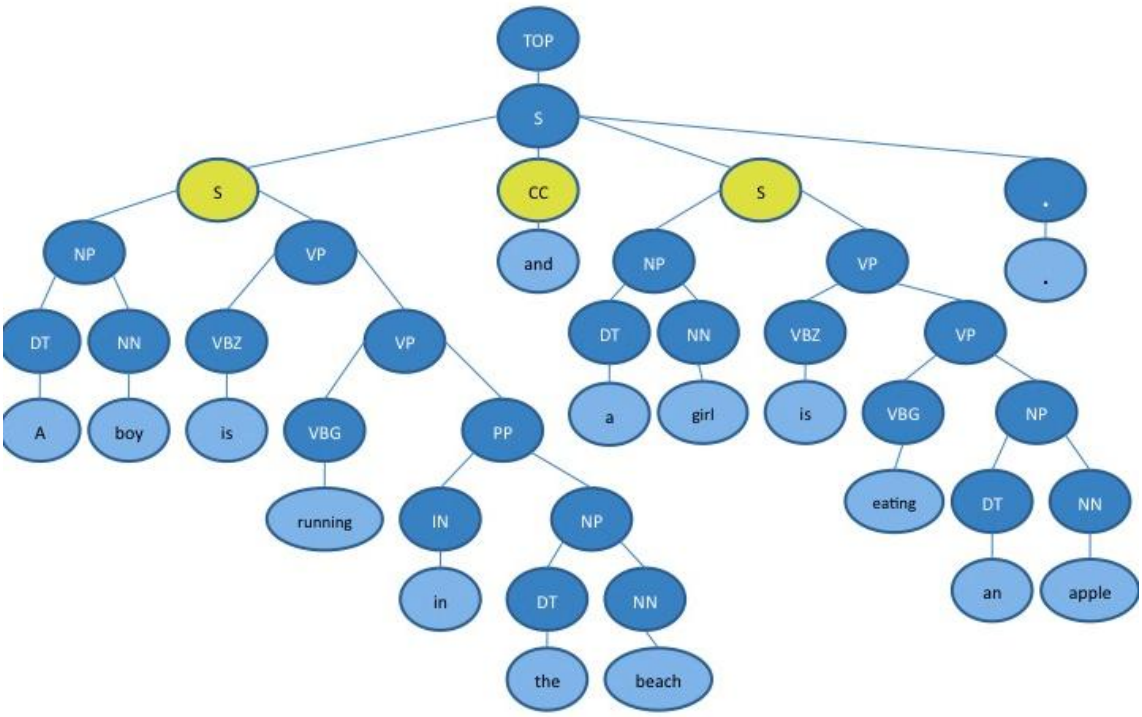

ILUSTRACIÓN 40: Árbol sintáctico tipo 1

Usamos este árbol para separar la frase que cumple la estructura deseada por la conjunción coordinada *and* y generar los dos árboles mediante los cuales obtendremos las oraciones simplificadas. En la ilustración 41 y en la ilustración 42 observamos los dos árboles resultantes del árbol de la ilustración 40. Ambas sentencias de la frase original tienen sujeto, se añade el punto al final de las oraciones y la etiqueta *TOP* que indica la raíz de la frase.

