eman ta zabal
zazu

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA INDUSTRIAL DE BILBAO

GRADO EN INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2014 / 2015

EDITOR DE TEXTOS CON CORRECTOR ORTOGRÁFICO PARA TEXTOS MÉDICOS

MEMORIA

DATOS DE LA ALUMNA O DEL ALUMNO	**DATOS DEL DIRECTOR O DE LA DIRECTORA**
NOMBRE: RAÚL	NOMBRE: KOLDO
APELLIDOS: MERINO TORRE	APELLIDOS: GOJENOLA GALLETEBEITIA
	DEPARTAMENTO: LENGUAJES Y SISTEMAS INFORMÁTICOS
FDO.:	FDO.:
FECHA:19-06-2015	FECHA:19-06-2015

Índice de capítulos

Índice de imágenes

Índice de tablas

Índice de figuras

GLOSARIO

DISTANCIA DE EDICIÓN. Se denomina distancia de edición o distancia de Levenshtein al número mínimo de operaciones necesarias para convertir una cadena de caracteres en otra. A continuación se muestran las operaciones.

· Inserción de un carácter: casa->cadsa

· Sustitución de un caracter: casa->cafa

· Eliminación de un carácter: casa->caa

La distancia de Damerau-Levenshtein considera la transposición, intercambio de dos caracteres, como una operación.

· Transposición: casa->caas

Por ejemplo la distancia de edición entre la palabra casa y cazar es de 2.

· Sustitución del carácter s por z: casa->caza

· Inserción del carácter r: caza->cazar

Como se puede comprobar, a través de solo dos operaciones conseguimos transformar una palabra en otra.

N-GRAMA. Es una subsecuencia de elementos dentro de una secuencia dada. Los n-gramas pueden estar formados en base a distintos tipos de elementos como fonemas, sílabas, pares de caracteres o palabras. Dependiendo del número de elementos que lo conforman se les puede denominar de maneras especiales, unigramas, bigramas o digramas y trigramas.

LEMA. Es la raíz o parte de la palabra básico, a través del cual se puede formar un conjunto de palabras con la adición de sufijos o prefijos. Por ejemplo, gato, gatos y gata son palabras con el mismo lema.

TIPO. Cada uno de los elementos que forman el vocabulario de una lengua.

TOKEN. Se trata de cada una de las instancias de un tipo de un texto dado

VOCABULARIO. Es el conjunto de tipos, de elementos, que componen una lengua. O en su defecto, la mayor cantidad posible de estos.

DICCIONARIO. Es la estructura de datos en la que ese encuentra almacenado todo el vocabulario, utilizado en consultas.

1. INTRODUCCION

1.1. Motivación

Actualmente no existe una herramienta de referencia orientada a la corrección de textos en el ámbito de la medicina en lengua castellana, que haga uso de un diccionario completo de términos médicos que cubra un rango amplio de palabras técnicas de la rama médica. Las herramientas de edición de textos cuentan en su gran mayoría con la posibilidad de añadir término a término palabras nuevas a sus diccionarios pero no cuentan con un diccionario inicial potente. De esta necesidad surge la idea de este proyecto, cuyo objetivo es crear una herramienta de edición de textos con un corrector adaptado a las necesidades de los profesionales médicos y sanitarios.

1.2. Propósito

El propósito de este proyecto consiste en la implementación de un editor de textos que cuente con un corrector ortográfico adaptado a las necesidades del lenguaje propio de la medicina. Este corrector deberá cubrir la mayor cantidad de palabras técnicas posible y se apoyará en la creación de un modelo de lenguaje propio para poder suministrar al usuario una lista de posibles palabras correctas coherente ante un error ortográfico.

1.3. Ámbito

Este proyecto se ha realizado haciendo uso del entorno de programación Eclipse. Para su desarrollo se ha utilizado lenguaje Java así como algunas librerías de código abierto disponibles de manera pública en la red, como el detector de decodificación *juniversalchardet* y las librerías para la creación de interfaces *JavaSwing* y *AWT*.

2. PLANTEAMIENTO INICIAL

2.1 Objetivos principales

El principal objetivo de este proyecto es el desarrollo de un software capaz de detectar errores ortográficos y sugerir una lista de posibles soluciones. Dicha lista no deberá ser muy extensa y mostrará las palabras que presenten una mayor probabilidad de ser la palabra correcta deseada por el usuario. Este corrector se ubicará dentro de una aplicación comúnmente conocida como editor de textos que permitirá al usuario la creación, corrección, impresión y guardado de archivos de texto plano.

En esencia, se trata de la clásica herramienta de edición y manipulación de texto plano en la que prima la funcionalidad sobre la apariencia. Algunas de las funciones por las que se conoce a estos editores son:

· Marcar región

· Búsqueda y reemplazo

· Copiar, cortar y pegar

· Deshacer y rehacer

· Importar

El aspecto más importante de la aplicación es el desarrollo de un corrector específico para lenguaje perteneciente al área médica y sanitaria. Incluyendo un vocabulario confeccionado especialmente para la herramienta, así como un sistema de detección y corrección de errores implementado únicamente para este proyecto.

2.2. Arquitectura

Para el desarrollo de este proyecto se utilizará una arquitectura Modelo-Vista-Controlador. La herramienta desarrollada se trata de una aplicación de escritorio, la cual hace uso de una parte gráfica que es la denominada Vista. Esta parte visual es aquella que el usuario visualiza y manipula. La parte gráfica necesita de una parte lógica, denominada Modelo, la cual se encarga de realizar los procesos necesarios para la corrección y creación de sugerencias de las palabras erróneas, así como la corrección automática de textos. Para poder coordinar el funcionamiento de ambas partes es necesario la creación de un tercer componente encargado de mantener la vista y el controlador independientes entre sí actuando él como intermediario

entre ambas. De esta manera la parte visual permanece en todo momento separada de la parte lógica.

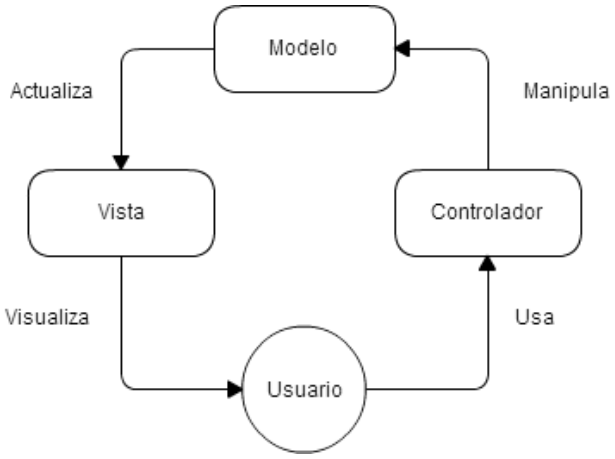

Imagen 1. Modelo-Vista-Controlador

2.3. Alcance

El ciclo de vida empleado para la realización del proyecto es de tipo incremental. Esta elección se produce debido a que se trata de un ciclo en el que se van generando versiones cada vez más completas, las funcionalidades se van ampliando en cada uno de los prototipos. Tras cada etapa del proceso se van satisfaciendo cada vez más requisitos.

Para la mejor comprensión de las tareas abarcadas en este proyecto se hará uso de una Estructura de Descomposición del Trabajo (EDT). Esta herramienta ayuda a la gestión del proyecto reflejando las distintas acciones necesarias de manera ordenada y descendente.

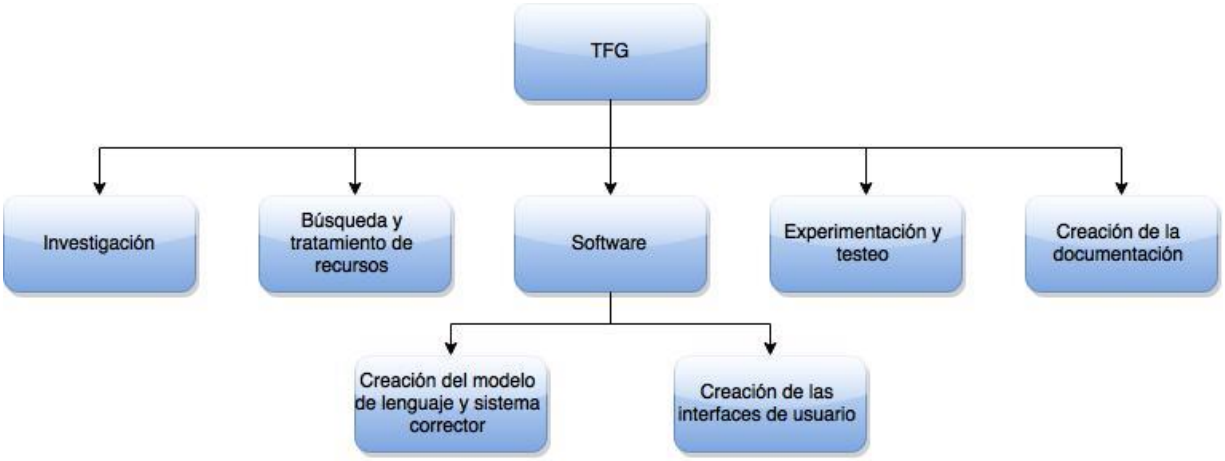

Imagen 2. EDT – Estructura de descomposición de trabajo

A continuación se detalla cada una de las tareas necesarias para la realización del proyecto, así como las entradas necesarias para la ejecución de cada una de ellas, las salidas obtenidas, una breve descripción y la duración estimada de la actividad.

Investigación.

Duración estimada: 70 horas.

Descripción: Realizar los trabajos de investigación y documentación necesarios para saber abordar los problemas que presenta el proyecto. Recoger Información y valorar trabajos anteriores realizados en el ámbito del proyecto en los que poder apoyarse para la toma de decisiones.

Entradas: Artículos, documentos e investigaciones realizadas en el ámbito en el que se enmarca el proyecto.

Salidas/Entregables: Esquemas y documentos con la información más destacada que servirán de apoyo para la elaboración del proyecto.

Recursos necesarios: Un equipo informático, navegador de internet, editor de textos Open Office.

Precedencias: Ninguna.

Búsqueda y tratamiento de recursos.

Duración estimada: 50 horas.

Descripción: Realizar los trabajos de búsqueda de los recursos necesarios para la creación de las herramientas. Debido a la naturaleza del proyecto es necesario la búsqueda de recursos para la creación del corpus, así como para la creación de un diccionario lo más completo posible.

Entradas: Esquemas y documentación realizada en la etapa de investigación, particularmente en el ámbito de los modelos de lenguaje.

Salidas/Entregables: Archivo de texto con la colección de palabras que componen el diccionario y una larga colección de textos destinados para procesamiento.

Recursos necesarios: Un equipo informático, navegador de internet.

Precedencias: Tarea de investigación finalizada.

En la EDT se indica que la tarea de software se descompone en dos sub-tareas bien diferenciadas: la creación del modelo de lenguaje y la creación de las interfaces de usuario. Se ha determinado que se debe realizar en primer lugar la creación del modelo de lenguaje ya que es la parte más compleja del proyecto y la que puede presentar mayor cantidad de problemas en su realización. Se ha considerado que la creación de la interfaz no supondrá tanta dificultad debido a que es un ámbito en el que el equipo humano acumula cierta experiencia.

Creación del modelo de lenguaje y sistema corrector.

Duración estimada: 90 horas.

Descripción: Realizar la creación del modelo de lenguaje. Tratamiento y procesamiento del corpus de texto conseguido en la etapa anterior para la creación del modelo de lenguaje. Realizar la comprobación de que la creación y almacenamiento del modelo de lenguaje se realiza correctamente.

Entradas: Corpus de textos.

Salidas/Entregables: Modelo del lenguaje.

Recursos necesarios: Un equipo informático, editor de textos Open Office, entorno de desarrollo Eclipse.

Precedencias: Tarea de búsqueda y tratamiento de recursos comenzada.

Creación de las interfaces de usuario.

Duración estimada: 80 horas.

Descripción: Diseño y creación de las interfaces que componen la aplicación. Desarrollo de todas y cada una de las funcionalidades que componen el editor de texto. Comprobación de todas y cada una de las funcionalidades de la aplicación y su correcto funcionamiento.

Entradas: Modelo de lenguaje accesible y utilizable.

Salidas/Entregables: Interfaces de usuario.

Recursos necesarios: Un equipo informático, plugin GUI para el entorno de desarrollo Eclipse como herramienta de maquetación, editor de textos Open Office, entorno de desarrollo Eclipse.

Precedencias: Tarea de creación del modelo de lenguaje finalizada.

Experimentación y testeo.

Duración estimada: 80 horas.

Descripción: Experimentación y realización de pruebas del correcto funcionamiento de la aplicación. Durante este proceso también se contempla la solución de estos y su mejora.

Entradas: Modelo de la aplicación plenamente operativo.

Salidas/Entregables: Aplicación finalizada.

Recursos necesarios: Un equipo informático, plugin GUI para el entorno de desarrollo Eclipse como herramienta de maquetación, editor de textos Open Office, entorno de desarrollo Eclipse.

Precedencias: Tarea de creación de las interfaces y del modelo de lenguaje finalizada.

Creación de la documentación.

Duración estimada: 80 horas.

Descripción: Redacción de la memoria del trabajo llevado a cabo y documentación completa del software producido. También se contempla la redacción de la hoja de viabilidad, así como el resumen del proyecto.

Entradas: Artículos, documentos e investigaciones realizadas en el ámbito en el que se enmarca el proyecto.

Salidas/Entregables: Memoria del proyecto.

Recursos necesarios: Un equipo informático, editor de textos Open Office, Visual Paradigm como herramienta de creación de modelos de dominio y herramienta de creación de diagramas de flujo.

Precedencias: Tarea de investigación en un punto avanzado.

2.4. Planificación Temporal

Una vez realizada la descomposición de las tareas necesarias para llevar a cabo el proyecto se procede a detallar la planificación temporal. Se debe recalcar que se trata de una estimación de las horas a ser empleadas en cada tarea que en la realidad no se ha visto satisfecha como se muestra en el apartado de conclusiones.

Esta planificación se ve representada mediante un diagrama de Gantt semanal en siguiente figura. Se ha estimado que la dedicación para la realización del proyecto será de 4 horas diarias. La dedicación semanal estimada será de 5 días por semana, lo que supone 20 horas de trabajo semanales aproximadamente.

Recalcar que la creación de la memoria no se ha de producir de manera continuada. De hecho se plantea realizar una parte de la memoria al comienzo y durante el proceso de investigación, y posponer el resto para cuando el software se encuentre prácticamente desarrollado. Dentro de la etiqueta de creación de lo memoria también se engloban las tareas de creación de la hoja de viabilidad y el resumen del proyecto.

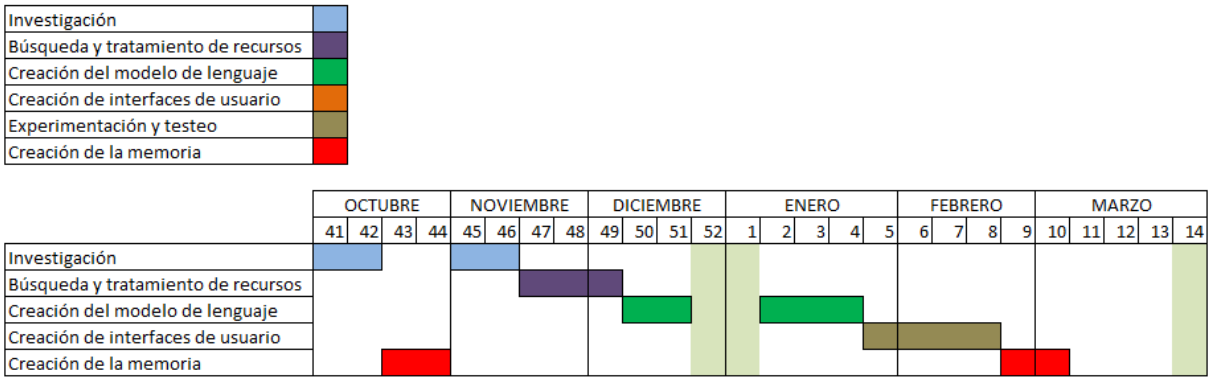

Imagen 3. Diagrama de Gantt, planificación temporal inicial del proyecto.

Según las estimaciones iniciales el proyecto debería estar terminado para la primera semana de mayo. Considerando como comienzo del mismo la primera semana de octubre lo que supone una duración aproximada de 20 semanas, en las que se ha de tener en cuenta que se observan dos semanas libres durante las fiestas navideñas.

2.5. Herramientas

Eclipse: Entorno de programación utilizado para el desarrollo del software en lenguaje Java, más información disponible sobre Eclipse y Java en la bibliografía.

Dropbox: Servicio gratuito para el almacenamiento de archivos online, más información disponible en el apartado de bibliografía.

Visual Paradigm: herramienta utilizada para la creación y diseño de diagramas de representación, para más información consultar la bibliografía.

Draw.i.o: Herramienta de diseño de diagramas online de uso libre utilizada para realizar algunos de los gráficos aparecidos en la memoria.

Libre Office Writer: herramienta utilizada para el procesamiento de textos.

Microsoft Office Excel: herramienta de tratamiento de hojas de cálculo, utilizada para la creación de documentación y tablas.

Cacoo: herramienta de diseño online para la creación de algunos de los diagramas aparecidos en la memoria.

2.6. Gestión de riesgos

En este capítulo de muestran los riesgos contemplados que se sufren en la realización de este proyecto. Se realiza una valoración de la probabilidad y el daño causado en caso de producirse. También se contemplan las vías para evitar estos riesgos y un plan de actuación en caso de que se sufra uno de estos problemas.

Pérdida o corrupción del corpus.

Descripción: Pérdida del corpus causado por daño, pérdida o virus.

Prevención: Realización de copias de seguridad tanto en la nube como en soportes físicos externos.

Plan de contingencia: Realizar el volcado de una de las copias de seguridad previamente realizadas.

Probabilidad: Bastante improbable.

Impacto: En caso de producirse, el impacto no será crítico ya que el corpus se encuentra disponible en la red.

Pérdida del software desarrollado.

Descripción: Pérdida del software desarrollado para la creación de la aplicación por daño, pérdida o virus.

Prevención: Realización de copias de seguridad con regularidad, tanto en la nube como en soportes físicos externos.

Plan de contingencia: Realizar el volcado de una de las copias de seguridad previamente realizadas.

Probabilidad: Bastante improbable.

Impacto: En caso de producirse, el impacto variará en función del volumen de datos perdidos, es decir, de los avances producidos después del último volcado.

Mala Planificación.

Descripción: Ya que la duración del proyecto es bastante larga, es conveniente tener en cuenta que la planificación realizada inicialmente puede ser errónea ya que no se contemplan posibles complicaciones que retrasen la buena marcha del proyecto. Problemas no contemplados que pueden demorar la ejecución de alguna de las fases.

Prevención: Se intentará realizar un cálculo lo más aproximado posible del tiempo que nos llevará a cabo la realización de cada tarea en base a la experiencia adquirida de proyectos previos similares.

Plan de contingencia: En caso de producirse un error en la planificación habrá que realizar un reajuste de ésta desde el instante en el que se detecta el error hasta la finalización del proyecto.

Probabilidad: Muy probable.

Impacto: En caso de producirse, el impacto variará en función de la gravedad del error producido.

Pérdida del equipo.

Descripción: Pérdida del equipo utilizado para la realización del proyecto por diferentes causas. Bien sea deterioro físico o error de funcionamiento, pérdida o robo.

Prevención: Se pondrá especial atención al cuidado exterior del equipo con fundas y carcasa adecuadas para su protección. Además, se hará un uso responsable del equipo, evitando la instalación de software malicioso que lo pueda dañar. Evidentemente se evitarán robos u olvidos.

Plan de contingencia: Se realizará el volcado de las copias de seguridad y se deberá adquirir un nuevo equipo para la realización del proyecto.

Probabilidad: Probable.

Impacto: El impacto de esta pérdida variará en función del volumen de datos perdidos y de su criticidad. Aunque el factor que determinará el impacto será si se trata de una pérdida transitoria o definitiva.

Pérdida del Servicio de Dropbox.

Descripción: Por algún motivo inesperado el servicio de almacenamiento en la nube utilizado podría verse interrumpido o cancelado.

Prevención: Para evitar la pérdida de los datos almacenados se descargará la aplicación del programa para asegurar que en caso de pérdida del servicio los archivos se encuentran almacenados en el equipo utilizado.

Plan de contingencia: En caso de pérdida se deberá buscar un servicio similar de almacenamiento gratuito, crear un perfil y cargar todos nuestros datos en la nueva aplicación.

Probabilidad: Poco probable.

Impacto: En caso de producirse el impacto no será muy elevado.

2.7. Evaluación económica

Teniendo en cuenta que no existe pretensión económica de ningún tipo por parte del autor ya que se trata de la realización del proyecto fin de grado, en este apartado se procede al análisis económico vinculado a la realización del mismo.

Para su elaboración se han establecido previamente los costes salariales:

· Cada hora empleada en trabajos de documentación supone un coste de 25€.

· Cada hora empleada en trabajos de diseño, análisis o implementación supone un coste de 35 €.

Así como los costes realizados equipos informáticos y licencias de software:

· Costes derivados de la adquisición de software.

· Costes derivados de la utilización de equipos informáticos.

Por último también se ha de tener en cuenta los gastos indirectos producidos como resultado de la elaboración de cualquier proyecto, como pueden ser el acceso a internet, suministro eléctrico, etc. En este caso estos costes se establecen en un 5% de los costes totales del proyecto.

Como costes derivados de la adquisición de software debemos tener en cuenta los costes de las licencias de uso y explotación de las herramientas necesarias para el desarrollo del proyecto. En nuestro caso estos gastos se elevan 348€. Si tenemos en cuenta que el período de amortización será de 3 años y que la duración de desarrollo del proyecto está programada para ser finalizado en 5 meses, supone un coste final de 65€.

Para el desarrollo de nuestra aplicación se hará uso de un equipo portátil Samsung valorado en 700€, al que se le asume una vida media útil de 5 años, de esta manera la amortización correspondiente se eleva a 97€.

Según la planificación temporal inicial se estima que se invertirán un total de 370 horas aproximadamente, de las cuales, aproximadamente 70 horas se invertirán en trabajos de investigación y las 300 horas restantes en trabajos de implementación de las diferentes partes del proyecto, testeo y documentación. Según los valores anteriormente establecidos, la suma asciende a 10850€.

A los gastos anteriormente mencionados ha de añadirse los denominados gastos indirectos. Son los gastos derivados del mantenimiento que se producen en cualquier proceso de creación. Son gastos tales como luz, agua, transporte, etc.

De esta manera los gastos totales del proyecto ascienden a 11124€. El desglose de cada gasto se muestra en la siguiente tabla.

Concepto	Coste
Costes Salariales	10850
Costes adquisición sw	65
Costes hw	97
Gastos indirectos	56
Total:	11124

Tabla 1. Costes totales de la aplicación.

3. ANTECEDENTES

En este apartado se presenta una descripción general del estado de los editores de textos, algunas de sus variantes y sus funciones más elementales. Además se realizará un análisis del estado actual y trabajos anteriores realizados en el ámbito de la corrección ortográfica basado tanto en distancia de edición como en modelos de lenguaje.

3.1. Editores de texto. Tipos y funcionalidades.

Entendemos como editor de textos al programa que permite crear y modificar archivos digitales compuestos solamente por un texto sin formato, conocidos de manera general como archivos de texto o texto plano. Se trata de herramientas muy útiles para la creación de documentos no demasiado extensos que permiten un uso sencillo y rápido para la edición de texto. Todos los sistemas operativos cuentan con distintas herramientas para este fin. Linux cuenta con Gedit y vi, Windows ofrece Notepad o Bloc de notas. Además existen herramientas de distribución abierta como el Notepad++, un editor potente y altamente popular en el ámbito profesional de la informática.

Se debe diferenciar entre editor de texto y procesador de texto. Los procesadores son aquellas aplicaciones que además de dar la oportunidad de crear contenido escrito, permiten al usuario administrar maquetación o diagramación al texto. Para administrar este formato a los textos el sistema hace uso de unos caracteres especiales no mostrados al usuario denominados caracteres de control. Mientras que en los documentos de texto plano estos se limitan únicamente al salto de línea, tabulación horizontal y retorno de carro, los procesadores de texto hacen uso de un mayor número para poder darle al texto unas características de terminadas.

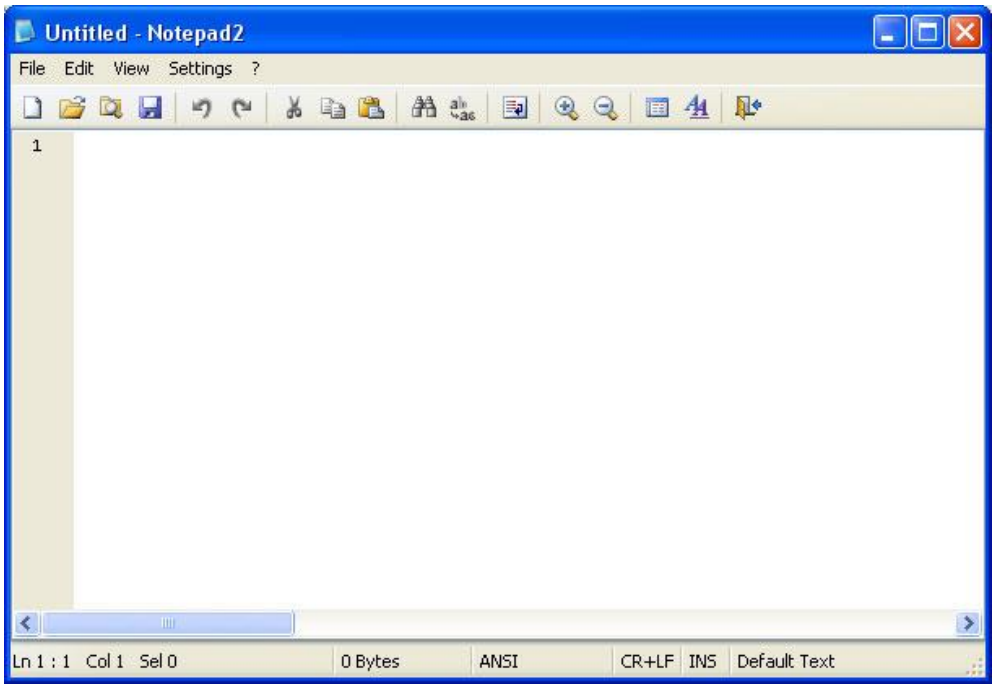

Imagen 4. Editor de texto Notepad2.