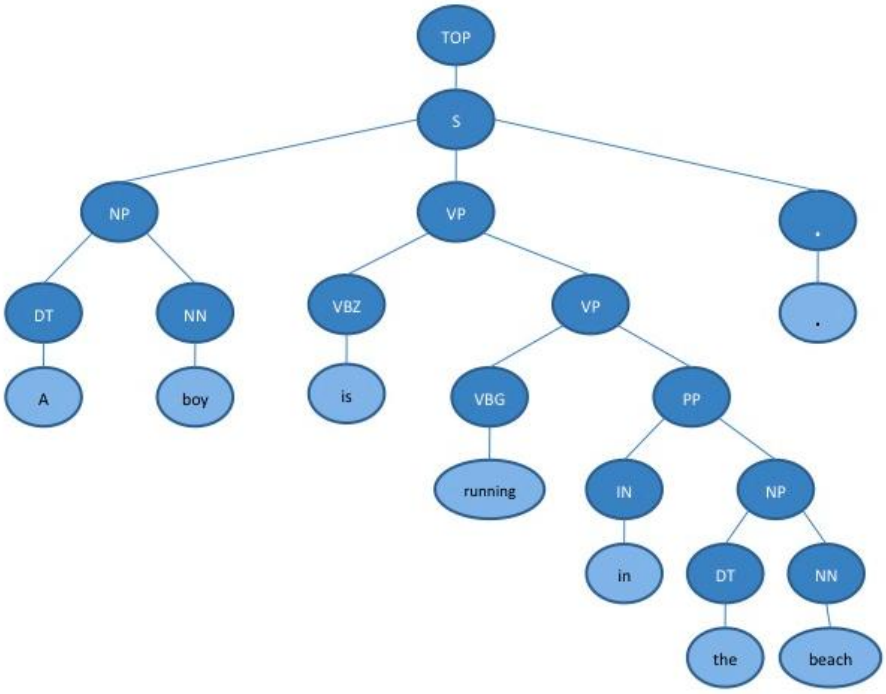

ILUSTRACIÓN 41: Árbol 1 resultante de la estructura tipo 1

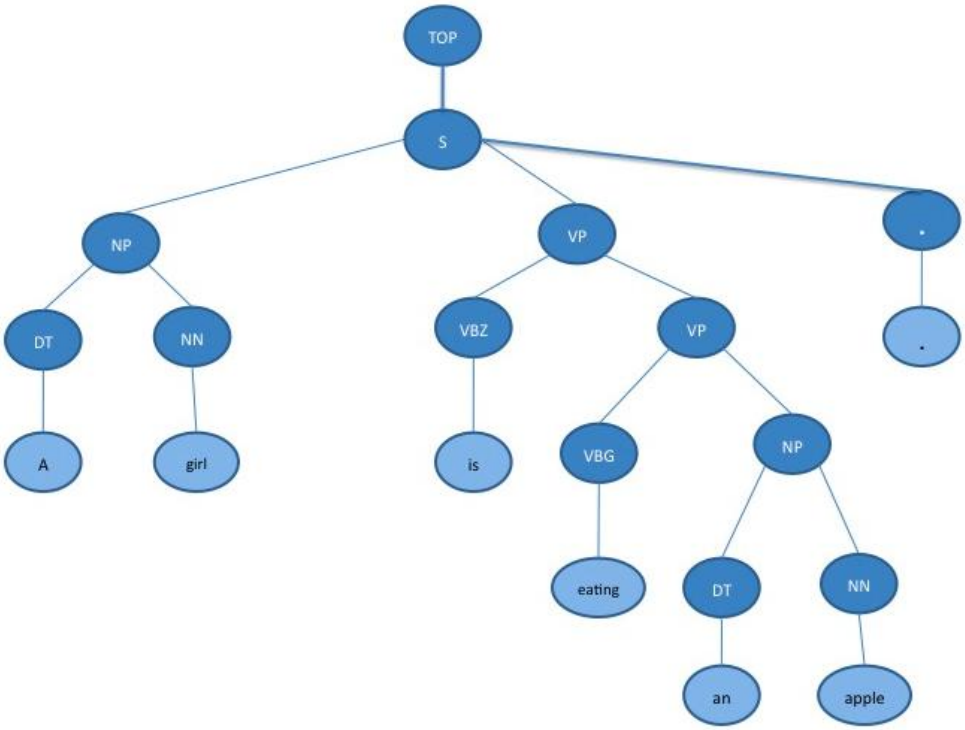

ILUSTRACIÓN 42: Árbol 2 resultante de la estructura tipo 1

El segundo tipo de oración reconocida es una oración cuyo parseo de OpenNLP contenga en sus niveles más altos (después del *TOP* y la sentencia) un verbo separado de otro verbo por una conjunción coordinada "*and*" (marcado en amarillo). En el ejemplo "*A boy is running and is eating an apple.",* el árbol sintáctico generado después del parseo es el mostrado en la ilustración 43.

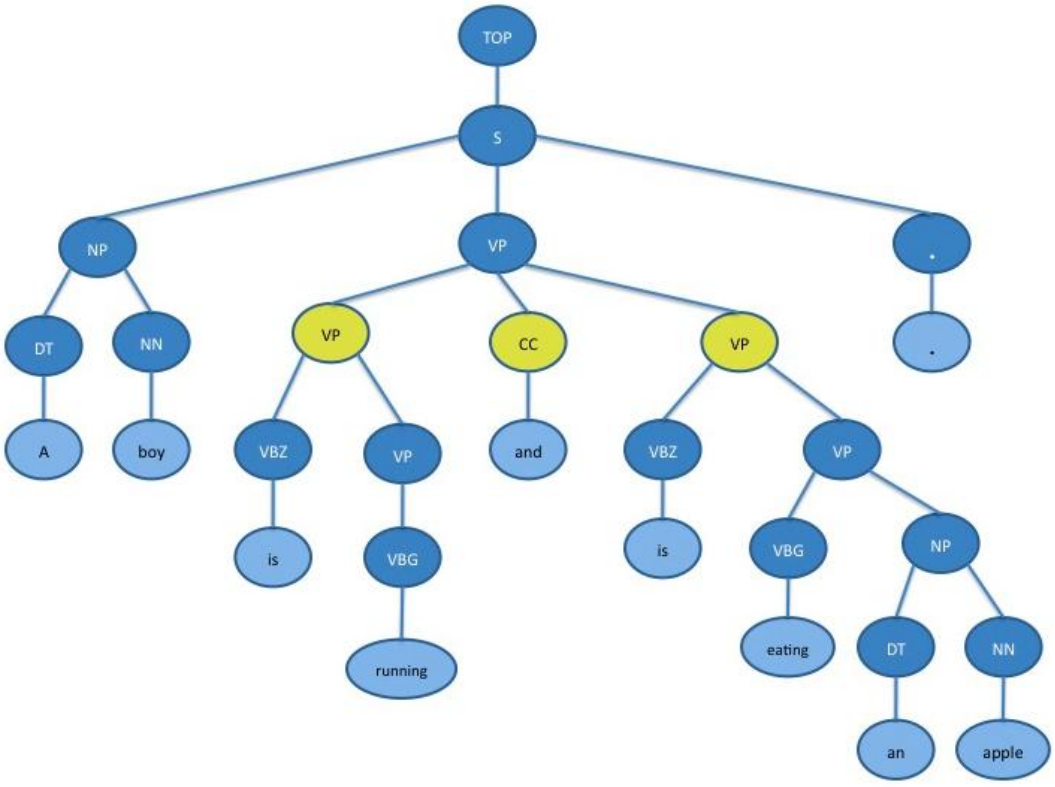

ILUSTRACIÓN 43: Árbol sintáctico tipo 2

Nuevamente observamos que la oración reconocida es la formada por las hojas del árbol sintáctico. En esta segunda estructura, la conjunción coordinada se encuentra entre dos verbos, lo que se hace es duplicar el sujeto debido a que el sujeto de la frase original hace referencia a ambos verbos utilizados para reconocer la estructura. En la ilustración 44 y en la ilustración 45 observamos los dos árboles sintácticos obtenidos del árbol de la ilustración 43.

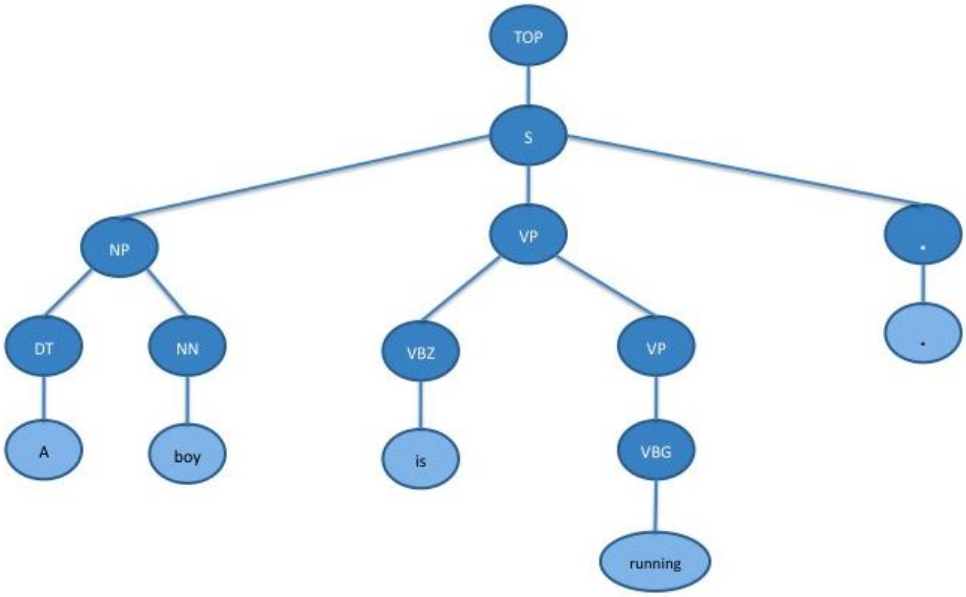

ILUSTRACIÓN 44: Árbol 1 resultante de la estructura tipo 2

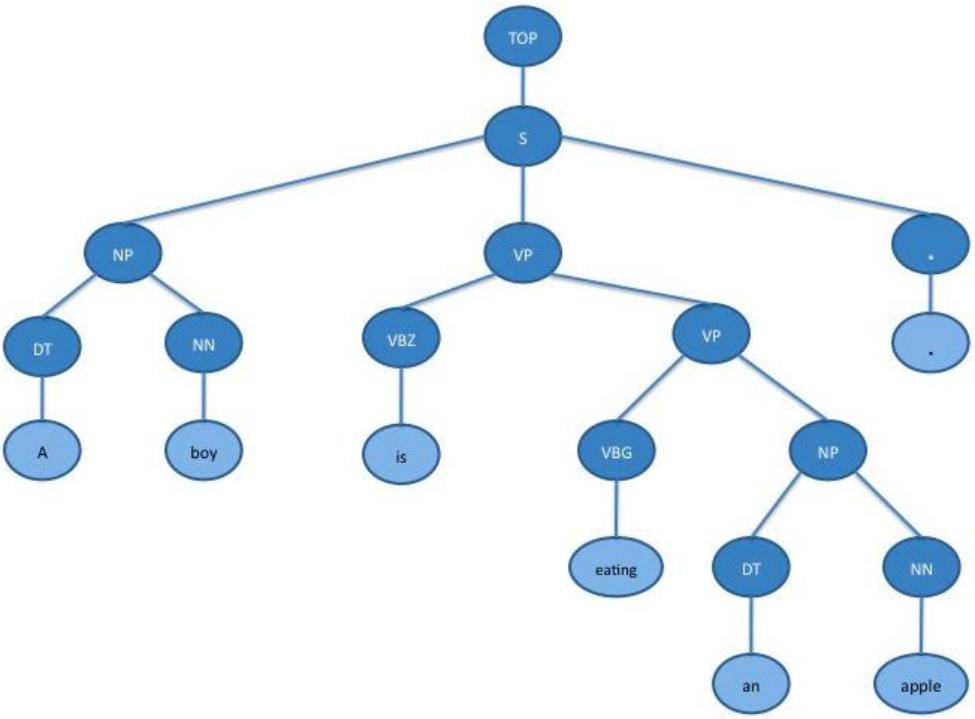

ILUSTRACIÓN 45: Árbol 2 resultante de la estructura tipo 2

6.3 Log

El log del plugin *Split* guarda cada uno de los cambios sintácticos que se han realizado sobre el archivo original. De la simplificación sintáctica es interesante almacenar la frase original y la resultante tras aplicar dicha simplificación.