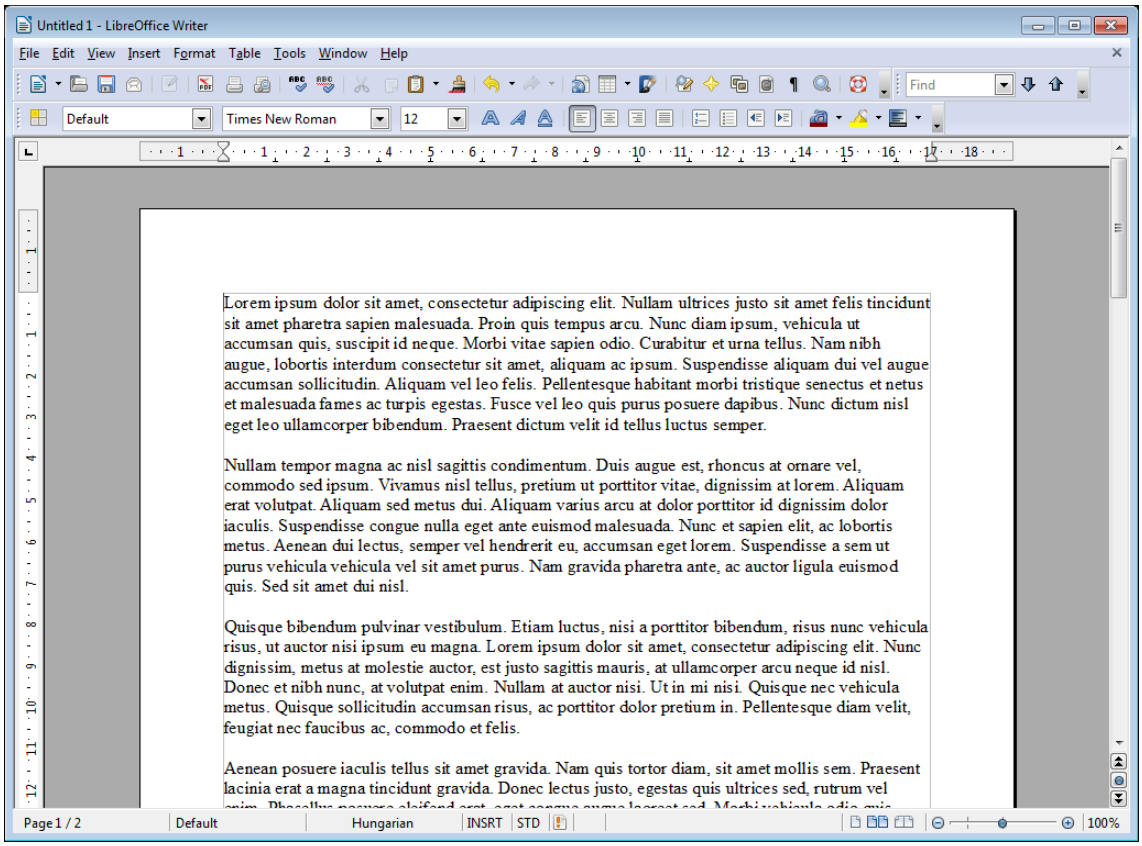

Imagen 5. Procesador de textos LibreOffice.

Hoy en día, existen gran variedad de editores, los cuales responden a distintas necesidades. Un caso especial son los editores de texto enfocados a la programación y tratamiento de código. Estos editores ofrecen multitud de funcionalidades como coloreado de sintaxis o regiones plegables, Y muchos de ellos suelen aparecer integrados en entornos de desarrollo que incluyen otra herramientas de trabajo.

En este punto se presenta la necesidad de decidir si nuestra aplicación debe o no soportar maquetación especial (interlineado, negrita, subrayado, etc.). Debido a las cualidades que presentan habitualmente los textos médicos se decide que no es necesaria la implementación de un procesador de textos ya que la elaboración de este tipo de documentos es meramente testimonial y la maquetación juega un papel irrelevante. El formato de los textos que la aplicación debe abordar será de tipo *txt*, por lo que la herramienta ha de encargarse solo de la integridad de los datos así como de su corrección pero no del estilo del texto.

3.2. Corrección ortográfica

Primeramente hemos de definir el objetivo de un corrector ortográfico y establecer la diferencia con correctores gramaticales. La tarea principal del corrector ortográfico es señalar al usuario las palabras del texto que se encuentran escritas de manera errónea, es decir, únicamente comprueba si las palabras se encuentran en el diccionario del lenguaje utilizado y de no ser así ofrece una serie de palabras candidatas reales. En cambio, el corrector gramatical cumple una función más complicada, ya que se encarga de comprobar la correcta construcción de una sentencia, como la concordancia de género y número, tiempos verbales, etc.

Para poder visualizar de una manera más sencilla el proceso de corrección ortográfica puede ser adecuado realizar una separación entre la detección de errores y la corrección de los mismos ya que son etapas y procesos muy distintos a ser analizados de forma independiente. Pero primeramente puede ser adecuado diferenciar las causas de los errores ortográficos.

Tipos de errores ortográficos

Los errores ortográficos producidos en la composición de un texto suelen ser causados primordialmente por dos razones:

Errores tipográficos. Estos errores se producen cuando el usuario es consciente de la correcta ortografía de la palabra pero comete un error durante la introducción de los caracteres. Este tipo de errores están más

relacionados con el método de entrada del texto y, por lo tanto, no siguen un criterio lingüístico. El estudio de Damerau plantea que el 80% de los errores de carácter tipográfico se encuentran en una de las siguientes 4 categorías.

- Inserción de un carácter (insertion) -> padso por paso.
- Omisión de un carácter (deletion) -> pso por paso.
- Sustitución de un carácter (substitution) -> pafo por paso.
- Transposición de dos caracteres adyacentes (transposition) -> psao por paso.

Errores cognitivos. Estos errores se producen cuando se desconoce la correcta ortografía de la palabra. En casos de error de tipo cognitivo la pronunciación de la palabra errónea es similar a la de la palabra correcta. Un ejemplo sería la introducción de la palabra clabo por la palabra clavo.

Detección de errores

La mayoría de correctores ortográficos actuales hacen uso de un diccionario. El primer uso indirecto de un diccionario para este objetivo es realizado por Riseman y Hanson en 1974, los cuales crean un diccionario con los trigramas (grupos de tres letras) que aparecen en un corpus, esta apreciación viene recogida en el artículo de Roger Mitton so Spellchecking by computer. De esta manera si durante el proceso de detección se encuentra un trigrama que no se encuentra en la tabla este se muestra como un error ortográfico.

Aunque existen otros métodos, como el análisis de n-gramas, la consulta en diccionario es el método más extendido. El análisis de n-gramas es utilizado específicamente para corrección de textos de gran volumen, no prestando atención a la palabra de manera aislada si no a la palabra dentro de un contexto, tratando un conjunto de ellas.

Aunque la mayoría de los correctores ortográficos hacen uso de diccionarios, estos pueden diferir en forma, ya que no es necesario el almacenamiento de todo el vocabulario de un idioma con todas sus palabras completas. Este tipo de almacenamiento surge en base a la separación entre la raíz de una palabra y sus posibles prefijos y sufijos. A este proceso se le denomina lematización o stemming. Este es el sistema utilizado por paquetes de herramientas como OpenOffice, en el que se distingue dos archivos de texto diferentes uno con la lista de palabras base junto

con las claves para su transformación, con extensión .dic, y otro archivo de extensión .aff en el que se establecen los códigos y los caracteres a introducir o eliminar.

```
PFX s 0 sub .
PFX t Y 1
PFX t 0 super .
PFX u Y 8
PFX u e tran es
PFX u 0 tran s
PFX u 0 trans [^es]
PFX u e trans e[^s]
PFX u e tra es
PFX u 0 tra s
PFX u 0 tras [^es]
PFX u e tras e[^s]
```

```
abocinar/P
abofeteador/GS
abofetear/ERDÁÀÂÅ
abofetee/GS
abogacía/S
abogad/O
abogadismo/S
abogador/S
abolengo/S
abolicionismo/S
abolicionista/S
abolid
```

Imagen 6. Archivo .aff y archivo .dic

Después, es el programa, el que mediante unos algoritmos produce la decodificación y formación de cualquier palabra. Este tipo de diccionarios también son muy utilizados por sistemas de búsqueda como Google.

Estos diccionarios surgen en base a una necesidad de ahorro de espacio de memoria ya que reduce ostensiblemente el espacio ocupado por el diccionario. En la red existen diferentes métodos de lematización basados mayoritariamente en el denominado algoritmo de Porter. Este algoritmo fue diseñado primeramente para la lengua inglesa pero más tarde se ha ido adaptando a otros idiomas.

Cabe destacar que el proceso de lematización no es igual de eficaz y no tiene el mismo nivel de complejidad para todos los idiomas. Por ejemplo, el inglés presenta unas reglas muy sencillas de formación de los distintos tiempos verbales en las que no influyen ni el género ni el número del sujeto, en el español sí se han de tener en cuenta estos factores, así como la irregularidad de las variaciones de las palabras aun teniendo raíces similares. Un ejemplo claro son estos tres verbos de la 3ª conjugación que en teoría deberían seguir un mismo comportamiento, los cuales son: medir, rendir y permitir. Al conjugar la primera persona del singular del presente de indicativo obtendríamos: mido, rindo y permito. Se puede observar que las terminaciones son totalmente distintas y en dos de los casos la raíz de la palabra desaparece prácticamente. Estas irregularidades de la lengua hacen de la lematización del español una tarea considerablemente más complicada.

Otro punto a destacar en la elección y elaboración del diccionario es la estructura de datos seleccionada para su almacenamiento. Hoy en día se presentan distintos tipos de

estructuras de datos para este cometido, que son principalmente la utilización de tablas hashmaps o el uso de árboles. Ambas estructuras presentan ventajas e inconvenientes. Más adelante en el apartado de desarrollo se justifica el uso de una estructura en detrimento de otra.

Corrección de errores

En la actualidad existen diversos métodos para la corrección de errores ortográficos entre los que se encuentran los siguientes.

Sistemas basados en reglas (Rule-based Techniques).

Se basan en la tenencia de un conjunto de reglas que recogen los errores tipográficos y ortográficos más comunes y la aplicación de estas reglas a la palabra errónea. Estas reglas son el proceso inverso a errores comunes. Cada una de las palabras obtenidas mediante la aplicación de estas reglas a una palabra errónea es tomada como una palabra correcta sugerida. Las reglas empleadas, también llevan asociadas unas probabilidades, haciendo posible la ordenación de las sugerencias en función de la suma de las probabilidades de cada una de las reglas aplicadas. Es un sistema realmente efectivo si se dispone de las reglas a emplear en el idioma deseado o al menos un corpus lo suficientemente grande como para poder inferir estas con un cierto grado de fiabilidad.

Redes neuronales.

A pesar de no ser el método más utilizado suponen un posible camino para el perfeccionamiento de la corrección de errores. Los métodos actuales se basan en redes de retropropagación. Se trata de un algoritmo de aprendizaje supervisado. Se utiliza un nodo de salida para cada palabra en el diccionario y uno de entrada para cada uno de los posibles n-gramas que pueden producirse en cada posición de la palabra, tratándose normalmente de bigramas o trigramas.

Normalmente solo uno de los nodos de salida suele estar activo, indicando que palabra del diccionario la red sugiere como corrección. Este método por el momento funciona para un número de palabras reducido, inferior a 1000. Los tiempos requeridos son demasiado grandes para el hardware ordinario actual.

Claves de similitud (Similarity Keys).

Se asigna una clave a cada una de las palabras que componen el diccionario y se realiza la comparación de la clave de la palabra errónea con las claves de las palabras que componen el diccionario. Las palabras correspondientes a las claves más parecidas a la clave generada por la palabra errónea son tomadas como sugerencias. Es un método rápido y con un adecuado algoritmo de transformación este método puede manejar problemas tipográficos.

Una vez presentados algunos de los métodos de corrección en uso en la actualidad, se profundizará en el método de distancia de edición y los modelos de lenguaje estadísticos. Se realiza especial hincapié en estos dos métodos ya que son los utilizados en el desarrollo de la aplicación debido a que su uso es ampliamente extendido ya que multitud de herramientas se basan en estos métodos para la creación de sus correctores.

Distancia de edición.

Como se ha apuntado en el apartado de tipos de errores, según la teoría de Damerau el 80% de los errores cometidos durante la introducción de una palabra en un texto se encuentran a una distancia de edición igual a uno. Este sistema se basa en la creación de todas las palabras posibles mediante la corrección de los tipos de errores anteriormente mencionados (inserción, omisión, sustitución y transposición). Es decir, si la palabra es errónea debido a la inserción de un carácter que no le pertenece, se tratará de generar todas las palabras correctas mediante la eliminación de cada uno de sus caracteres.

Palabra errónea introducida: cadsa

Palabras generadas mediante eliminación: adsa, cdsa, casa, cada, cads

Estas serían las palabras con distancia de edición uno, generadas por el método de eliminación a partir de la palabra errónea.

Como a priori, se desconoce el error cometido en el momento de la introducción del texto, se realiza la creación de todas las palabras posibles teniendo en cuenta los tipos de errores posiblemente cometidos.

Palabra errónea introducida: cadsa.

Palabras generadas mediante eliminación: adsa, cdsa, casa, cada, cads.

Palabras generadas mediante transposición: acdsa, cdasa, casda, cadas.

Palabras generadas mediantes inserción: acadsa, bcadsa, ccadasa, dcadsa … cdasay, cadsaz.

Palabras generadas mediantes sustitución: aadsa, badsa, dadsa, eadsa, badsa … cadsx, cadsy, cadsz.

Este proceso puede dar lugar a un conjunto muy grande de palabras. Para una palabra de longitud n, se producirán n eliminaciones, n-1 transposiciones, 26n sustituciones y 26(n+1) inserciones lo que supone un total de 54n+25 palabras generadas con distancia de edición igual a uno. Evidentemente de este conjunto de palabras solo un pequeño número son palabras reales, lo cual se comprueba mediante la utilización de un diccionario.

Con la necesidad de realizar una ordenación de estas sugerencias en función de la probabilidad de ser la correcta surge la denominada ponderación. Este método, relativamente parecido al basado en reglas, realiza una asociación del error cometido en relación a la probabilidad del error producido, dando más peso a los errores cometidos más comúnmente. Para llevar a buen puerto esta ponderación es necesario tener en cuenta los errores tipográficos (posicionamiento de los caracteres en el dispositivo de entrada) y errores ortográficos más comunes de la lengua a tratar.

Modelo de lenguaje

La función de un modelo de lenguaje estadístico es asignar una probabilidad a una secuencia de palabras en base a una distribución de probabilidad. Esta técnica se utiliza en un amplio número de campos, además de en el de la corrección ortográfica, como pueden ser el reconocimiento de voz, etiquetado de discurso, reconocimiento de escritura y especialmente en recuperación de información.

Este método se basa en calcular la probabilidad de una secuencia de palabras en base a la probabilidad obtenida en un corpus de entrenamiento. En este proyecto se hará uso de los modelos de lenguaje basados en n-gramas. En este tipo de modelos se asume que la probabilidad de aparición de un término viene condicionada por las palabras que lo preceden. Estas probabilidades se obtienen en base a los recuentos de frecuencia registrados durante el entrenamiento o tratamiento del corpus.

Estas teorías se fundamentan apoyándose en la propiedad de Markov. Esta propiedad es una teoría de probabilidad y estadística que establece la propiedad de ciertos procesos estocásticos (sucesión de variables aleatorias que evolucionan en función de otra variable) carecen de memoria. Esto significa que la distribución de probabilidad de un valor futuro depende de su valor actual, pero es independiente del historial de esa variable.

Este concepto se puede consultar más en profundidad en los trabajos de Michael Collins realizados en la Universidad de Columbia. Más adelante, en el apartado de desarrollo del modelo de lenguaje se profundiza en los conceptos matemáticos en los que se apoya esta teoría y su utilización.

Actualmente existen otros modelos de lenguaje destacables como puede ser el modelo de lenguaje factorizado o el modelo de lenguaje posicional, que establece que las palabras ocurren cerca unas de otras en un texto, pero no necesariamente han de ser adyacentes.

En el modelo de lenguaje basado en n-gramas las probabilidades no son calculadas directamente, ya que se presenta el problema de calcular probabilidades de n-gramas que no han sido vistos explícitamente por el modelo. Para ello es necesaria la utilización de alguna manera de suavizado, técnicas de smoothing, desde el método add-one, a los más complejos Good-Turing y modelos de back-off.

3.3. Estado del arte

Desde hace años se han llevado a cabo diversos estudios y varios proyectos impulsados desde distintos colectivos.

Un ejemplo es el proyecto llevado a cabo desde la universidad de Stanford, el cual ha desarrollado multitud de software dedicado a esta rama, el cual engloba desde clasificadores, a tokenizadores y equitadores, así como creadores de modelos de lenguaje. En este centro existe un grupo mixto de lingüistas e informáticos enteramente dedicado al estudio de esta área de la informática.

Otro proyecto de carácter colaborativo el OpenNLP de Apache. Se trata de una librería que incluye una herramienta para el procesamiento de texto de lenguaje natural. Realiza las tareas más comunes relacionadas con NLP como la tokenización, segmentación de frases, etiquetado,

etc. A diferencia del grupo perteneciente a Stanford, los cuales ofrecen infinidad de recursos teóricos sobre la materia, este proyecto es más hermético y más orientado a la creación de un software.

En lo que se refiere a tratamiento de lenguaje médico, si bien es cierto que existe multitud de material y trabajos realizados en esta área hay que destacar que la mayoría están más enfocados a recuperación de información y speech recognition, los cuales también se apoyan en modelos de lenguaje pero no supone ningún aporte para la realización de este proyecto. En resumen no aportan nuevas teorías para lenguaje medico en español que puedan resultar útiles para el editor a desarrollar.

3.4. Estudio de antecedentes

Para el desarrollo de este proyecto se ha tomado como base y punto de partida los conocimientos desarrollados por los profesores de Standford, Jurafsky y Manning en un curso sobre procesamiento de lenguaje natural, NLP. En este curso impartido por estos dos expertos se establecen los conceptos básicos para poder afrontar los retos de desarrollo de un corrector ortográfico básico. Se abordan los concetos de tokenización, edición de distancia y modelos de lenguaje, incluidas las diferentes técnicas de smoothing.

Aunque de menos calado, también es importante recalcar la utilización de un curso perteneciente a la universidad de Columbia realizado por el profesor Michael Collins en el que en el primero de sus apartados se hace un extenso repaso y explicación del modelado de lenguajes, estableciendo las bases de su funcionamiento y su fundamentación teórica.

Además de este curso ha sido de gran ayuda un artículo realizado por Peter Norvig, en el cual, este experto, realiza un pequeño experimento de un corrector ortográfico rudimentario desarrollado en Python para inglés. En este artículo se establecen algunas pautas para la utilización de las medidas de distancia y creación de un pequeño modelo de lenguaje.

En base a este material se puede partir con unos objetivos claros a los que poder aspirar. Aunque para la resolución del proyecto ha sido necesaria la documentación de otros aspectos como pueden ser las estructuras de datos empleadas para el almacenamiento del vocabulario del diccionario así como de los datos inferidos a partir del corpus. Para poder decidir qué estructuras se pueden adecuar a nuestro volumen de datos y nuestros requerimientos nos hemos basado en los trabajos realizados por Daniel Robenek, Jan Platos, Vaclav Snasel sobre Ternary search Tree y en los estudios llevados a cabo por los mismos autores acerca de estructuras de datos en memoria eficientes para indexación de n-gramas.

4. CAPTURA DE REQUISITOS

Se debe conseguir una aplicación que sea capaz de editar un texto de manera cómoda. Pero principalmente, deberá detectar las palabras que contengan errores ortográficos y suministrar al usuario una serie de palabras reales similares basadas en la probabilidad obtenida a través de un modelo de lenguaje.

4.1 Casos de uso

La aplicación consta de un solo actor y se trata del usuario de la aplicación. El usuario hace uso de la aplicación a través de su computadora. Las funcionalidades que le corresponden se ven representadas en el siguiente diagrama.

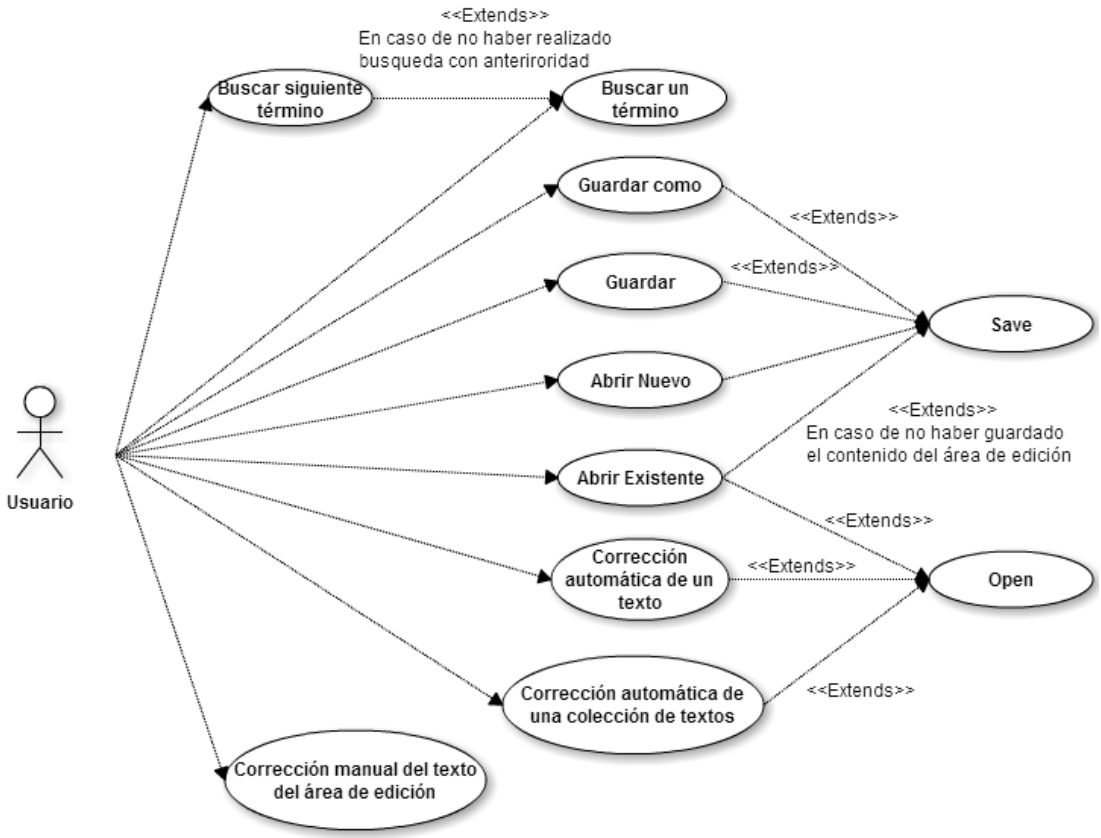

Imagen 7. Diagrama de casos de uso

En este diagrama se han obviado algunos casos de uso que no son relevantes para la aplicación y que vienen explicados con detalle en el apartado de casos de uso extendido.

4.2. Casos de uso extendidos

Nombre: **Modificar área de texto**

Descripción: El usuario modifica (escribir, eliminar, cortar o pegar) el texto del área de edición

Actores: Usuario

Precondiciones: Ninguna (haber abierto la aplicación)

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario abre la aplicación.
2. Modifica los valores del texto del área de edición.

PostCondiciones: Ninguna

Interfaz Gráfica:

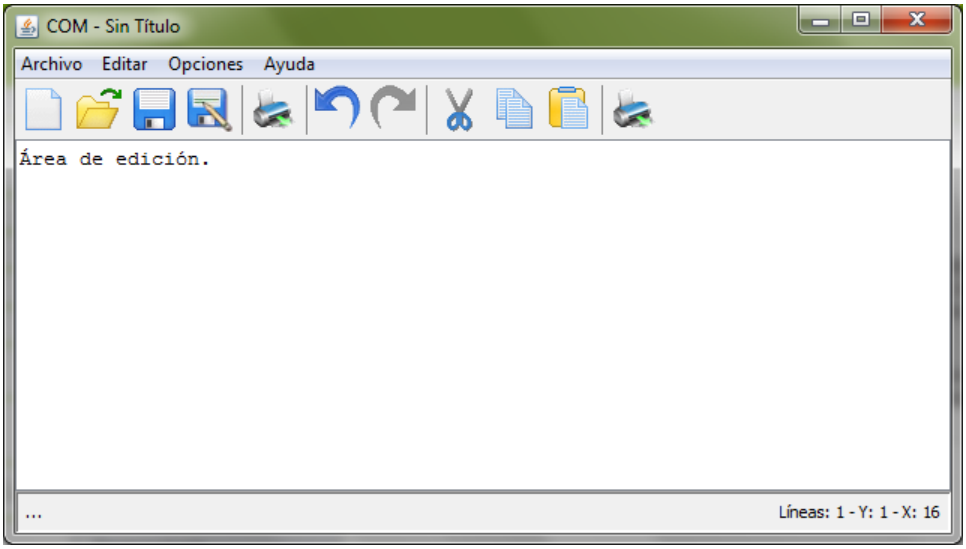

Imagen 8. Interfaz del área de edición.

Action Save

1. Se abre un nuevo cuadro diálogo en el que al usuario se le permite seleccionar de manera manual la ubicación en la que desea guardar el archivo y el nombre del mismo.

[El usuario no selecciona una ruta]

2a. La ruta por defecto en la que se guardará el archivo es MisDocumentos.

[El usuario no completa el campo de nombre del archivo]

[El usuario pulsa aceptar]

3aaa. No permite guardar el archivo.

[El usuario pulsa cancelar]

3aab. El archivo no se guarda y se muestra el área de texto vacía.

2b. [El usuario completa el campo de nombre del archivo]

[El usuario pulsa aceptar]

3aba. El archivo se guarda y se muestra el área de texto vacía.

[El usuario pulsa cancelar]

3abb. El archivo no se guarda y se muestra el área de texto vacía.

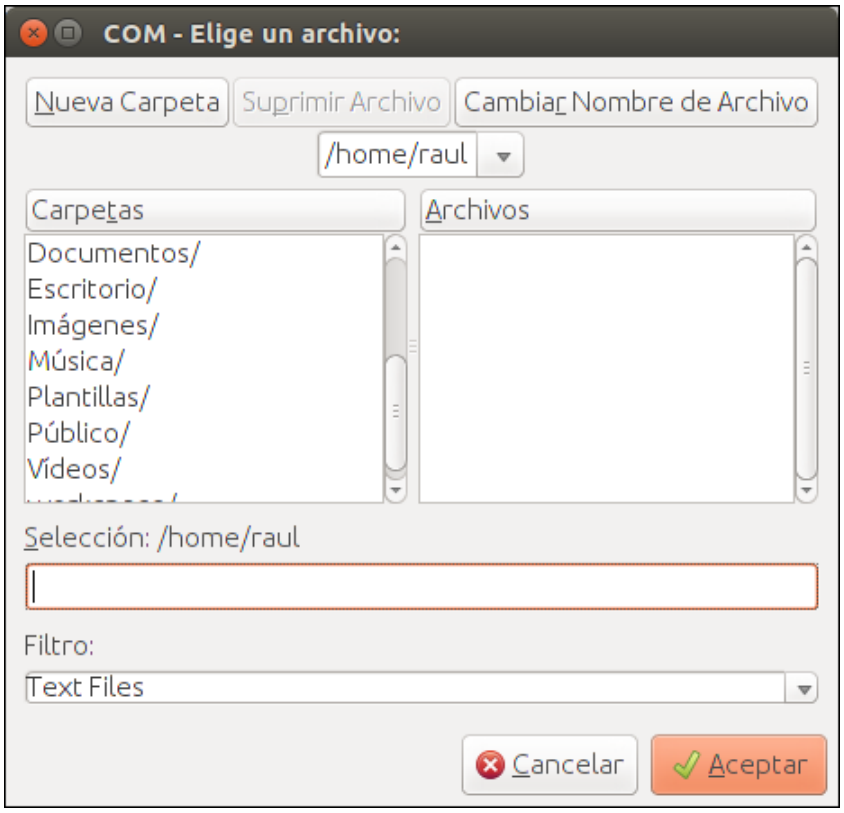

Imagen 9. Interfaz de guardado de fichero.

Action Open

1. Se abre un nuevo cuadro de diálogo en el que al usuario se le permite acceder de manera manual a la ubicación donde se alberga el archivo y seleccionar tanto de manera manual como escribiendo el archivo requerido.

[El usuario no selecciona con el ratón ningún archivo y pulsa Abrir]

2a. No ocurre nada, se sigue mostrando la misma interfaz.

[El usuario escribe de manera errónea el nombre del archivo y pulsa Abrir]

2b. Se muestra una ventana emergente con un mensaje de error.

[El usuario escribe de forma correcta el nombre del archivo o lo selecciona con el ratón y pulsa Abrir]

2c. Se cierra el cuadro de diálogo y se muestra el editor en el que el área de edición contiene el contenido del archivo seleccionado.

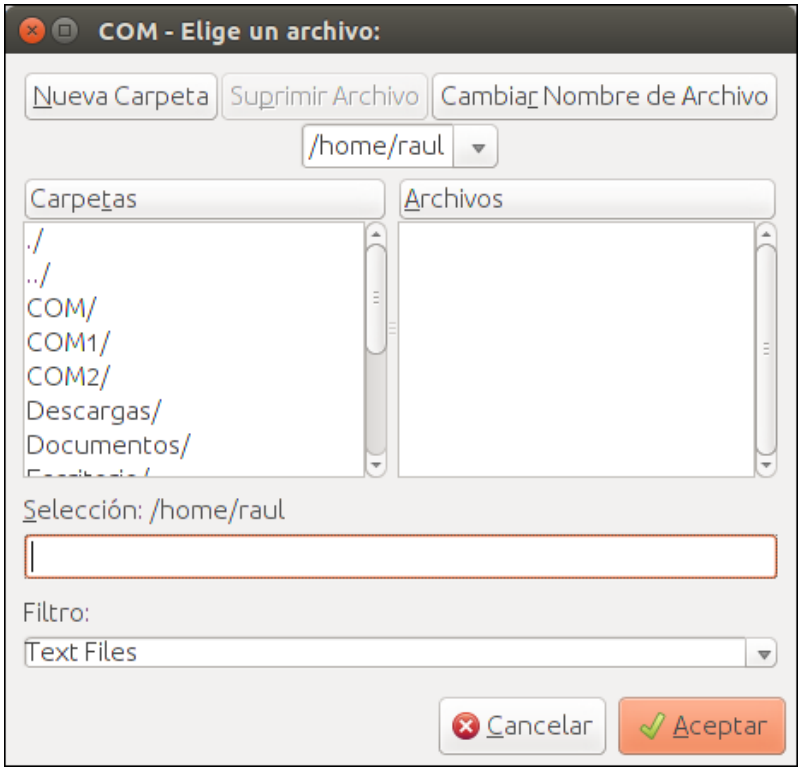

Imagen 10. Interfaz de apertura de fichero.

Nombre: **Nuevo**

Descripción: El usuario realiza la apertura de un nuevo archivo de texto.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “Nuevo” o selecciona la opción “Nuevo” dentro del menú Archivo o pulsa la combinación de teclas CTRL+N .

   [El usuario ha realizado algún cambio en el área de edición (de cualquier tipo) y no lo ha guardado.]

   2a. Se muestra un cuadro de diálogo donde se le pregunta si desea guardar los cambios realizados.

   [El usuario pulsa el botón No]

   3aa. Se cierra el cuadro de diálogo y se muestra el área de edición totalmente en blanco.

   [El usuario pulsa el botón Cancelar]

   3ab. Se cierra el cuadro de diálogo y se muestra el área de edición tal y como estaba.

   [El usuario pulsa el botón Sí]

   3ac. Action Save

   [El usuario ha guardado cualquier cambio que se haya producido en el área de edición]

   2b. Se muestra una nueva área de edición totalmente en blanco.

PostCondiciones: Ninguna

Interfaz Gráfica:

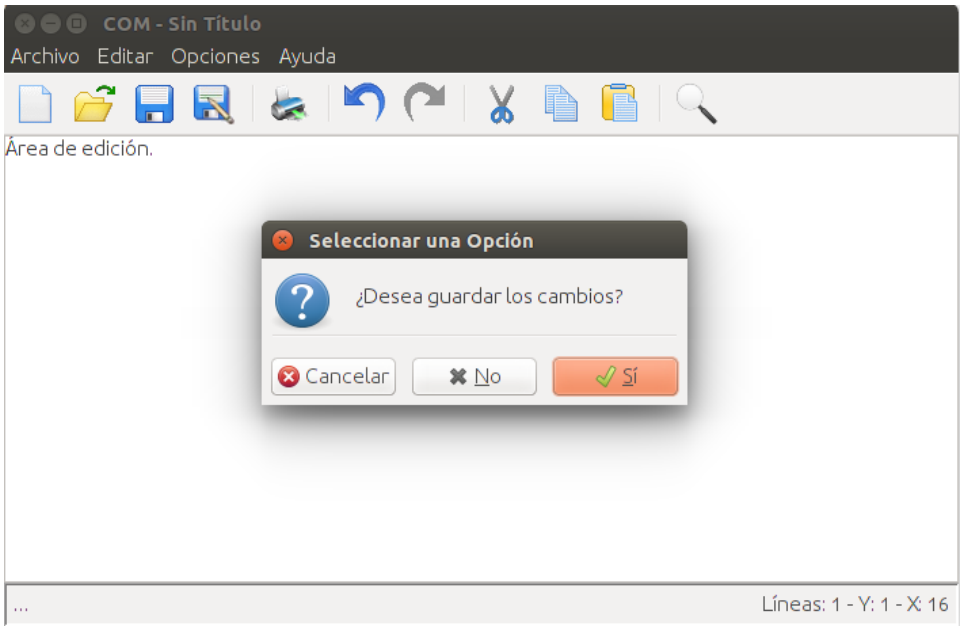

Imagen 11. Cuadro de diálogo de consulta de guardado de cambios.

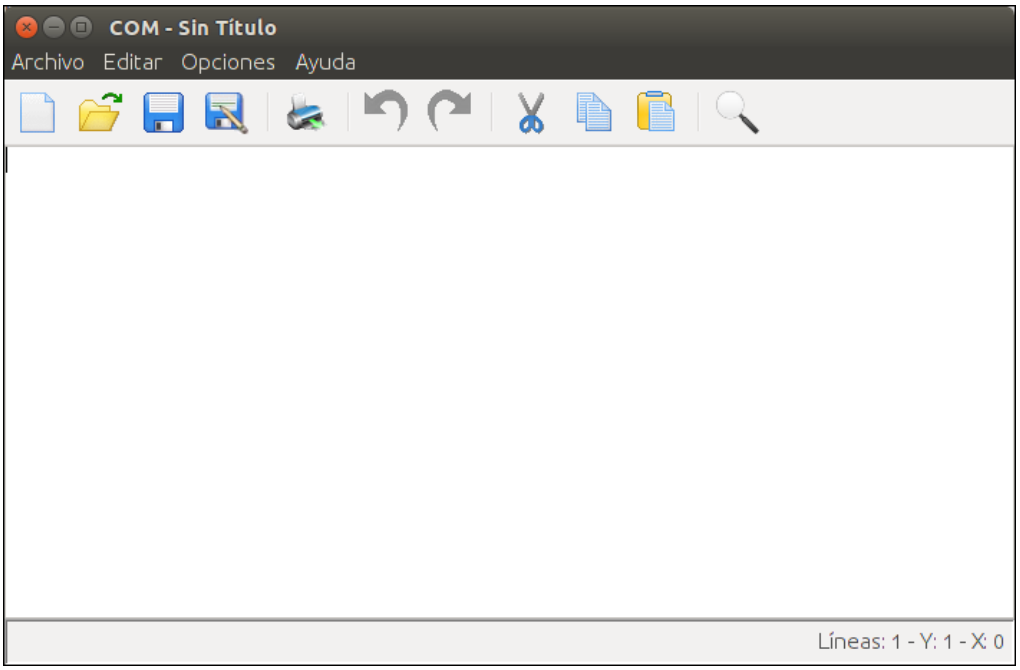

Imagen 12. Área de edición. (Nuevo documento).

Nombre: **Abrir**

Descripción: El usuario realiza la apertura de un archivo de texto ya existente.

Actores: Usuario

Precondiciones: Es necesario que el archivo exista en los directorios del equipo.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “Abrir” o selecciona la opción “Abrir” dentro del menú Archivo o pulsa la combinación de teclas CTRL+O.
   [El usuario ha realizado algún cambio en el área de edición (de cualquier tipo) y no lo ha guardado.]

   2a. Se muestra un cuadro de diálogo donde se le pregunta si desea guardar los cambios realizados.

   [El usuario pulsa el botón No]

   3aa. Action Open

   [El usuario pulsa el botón Cancelar]

   3ab. Se cierra el cuadro de diálogo y se muestra el área de edición tal y como estaba.

   [El usuario pulsa el botón Si]

   3ac. Action Save

   4ac. Action Open

   [El usuario ha guardado cualquier cambio que se haya producido en el área de edición]

   2b. Action Open

PostCondiciones: Ninguna

Interfaz Gráfica:

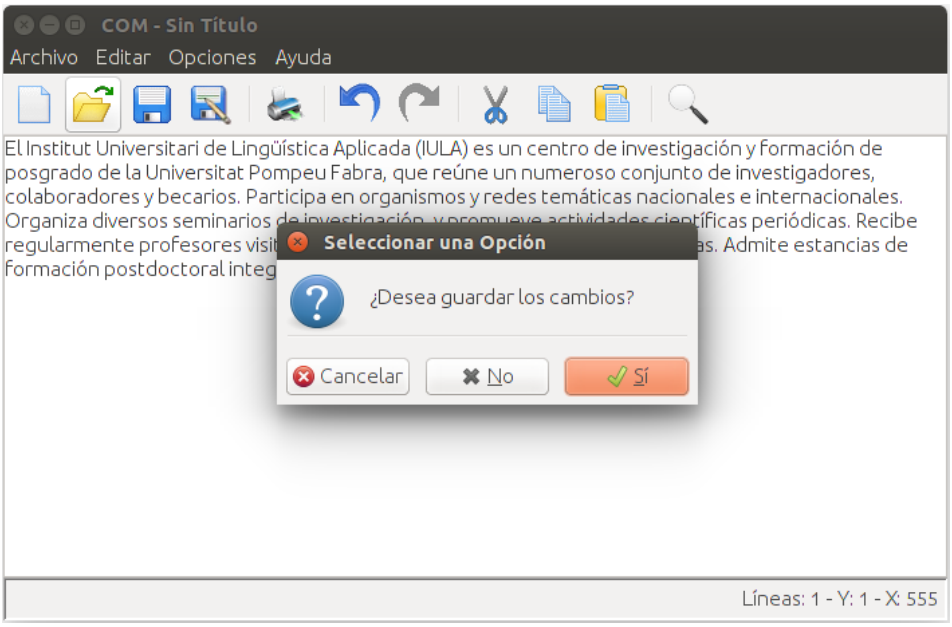

Imagen 13. Cuadro de diálogo de confirmación de apertura de un archivo

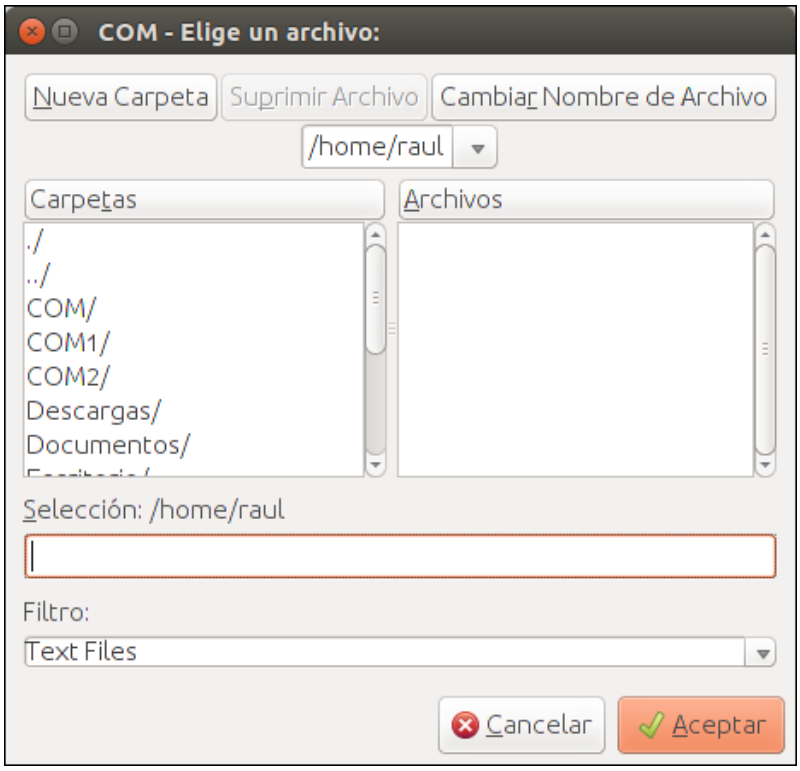

Imagen 10. Interfaz de apertura de fichero.

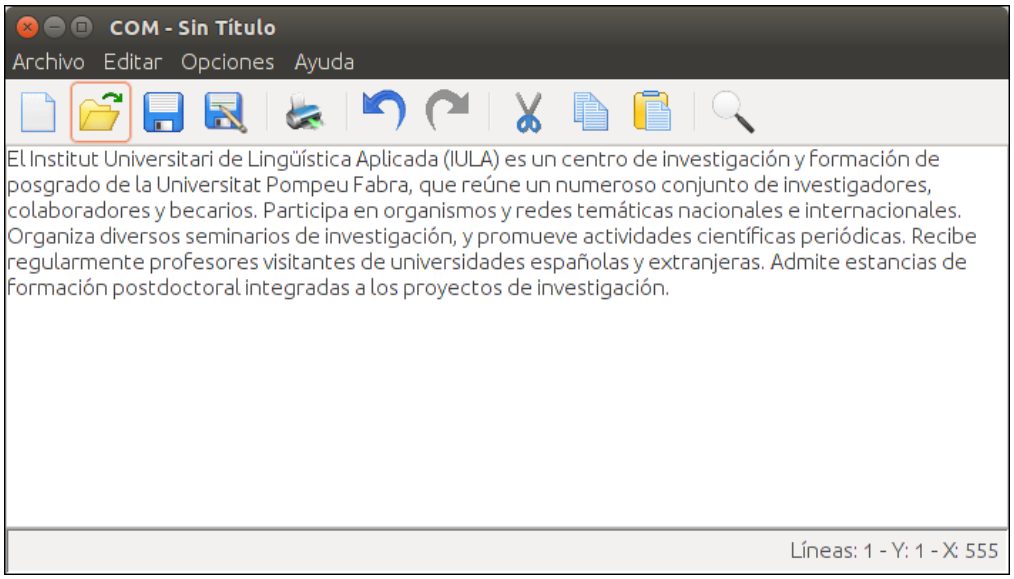

Imagen 14. Archivo de texto abierto.

Nombre: **Guardar como**

Descripción: El usuario realiza el guardado del archivo de texto en la ruta deseada.

Actores: Usuario

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “Save as” o selecciona la opción “Guardar como” dentro del menú Archivo.
2. Action Save

PostCondiciones: Ninguna

Interfaz Gráfica:

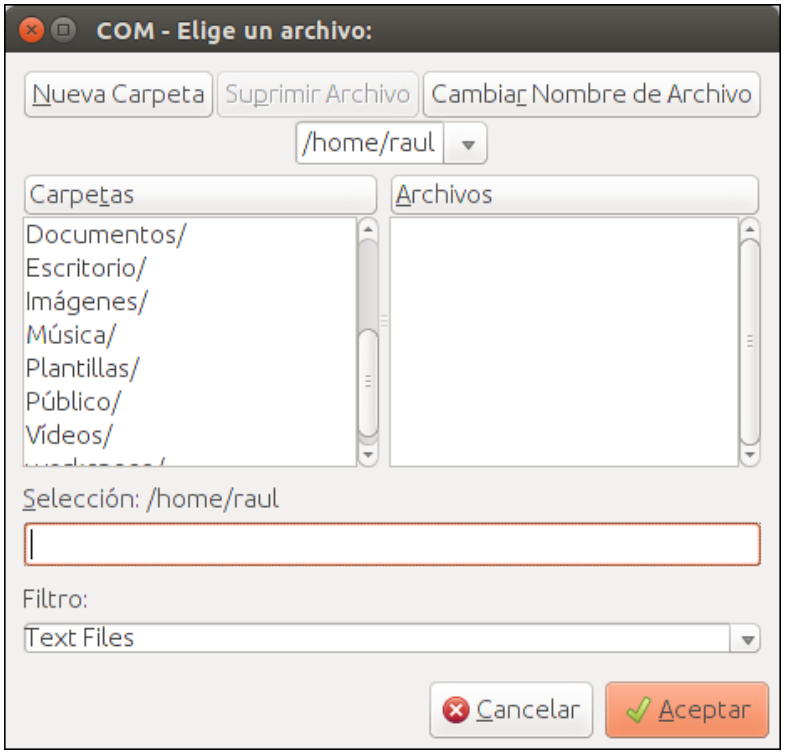

Imagen 9. Interfaz de guardado de fichero.

Nombre: **Guardar**

Descripción: El usuario realiza el guardado del archivo de texto ya existente.

Actores: Usuario

Precondiciones: El archivo que se está modificando en el área de edición ya existía y se ha abierto para su modificación o ha sido guardado anteriormente.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “save” o selecciona la opción “guardar” dentro del menú Archivo o pulsa la combinación de teclas CTRL+S.
   [El archivo no se encuentra almacenado en el sistema]
   2a. Action Save.
   [El archivo ya se encuentra almacenado en el sistema]
   2b. Se realiza el guardado del archivo.

PostCondiciones: Ninguna

Interfaz Gráfica:

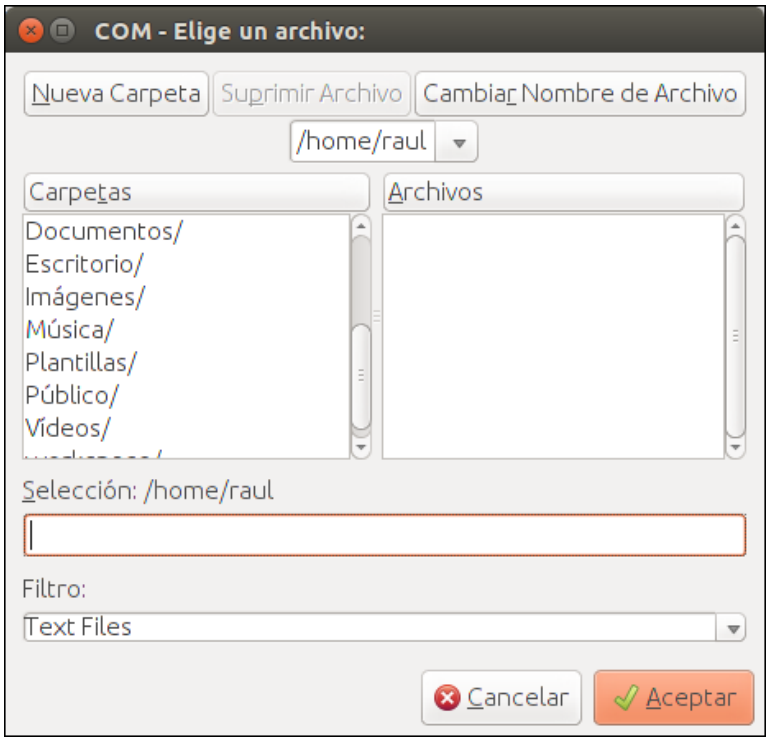

Imagen 9. Interfaz de guardado de fichero.

Nombre: **Deshacer**

Descripción: El usuario deshace la última acción realizada.

Actores: Usuario

Precondiciones: El usuario debe haber realizado un cambio de cualquier tipo en el archivo de texto, ya sea de inserción o eliminación de caracteres.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “undo” o selecciona la opción “deshacer” dentro del menú Archivo o pulsa la combinación de teclas CTRL+Z.
2. Se realizan los cambios necesarios en el archivo para que volver al estado directamente anterior y se muestran en el área de Edición.

PostCondiciones: Ninguna

Interfaz Gráfica:

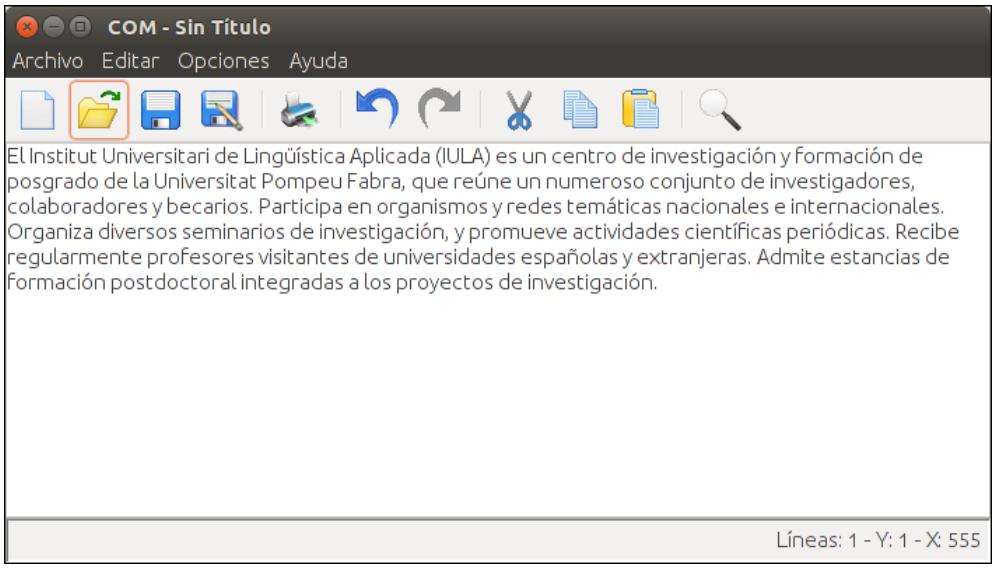

Imagen 14. Archivo de texto abierto.

Nombre: **Seleccionar todo**

Descripción: El usuario selecciona todo el texto contenido en el área de edición.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción “Seleccionar todo” dentro del menú Editar o pulsa la combinación de teclas CTRL+E.
2. Se muestra todo el texto contenido en el área de edición como seleccionado(resaltado en color azul).