En el plugin *Split* para cada frase sustituida se genera un elemento XML de la siguiente forma:

```
<Split>
  <OriginalPhrase>
    Frase original que se quiere simplificar.
  </OriginalPhrase>
  <SplitPhrase>
    Frase resultante de aplicar la simplificación sintáctica.
  </SplitPhrase>
  <SuggestedSplit>
    Frase simplificada sugerida por la aplicación.
  </SuggestedSplit>
  <Edited>
    Indica si el usuario ha editado la frase a simplificar:
    Yes / No.
  </Edited>
  <Order>
    Orden en la lista de sustituciones.
  </Order>
</Split>
```

Cada elemento *Split* consta de cinco campos: el campo *OriginalPhrase* en el que se almacena la frase original que se quiere simplificar; el campo *NewPhrase* en el que se almacena la frase resultante de aplicar la simplificación sintáctica; el campo *SuggestedSplit* en el que se almacena la frase simplificada sugerida por la aplicación; el campo *Edited* nos indica si el usuario ha decidido editar la frase original “Yes”, o si decide usar la frase simplificada sugerida por la aplicación “No”; y el campo *Order*, en el que se almacena el orden que ocupa esta sustitución en la lista de sustituciones.

Para obtener los campos *OriginalPhrase* y *NewPhrase* usamos el detector de frases de OpenNLP, explicado en el apartado 6.2.1. Cuando seleccionamos la palabra *“and”* contenida en la frase para la que buscamos una simplificación sintáctica, guardamos el offset de la palabra seleccionada. Este offset nos sirve para obtener la frase original dentro del texto y la nueva frase.

Por cada una de las frases simplificadas se crea un elemento *Split* en el documento XML, cuya etiqueta raíz es *SyntacticLog*. El documento XML tiene el siguiente aspecto:

```
<SyntacticLog>
  <Split>
    <OriginalPhrase> frase1 </OriginalPhrase>
    <SplitPhrase> resultado1 </SplitPhrase>
    <SuggestedSplit> … </SuggestedSplit>
    <Edited> … </Edited>
```

```
      <Order> … </Order>
    </Split>

    <Split>
      <OriginalPhrase> frase2 </OriginalPhrase>
      <SplitPhrase> resultado2 </SplitPhrase>
      <SuggestedSplit> … </SuggestedSplit>
      <Edited> … </Edited>
      <Order> … </Order>
    </Split>
  </SyntacticLog>
```

En este ejemplo concreto, existirían dos frases que han sido simplificadas, *frase1* y *frase2*. La primera de ellas habría sido sustituida por *resultado1* y la segunda por *resultado2*, como se puede observar se crea un elemento *Split* para cada una de las frases.

6.4 Diagrama de arquitectura

La organización interna del plugin Split es la que se muestra en la ilustración 46, en la que podemos observar que se dispone de una clase SplitPlugin que es necesaria para poder desarrollar un plugin para jEdit, tiene que heredar por requerimiento de jEdit de la clase EditPlugin. También consta de una simplificación de la herramienta de resaltado como es el caso de HighlightPluginSplit. La clase Split es la encargada de gestionar la interfaz del plugin además de la gestión de la herramienta de Procesamiento de Lenguaje Natural. Finalmente se dispone de un gestor de cambios a nivel léxico, estos cambios se ven reflejados en el log físico.

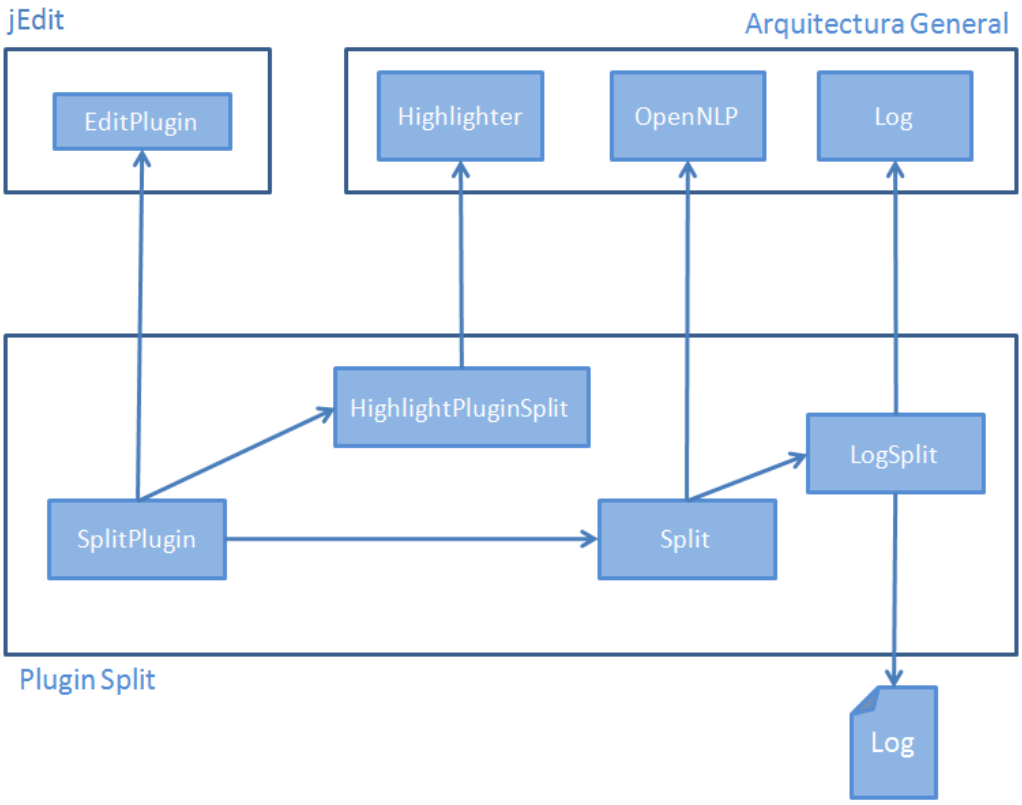

ILUSTRACIÓN 46: Diagrama de arquitectura del plugin Split

Capítulo 7

Conclusiones y Trabajo Futuro

Capítulo 7 – Conclusiones y Trabajo Futuro

Finalizado el proyecto podemos comentar las conclusiones a las que hemos llegado y el trabajo que esperamos se desarrolle en un futuro.

7.1 Conclusiones

Una vez realizado el estudio de las necesidades de los expertos en el momento de realizar la simplificación de textos, de las herramientas existentes que facilitan esta labor, de las tecnologías de Procesamiento de Lenguaje Natural y de las directrices existentes para la lectura fácil, se ha comprendido la importancia del desarrollo de una herramienta de apoyo para los expertos y así facilitar la labor de la simplificación de textos dado que son pocos los avances que se han realizado en este área. Debido a que las capacidades de comprensión lectora varían de una persona a otra fue difícil decidir con qué funcionalidades debían de contar las herramientas que íbamos a desarrollar y finalmente concluimos que eran necesarias dos tipos de simplificaciones: una a nivel léxico y otra a nivel sintáctico, de tal manera que se desarrolló una herramienta para la simplificación léxica (Synonyms) y otra para la simplificación sintáctica (Split).

Para conseguir diseñar e implementar una arquitectura que permitiese integrar las herramientas de PLN existentes se realizó un estudio de distintos editores de texto. Para ello tuvimos en cuenta que necesitábamos un editor que cumpliese dos características: que estuviese escrito en Java y que fuese un software de código libre y multiplataforma. Contemplando estas dos características, jEdit fue el editor de texto que mejor se adaptó a nuestras necesidades, ya que además de cumplirlas permitía de una forma simple añadir funcionalidades al mismo mediante el desarrollo de plugins y de esta manera las herramientas de simplificación léxica y sintáctica se añadirían como plugins de jEdit. El proceso de familiarización con el entorno de jEdit y de lo necesario para implementar los plugins que permitieran añadir funcionalidades al mismo se realizó basándonos en otros plugins desarrollados y mediante pruebas sucesivas en las que íbamos incluyendo las herramientas de PLN de las que disponíamos, tras las cuales íbamos adquiriendo nuevos conocimientos que finalmente nos permitieron desarrollar completamente la herramientas léxica y la herramienta sintáctica.

En cuanto al desarrollo de la herramienta léxica y la herramienta sintáctica, se necesitaba disponer de una interfaz que fuese fácil de usar para el experto. Por ello se decidió reutilizar y simplificar un plugin existente para jEdit que hacía más amigable la visualización de nuestros plugins.

Finalmente después de todo el esfuerzo realizado documentándonos sobre todas las herramientas existentes que ayudaran a desarrollar el proyecto se ha conseguido diseñar e implementar una arquitectura que permite integrar las herramientas de PLN para su uso en un editor de texto ya existente, incluyendo en ella funcionalidades para conseguir en un futuro automatizar el proceso de simplificación, que se contemplan en el log. Esta arquitectura puede

ser extendida de una manera sencilla con otras herramientas de simplificación que se implementen en el futuro.

Además todo el proyecto está publicado en SourceForge como software libre multiplataforma para que pueda ser usado por todo aquel que desee utilizar y/o mejorar la herramienta. Se puede encontrar en las páginas https://sourceforge.net/projects/synonymsp/ y https://sourceforge.net/projects/splitp/ .

A la hora del término de esta memoria la publicación de los plugins en la página oficial de jEdit se encontraba en proceso.

7.2 Trabajo Futuro

El campo de la lectura fácil es un campo muy amplio donde existen multitud de herramientas poco accesibles a los expertos que simplifican los textos. Con el fin de ampliar las herramientas desarrolladas en este proyecto se puede pensar en un trabajo futuro que contemple distintas propuestas.

En el ámbito léxico podemos pensar en ampliar la búsqueda de sinónimos para adjetivos y verbos en distintos tiempos verbales. El diccionario del que disponemos en este proyecto no cuenta con una base de datos para adjetivos ni para distintos tiempos verbales. Para poder trabajar con adjetivos será necesario incorporar un nuevo diccionario, mientras que para los verbos existen herramientas que permiten la transformación de éstos a otro tiempo verbal.

Otra ampliación léxica que se puede realizar es la simplificación de expresiones en vez de trabajar con la unidad mínima de la palabra.

Sintácticamente se puede contemplar la eliminación de palabras, expresiones o complementos circunstanciales que no aporten información necesaria al texto. También podemos pensar en separar las frases, volviendo al concepto de una sola idea por oración, que contengan otro tipo de conjunciones distintas de "*and*". Además se puede incluir una funcionalidad que analice todo el texto y sugiera qué oraciones podrían ser simplificadas.

Aprovechando los resultados almacenados en el log se podría crear una base de datos que recuerde y almacene las distintas simplificaciones que se han realizado de tal forma que si se reconoce en otro texto frases o palabras similares se sugiera al usuario esa simplificación.

El proceso de simplificación de un texto es una tarea ardua en la que hay que tener en cuenta muchos aspectos desde el cliente final del texto hasta el tipo de texto que se simplifica. No podemos olvidar que el objetivo futuro de todo el trabajo que se realiza en este campo es la automatización de todo el proceso de simplificación.

Anexos

Anexo A: OpenNLP

Para poder explicar las distintas funcionalidades de OpenNLP utilizadas se presenta un ejemplo para cada una de ellas.