PostCondiciones: Ninguna

Interfaz Gráfica:

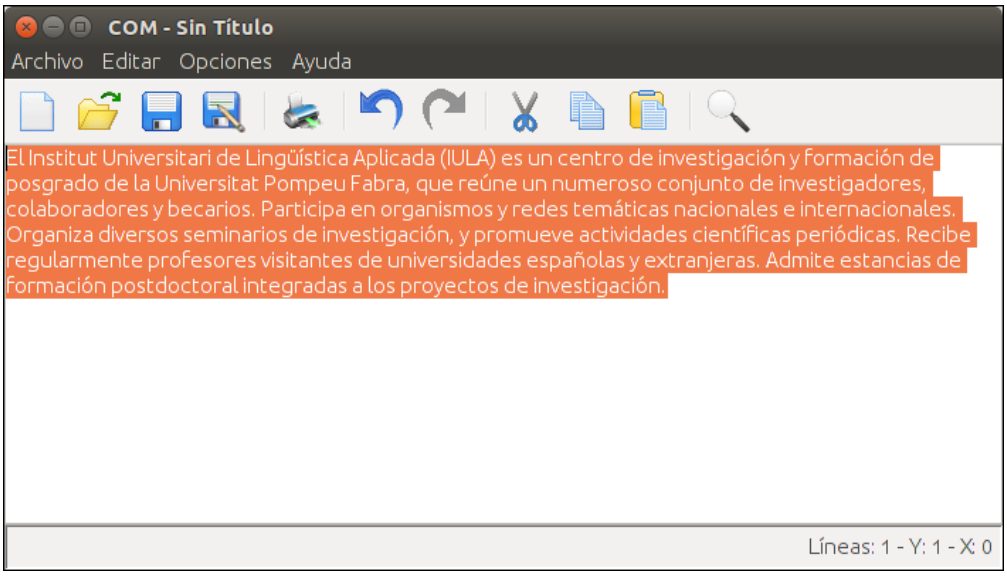

Imagen 15. Área de edición, texto seleccionado.

Nombre: **Rehacer**

Descripción: El usuario rehace la última acción deshecha.

Actores: Usuario

Precondiciones: El usuario debe haber realizado antes la acción de Deshacer un cambio de cualquier tipo en el archivo de texto, ya sea de inserción o eliminación de caracteres.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “undo” o pulsa la combinación de teclas CTRL+Y.
2. Se realizan los cambios necesarios en el archivo para volver al estado directamente anterior y se muestran en el área de Edición.

PostCondiciones: Ninguna

Interfaz Gráfica:

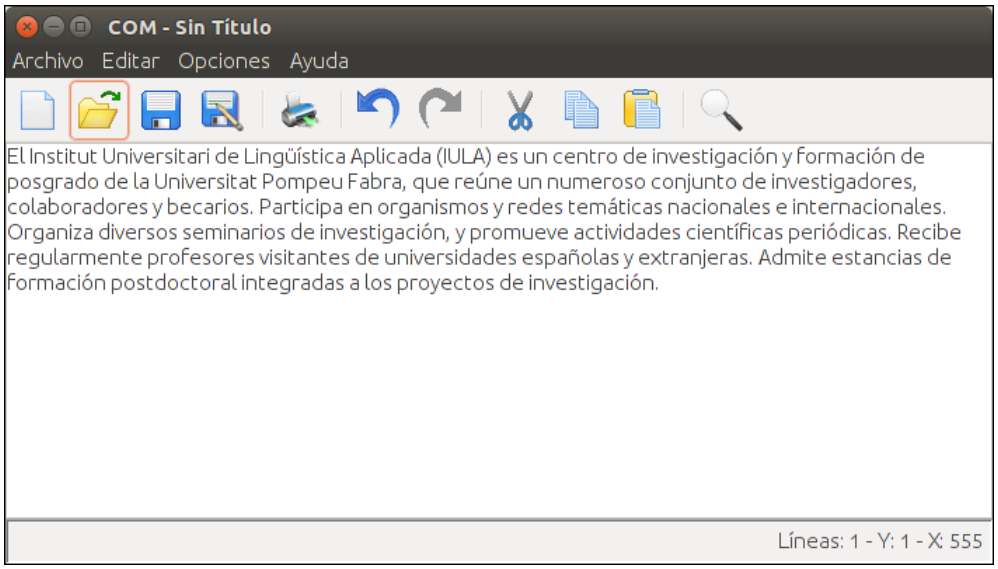

Imagen 14. Archivo de texto abierto.

Nombre: **Ajuste de línea**

Descripción: El usuario selecciona se desea que el texto se muestre ajustado a las dimensiones del cuadro de diálogo o no.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción “Ajuste de línea” dentro del menú.
   [El texto ya se encuentra ajustado al área de edición(Opción marcada en el menú)]
   2a. El texto se muestra en función de los saltos de línea introducidos por el usuario, de manera que puede quedar texto en la parte derecha no visible para el que será necesario desplazarse.
   [El texto no se encuentra ajustado al área de edición(Opción no marcada en el menú)]
   2b. El texto se muestra ajustado al área de edición. No es necesario el desplazamiento lateral.

PostCondiciones: Ninguna

Interfaz Gráfica:

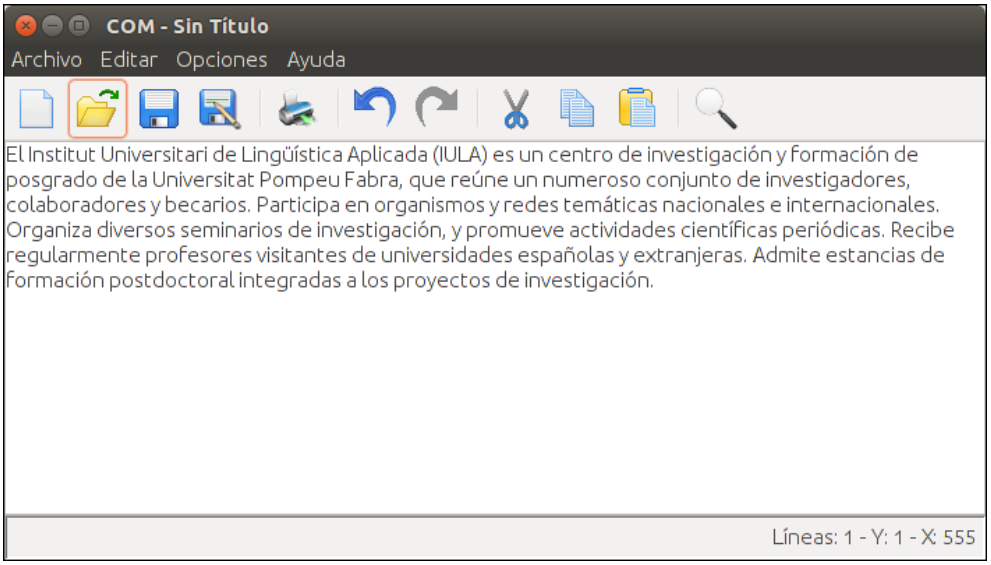

Imagen 14. Archivo de texto abierto.

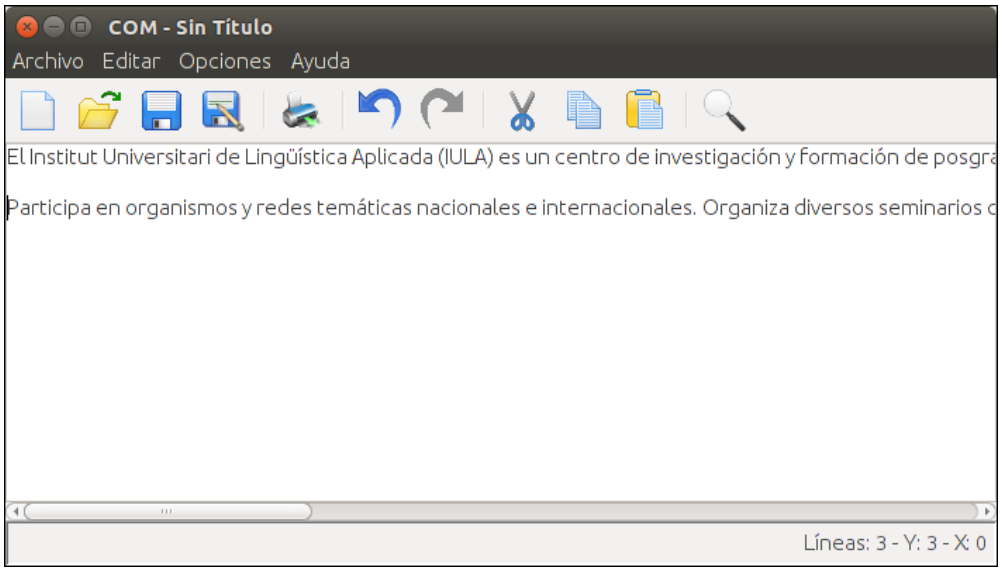

Imagen 16. Área de edición sin ajuste de línea.

Nombre: **Ver barra de herramientas**

Descripción: El usuario selecciona si desea que se muestre la barra de herramientas o no.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción "Ver barra de herramientas" dentro del menú Opciones.
   [La barra de herramientas se encuentra visible (Opción marcada en el menú)]
   2a. Se muestra la barra de herramientas con los botones de acción en la ventana principal.
   [La barra de herramientas no se encuentra visible (Opción no marcada en el menú)]
   2b. La barra de herramientas no se visualiza en la ventana principal.

PostCondiciones: Ninguna

Interfaz Gráfica:

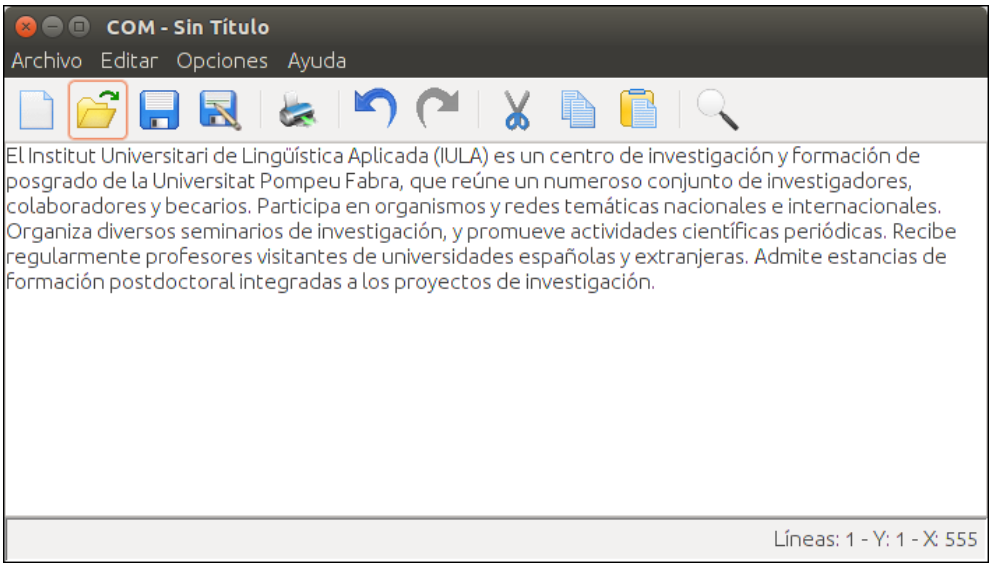

Imagen 14. Archivo de texto abierto.

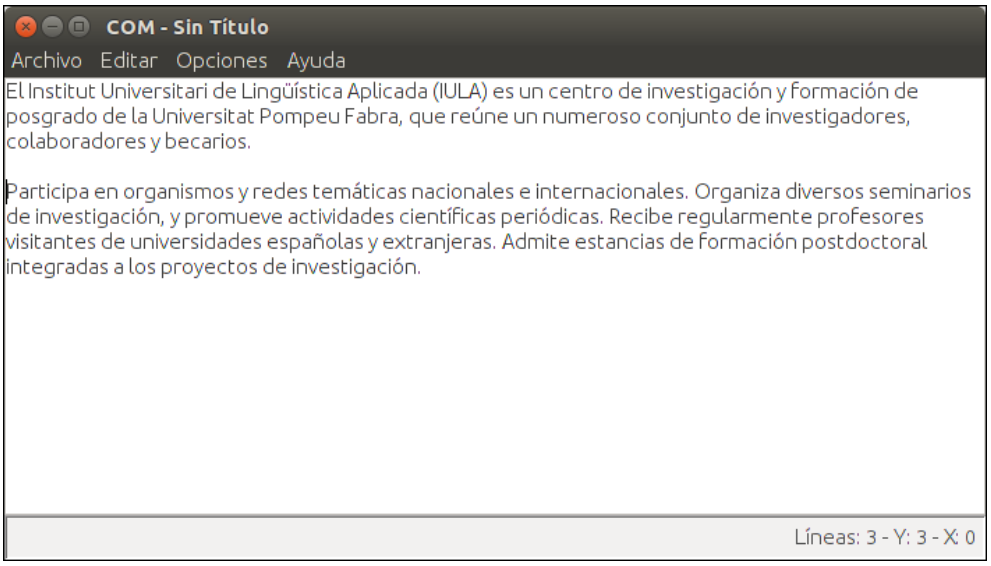

Imagen 17. Área de edición, no se muestra barra de herramientas.

Nombre: **Ver barra de estado**

Descripción: El usuario selecciona si desea que se muestre la barra de estado o no.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción “Ver barra de estado” dentro del menú “Opciones”.
   [La barra de estado se encuentra visible (Opción marcada en el menú)]
   2a. Se muestra la barra de estado en la parte inferior de la ventana principal, por debajo del área de edición.
   [La barra de estado no se encuentra visible (Opción no marcada en el menú)]
   2b. La barra de estado no se visualiza en la ventana principal.

PostCondiciones: Ninguna

Interfaz Gráfica:

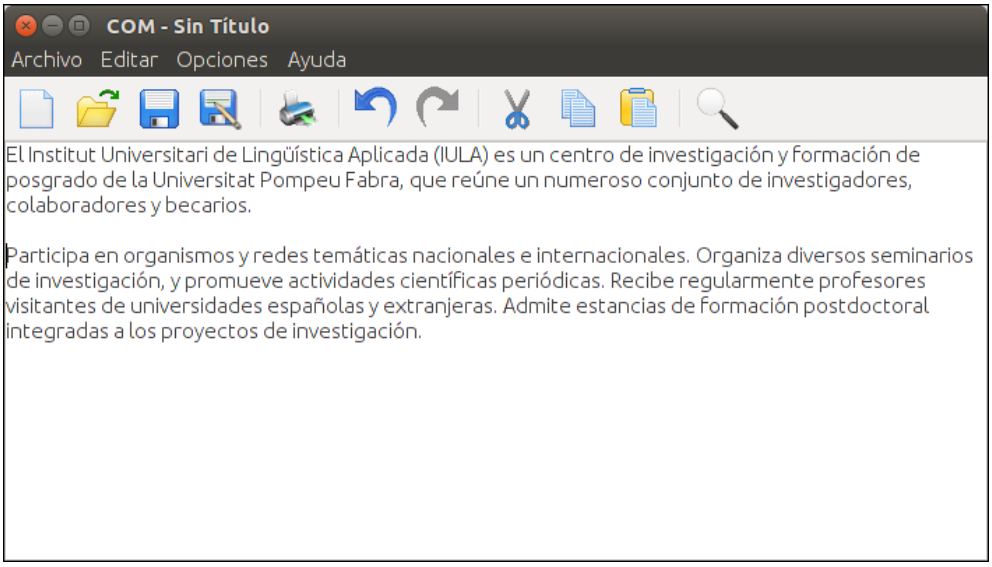

Imagen 18. Área de edición, no se muestra la barra de estado.

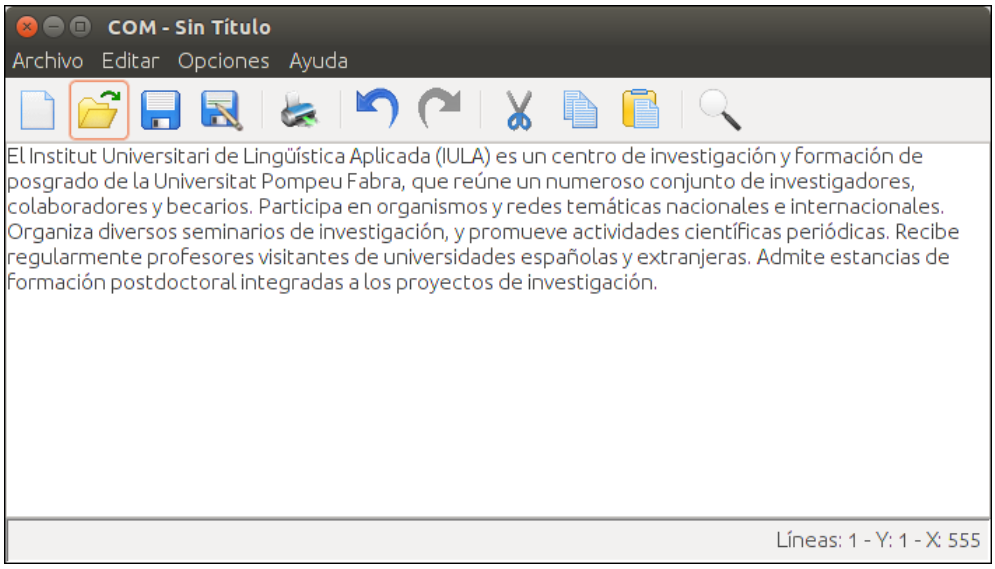

Imagen 14. Archivo de texto abierto.

Nombre: **Fijar barra de herramientas**

Descripción: El usuario cambiar el lugar en el que se ubica la barra de herramientas.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción “Fijar barra de herramientas” dentro del menú “Opciones”.
   [La barra de herramientas se encuentra fija (Opción marcada en el menú)]
   2a. Se muestra una marca en la parte izquierda o superior, dependiendo de la posición que ocupa la barra de herramientas, que indica que la barra de herramientas se puede cambiar de ubicación. Se permite su reubicación.
   [La barra de estado no se encuentra fija (Opción no marcada en el menú)]
   2b. La marca que indica que la barra de herramientas se puede mover desaparece y no se permite la reubicación de la barra de herramientas.

PostCondiciones: Ninguna

Interfaz Gráfica:

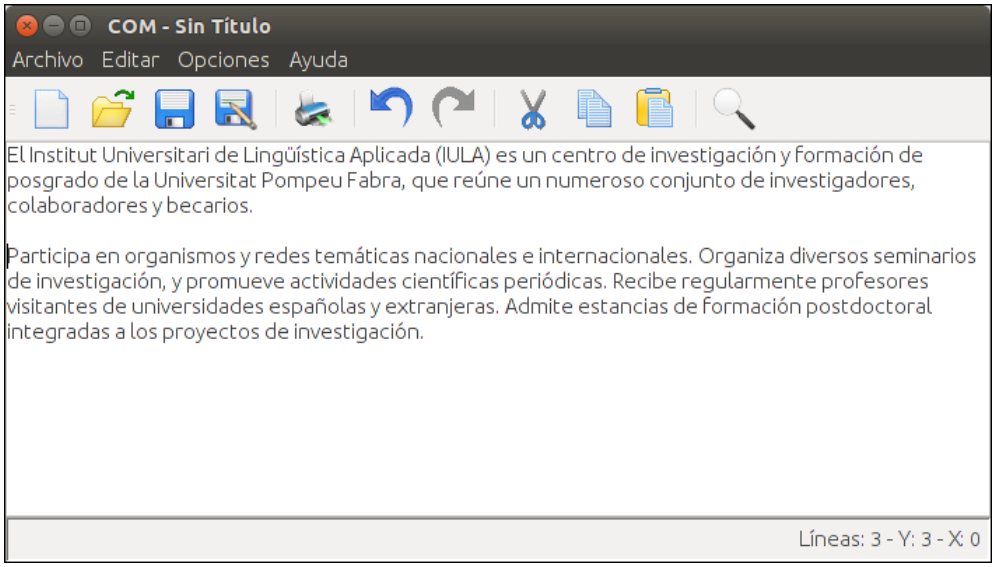

Imagen 19. Área de edición, barra de herramientas no fija.

Nombre: **Mover ubicación de la barra de herramientas**

Descripción: El usuario cambia la ubicación de la barra de herramientas.

Actores: Usuario

Precondiciones: La opción “Fijar barra de herramientas” dentro del menú opciones debe de encontrarse no marcada.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario clicka en el área de puntos de la barra de herramientas y arrastra a la zona donde dese ubicar la barra.
   [El usuario deposita la barra de herramientas en la parte superior de la ventana principal]
   2a. La barra de herramientas se deposita en la parte superior con los botones alineados de manera horizontal.
   [El usuario deposita la barra de herramientas en el lateral derecho de la ventana principal]
   2b. La barra de herramientas se deposita en la parte derecha con los botones alineados de manera vertical.
   [El usuario deposita la barra de herramientas en el lateral izquierdo de la ventana principal]
   2c. La barra de herramientas se deposita en la parte izquierda con los botones alineados de manera vertical.
   [El usuario deposita la barra de herramientas fuera de la ventana principal]
   2d. La barra de herramientas se deposita en una ventana independiente a la principal en la zona donde se haya depositado.
   [El usuario pulsa el botón cerrar X de la parte superior derecha de la ventana que contiene la barra de herramientas]
   3da. La barra de herramientas se muestra en la última posición en la que se encontraba depositada dentro de la ventana principal.

PostCondiciones: Ninguna

Interfaz Gráfica:

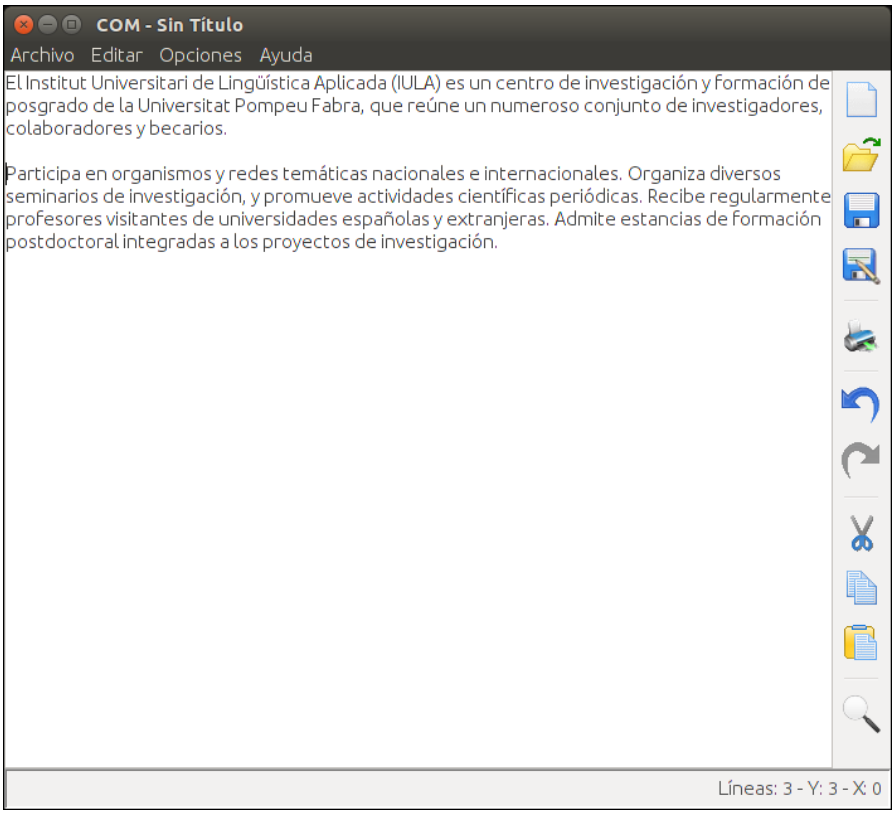

Imagen 20. Área de edición, barra de herramientas fijada a un lateral.

Nombre: **Imprimir**

Descripción: El usuario gestiona las opciones de impresión del archivo.

Actores: Usuario

Precondiciones: El área de edición no se puede encontrar en blanco. Debe de tener algún contenido

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “print” o selecciona la opción “Imprimir” dentro del menú Archivo o pulsa la combinación de teclas CTRL+P.
   [Si el área de edición se encuentra vacía]
   2a. Se abre un nuevo cuadro de diálogo en el que se le muestran las impresoras configuradas para el equipo y se permite al usuario sleccionar las diferentes opciones de impresión.
   [El usuario pulsa Imprimir]
   3aa. Se produce la impresión y se vuelve a mostrar el área de edición.
   [El usuario pulsa Cancelar]
   3ab. Se vuelve a mostrar el área de edición y no se produce la impresión.
   [Si el área de edición no se encuentra vacía]
   2b. No se muestra nuevo cuadro de diálogo ni mensaje de alerta.

PostCondiciones: Ninguna

Interfaz Gráfica:

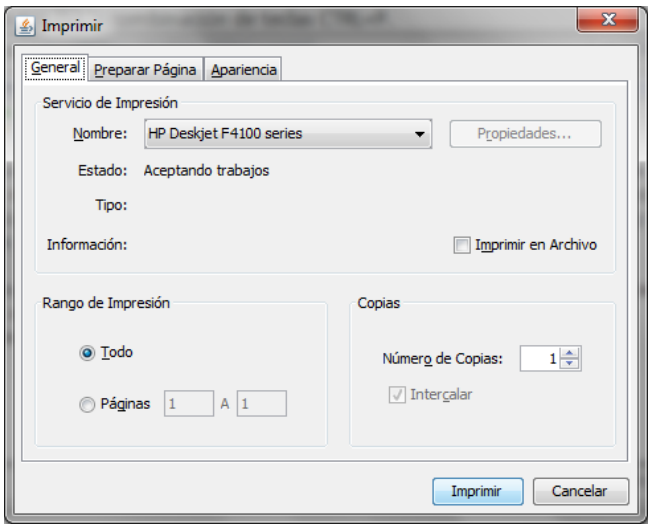

Imagen 21. Cuadro de diálogo para la impresión.

Nombre: **Salir**

Descripción: El usuario cierra la aplicación.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario pulsa el botón X de la esquina superior derecha o selecciona la opción "Deshacer" dentro del menú Archivo o pulsa la combinación de teclas CTRL+Q.
   [El usuario ha realizado algún cambio en el área de edición (de cualquier tipo) y no lo ha guardado]
   2a. Se abre un cuadro de diálogo de alerta para consultar al usuario si desea guardar los cambios antes de abandonar la aplicación.
   [El usuario pulsa el botón Sí]
   3aa. Action Save.
   [El usuario pulsa el botón No]
   3ab. Se cierra la aplicación y los cambios realizados en el área de edición después del último guardado no se almacenan en el archivo.
   [El usuario pulsa el botón Cancelar]
   3ac. Se cierra el cuadro diálogo y se vuelve a mostrar el área de edición con el contenido que tenía en el momento de pulsar X.
   [El usuario ha guardado cualquier cambio que se haya producido en el área de edición]
   2b. Se cierra la aplicación.

PostCondiciones: Ninguna

Interfaz Gráfica:

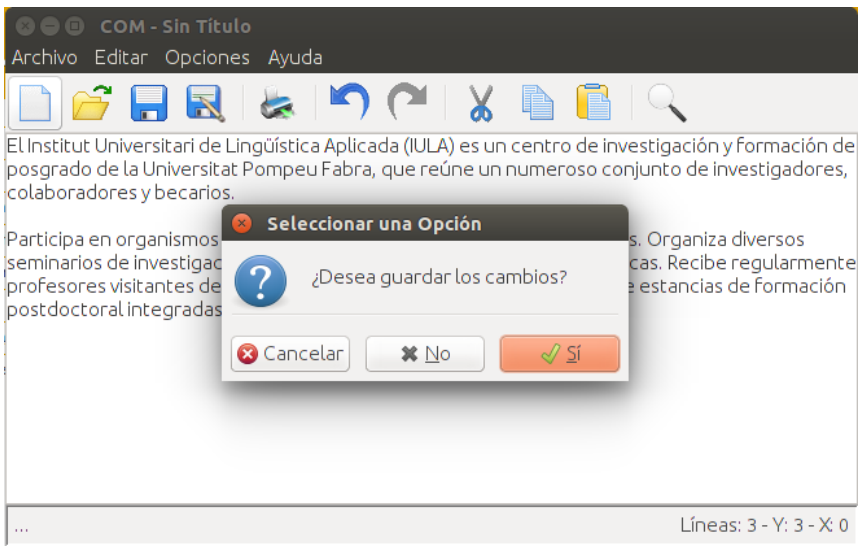

Imagen 22. Cuadro de diálogo de confirmación de cerrado de un archivo.

Nombre: **Corregir**

Descripción: El usuario realiza la revisión ortográfica del texto contenido en el área de edición.

Actores: Usuario

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón “check” o selecciona la opción “Corregir” dentro del menú Archivo o pulsa la combinación de teclas CTRL+R.

[El área de edición se encuentra vacía]

2a. Se muestra un cuadro de diálogo con el mensaje:”Revisión ortográfica finalizada”.

[El área de edición no se encuentra vacía]

[No existen palabras erróneas]

3aa. Se muestra un cuadro de diálogo con el mensaje:”Revisión ortográfica finalizada”.

[Existen palabras erróneas]

3ab. Se muestra un nuevo cuadro de diálogo en el que se van mostrando una por una las palabras erróneas encontradas junto a las sugerencias que la aplicación ofrece.

[El usuario edita la palabra errónea]

4aba. La apariencia de la ventana cambia y los botones “Añadir a diccionario”, “Omitir todas” y “Cambiar todas” se deshabilitan. El botón “Omitir” cambia a “Deshacer”.

[El usuario pulsa el botón Deshacer]

5abaa. En el campo de texto se vuelve a mostrar la palabra errónea original. Se desechan los cambios.

[El usuario pulsa el botón Cambiar]

5abab. La aplicación guarda como corrección la palabra que aparece en el campo de texto que ha sido editada por el usuario.

[Existen más palabras erróneas]

6ababa. Muestra la siguiente palabra errónea junto a las sugerencias.

[No existen más palabras erróneas]

6ababb. Se muestra un cuadro de diálogo con el mensaje: ”Revisión ortográfica finalizada”.

7ababb. Reemplaza en el área de edición las palabras erróneas por las elecciones realizadas por el usuario en cada caso.

[El usuario no edita la palabra errónea]

[El usuario pulsa el botón Omitir]

5abba. La aplicación toma como válida la palabra errónea.

[El usuario pulsa el botón Omitir todas]

5abbb. La aplicación toma como válida la palabra errónea y todas las palabras iguales que puedan aparecer en el texto.

[El usuario pulsa el botón Añadir a Diccionario]

5abbc. La aplicación añade al diccionario permanente la palabra errónea .

[El usuario pulsa el botón Cambiar]

5abbd. La aplicación guarda la preferencia elegida de la lista de sugerencias como corrección para esa palabra errónea.

[El usuario pulsa el botón cambiar todas]

5abbe. La aplicación guarda la preferencia elegida de la lista de sugerencias como corrección para esa palabra errónea y todas las similares que puedan aparecer en el texto.

PostCondiciones: Ninguna

Interfaz Gráfica:

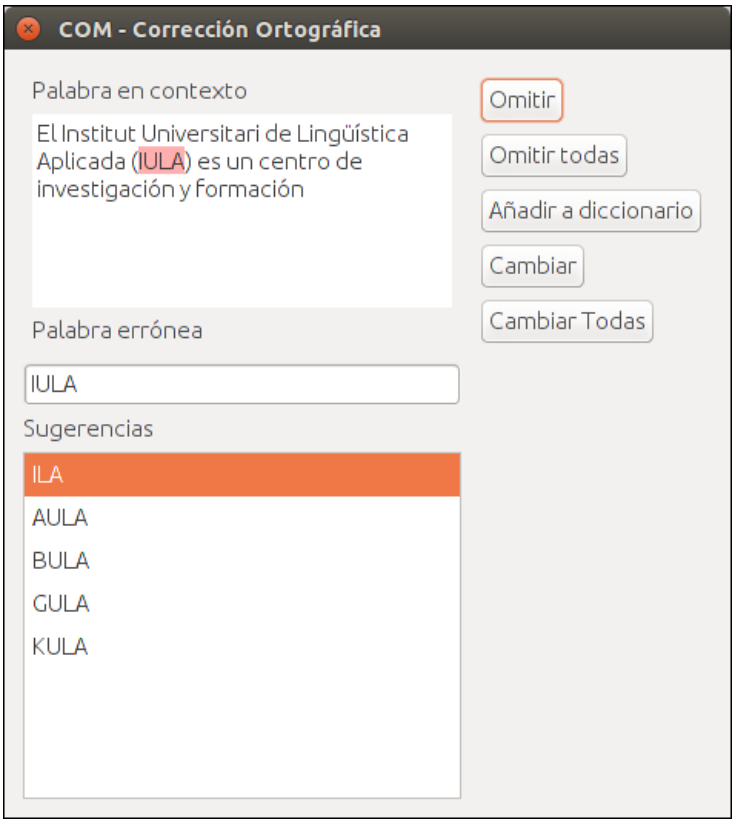

Imagen 23. Interfaz de corrección manual de errores.

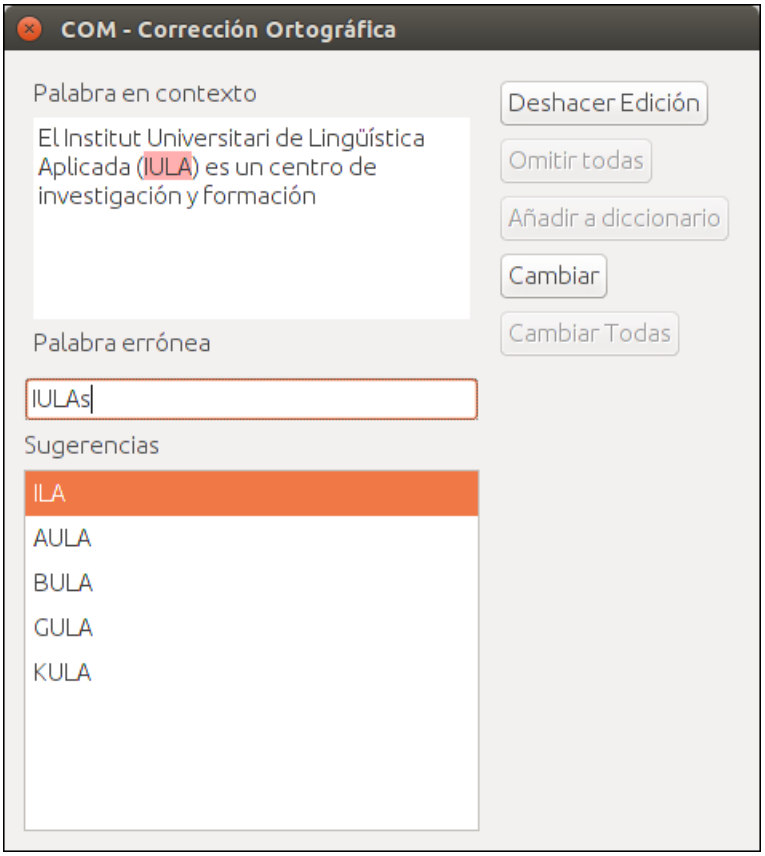

Imagen 24. Interfaz de corrección manual de errores, palabra editada.

Nombre: **Buscar**

Descripción: El usuario realiza la búsqueda de una o varias palabras dentro del área de edición.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción "Buscar" dentro del menú Editar o pulsa la combinación de teclas CTRL+F.
2. Se muestra un nuevo cuadro de diálogo.
   [El usuario introduce una cadena de texto en el campo de texto que el cuadro de diálogo que se le ofrece]
   [El usuario pulsa el botón Aceptar]
   [La cadena se encuentra en el texto]

   3aaa. Se cierra el cuadro de diálogo y se muestra el área de edición con la primera aparición de la cadena introducida por el usuario resaltada en el área de edición.

   [La cadena no se encuentra en el texto]

   3aab. Se cierra el cuadro de diálogo y se muestra el área de edición.

   [El usuario no introduce una cadena de texto en el campo de texto que el cuadro de diálogo que se le ofrece]

   [El usuario pulsa el botón Aceptar]

   3aba. Se cierra el cuadro de diálogo y se muestra el área de edición.

   [El usuario pulsa el botón Cancelar]

   3abb. Se cierra el cuadro de diálogo y se muestra el área de edición.

PostCondiciones: Ninguna

Interfaz Gráfica:

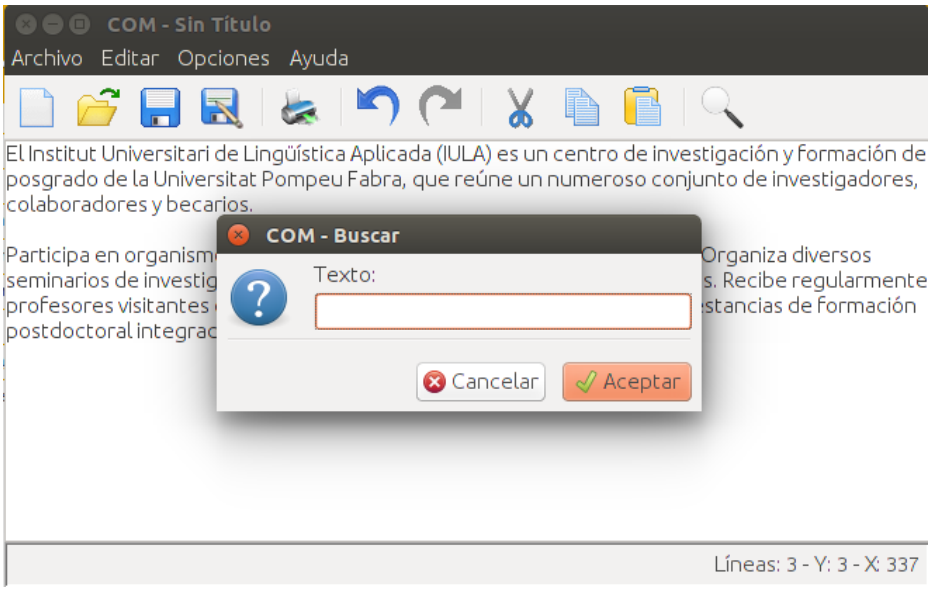

Imagen 25. Interfaz de búsqueda de un término en el área de edición.

Nombre: **Buscar siguiente**

Descripción: El usuario realiza la búsqueda de una o varias palabras que han sido buscadas anteriormente.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción "Buscar siguiente" dentro del menú Editar o pulsa la combinación de teclas F3.
   [El usuario no ha realizado una búsqueda anteriormente]
   2a. Caso de uso: Buscar
   [El usuario ha realizado una búsqueda anteriormente]
   [Existen más apariciones de la cadena de texto a buscar en el área de edición]
   3ba. Se muestra la siguiente aparición de la cadena de texto buscada resaltada en el área de edición.
   [No existen más apariciones de la cadena de texto a buscar en el área de edición]
   3bb. Se muestra la última aparición de la cadena de texto encontrada resaltada en el área de edición.

PostCondiciones: Ninguna

Interfaz Gráfica:

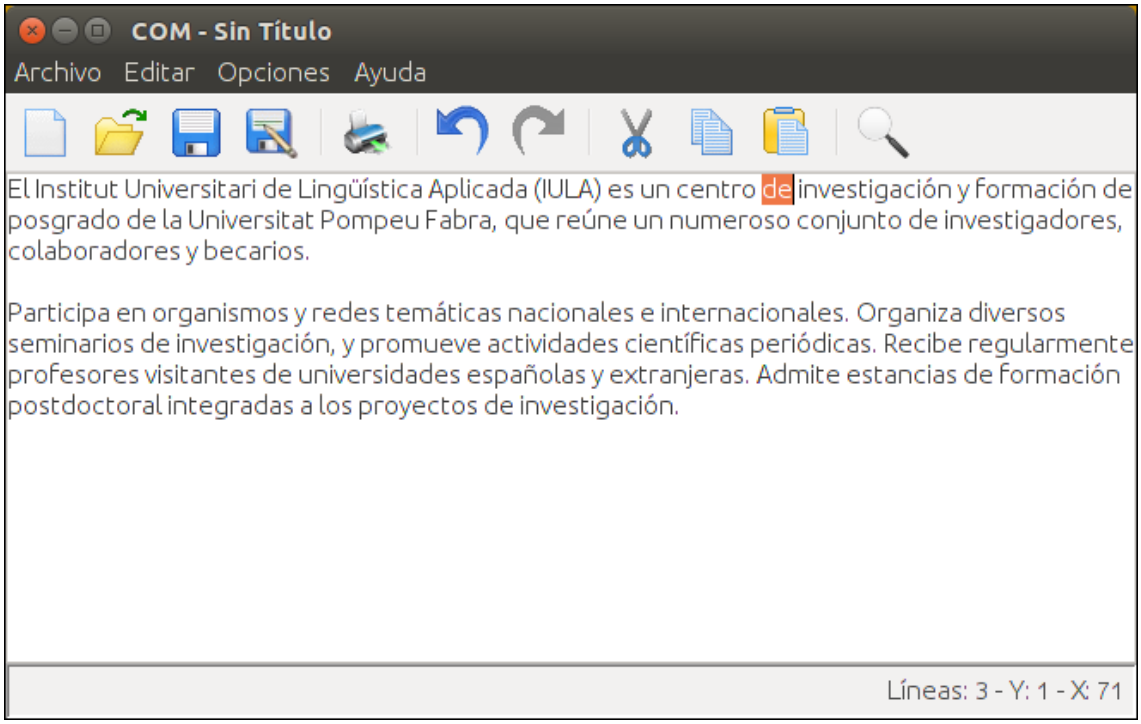

Imagen 26. Búsqueda de siguiente término en el área de edición.

Nombre: **Ajustar Color de Fondo**

Descripción: Permite al usuario seleccionar el color de fondo del área de edición.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción “Color de fondo” dentro del menú Opciones.
2. Se abre un nuevo cuadro de diálogo en el que se muestra al usuario una paleta de colores de la que poder seleccionar el color deseado y un texto de muestra para poder visualizar el posible resultado.

[El usuario clicka sobre un color]

3a. El color seleccionado se muestra en la paleta de Reciente.
4a. En la parte inferior se muestra al usuario el texto de muestra con el color de fondo seleccionado.

[El usuario pulsa el botón Aceptar]

5aa. Se cierra el cuadro de diálogo.
6aa. El área de edición se muestra con el color que el usuario ha seleccionado por última vez.

[El usuario pulsa el botón Cancelar]

5ab. Se cierra el cuadro de diálogo.
6ab. El área de edición se mantiene con el mismo color con el que se encontraba.

[El usuario pulsa el botón Restablecer]

5ac. El texto de muestra con el mismo color con el que se encontraba el área de edición.

[El usuario no clicka sobre un color]

[El usuario pulsa el botón Aceptar]

3ba.Se cierra el cuadro de diálogo.
4ba. El área de edición se mantiene con el mismo color con el que se encontraba.

[El usuario pulsa el botón Cancelar]

3bb. Se cierra el cuadro de diálogo.
4bb. El área de edición se mantiene con el mismo color con el que se encontraba.

[El usuario pulsa el botón Restablecer]

3bc. El cuadro de diálogo permanece igual. No se produce ningún cambio

PostCondiciones: Ninguna

Interfaz Gráfica:

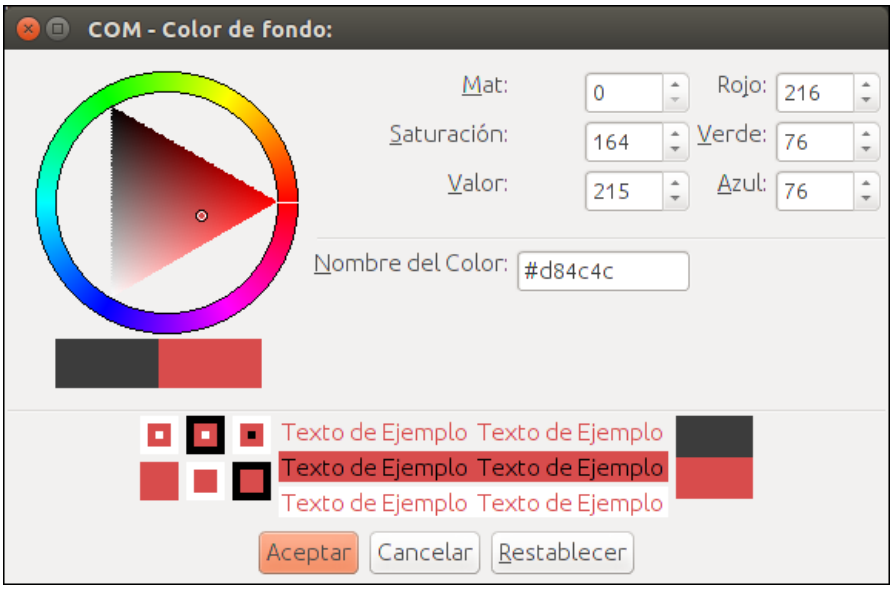

Imagen 27. Cuadro de diálogo de selección de color de fondo.

Nombre: **Ajustar fuente de letra**

Descripción: Permite al usuario seleccionar la fuente, estilo y el tamaño de letra del texto del área de edición.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción "Fuente de letra" dentro del menú Opciones.
2. La aplicación muestra un nuevo cuadro de diálogo en el que se muestra tres listas para que el usuario seleccione la fuente, el estilo y el tamaño de la fuente. De manera automática el cuadro muestra las opciones establecidas en ese momento.

   [El usuario selecciona cualquier opción dentro de los desplegables]

   [El usuario pulsa Ok]

   3aa. Los cambios se efectúan a todos los caracteres del área de edición.

   [El usuario pulsa Cancel o cierra la ventana]

   3ab. No se efectúa ningún cambio en los caracteres del área de edición.

   [El usuario no selecciona ninguna opción dentro de los desplegables]

   [El usuario pulsa Ok o Cancel o cierra la ventana]

   3ba. No se efectúa ningún cambio en los caracteres del área de edición.

PostCondiciones: Ninguna

Interfaz Gráfica:

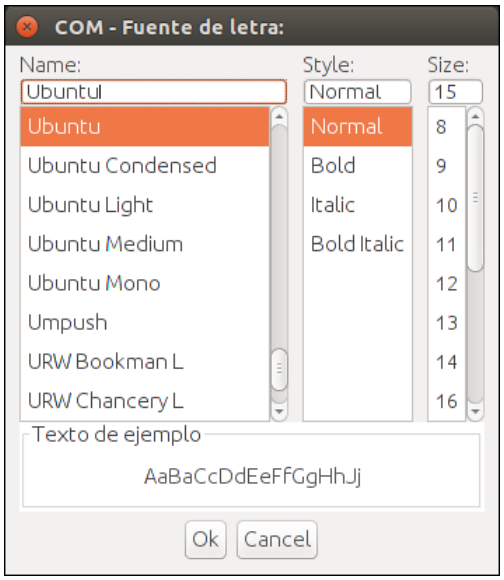

Imagen 28. Cuadro de diálogo de selección de fuente.

Nombre: **Ajustar Color de letra**

Descripción: Permite al usuario seleccionar el color de letra del área de edición.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario selecciona la opción “Color de letra” dentro del menú Opciones.
2. Se abre un nuevo cuadro de diálogo en el que se muestra al usuario una paleta de colores de la que poder seleccionar el color deseado y un texto de muestra para poder visualizar el posible resultado.

[El usuario clicka sobre un color]

3a. El color seleccionado se muestra en la paleta de Reciente.
4a. En la parte inferior se muestra al usuario el texto de muestra con el color de letra seleccionado.

[El usuario pulsa el botón Aceptar]

5aa. Se cierra el cuadro de diálogo.
6aa. El texto del área de edición se muestra con el color que el usuario ha seleccionado por última vez.

[El usuario pulsa el botón Cancelar]

5ab. Se cierra el cuadro de diálogo.
6ab. El texto del área de edición se mantiene con el mismo color con el que se encontraba.

[El usuario pulsa el botón Restablecer]

5ac. El texto de muestra con el mismo color con el que se encontraba el área de edición.

[El usuario no clicka sobre un color]

[El usuario pulsa el botón Aceptar]

5ba. Se cierra el cuadro de diálogo.
6ba. El texto del área de edición se mantiene con el mismo color con el que se encontraba.

[El usuario pulsa el botón Cancelar]

5bb. Se cierra el cuadro de diálogo.
6bb. El texto del área de edición se mantiene con el mismo color con el que se encontraba.

[El usuario pulsa el botón Restablecer]

5bc. El cuadro de diálogo permanece igual. No se produce ningún cambio

PostCondiciones: Ninguna

Interfaz Gráfica:

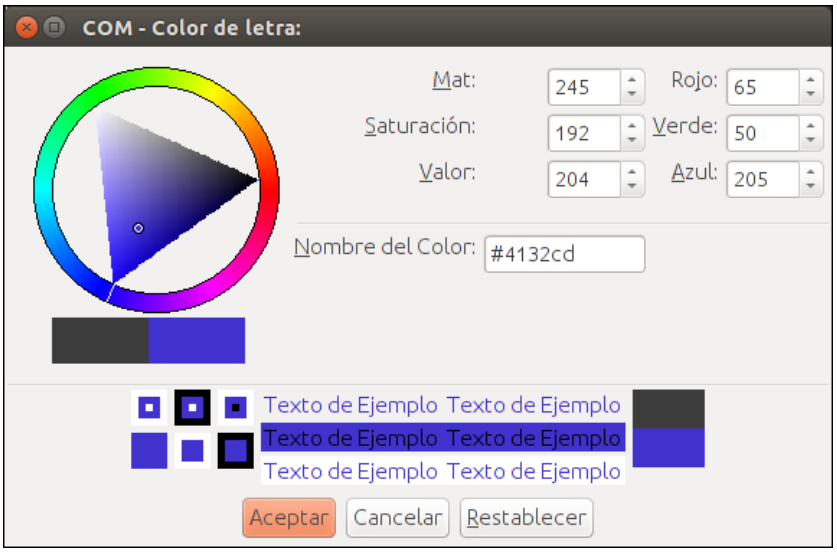

Imagen 29. Cuadro de diálogo de selección de color de letra.

Nombre: **Ampliar diccionario**

Descripción: Permite al usuario introducir su propio diccionario con términos propios o demasiado específicos que el diccionario de la aplicación no contempla.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: La lista de palabras que componen el diccionario del usuario debe de estar en un formato determinado.

Flujo de eventos:

1. El usuario selecciona la opción “Agregar diccionario” dentro del menú “Archivo”.
2. La aplicación muestra un cuadro de diálogo informativo en el que se le indica el formato de los archivos requerido.
3. Action Open.

   [En caso de error por formato incorrecto de ficheros]

   4a. La aplicación muestra un cuadro de diálogo informando del problema.

   5a.Se produce la cancelación del proceso de adición de palabras al diccionario.

   6a.La aplicación muestra de nuevo la pantalla principal que contiene el área de edición.

   [En caso de no producirse error en la lectura del archivo]

   4b. La aplicación muestra un cuadro de diálogo informando de que el resultado obtenido es satisfactorio.

   5b. La aplicación muestra de nuevo la ventana principal que contiene el área de edición.

PostCondiciones: Ninguna

Nombre: **Corrección automática de un texto**

Descripción: Permite al usuario indicar un fichero de texto para que la aplicación realice los cambios de las palabras erróneas de manera automática.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario selecciona la opción “Corrección automática” dentro del menú “Archivo”.
2. La aplicación muestra un cuadro de diálogo informativo en el que se le indica el formato de los archivos requerido.
3. Action Open

[En caso de error por formato incorrecto de fichero]

4a. La aplicación muestra un cuadro de diálogo informando del problema.
5a. Se produce la cancelación del proceso de corrección automática.
6a. La aplicación muestra de nuevo la pantalla principal que contiene el área de edición.
7a. No se crea el fichero con las correcciones aplicadas.