Dado el siguiente texto:

> *A boy was riding a bike. He was also eating an apple. The road was empty.*

Si aplicamos la detección de frases de OpenNLP resultan las siguientes frases:

> *A boy was riding a bike.*
> *He was also eating an apple.*
> *The road was empty.*

También podemos obtener el offset donde comienzan y donde terminan cada frase del texto. En el ejemplo anterior:

> 0..24 - *A boy was riding a bike*.
> 25..53 - *He was also eating an apple.*
> 54..73 - *The road was empty*.

Dada la siguiente frase:

> *A boy was riding a bike while he was eating an apple.*

Si aplicamos el detector de tokens obtenemos, al igual que con el detector de frases, tanto los tokens como el offset donde comienzan y terminan:

A	0..1 - *A*
boy	2..5 - *boy*
was	6..9 - *was*
riding	10..16 - *riding*
a	17..18 - *a*
bike	19..23 - *bike*
while	24..29 - *while*
he	30..32 - *he*
was	33..36 - *was*
eating	37..43 - *eating*
an	44..46 - *an*
apple	47..52 - *apple*
.	52..53 - .

Aplicando el etiquetado a la misma frase se obtiene:

A (DT)	Determinante
boy (NN)	Sustantivo singular
was (VBD)	Verbo en tiempo pasado
riding (VBG)	Verbo en gerundio o en presente del participio
a (DT)	Determinante
bike (NN)	Sustantivo singular
while (IN)	Preposición o conjunción subordinada
he (PRP)	Pronombre personal
was (VBD)	Verbo en tiempo pasado
eating (VBG)	Verbo en gerundio o en presente del participio
an (DT)	Determinante
apple (NN)	Sustantivo singular
. (.)	Sentencia de terminación

Aplicando el parseo a la frase se obtiene:

(TOP	raíz
(S	sujeto
(NP	nombre propio en singular
(DT *A*)	determinante
(NN *boy*)	sustantivo singular
)	
(VP	verbo de la frase
(VBD *was*)	verbo en pasado
(VP	verbo de la frase
(VBG *riding*)	verbo en gerundio o presente del participio
(NP	nombre propio en singular
(DT *a*)	determinante
(NN *bike*)	sustantivo singular
)	
(SBAR	cláusula introducida para la conjunción subordinada
(IN *while*)	preposición o conjunción subordinada
(S	sujeto
(NP	nombre propio en singular
(PRP *he*)	pronombre personal
)	
(VP	verbo de la frase
(VBD *was*)	verbo en pasado
(VP	verbo de la frase
(VBG *eating*)	verbo en gerundio o presente del participio
(NP	nombre propio en singular
(DT *an*)	determinante
(NN *apple.*)	sustantivo singular
)	

```
                    )
                  )
                )
              )
            )
          )
        )
      )
    )
)
```

Anexo B: Textos Simplificados

La simplificación de textos que han realizado cuatro personas nativas de habla inglesa y que han servido para realizar el estudio explicado en el capítulo 4 son los siguientes:

Texto Original	*Texto Simplificado*
Meanwhile, attention focused on the captain, who was spotted by Coast Guard officials and passengers fleeing the scene even as the chaotic and terrifying evacuation was under way.	Everyone thought about the captain, who was seen by boat workers and people on the ship, leaving the accident during the crazy and scary rescue.
The ship's Italian owner, a subsidiary of Carnival Cruise lines, issued a statement late Sunday saying there appeared to be "significant human error" on the part of the captain, Francesco Schettino, "which resulted in these grave consequences."	The owner of the ship, part of the company called Carnival Cruise lines, said on Sunday night that it seems there was “a lot of things done wrong “ by the captain, named Francesco Schettino, “which is why these terrible things happened.”
Authorities were holding Schettino for suspected manslaughter and a prosecutor confirmed Sunday they were also investigating allegations the captain abandoned the stricken liner before all the passengers had escaped. According to the Italian navigation code, a captain who abandons a ship in danger can face up to 12 years in prison.	Officers had Schettino in jail for possibly killing people and the lawyer against him said Sunday that they are going to find out if the captain left the damaged ship before all the people on the ship had left. The rules of the Italian boat workers says that a captain who leaves a ship in a problem can go to jail for as many as 12 years.

Texto Original	*Texto Simplificado*
An Indonesian girl swept away in the 2004 tsunami has been reunited with her parents seven years on, the family say.	In 2004 a tsunami swept away an Indonesian girl. She was separated from her family. After seven years her family says that she was reunited with them.
Meri Yulanda turned up at a cafe in Meulaboh, Aceh, remembering only the name of her grandfather, Ibrahim.	Her name was Meri Yulanda and recently she appeared at a cafe in Meulaboh, Aceh, and could only remember the name of her grandfather, Ibrahim.
Someone at the cafe knew a man by that name and got in touch.	Someone who knew her grandfather in the cafe called him.

Meri says she was beaten and forced to work as a beggar before being set free last week.

Meri says she was beaten and forced to work as a beggar before being set free last week.

Texto Original

If you are a Football Association bigwig or a Tottenham fan brace yourself for an uncomfortable statistic. Since Fabio Capello resigned as England coach, Spurs have picked up only 0.8 points per game. Before the Italian left these shores - and the Tottenham manager Harry Redknapp became a shoo-in to replace him - Spurs had been averaging 2.1 points per game. That impressive points average had the media salivating over the style and swagger of Redknapp's side as Spurs briefly threatened to break up the Mancunian duopoly of the Premier League title race. Even when Spurs's title bid faltered, Redknapp's side still looked strong favourites to finish third, even more so when Tottenham led Arsenal 2-0 just before half-time at Emirates Stadium on 26 February. We all know what happened next. Spurs shipped five goals to lose that game 5-2 and have managed to pick up just one point in 12, while they only narrowly avoided a fourth successive league defeat on Wednesday, grabbing a late equaliser against Stoke.