[En caso de no producirse error en la lectura del fichero]

4b. La aplicación muestra un cuadro de diálogo informando de que el resultado obtenido es satisfactorio.
5b. La aplicación muestra de nuevo la ventana principal que contiene el área de edición.
6b. Se crea un fichero en la misma carpeta que contiene el fichero a corregir, con el nombre del fichero seguido de la palabra -new.

PostCondiciones: Ninguna

Nombre: **Corrección automática de una colección de textos**

Descripción: Permite al usuario realizar la corrección automática de todos los textos contenidos en una carpeta.

Actores: Usuario

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario selecciona la opción “Corrección automática” dentro del menú “Archivo”.
2. La aplicación muestra un cuadro de diálogo informativo en el que se le indica el formato de los archivos requerido.
3. Action Open

   [En caso de error por formato incorrecto de fichero]

   4a. La aplicación muestra un cuadro de diálogo informando del problema.

   5a. Se produce la cancelación del proceso de corrección automática.

   6a. La aplicación muestra de nuevo la pantalla principal que contiene el área de edición.

   7a. No se crean el ficheros con las correcciones aplicadas.

   [En caso de no producirse error en la lectura del fichero]

   a. La aplicación muestra un cuadro de diálogo informando de que el resultado obtenido es satisfactorio.
   b. La aplicación muestra de nuevo la ventana principal que contiene el área de edición.
   c. Se crea un fichero en la misma carpeta que contiene el fichero a corregir, con el nombre del fichero seguido de la palabra -new.

PostCondiciones: Ninguna

4.3. Diagrama de clases

A continuación se ofrece una breve descripción de cada una de las clases y su función dentro de la aplicación. Primeramente, analizaremos las clases correspondientes a las vistas.

TPEditor. Clase principal que construye la interfaz del Editor. En esta interfaz se construyen y configuran todas las opciones presentes en la aplicación además de mostrarse el área de edición empleada para la composición de textos.

JFontChooser. Clase que implementa la interfaz encargada de la selección de fuente y tamaño del texto. También se encarga de la recogida y envío de las preferencias del usuario a la clase TPEditor.

JChecker. Clase que implementa la interfaz encargada de la administración de los errores pertenecientes al texto contenido en el área de edición.

PrintAction: Clase que implementa la interfaz encargada de la impresión del documento. Permite al usuario configurar algunos aspectos de la impresión, construye un trabajo de impresión y los atributos correspondientes.

Debido a la utilización del patrón modelo-vista-controlador es necesario el uso de clases que se encarguen de regular el funcionamiento de las interfaces. Debido a la nula complejidad de funcionamiento de las interfaces JFontChooser y PrintAction se desestimó la creación de una clase controlador para ellas, en cambio sí se crearon para TPEditor y JChecker, las cuales se muestran a continuación.

MainController. Es la clase controladora de la interfaz principal, TPEditor, que se encarga de ejecutar las acciones necesarias en caso de afectar a la interfaz y de realizar las llamadas al módelo lógico en caso de ser necesario el uso de éstas.

CheckerController. Es la clase controladora de la interfaz JChecker, se encarga de gestionar la apariencia de la interfaz así como de coordinar las llamadas al módelo lógico encargado de la detección y corrección de los errores.

Además de estas clases también se dispone de clases pertenecientes al modelo lógico, las cuales implementan todas la funciones necesarias para la ejecución del corrector.

Dictionary. Esta clase se encarga de implementar la estructura de datos necesaria para almacenar las palabras que forman el vocabulario de nuestro diccionario. Además realiza acciones de búsqueda, inserción y eliminación de palabras. Esta clase hace uso de la clase TSTNode, necesaria para la representación de cada uno de los nodos del árbol.

TSTNode. Clase que implementa cada uno de los nodos que componen el árbol de búsqueda ternario o TST. No contiene ninguna funcionalidad, solo la de representar mediante la posesión de ciertos atributos cada uno de los caracteres de una palabra dentro del árbol.

LanguageModel. Esta clase implementa las funcionalidades necesarias de un modelo de lenguaje. Desde la tokenización, tratamiento de un corpus y creación de los n-gramas y sus frecuencias al cálculo de probabilidad de un término dentro de una sentencia.

ErrorCaseList: Lista de casos de palabras erróneas. Esta clase se encarga del tratamiento del texto contenido en el documento, detección de las palabras erróneas y sustitución de las mismas en función de las decisiones tomadas por el usuario. Coordina las consultas probabilísticas al modelo de lenguaje y ofrece las palabras correctas más probables en función de los resultados obtenidos

ErrorCase: Caso de palabra errónea. Implementa cada uno de los casos de error ortográfico, conteniendo atributos esenciales para su representación y fácil manejo.

MakeCandidates. Clase que implementa la creación de las palabras reales basados en la distancia de edición de Damerau-Levenshtein. Porporciona la lista de palabras con distancia de edición uno y dos de una palabra dada.

EncodingDetector. Clase implementada haciendo uso de la librería jchardetuniversal capaz de identificar la codificación en la que se encuentra un archivo de texto.

Por último, se muestran dos clases adicionales que han sido implementadas para el tratamiento del corpus así como para obtención de vocabulario a partir de los recursos de Snomed, que no se incluyen dentro de la aplicación pero fueron desarrolladas dentro del proyecto. Snomed-CT (Systematized Nomenclature of Medicine – Clinical Terms) es la terminología clínica multilingüe y codificada de mayor amplitud e importancia desarrollada en el mundo. Nace en el Reino Unido y aporta una terminología de referencia que permite representar la información clínica de forma precisa e inequívoca.

GetURLContent. Clase que obtiene el contenido de una página de la red y realiza el proceso de filtrado y limpieza del texto obtenido, desestimando todo el etiquetado propio del lenguaje de programación de la página para poder registrar el contenido interesante para la aplicación.

Extraction. Clase que implementaba la descompresión y selección de los archivos adecuados que componen el wikicorpus. Esta tarea necesitaba ser desempeñada por el sistema ya que de haberse realizado de manera manual habría supuesto un coste en tiempo demasiado elevado.

Seguidamente, se ofrece un diagrama de las clases existentes en la aplicación y su relación entre ellas.

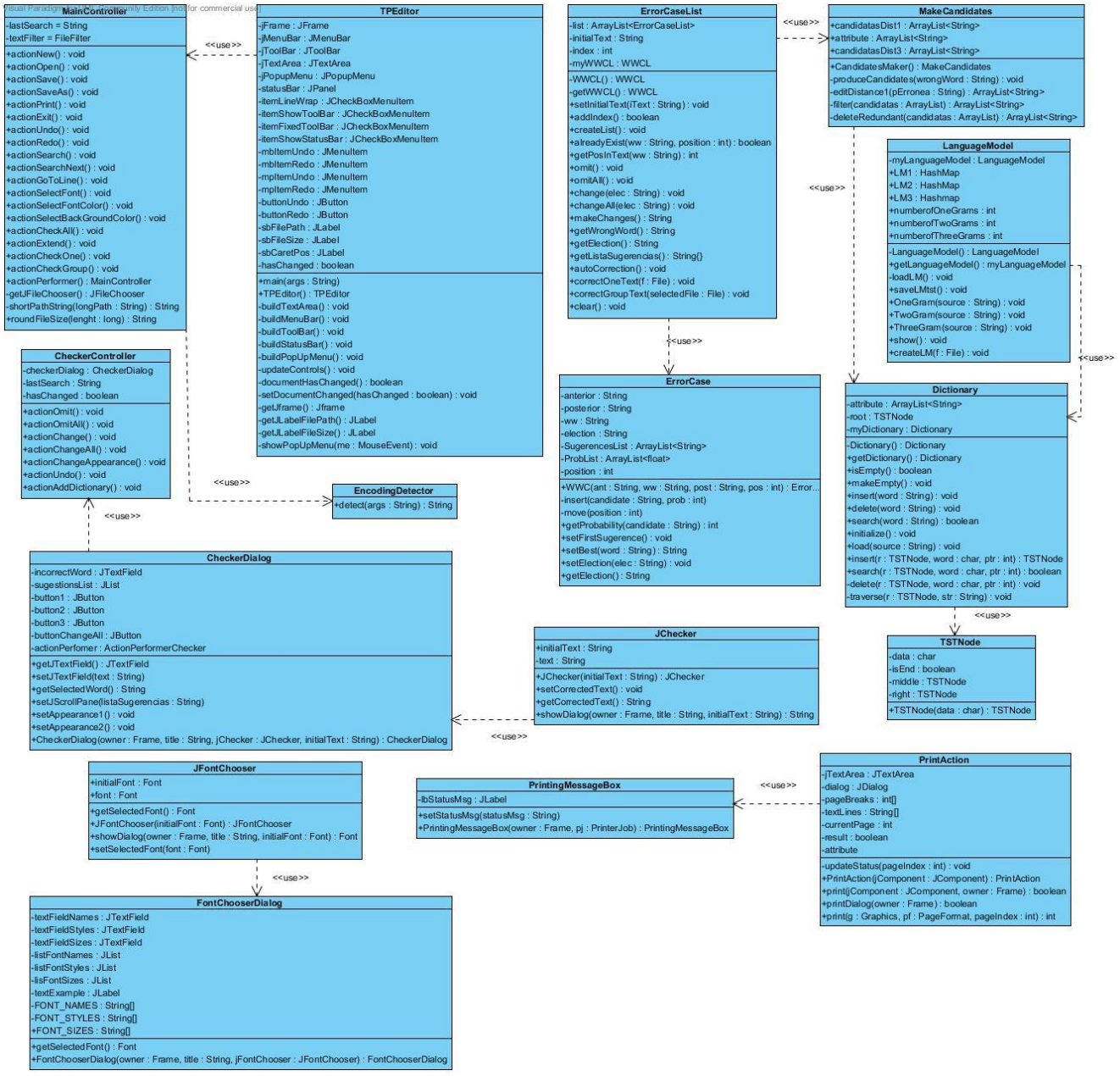

Imagen 30. Diagrama de clases.

Hay que remarcar que algunas de las clases representadas siguen el patrón singleton debido a que se considera que ha de existir una única instancia del objeto y además debe ser accesibles de manera global por las demás clase. Estas clases de tipo MAE son:

·Languagemodel

·Dictionary

·ErrorCaseList

4.4. Diagrama de secuencia

A continuación se muestran los diagramas de secuencia de los métodos y acciones que contemplan una mayor complejidad y que pueden ser representados claramente a través de estos diagramas su funcionamiento. Debido a que se considera que el código del programa que se adjunta junto a esta memoria se encuentra debidamente comentado y la mayoría de los métodos no son demasiado complejos se ha preferido obviar la representación de todas las acciones realizadas por la herramienta, para no saturar al lector.

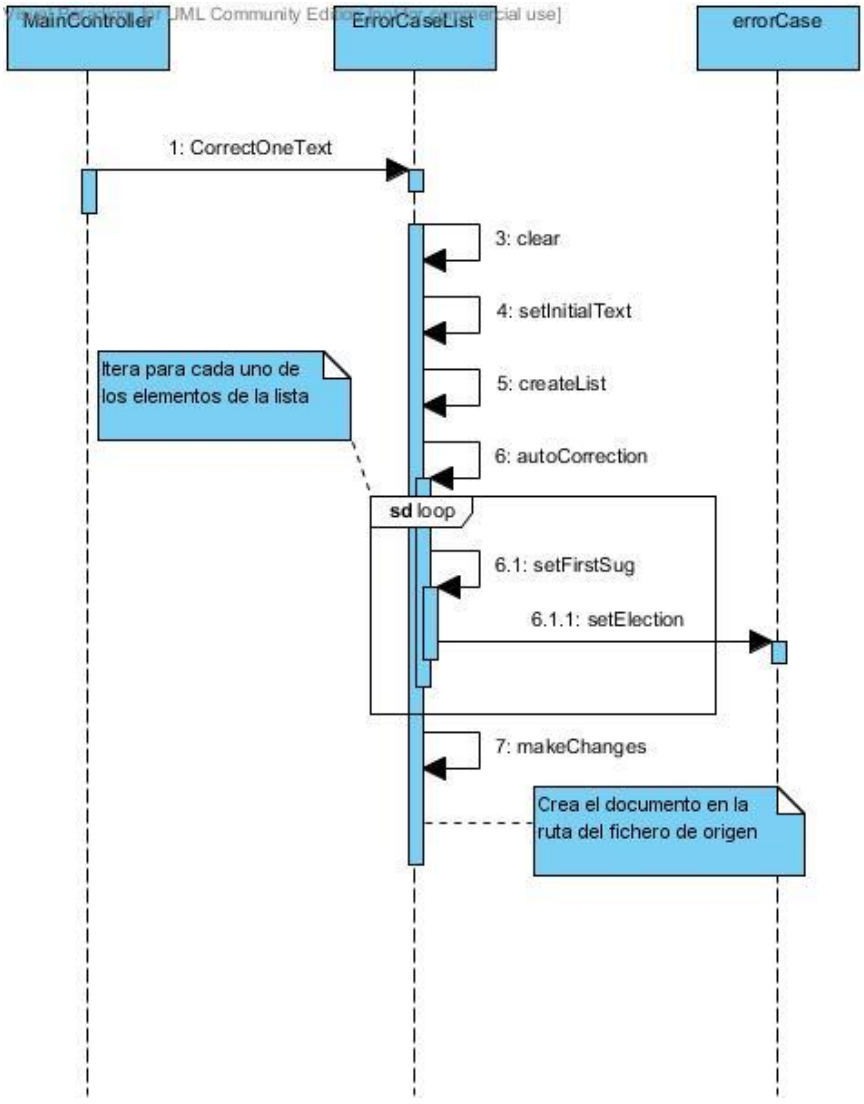

Imagen 31. Diagrama de secuencia de corrección automática de un texto.

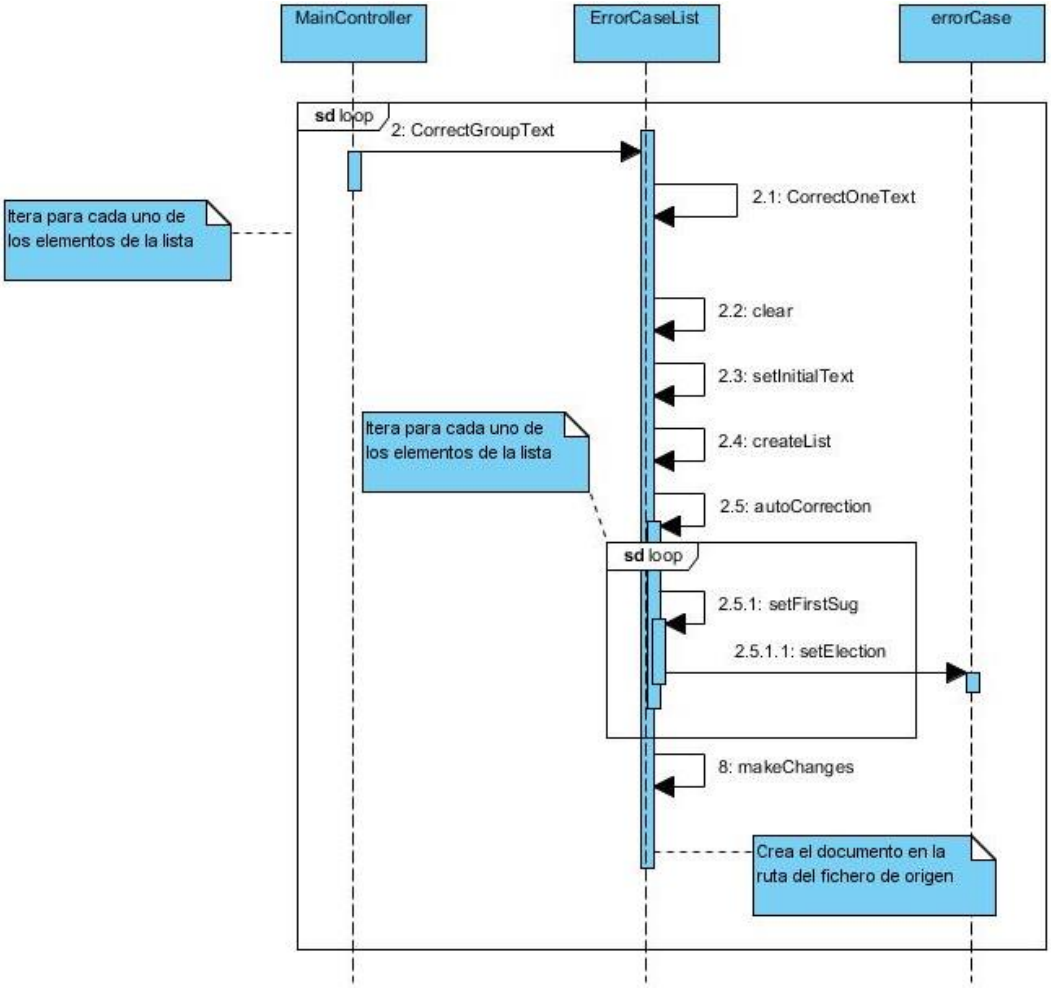

Imagen 32. Diagrama de secuencia de corrección automática de una colección de textos.

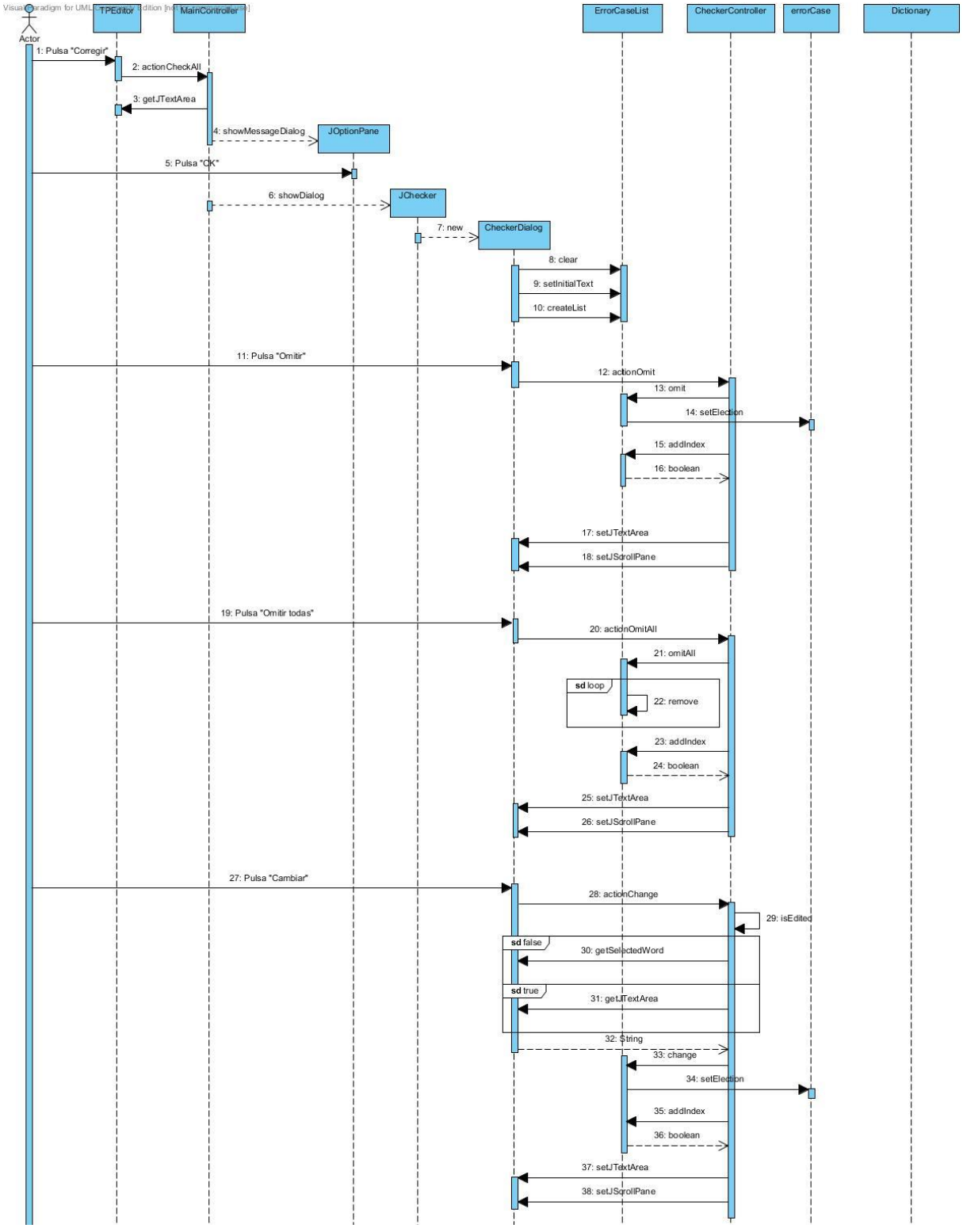

Imagen 33. Diagrama de secuencia de corrección manual de un texto (Parte 1).

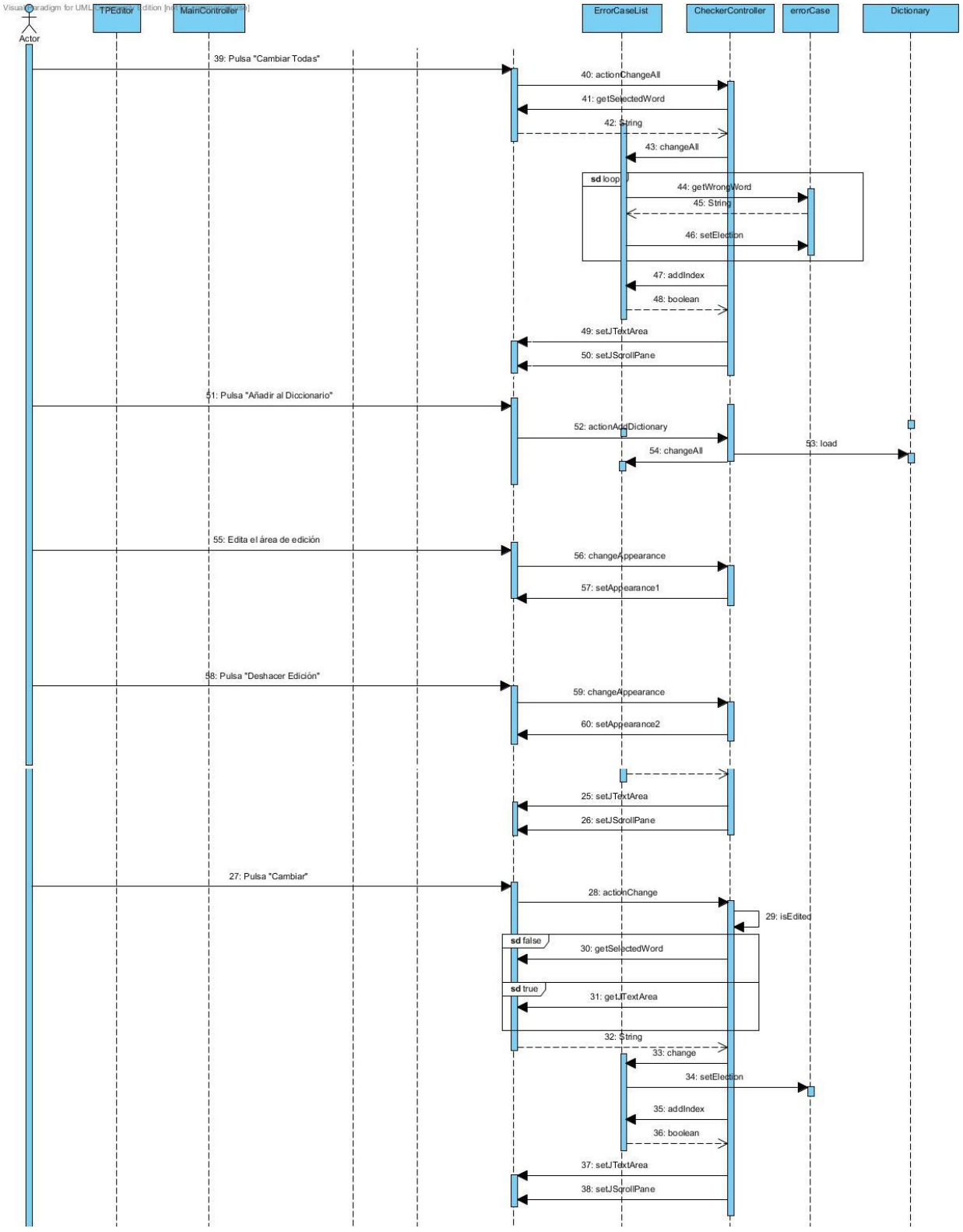

Imagen 34. Diagrama de secuencia de corrección manual de un texto (Parte 2).

5. DESARROLLO

En este apartado se analiza de manera detallada el desarrollo e implementación de cada una de las partes primordiales que componen la aplicación. El orden en el que se produjeron cada una de las partes es el mismo en el que se exponen en este apartado.

Primeramente se realizó la creación del modelo de lenguaje ya que se consideró que podía ser el proceso que podría suponer una carga de horas más elevada. Se entendió así debido a la inexperiencia en este área y la alta probabilidad de que surgieran problemas durante su realización, bien sean técnicos o de tipo conceptual. Una vez realizada la creación y comprobación del modelo de lenguaje se realizó el desarrollo del módulo de creación de palabras correctas candidatas basado en las distancias de edición. Las dos primeras acciones eran independientes entre sí pero ambas eran necesarias para el desarrollo y comprobación del módulo de cálculo de probabilidades ya que este hace uso de ambas. Por último se produjo la creación de la interfaz de usuario, realizada en último lugar ya que se consideró que esta tarea no implicaba la misma complejidad que las tareas anteriores.

5.1. Diccionario

El diccionario es la parte de la aplicación no visible para el usuario que se encarga de almacenar todas las palabras que se toman como correctas. Además de almacenamiento también presta servicios de consulta e inserción.