Texto Simplificado

For both powerful decision makers at the Football Association and Tottenham supporters it is concerning that since Fabio Capello resigned as England Coach Spurs have averaged just 0.8 points per game played. Before this and the strong rumour connecting Harry Redknapp to the England job started, Spurs had averaged 2.1 points per game. Even after Spurs lost the chance to challenge for the title it looked very likely that they would finish third especially at the point they were leading Arsenal 2 v 0 just before half time away at Arsenal. However Spurs eventually lost that game 5 v 2 and have only achieved 1 point in the last 4 games, and almost lost again on Wednesday, only gaining a draw with Stoke with a very late goal.

Texto Original

Five beautiful mermaids were living in the depths of bluest ocean. The little mermaid who was the youngest and have a wonderful voice spent her time thinking on the surface, the world of the men.

- “Oh!!! How I wish I could go to the surface and see the sky!!!

Texto Simplificado

Five beautiful mermaids lived in the bluest ocean. The little mermaid was the youngest and had a wonderful voice. She spent her time thinking about the world of the men.

- “Oh!!! How I wish I could go to the surface and see the sky!!!

- “You are still too young“, her grandmother used to say.

When she grew up older her father said:

- “Well, now you can go to the surface to breathe the air and see the sky! But remember that the upper world is not ours; we are only to admire it! We are children of the sea and we don’t have our soul like men up there. Be careful and don’t come too close to them or you will regret it!”

And the little mermaid emerged to the surface.

But a ship came by the reef where she was laying. On board, the activity of the men made believe they were strangely possessed by something.

And the mermaid saw a young man. She could not help looking at him. She was not familiar with this feeling of joy but of suffering at the same time; a feeling that forced upon heart.

- "... therefore you want to get rid off you fishtale!! And I supossed you want two legs. You will have to suffer horribly and everytime you step on the ground you will feel a terrible pain. You will have to give me your wonderful voice and you will remain dumb forever! And remember, if the man you love is ever to marry someone else your body will vanish in the water as the spume of a wave."
- "I accept" were the last words she said.

She swam toward the beach and went onto the surface.

- “You are still too young“, her grandma used to say.

When she got older her father said:

- “Well, now you can go to the surface to breathe the air and see the sky! But remember that the upper world is not ours. We are only to admire it! We are children of the sea and we don’t have souls like the men up there. Be careful and don’t get too close to them or you will be sorry!”

And the little mermaid went up to the surface.

But a ship came by the reef where she was laying. On the ship, the way the men acted made her believe they were strangely possessed by something.

The mermaid saw a young man. She could not help looking at him. She had not known this feeling of joy and sadness at the same time; a feeling that came into her heart.

- "... you want to get rid of your fishtale!! And I guess you want two legs. You will go through lots of bad things and everytime you step on to the ground you will feel a terrible pain. You will have to give me your wonderful voice. You will stay dumb forever! And remember, if the man you love ever marries someone else your body will disappear in the water like the foam of a wave."
- "I accept" were the last words she said.

She swam toward the beach and went onto the surface.

She felt such a strong pain that she felt into a faint. When she wakes up, she saw this young man she loved looking at her, smiling.

- "I will take you to the castle and look after you".

One day a big ship could be seen from the tower of the castle. As it was coming closer the prince decided to welcome the crew.

The stranger who went off the ship was actually the girl the prince held in his heart.

She felt such a strong pain that she fainted. When she woke up, she saw the young man she loved looking at her, smiling.

- “I will take you to the castle and look after you".

One day they saw a big ship from the tower of the castle. As it came closer the prince decided to welcome the crew.

The stranger that left in the ship was the girl the prince held in his heart.

Anexo C: Log

El log es una parte muy importante de nuestro proyecto sobre todo de cara al futuro. En él se muestran todos los cambios que se realizan sobre un documento gracias a los plugins. A continuación se muestra un ejemplo real de log que se ha conseguido gracias a los cambios que se han generado del último de los textos que se muestran en el anexo B.

```
<Changes>
  <Counter>5</Counter>
  <SynonymsLog>
    <OriginalWord word="remain">
      <Synonym word="stay">
        <Substitution>
          <OriginalPhrase>
                You will remain dumb forever.
          </OriginalPhrase>
          <NewPhrase>
                You will stay dumb forever.
          </NewPhrase>
          <SuggestedSynonyms>stay,rest,remain</SuggestedSynonyms>
          <Edited>No</Edited>
          <Order>5</Order>
          <Action>replace</Action>
        </Substitution>
      </Synonym>
    </OriginalWord>
    <OriginalWord word="regret">
      <Synonym word="be sorry">
        <Substitution>
          <OriginalPhrase>
                Be careful and don’t come too close to them or you
                will regret it!”
          </OriginalPhrase>
          <NewPhrase>
                Be careful and don’t come too close to them or you
                will be sorry it!”
          </NewPhrase>
          <SuggestedSynonyms>
                rue,regret,ruefulness,sorrow
          </SuggestedSynonyms>
          <Edited>Yes</Edited>
          <Order>4</Order>
          <Action>replace</Action>
        </Substitution>
      </Synonym>
    </OriginalWord>
    <OriginalWord word="grandmother">
      <Synonym word="grandma">
        <Substitution>
          <OriginalPhrase>
```

```
                - "You are still too young", her grandmother used to
                say.
            </OriginalPhrase>
            <NewPhrase>
                 - "You are still too young", her grandma used to
                say.
            </NewPhrase>
            <SuggestedSynonyms>
                granny,grandma,gran,grandmother,grannie
            </SuggestedSynonyms>
            <Edited>No</Edited>
            <Order>3</Order>
            <Action>replace</Action>
          </Substitution>
        </Synonym>
      </OriginalWord>
    </SynonymsLog>

    <SyntacticLog>
      <Split>
        <OriginalPhrase>
          We are children of the sea and we don't have our soul like
          men up there.
        </OriginalPhrase>
        <SplitPhrase>
          We are children of the sea. We don't have our soul like men
          up there.
        </SplitPhrase>
        <SuggestedSplit>
          We are children of the sea. We don't have our soul like men
          up there.
        </SuggestedSplit>
        <Edited>No</Edited>
         <Order>1</Order>
      </Split>
      <Split>
        <OriginalPhrase>
          You will have to give me your wonderful voice and
          you will remain dumb forever!
        </OriginalPhrase>
        <SplitPhrase>
          You will have to give me your wonderful voice. You will
          remain dumb forever.
        </SplitPhrase>
        <SuggestedSplit>
          You will have to give me your wonderful voice. You will
          remain dumb forever.
        </SuggestedSplit>
        <Edited>No</Edited>
        <Order>2</Order>
      </Split>
    </SyntacticLog>
  </Changes>
```