El diccionario de nuestra aplicación se compone de un compendio de archivos de texto adquiridas de tres fuentes fundamentales:

- El diccionario obtenido con la utilización de la herramienta unmunch perteneciente al paquete Hunspell de Linux a partir del diccionario integrado en la plataforma de software libre OpenOffice. Hunspell es el corrector ortográfico utilizado por herramientas como Google Chrome, Firefox u Opera.
- La colección de medicamentos existente en la página http://www.vademecum.es. Para la cual fue necesaria la implementación de un método específico dentro del proyecto para la obtención del contenido de cada una de las páginas que componen la lista de medicamentos del sitio web. Así como de un algoritmo específico para limpiar y desechar las partes del código no útiles para la aplicación. De esta fuente se obtienen un total aproximado de 4700 nombres de medicamentos.

- La utilización de algunos subconjuntos del vocabulario normalizado SNOMED CT ofrecida por el Sistema Nacional de Salud del Gobierno de España. Se han añadido todos aquellos términos no abarcados por el diccionario original. Se ha considerado oportuno la adición de estas palabras debido a que se puede aceptar la validez de éstas debido a que se encuentran respaldadas por un sistema internacional de terminología médica. Tras el tratamiento de estos textos la aplicación extrae un total de 70000 nuevos términos.

El diccionario que usa la aplicación se basa en una estructura de datos de tipo Ternary Search Tree. Se decide la utilización de este tipo de árbol primordialmente por dos causas. La primera es que este tipo de estructuras evitan la aparición de colisiones como puede ocurrir en las Hashtable y tampoco permite la existencia de duplicados. Además, se considera oportuno recalcar que gracias a la estructura del TST, éste permite realizar la búsqueda de una palabra carácter a carácter. De esta manera se puede ir comprobando la existencia de la palabra introducida en el momento en el que el usuario va introduciendo el contenido por teclado. El TST es un tipo de árbol, también llamado Prefix Tree, en el que cada nodo contiene:

- Un carácter perteneciente a una palabra
- Un valor numérico, 0 o 1. El cual indica si el carácter marca final de palabra.
- Tres punteros, de los cuales:
  - El puntero izquierdo apunta a un nodo cuyo valor es menor que el del propio nodo.
  - El puntero central apunta a otro nodo cuyo valor es igual al del propio nodo. Es decir, para los caracteres que pertenecen a la palabra.
  - El puntero derecho apunta a un nodo cuyo valor es superior al del propio nodo.

El TST es muy similar al Binary Search Tree salvo por el hecho de que el TST necesita de la utilización de un nodo para el almacenamiento de cada uno de los caracteres que componen una palabra mientras que el BST almacena cada palabra en un nodo del árbol. En el siguiente gráfico se muestra cómo son almacenadas las palabras en el TST.

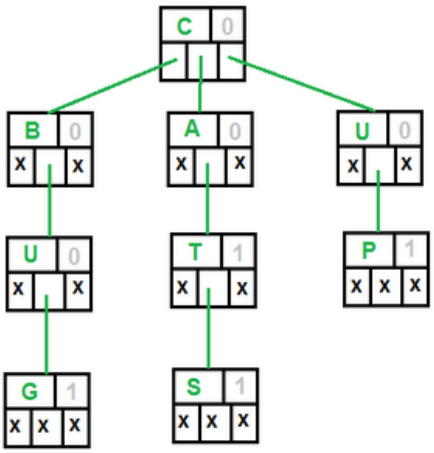

Imagen 35. Ejemplo de estructura TST.
Palabras almacenadas: CAT, BUGS, CATS, UP.
Fuente: http://www.geeksforgeeks.org/ternary-search-tree/ (2015)

Por lo tanto una vez obtenido el vocabulario necesario y haber sido almacenado en los correspondientes archivos de texto plano que contiene una palabra por línea se puede realizar el cargado de estas palabras en la estructura de datos implementada en las clases Dictionary y TSTNode. Es casi evidente la necesidad de utilización de una clase secundaria denominada TSTNode que representa cada uno de los nodos explicados anteriormente con todos los elementos necesarios.

Aunque tradicionalmente en los TST se suele utiliza un valor numérico para indicar el final de palabra en este proyecto se ha decidido utilizar un valor booleano que desempeña la misma función y no afecta al funcionamiento del sistema. Además se considera que no supone un problema en caso de realizarse mejoras en la aplicación.

```
package spellChecker3;
 ** Java Program to Implement Ternary Search Tree

import java.util.Scanner;

/** class TSTNode **/
class TSTNode
{
    char data;
    boolean isEnd;
    TSTNode left, middle, right;

    /** Constructor **/
    public TSTNode(char data)
    {
        this.data = data;
        this.isEnd = false;
        this.left = null;
        this.middle = null;
        this.right = null;
    }
}
```

Figura 1. Código de la clase TSTNode

Se decide la utilización del TST debido a la eficiencia en el tiempo de consulta de palabras y el reducido tamaño de ocupación en memoria. Siendo un método ágil y liviano para los procesos de inserción y consulta generados por la aplicación. La búsqueda de un String de longitud k en un TST que contenga n Strings requerirá como mucho O(log n+k) comparaciones.

5.2. Modelo de lenguaje.

En este apartado se detalla la composición, búsqueda y tratamiento del corpus para la creación de los n-gramas en los que el modelo probabilístico se apoyará para realizar los cálculos de probabilidad de una palabra dentro de una sentencia.

Cuando la herramienta desarrollada encuentra una palabra errónea la clase CandidatesMaker se encarga de generar posibles palabras y comprobará que estas se encuentran en el diccionario, dándolas por válidas o por el contrario descartándolas. Cuando obtenemos una lista de posibles palabras correctas el sistema ha de poder discriminar de alguna manera la probabilidad de que una de estas palabras generadas sea más probable frente al resto. Para ello se tiene en cuenta el contexto de la palabra mal escrita. En función de las palabras que rodean a la palabra errónea el Modelo de Lenguaje se encarga de realizar los cálculos apropiados en función a las tablas de unigramas, bigramas y trigramas previamente creadas y almacenadas.

Descripción de las muestras del corpus

Para la elaboración del modelo de lenguaje de la aplicación se debió realizar una búsqueda exhaustiva por la red de textos de carácter médico. La elección de textos relacionados con este campo se debe al hecho establecido de que un modelo de lenguaje ofrecerá resultados más precisos cuánto más parecidos sean los textos que componen el corpus a los textos a tratar a posteriori.

Debido a la incapacidad de encontrar un número tan elevado de textos relacionados con la medicina, de carácter público y en el formato deseado, se optó por la utilización de un conjunto de textos utilizado muy habitualmente en la actualidad. Este corpus consta de más de 230.000 artículos de temática variada procedentes de la Wikipedia, obtenido de la plataforma de recursos de IULA. Además de este recurso también se hace uso de uno de los subcorpus técnicos pertenecientes a la IULA, ofrecido a través de su plataforma, compuesto de más de cuatro millones de palabras. Este subcorpus del área de medicina contiene estructuras y vocabulario más similar al que se quiere abordar.

Preproceso de las muestras

Tratamiento

Antes de cualquier acción de procesamiento de los textos se debió realizar la extracción y descompresión de los archivos de texto plano necesarios para lo cual se diseñó una clase que no se abarca dentro del instalable pero que fue necesaria para unificar y reunir todos los textos que componen el corpus. Esto es debido a que el corpus perteneciente a la Wikipedia se componía de archivos en diverso formato para cada uno de los artículos. Formatos que no resultaban de valor para el proyecto.

Preprocesamiento

En la actualidad existen diferentes modalidades de software definidas con el objetivo de tokenización de textos, es decir, la separación de cada una de las palabras que los componen. Algunos ejemplos pueden ser las herramientas desarrolladas por IBM y la universidad de Standford. Las cuales están diseñadas para habla inglesa en su gran mayoría.

En el caso de este proyecto se decidió la realización de nuestro propio algoritmo de tokenización. Este algoritmo se encargaría de diferenciar y separar cada una de las palabras. Su funcionamiento se basa principalmente en la separación de las palabras por espacios y por caracteres especiales, entendiéndose por estos caracteres:

¡”·$%&/()=?¿ªº\|@#¬’¡[]{};:,.^*

Para la creación de unigramas estos caracteres se tratan como separadores de sentencias o como caracteres especiales, pero nunca se toman como palabras, ni como parte de ellas. A pesar de ello, para la creación de trigramas y bigramas sí se han de tener en cuenta debido a que estos caracteres pueden marcar el comienzo o el final de una frase. Por ello, algunos se sustituyen por el String <s> que significa comienzo o final de línea. Estos caracteres son los siguientes:

¡”&/()?¿\[]{};:,.

De manera que por ejemplo la línea:

> Los arcos están unidos por un puente, la glabela; la depresión debajo es el nasión.

Se convierte en la línea:

<s> Los arcos están unidos por un puente <s> la glabela <s> la depresión debajo es el nasión <s>

Dando como resultado la creación de los bigramas

<s> los - los arcos - arcos están - están unidos - unidos por – por un – un puente – puente <s> - <s> la – la glabela- glabela <s> …

En resumen, los inicios y finales de línea se tienen en cuenta a la hora de construir los n-gramas. También se tiene en cuenta el contenido de la palabra ya que se ha de diferenciar entre números, si contiene algún tipo de cifra, o palabras. Sustituyendo de igual manera las palabras numéricas por el String <tokenNumber>.

También se ha de remarcar que a la hora de procesar y almacenar cada una de las palabras del texto se realiza el cambio de todos los caracteres en mayúsculas a minúsculas ya que, de otra manera, una diferencia de un carácter en el texto supone un n-grama distinto.

Creación y almacenamiento

Se decide restringir el uso del modelo de lenguaje unigramas, bigramas y trigramas. Esta decisión se toma debido a que se intuye que el volumen de palabras contenidas en el corpus no es lo suficientemente grande como para que la utilización de 4-gramas sea realmente útil, a excepción de unas pocas combinaciones inusualmente frecuentes. El uso de 4-gramas supondría un coste en memoria, tiempo de consulta y cargado de la aplicación que no supondrían un aumento significativo en la calidad de las probabilidades calculadas.

Debido que las estructuras de datos en Java se almacenan en la memoria volátil y el sistema no ofrece una opción de almacenamiento en disco si no es con el uso de librerías externas se plantean dos alternativas. Por un lado, se ofrece la opción guardar los pares n-grama+contador de apariciones en un fichero de texto plano y realizar su lectura y carga en un Hashmap durante el inicio de la aplicación. Por otro, se valora la opción de guardar los pares en tablas de datos e introducir una base de datos embebida en la aplicación, utilizando alguna de las librerías de tratamiento de bases de datos de distribución libre como h2.

Finalmente se decide que el método de almacenamiento de los n-gramas sea un hashmap que se cargue al principio de la aplicación. Esta decisión se toma apoyada en el artículo "Efficient in-memory data structures for n-grams indexing" en el que se realiza una comparación de las estructuras de almacenamiento más utilizadas en la actualidad, comparando su rendimiento por tiempos de consulta, almacenamiento y espacio que ocupa en memoria.

También se considera que la naturaleza de la tabla hash encaja de manera perfecta con el concepto de n-grama ya que se trata de un valor en este caso de tipo String conformado por la porción de texto que compone el n-grama y que actúa como clave, asociado a un valor de tipo numérico que se encarga de contabilizar las apariciones de ese n-grama en el corpus.

Una vez establecida esta decisión se pasa al proceso de creación de los n-gramas y almacenamiento de estos en hashmaps en base al artículo publicado los profesores de la universidad de Berkeley, Adam Pauls y Dan Klein, en el que se realiza un estudio de la utilización de estructuras de datos hashmap para el almacenamiento de los n-gramas de modelos de lenguaje. Se decide la utilización de un hasmap para cada uno de los tipos de n-gramas (1, 2 o 3 palabras). De manera que se recorre el corpus y se van añadiendo los n-gramas en función de su aparición. Si el n-grama ya se encuentra almacenado en la tabla se realiza la adición de una aparición a su contador de lo contrario se añade a la tabla. Después de la creación de todas las tablas se crea un método encargado de realizar la exportación del contenido de cada una de las tablas a tres ficheros de texto, los cuales serán los utilizados durante la ejecución del programa. Estos ficheros se encuentran alojados junto al resto de la aplicación ya que son necesarios para su funcionamiento, se muestran de la siguiente manera:

movimiento inicial:1
selección estudiada:7
exista mejora:1
página en:44
tarde vinieron:1
en prosaísmo:1
página es:4
contratar al:5
motor y:3
denominada química:1

Para realizar el cargado de estos ficheros durante la ejecución de la aplicación se realiza la creación de una clase que será ejecutada al inicio de la aplicación que se encarga de recorrer y guardar en las tablas Hash los pares contenidos en el fichero. Estas serán las tablas que la aplicación utilizará para realizar las consultas del número de apariciones de cada n-grama.

Para ofrecer una imagen general del corpus con el que se trabaja y los resultados obtenidos a partir de su tratamiento se muestran las siguientes cifras. El corpus de artículos de la Wikipedia se compone de un total de 10 millones de tokens (instancia de un texto) y la herramienta obtiene 237.000 palabras distintas. Por otro lado, el subcorpus médico del IULA cuenta con alrededor de medio millón de palabras y se obtienen unas 60.000 palabras distintas.

5.3. Módulo de cálculo de probabilidades (LM y ED).

Debido a la imposibilidad de crear un sistema de distancia de edición ponderado lo suficientemente consistente ya que no se contaba con un criterio claro de ponderación de los errores cometidos en lengua española, se decide la utilización de un modelo de lenguaje estadístico para hacer posible una cierta ordenación de las candidatas obtenidas mediante las operaciones básicas de edición.

Este módulo es el encargado de calcular la probabilidad de las palabras candidatas dentro de la sentencia que contiene una palabra errónea. Es decir calcular la probabilidad de cada una de las posibles soluciones para el entorno que la rodea.

Para la creación de este módulo se toma como punto de partida los trabajos actuales realizados sobre la materia que los autores Dan Jurafsky y Christopher Manning establecen en su curso de la Univerisdad de Stanford. Se trata entonces de un modelo estadítico basado en n-gramas. El cual se basa en la teoría de la probabilidad condicionada, que establece:

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

De la misma manera si aplicamos cambios operacionales lógicos se obtiene:

$$P(A,B) = P(A|B) \cdot P(B)$$

Además esta teoría se puede aplicar a más términos, por ejemplo:

$$P(A,B,C,D) = P(A) \cdot P(B|A) \cdot P(C|A,B) \cdot P(D|A,B,C)$$

Esto es lo que establece la ley de la cadena, que se puede utilizar para calcular la probabilidad de una combinación de términos.

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

Una vez establecidos los fundamentos básicos para el cálculo de las probabilidades de n-gramas se ha de determinar el número palabras de los n-gramas a ser utilizado en el modelo de lenguaje aplicado ya que se pueden aplicar modelos de lenguaje basados en unigramas, bigramas, trigramas y así sucesivamente. Por ejemplo dependiendo del modelo de lenguaje empleado el cálculo probabilístico para la frase:

"Yo veo la casa azul."

La cual, una vez tratada pasa a ser:

<s> Yo veo la casa azul<s>

Según el modelo de unigramas obtendríamos:

$$P(Yo\ veo\ la\ casa\ azul.) = P(<s>) \cdot P(Yo) \cdot P(veo) \cdot P(la) \cdot P(casa) \cdot P(azul) \cdot P(<s>)$$

De manera que sigue la ecucación:

$$P(w_1 w_2 w_3 \dots w_n) = \prod_i P(w_i)$$

Este modelo presenta algunos problemas debido a que no es necesariamente cierto que por tener una mayor probabilidad individual una sucesión de ellas sea lógica. Por ejemplo, la probabilidad de la sentencia "es es es es" será muy elevada debido a que la palabra es tiene una frecuencia realmente alta de aparición, pero la sucesión del mismo término cuatro veces es realmente improbable en una sentencia real. Este es un pequeño ejemplo que demuestra que este modelo no es del todo competente.

Según un modelo basado en bigramas el cálculo de la probabilidad de la frase anterior sería:

$$P(Yo\ veo\ la\ casa\ azul.) = P(Yo|<s>) \cdot P(veo|Yo) \cdot P(la|veo) \cdot P(casa|la) \cdot P(azul|casa) \cdot P(<s>|casa)$$

Para ello se decide que la mejor opción es implementar el cálculo de varios modelos y considerar cuál de ellos presenta mejores resultados frente a casos reales. Estos resultados se exponen en el apartado de pruebas. Para ello se realiza la implementación de del cálculo

probabilístico de un mdelo basado en unigramas, otro en bigramas y otro haciendo uso de la teoría de la cadena. El método de la cadena calcularía la probabilidad de la frase anterior, de la siguiente manera:

$$P(Yo\ veo\ la\ casa\ azul.) = P(<s>) \cdot P(<s>, Yo) \cdot P(<s>, Yo, veo) \cdot P(<s>, Yo, veo, la) \cdot P(<s>, Yo, veo, la, casa) \cdot P(<s>, Yo, veo, la, casa, azul) \cdot P(<s>, Yo, veo, la, casa, azul, <s>)$$

Para eliminar el denominado underflow se decide aplicar la utilización de logaritmos. El underflow es un problema producido durante la multiplicación de las probabilidades condicionadas de cada uno de los n-gramas. Estos productos ofrecen resultados tan bajos que emiten error debido a que el programa no es capaz de abordar datos decimales tan largos. Con la aplicación de logaritmos se consigue obtener datos enteros no muy elevados y ofrecer probabilidades finales manejables.

```
probLog=(int) -(Math.log(prob)+Math.log(prob2)+Math.log(prob3));
```

Otro problema que se presenta a la hora del cálculo probabilístico es la posibilidad de que no exista el n-grama buscado. Es decir, su número de apariciones sería cero. Esto produciría que la probabilidad resultante sería cero o, en caso de haberse aplicado logaritmos, un error operacional al intenta calcular el log(0). Para ello se utiliza la denominada técnica de Smoothing. Algunas de las más destacadas son Good-Turing y Knesser-Ney. En este proyecto se ha utilizado la denominado Add-one, también denominado Additive o de Laplace. Esta técnica asume que cada n-grama aparece en el corpus por lo menos una vez. Por lo tanto se suma una aparición a todos los n-gramas. Esta adición supone tener que sumar de la misma manera en el denominador el número de n-gramas distintos que componen nuestro corpus. La siguiente ecuación muestra el cálculo de un término en función de sus predecesoras:

$$P(w_i|w_1 w_2 \dots w_{i-1}) = \frac{count(w_1, w_2 \dots w_{i-1}, w_i)}{count(w_1, w_2 \dots w_{i-1})}$$

De esta manera, para realizar el smoothing, hemos de sumar una unidad al denominador y el número de n-gramas que componen nuestro modelo.

$$P(w_i|w_1 w_2 \dots w_{i-1}) = \frac{count(w_1, w_2 \dots w_{i-1}, w_i) + 1}{count(w_1, w_2 \dots w_{i-1}) + V}$$

Aunque existen otros métodos que ofrecen mejores resultados, se ha decidido la utilización de este, debido a que como se apunta en el curso impartido por Jurafsky y Manning, es el modelo más utilizado en la actualidad para modelos de lenguaje que contienen un gran número de

ceros. Se ha considerado, por lo tanto que sería el ideal para esta aplicación debido a su sencillez y las características de los n-gramas obtenidos del corpus disponible.

5.4. Módulo de creación de candidatas (ED).

Esta parte del programa se encarga de producir todas las posibles palabras con distancia de edición 1 y 2 a partir de una palabra errónea. También se encarga del filtrado de éstas a través del diccionario, ofreciendo de esta manera solo aquellas palabras existentes que pueden sustituir a la palabra errónea.

Para realizar el módulo de creación de candidatas se ha de tener en cuenta los conceptos de distancia de edición entre dos palabras que se entiende como el número de operaciones mínimas necesarias para transformar una cadena de caracteres en otra. En el caso de la distancia de edición de Leventhstein se entiende por operaciones: la inserción, eliminación y sustitución de un carácter. Por otro lado la teoría de Damerau-Leventhstein incluye además como distancia de edición la transposición de dos caracteres. En nuestro caso se decide utilizar la segunda debido a que se considera que la transposición es un error muy común durante la introducción de texto en documentos.

Según se establece en el artículo de Norvig entre el 80 y el 95% de las palabras erróneas en un texto se encuentra dentro de la distancia de edición 1, es decir, solo es necesaria una operación para obtener la palabra correcta. En base a esta afirmación se considera adecuada la creación de un módulo que cree todas las palabras existentes en el diccionario con distancia de edición uno. Este módulo es de carácter reutilizable de manera que cuando se desee crear todas las palabras con distancia de edición 2 de una palabra, se deberá realizar primeramente la creación de las palabras con distancia de edición 1 y volver a realizar esta creación para cada una de las palabras obtenidas.

Se desestima la creación de palabras de distancia de edición 3 por dos motivos principales. El primero de ellos es el coste en tiempo que supone, ya que se debe crear y comprobar en el diccionario un número de palabras demasiado elevado. Para una palabra de 4 letras se produciría un total de 17500 posibles palabras. Este motivo se ve apoyado por el hecho de que la probabilidad de que la palabra deseada se encuentre en este grupo. Se trata de un coste en tiempo demasiado elevado que no se ve respaldado por un motivo de peso para su realización.

5.5. Interfaz de usuario.

Estructura y componentes

Se trata de una aplicación de escritorio desarrollada utilizando el entorno de desarrollo eclipse y lenguaje Java.

Su funcionamiento se basa en el uso del patrón modelo-vista-controlador (Model-View-Controler). Este sistema permite tener separado la lógica de negocio y los datos de la interfaz de usuario.

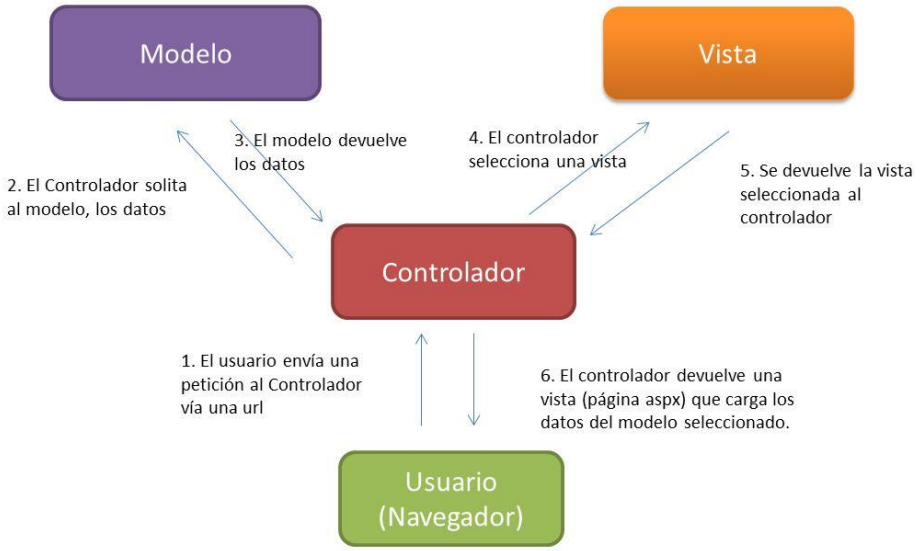

Imagen 36. Diagrama de funcionamiento MVC.

Fuente: http://mind42.com (2015)

En todo momento el controlador se mantiene como mediador entre la parte del modelo y la vista. Se encarga de gestionar en cada momento la información solicitada por la vista para ser mostrada, así como de incluir los datos necesarios en el modelo para el correcto funcionamiento de este.

Para poder facilitar la creación de las vistas se hace uso del plugin WindowsBuilder para Eclipse. Esta herramienta permite la creación y diseño de interfaces gráficas mediante Swing de una manera visual, disminuyendo considerablemente la dificultad de la tarea y la carga de trabajo necesaria para llevarla a cabo.

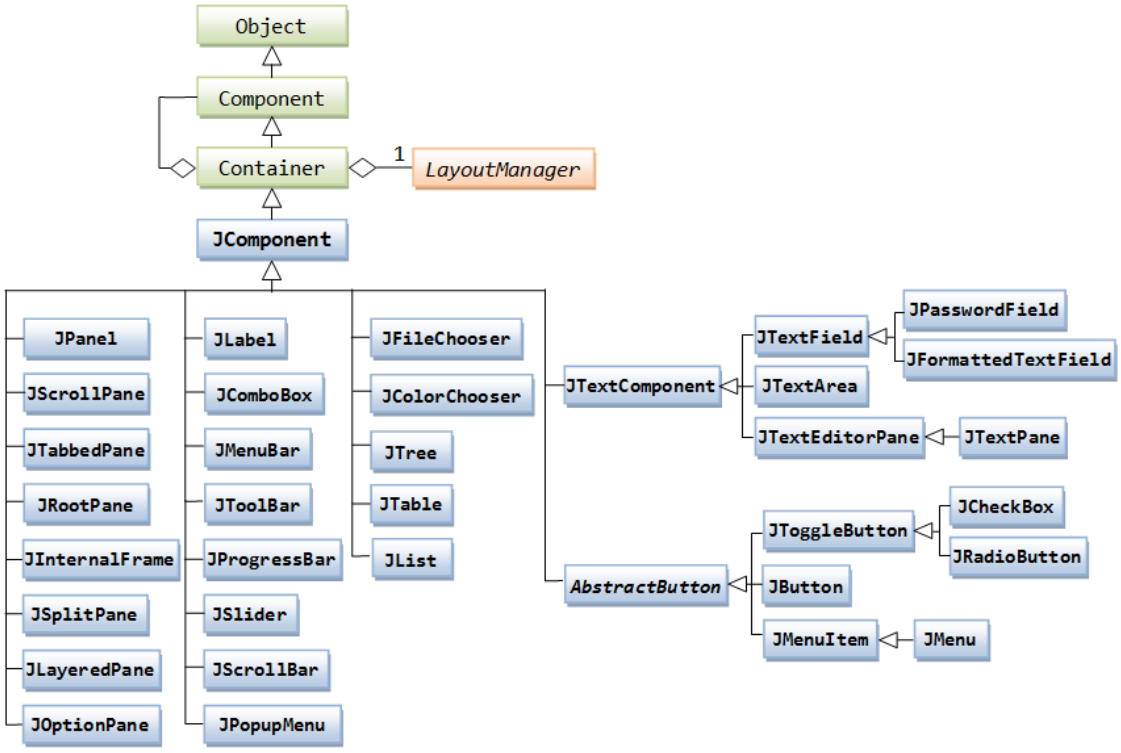

Imagen 37. Componentes de JavaSwing.

Java ofrece, principalmente, dos tipos de vistas denominadas JFrame y JDialog. Existen algunas diferencias significativas que han de ser tenidas en cuenta a la hora de tomar una decisión sobre su uso. Entre las más destacables se encuentran el hecho de que Jframe no admite padres en su constructora mientras Jdialog sí admite otra ventana como padre (bien sea JFrame o JDialog). A esto se suma el hecho de que JDialog puede ser modal mientras que JFrame no. Por estas razones se ha decidido definir la ventana principal como JFrame y las ventanas secundarias como JDialog.

Se ha hecho uso de las clases de tratamiento de eventos AWT incluidas en la librería java.awt.event. Además de estas aplicaciones, también se ha hecho uso de las aplicaciones Swing contenidas en la librería javax.swing.event, aunque no son tan utilizadas como las AWT.

Siguiendo el estilo de los editores de texto existentes se ofrece al usuario la oportunidad de realizar las mismas acciones mediante distintos métodos. La mayoría de las acciones se pueden originar de tres maneras distintas: pulsando el botón correspondiente, mediante atajos con combinación de teclas y mediante las opciones contenidas en cada uno de los desplegables que componen el menú.

Cada uno de los botones de la aplicación así como cada una de las opciones contenidas en los desplegables del menú utilizan el evento ActionEvent-ActionPerformed. De

manera que cuando un botón o una opción del menú son presionadas se hace uso de este evento y se realiza una llamada a la clase ActionPerformer correspondiente. La cual actúa como controlador entre el modelo de datos y la vista mostrada al usuario.

Como se ha comentado anteriormente, la aplicación asocia una combinación de teclas a algunas de las acciones ofrecidas. Estas combinaciones o atajos utilizan el evento KeyEvent-ActionPerfomed, las cuales actúan de la misma manera que si esa opción hubiese sido pulsada, solo que su activación se produce debido a la pulsación por teclado de dos teclas determinadas.

Aunque inicialmente se consideró la utilización de un único controlador finalmente se decidió la utilización de dos controladores distintos diferenciados por la interfaz que controlan y la naturaleza de las acciones llevadas a cabo.

El primero de ellos es el encargado de gestionar los eventos que se producen en el área de edición y la pantalla principal, denominado ActionPerformer.java. Esta clase gestiona principalmente los eventos y funcionalidades del sistema que no tienen interacción con el modelo de datos. Las únicas acciones relacionadas con el modelo de datos que esta clase coordina son la carga inicial del modelo de lenguaje y del diccionario. Como síntesis se podría decir que se encarga de gestionar los eventos producidos en la ventana principal de la aplicación, las acciones más implicadas en la edición de texto, almacenamiento e impresión.

El segundo es el encargado de gestionar la búsqueda y corrección de las palabras erróneas encontradas en el texto, denominado ActionPerformerChecker.java. Controla los eventos producidos en la interfaz correspondiente a “Corrección ortográfica” y realiza todas las llamadas necesarias al modelo de datos: llamada al módulo de creación de candidatas y llamada al módulo de cálculo de probabilidades.

Otro de los posibles aspectos a destacar en el desarrollo de la parte visual de la aplicación es la agrupación de las acciones que la herramienta puede llevar a cabo. Se consideró oportuno, ya que algunas acciones se pueden lanzar de distintas maneras (pulsación de un botón, pulsación en una opción de los menús desplegables o combinación de teclas) que se debería unificar esas llamadas a través de la asociación de cada elemento funcional de la aplicación a un comando de acción. De esta manera cuando se pulsa un elemento se genera un evento y este se encarga de leer el comando que ha producido esa acción y en función de éste realizar una acción u otra.

```
JMenuItem itemPrint = new JMenuItem("Imprimir");
itemPrint.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_P, KeyEvent.CTRL_MASK));
itemPrint.setActionCommand("cmd_print");
```

Figura 2. Creación de un botón.

```
if (ac.equals("cmd_new") == true) {    //opción seleccionada: "Nuevo"
    actionPerformer.actionNew();
} else if (ac.equals("cmd_open") == true) {    //opción seleccionada: "Abrir"
    actionPerformer.actionOpen();
} else if (ac.equals("cmd_save") == true) {    //opción seleccionada: "Guardar"
    actionPerformer.actionSave();
} else if (ac.equals("cmd_saveas") == true) {    //opción seleccionada: "Guardar como"
    actionPerformer.actionSaveAs();
} else if (ac.equals("cmd_print") == true) {    //opción seleccionada: "Imprimir"
    actionPerformer.actionPrint();
} else if (ac.equals("cmd_checkall") == true) { //opción seleccionada: "Corregir"
```

Figura 3. Tratamiento del mensaje generado.

Apertura y guardado de ficheros

Esta interfaz permite la exploración de las carpetas existentes en la máquina en la que se encuentra instalada la aplicación con la finalidad de realizar la apertura de un fichero para su edición, así como del guardado del contenido producido en la aplicación en nuevos ficheros(Guardar como) y ficheros existentes(Guardar).

Esta interfaz hace uso de la clase del componente JFileChooser. Ésta es una clase que ofrece el conjunto de librerías Swing que facilita en gran manera la creación de interfaces para la elección de ficheros y directorios. Y que muestra de manera gráfica las carpetas contenidas en el equipo.

Debido a la naturaleza de algunas de las funcionalidades se aplican filtros para evitar situaciones críticas.

```
fc.setFileFilter(textFileFilter); //aplica un filtro de extensiones
```

De esta manera se indica que en la interfaz solamente se han de mostrar los ficheros de texto, aquellos ficheros cuya extensión sea del tipo .txt. Así el usuario ve restringida su capacidad de elección a aquellos elementos que cumplen los requisitos.

Además de esta situación, también se debe controlar que el usuario no seleccione más de un fichero para su apertura ya que la aplicación no está diseñada para tal uso. Para evitar esta situación se deshabilita esta característica del JFileChooser con la siguiente sentencia:

```
fc.setMultiSelectionEnabled(false); //desactiva la multi-selección
```

En el caso particular de la corrección de un conjunto de textos, en la que se realiza la corrección de todos los archivos de textos contenidos en un directorio (carpeta), se establece que el selector de archivos estará enfocado únicamente a directorios. Esto se realiza mediante la sentencia:

```
fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
```

El selector solo muestra directorios y carpetas, evitando de esta manera la elección de cualquier otro tipo de elemento que no sea directorio.

Durante el proceso de prueba se hizo patente la necesidad de controlar la codificación que el archivo de texto posee debido a que la lectura a través del método BufferedReader de java puede originar caracteres extraños si el archivo no se encuentra codificado en código ANSI. Para ello se hace uso de la librería de código abierto juniversalchardet, la cual tiene como objetivo la detección de la codificación de un texto. Para llevar a cabo esta comprobación se realizó la creación de la clase Detector, la cual toma como entrada el fichero y devuelve un String con el formato detectado. Así, dependiendo del formato detectado se configura el BufferedReader para la lectura de un formato u otro.

```
EncodingDetector ed=new EncodingDetector();
String encoding=ed.Detect(f.toString());
String s="";
try {
    BufferedReader br;
    if (encoding.equals("UTF-8")){
        br = new BufferedReader(new InputStreamReader(new FileInputStream(f), "utf-8"));
    }
    else
    {
        br = new BufferedReader(new InputStreamReader(new FileInputStream(f), "Cp1252"));
    }
```

Figura 4. Detección de la codificación del archivo.

Selección de color de fondo y fuente

Así como para la apertura y guardado de ficheros para la interfaz de selección de color de fondo y color de fuente se hace uso de una de los componentes ofrecidos por la librería Swing, JColorChooser. Esta librería ofrece la posibilidad de mostrar una interfaz pre-diseñada con tres pestañas distintas en la que cada una nos ofrece una manera distinta de elegir los colores.

```
public void actionSelectFontColor() {
    //presenta el dialogo de selección de colores
    Color color = JColorChooser.showDialog(tpEditor.getJFrame(),
                                           "Corrector Médico - Color de letra:",
                                           tpEditor.getJTextArea().getForeground());
    if (color != null) {    //si un color fue seleccionado
        //se establece como color del fuente y cursor
        tpEditor.getJTextArea().setForeground(color);
        tpEditor.getJTextArea().setCaretColor(color);
    }
}
```

Figura 5. Selección de color de fuente.

Con esta librería se consigue de manera sencilla obtener el color elegido por el usuario de manera que el sistema solo ha de hacer llamada al método showDialog de este componente y recoger en una variable el color seleccionado. Después, se establecerá este color como el de fondo o el de fuente dependiendo de la opción seleccionado en la interfaz principal.

Corrección manual de errores

Esta interfaz de diálogo tiene como objetivo gestionar la detección de palabras erróneas, presentación de sugerencias y elección de correcciones. Como se ha comentado anteriormente, para la ventana de corrección manual de errores se ha realizado la creación de un controlador independiente al principal ya que este apartado presenta una complejidad particular y diferente al resto de la aplicación.

Se ha decidido crear una clase de tipo static, denominada JChecker, de manera que sea accesible desde la inicialización del programa. Ésta contiene un método denominado showDialog que es el encargado de mostrar el cuadro de diálogo correspondiente a la corrección manual. A este cuadro de diálogo se le suministrará un String con el texto contenido en el área de edición y devolverá otro String con el texto anterior modificado de acuerdo con las indicaciones realizadas por el usuario.

Para ello la el módelo lógico realiza un filtrado de todas las palabras contenidas en el String de entrada, añadiendo a un ArrayList<> los casos de palabras erróneas encontradas en la clase WWCList. Cada uno de estos casos se almacena en un objeto denominado WWC que contiene:

· String de la palabra errónea

· String de la palabra anterior

· String de la palabra posterior

Se ha decidido almacenar estas palabras como atributos para poder realizar de manera más rápida el cálculo probabilístico de las posibles palabras candidatas sin necesidad de volver a procesar el String para realizar la tokenización.

· Int posición

La posición en el texto original también se registra en cada WWC cuando se realiza la creación de la lista. Esto es debido al hecho de que para la tokenización se produce el reemplazamiento de algunos caracteres por Strings de mayor longitud lo que provoca que la posición que ocupa una palabra en el String original no sea la misma que en el String tokenizado. Además es necesario el uso de esta posición ya que una misma cadena de caracteres puede aparecer varias veces en un texto y no siempre ha de ser reemplazada de la misma manera. Es necesaria la diferenciación de cada una de las palabras erróneas, aunque sea trate del mismo error.

· Lista de sugerencias

Será la lista que se utilice para almacenar las palabras sugeridas que serán mostradas al usuario para que él decida cuál es la más adecuada. Esta lista de sugerencias lleva asociada una Lista de probabilidades similar, la cual será usada para ordenar las palabras generadas según la probabilidad obtenida en base al modelo de lenguaje.

La palabra que el usuario establece como corrección se guardará en otro atributo de la clase, String de la palabra seleccionada. De esta manera cuando el usuario termina la supervisión de todas las palabras erróneas el sistema recorre la lista de todos los casos de WWC y realiza el cambio de la palabra errónea que se encuentra en una determinada posición por la palabra seleccionada y devuelve el String corregido a la interfaz principal.

6. PRUEBAS Y VALIDACIÓN

La fase del proyecto que se documenta en este apartado se divide en dos secciones bien diferenciadas. Por una parte se presenta la necesidad de probar cada una de las funcionalidades de los elementos de la interfaz y por otro la de probar la eficacia de los sistemas de corrección implementados.

Para realizar la validación del correcto funcionamiento de las interfaces es necesario la utilización de estas en el mayor número de casos posibles, cambiando las circunstancias en las que se ejecuta cada acción y comprobando que la respuesta del sistema es la deseada. Para ello se realiza una recogida de todos los posibles casos y se lleva a cabo la realización de cada uno de ellos. Evidentemente durante este proceso se han encontrado fallos que han debido ser solucionados. Además se ha tenido en cuenta la situación en la que el error se ha producido y se ha tenido en cuenta para la ejecución de otras funciones, dando como resultado nuevos casos de prueba.

Se presta especial atención a los casos en los que el sistema debe mostrar mensajes de error o alerta para comprobar que estos casos quedan resueltos y se ejecutan con corrección en cada una de las posibles situaciones.

Para probar la eficacia del corrector implementado se procede a recoger textos con erratas de distintos sitios web enfocados a la medicina. Se ha decidido que la variedad en la procedencia de los textos es un factor importante para asegurar que la naturaleza de los errores no es siempre la misma.

Principalmente los textos recogidos proceden de dos fuentes. Una de ellas es el sitio web http://www.portalesmedicos.com que cuenta con un foro dedicado a medicina, slaud y enferemería con temáticas variadas. La otra es un pequeño corpus de textos de foros de medicina recogids en internet a disposición en el área de recursos la Universidad Carlos III de Madrid.

En total se trata de un corpus de unas 6000 palabras en las que se ha realizado una búsqueda de los términos erróneos que deseamos que el corrector identifique. Seguidamente se ha pasado a la corrección automática del fichero a través de la funcionalidad implementada en el programa y se ha procedido a la comprobación de las correcciones adjudicadas por el sistema. Como se comentó en el apartado de desarrollo, se realiza la corrección a través de los tres métodos implementados, el primero de los métodos realiza el cálculo de la probabilidad de la palabra candidata basándose en la regla de la cadena, el segundo lo realiza a través de un

modelo basado en unigramas y el tercero actúa como un modelo de lenguaje basado en bigramas. A través de este proceso, se trata de observar las diferencias en los resultados de los tres modelos y evaluar cuál puede resultar más eficaz.

En este corpus se presentan un total de 213 errores de tipo ortográfico y tras realizar la corrección automática del texto mediante la utilización de los tres métodos se observan los siguientes resultados porcentuales.

Módelo	Porcentaje
Cadena	94.24%
Unigrama	87.6%
Bigrama	94.7

Tabla 2. Porcentajes de acierto según modelo de lenguaje.

Mediante el análisis de las correcciones realizadas por el sistema se puede destacar que los modelos basados en la teoría de la cadena y el basado en bigramas, ofrecen mejores resultados ante palabras sencillas. Por ejemplo, el modelo basado en unigramas ante la palabra “mas” ofrece como corrección la palabra “las” cuando es evidente que en la mayoría de los casos la solución adecuada será “más”, gracias a que los otros dos modelos hacen uso del contexto que rodea la palabra errónea ofrece soluciones más coherentes con el sentido de la sentencia.

También se observa que el corrector carece de eficacia ante palabras cuya distancia de edición es superior a 2, independientemente del modelo de lenguaje implementado, ya que nuestro creador de candidatas no crea las palabras con distancia de edición igual o superior a 3 debido a los costes de tiempo que supondría para el sistema.

Para completar el apartado de pruebas también se realiza la corrección del texto a través de la interfaz de corrección manual, comprobando que la palabra correcta deseada se encuentra en la lista de las 5 palabras sugeridas. Obteniendo como resultado las siguientes cifras porcentuales.

	Cadena	Bigrama	Unigrama
1ª Opción	94.24	94.7	87.6
2ª Opción o superior	95.15	95.3	88.26
3ª Opción o superior	95.43	95.64	88.52
4ª Opción o superior	95.84	95.87	88.67
5ª Opción o superior	96.23	96.54	88.86

Tabla 3. Porcentajes de acierto en las cinco primeras opciones, según modelo de lenguaje.

Hay que destacar que la mayoría de los casos que forman ese porcentaje residual de no aparición en las 5 primeras sugerencias es debido a que se trata de palabras no contempladas en el diccionario o palabras con distancia de edición superior a 2.

En general se pueden considerar satisfactorios los resultados obtenidos, pudiendo destacarse la posibilidad de márgenes de error superiores ante la aparición de un corpus con un número de abreviaturas elevado. Este fenómeno se produce debido a la irregularidad y la falta de estandarización en el uso de abreviaturas en el área de la medicina.

A continuación se muestra un pequeño ejemplo de los mensajes impresos por pantalla para la comprobación del funcionamiento de la herramienta. Merece la pena recalcar la complejidad de la corrección en textos de esta naturaleza por que no se cumplen unos estándares de utilización de abreviaturas lo que hace tremendamente difícil la corrección de éstas. Se puede apreciar que para el término LsCsSs la aplicación la ha mantenido como estaba ante la imposibilidad de encontrar una palabra sustituta posible.

La palabra errónea es:VARON
Pone el cambio a: VARÓN
La palabra errónea es:ATUTXA
Pone el cambio a: ATUTXA
La palabra errónea es:EXPLORACION
Pone el cambio a: EXPLORACIÓN
La palabra errónea es:LsCsSs
Pone el cambio a: LsCsSs
La palabra errónea es:ANALITICA
Pone el cambio a: ANALÍTICA
La palabra errónea es:EVOLUCION
Pone el cambio a: EVOLUCIÓN
La palabra errónea es:ANTIBIOTICO
Pone el cambio a: ANTIBIÓTICO

7. CONCLUSIONES

Una vez finalizado el proyecto y analizando los resultados obtenidos, se puede considerar que los objetivos iniciales se han visto satisfechos. Si bien es cierto que los objetivos principales y las funcionalidades requeridas se han visto cubiertas es en el área de la planificación donde se ha detectado una clara inexperiencia. Queda demostrado que la falta de experiencia a la hora de llevar a cabo proyectos de este tamaño repercute en los plazos y las estimaciones previas.

Si realizamos una pequeña comparación de las horas estimadas para la ejecución de cada una de las partes con el tiempo aproximado real se observan notables diferencias.

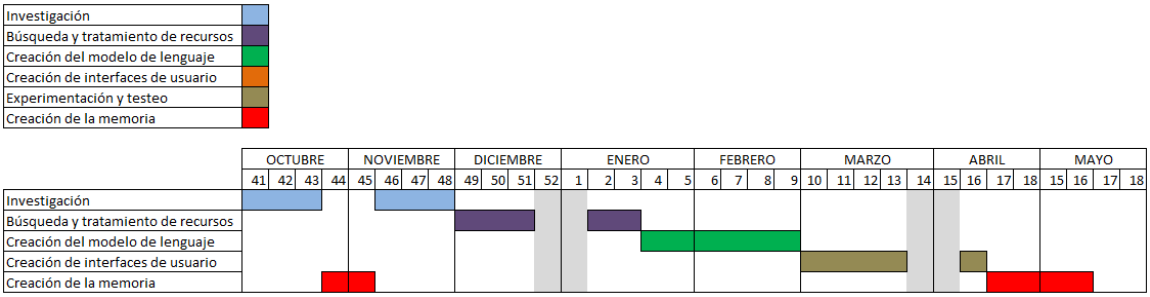

Imagen 38. Diagrama de Gantt, planificación temporal del proyecto final.

Es evidente que las estimaciones iniciales se vieron claramente violadas durante la elaboración del proyecto. Una de las causas principales es error de estimación de las horas semanales empleadas en el desarrollo del proyecto. En una fase inicial se estimó una inversión de 20 horas semanales cuando en la realidad no fue posible la inversión de más de 15 horas semanales lo cual retraso ostensiblemente la finalización del proyecto.

También hay que recalcar que algunas de las tareas se estimaron en un número de horas inferior al que realmente se han empleado. Se achaca este fallo a la inexperiencia en la realización de proyectos de esta envergadura. En la siguiente tabla se muestra la diferencia entre las horas estimadas y las reales empleadas.

Tarea	Horas estimadas	Horas empleadas reales
Investigación	70	90
Búsqueda y tratamiento de recursos	50	70
Creación del modelo de lenguaje	90	90
Creación de interfaces de usuario	80	80
Experimentación y testeo	80	90
Creación de la memoria	80	90
Total	450	510

Tabla 4. Tabla comparativa de tiempo estimado y tiempo real empleado.

Además, este proyecto presentaba un reto especial ya que se trataba de adentrarse en un área de la informática totalmente desconocida para el autor del proyecto. La parte visual de la aplicación no ha supuesto un esfuerzo demasiado grande ya que durante la realización del grado se han adquirido multitud de conocimientos de diseño y programación. Aunque la creación de las interfaces ha sido en principio sencilla, los primeros pasos en cuanto a la asimilación de los conceptos de Procesamiento de Lenguaje Natural y Modelos de lenguaje no lo fue tanto.

Si bien es cierto que las técnicas de implementación utilizadas en el Modelo de Lenguaje que usa la aplicación no son las más sofisticadas de las existentes, hay que recalcar que el trabajo de documentación ha sido muy arduo. Como ya se ha citado en los apartados de Estado del arte y Antecedentes la realización de este proyecto se basa principalmente en las enseñanzas impartidas en el curos de Dan Jurafsky y Christopher Manning ha servido de punto de partida para comenzar a trabajar. El problema que se presenta es que la mayoría de los recursos son de carácter muy limitado, no existen grandes trabajos acerca de la materia de carácter libre.

Otro punto que ha supuesto ciertos problemas ha sido la tokenización y tratamiento de los textos que forman el corpus. Debido a la inexperiencia, se ha tenido que realizar un proceso de prueba y repetición hasta conseguir los resultados deseados, ya que la variedad en los textos (tabulaciones, saltos de página, gran variedad de caracteres) supone una gran variedad de casos no contemplados inicialmente y que pueden suponer errores en la creación de los n-gramas.

Merece detenerse a recalcar un tema que a priori no había sido demasiado meditado y que después tuvo que ser solucionado como es el del formato de los datos. Es un punto que normalmente no se toma en cuenta pero de vital importancia en este proyecto en tres de sus

apartados la creación del diccionario, el procesado del corpus para la creación de los n-gramas y la apertura y edición de archivos de texto existentes. Para ello se ha tenido que hacer uso de una librería de uso libre jchardetuniversal que suministraba una vía útil para averiguar la codificación de un fichero de texto.

8. TRABAJO FUTURO

Como trabajo futuro se plantea la mejora de aspectos relacionados con la experiencia del usuario en el área de detección y corrección de errores. Por ejemplo, se considera interesante realizar un cambio en la manera en la que el usuario visualiza las palabras durante el proceso de corrección manual. Sería adecuado mostrar el contexto en el que se enmarca la palabra y, o bien mostrar la frase en la que se engloba la palabra errónea en la misma interfaz de corrección o bien, mostrar en segundo plano la palabra en el área de edición, resaltada de alguna manera. Es decir, mostrar este apartado siguiendo los convencionalismos de los correctores actuales más establecidos.

Otro cambio que se podría realizar, es el de que la corrección de las palabras se produzca al tiempo que van siendo introducidas por teclado. Con esta mejora se podría realizar la corrección a la vez que el texto se completa y las palabras erróneas se mostrarían resaltadas de alguna manera, permitiendo al usuario visualizar las sugerencias realizando click derecho en la palabra deseada. Una de las ventajas de haber utilizado un TST como estructura de datos para nuestro diccionario es que permitiría esta ampliación de una manera mucho más sencilla, ya que esta estructura permite realizar la búsqueda carácter a carácter.

El trabajo futuro se podría resumir en la mejora de la experiencia del usuario con la aplicación, facilitando y haciendo más sencillo su uso. Adaptando la herramienta desarrollada a los estándares actuales para que el usuario no note que se trata de una herramienta distinta a las acostumbradas.

9. BIBLIOGRAFÍA

[Java (2015)] Java (2015), URL https://www.java.com.

[Eclipse (2015)] Eclipse (2015), URL https://www.eclipse.org/.

[juniversalchardet (2015)] juniversalchardet(2015), URL https://code.google.com/p/juniversalchardet/.

[Dropbox (2015)] Dropbox (2015), URL https://www.dropbox.com/.

[Draw.i.o (2015)] Draw.i.o (2015), URL https://www.draw.io/.

[Visual Pradigm (2015)] Visual Paradigm (2015), URL http://www.visual-paradigm.com/.

[Cacoo (2015)] Cacoo (2015), URL https://cacoo.com

[Peter Norvig (2007)] How to write a spelling corrector, URL http://norvig.com/spell-correct.html.

[Dan Jurafsky,Christopher Manning (2015)] Stanford University, Natural Language Processing, URL https://class.coursera.org/nlp/lecture/preview.

[Daniel Robenek, JanPlatos, Vaclav Snásel (2014)] Ternary Tree Optimalization for n-gram indexing, URL http://ceur-ws.org/Vol-1139/paper6.pdf.

[Daniel Robenek, JanPlatos, Vaclav Snásel (2014)] Efficient in-memory data structures for n-grams, indexing, URL http://ceur-ws.org/Vol-971/paper21.pdf.

[Roger Mitton (1996)] Spellchecking by computer, Birkbeck College, URL http://www.dcs.bbk.ac.uk/~roger/spellchecking.html.

[Michael Collins (2013)] Course notes for NLP by Michael Collins, Columbia University, URL http://www.cs.columbia.edu/~mcollins/lm-spring2013.pdf.

[Adam Pauls, Dan Klein (2011)] Faster and Smaller N-Gram Language Models , URL http://nlp.cs.berkeley.edu/pubs/Pauls-Klein_2011_LM_paper.pdf.

[The Stanford natural Language Processing Group (2015)] The Stanford natural Language Processing Group, URL http://nlp.stanford.edu/index.shtml.

[Apache OpenNLP (2015)] The Apache Software Foundation, URL https://opennlp.apache.org/.

[Hunspell (2015)] Huspell, URL http://hunspell.sourceforge.net/.

[Plataforma de recursos IULA(2015)] Plataforma de recursos IULA, Universitat Pompeu Fabra, URL http://www.iula.upf.edu/recurs01es.htm.

[H2 Database Engine (2015)] H2 the Java SQL Dtabase, URL
http://www.h2database.com/html/main.html.