Bibliografía y Referencias

Bibliografía y Referencias

AbiSource Community. *AbiWord.* Abril de 2002. http://www.abisource.com/ (último acceso: 20 de Junio de 2012).

Aluísio, Sandra M. *Proyecto PorSimples.* Noviembre de 2007. http://caravelas.icmc.usp.br/wiki/index.php/Principal (último acceso: 6 de Marzo de 2012).

Associació Lectura Fàcil. *Asociación Lectura Fácil.* 2002. http://www.lecturafacil.net/content-management-es/ (último acceso: 20 de Junio de 2012).

Educational Facilita. http://caravelas.icmc.usp.br:3003/facilita/presentation (último acceso: 6 de Marzo de 2012).

Freyhoff, Geert, Gerhard Hess, Linda Kerr, Elizabeth Menzel, Bror Tronbacke, y Kathy Van Der Veken. «El Camino Más Fácil.» Junio de 1998. http://www.uc3m.es/portal/page/portal/orientacion_personal_participacion/PIED1/lectura_facil/documentos/ManualdeLecturaFacil.pdf (último acceso: 20 de Junio de 2012).

Fundación Sidar. *Seminario Iberoamericano sobre Discapacidad y Accesibilidad en la Red.* 1997. http://www.sidar.org/ (último acceso: 20 de Junio de 2012).

Ho, Don. *Notepad++.* http://notepad-plus-plus.org/ (último acceso: 20 de Junio de 2012).

IFLA. *IFLA.* 1971. http://www.ifla.org/ (último acceso: 20 de Junio de 2012).

Inclusion Europe aisbl. *Inclusion Europe.* 1988. http://www.inclusion-europe.org/ (último acceso: 20 de Junio de 2012).

Inclusion International. *Inclusion International.* http://www.inclusion-international.org/?lang=es&lang=es (último acceso: 20 de Junio de 2012).

JDOM Project. *JDom.* 2000. http://www.jdom.org/downloads/index.html (último acceso: 20 de Junio de 2012).

KOffice Community. *KWord.* http://www.koffice.org/kword/ (último acceso: 20 de Junio de 2012).

Microsoft Corporation. *Microsoft Word.* http://office.microsoft.com/es-es/word/ (último acceso: 20 de Junio de 2012).

Miller, G.A. «WordNet: a lexical database for English.» *Communications of the ACM* 38, nº 11 (1995): 39-41.

Northwestern University. *MorphAdorner.* 2007. http://morphadorner.northwestern.edu/morphadorner/ (último acceso: 20 de Junio de 2012).

OpenNLP Community. *OpenNLP.* http://sourceforge.net/apps/mediawiki/opennlp/index.php?title=Main_Page#OpenNLP_Documentation (último acceso: 20 de Junio de 2012).

Pestov, Slava. *jEdit.* http://www.jedit.org/ (último acceso: 20 de Junio de 2012).

PorSimples. *Simplifica.* 2009. http://nilc.icmc.usp.br/porsimples/simplifica/ (último acceso: 20 de Junio de 2012).

Seco, N. *LightWeight WordNet.* 2005. http://eden.dei.uc.pt/~nseco/lwwn.tar.gz (último acceso: 20 de Junio de 2012).

Serra Milà, Jaume. *La fletxa negra.* 2007. http://xtec.cat/~jserra18/FNLF.pdf (último acceso: 20 de Junio de 2012).

Serra Milà, Jaume. *La lectura fàcil: una necessitat per a la inclusió de l'alumnat nouvingut d'ESO.* 2007. http://www.xtec.es/sgfp/llicencies/200708/memories/1740m.pdf (último acceso: 20 de Junio de 2012).

The Apache Software Foundation. *OpenOffice Writer.* 2000. http://about.openoffice.org/index.html (último acceso: 20 de Junio de 2012).

The Document Foundation. *LibreOffice Writer.* Septiembre de 2000. http://www.libreoffice.org/features/writer/ (último acceso: 20 de Junio de 2012).

Vive la fácil lectura. http://www.facillectura.es/ (último acceso: 20 de Junio de 2012).