eman ta zabal
zazu

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA INDUSTRIAL DE BILBAO

GRADO EN (TITULACIÓN)

TRABAJO FIN DE GRADO

2014 / 2015

EDITOR DE TEXTOS CON CORRECTOR ORTOGRÁFICO PARA TEXTOS MÉDICOS

RESUMEN

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

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 necesaria 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.

3. ANTECEDENTES

3.1. Editores de texto

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.

3.2. Corrección ortográfica

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

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.

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).

Redes neuronales.

Claves de similitud (Similarity Keys).

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. Por operaciones se entienden sustitución, transposición, inserción y eliminación.

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.

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. 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.

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.

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 menor 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.

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. DESARROLLO

Para llevar el proyecto se decidió comenzar por el tratamiento del corpus y creación de los unigramas, bigramas y trigramas utilizados por el modelo de lenguaje. Para ello se realizó una búsqueda exhaustiva de textos que encajen con las necesidades de nuestro modelo. Finalemente se decidió la utilización de un corpus de unos 11 millones de palabras formados mayormente por un corpus perteneciente a artículos de Wikipedia y en menor medida por un pequeño subcorpus de textos médicos perteneciente a la IULA. Mediante métodos de tokenización se llevó a cabo la separación de cada uno de los elementos de los textos y la creación y conteo de los ngramas.

Además de la creación de los ngramas se ha de decidir la estructura de datos a utilizar para su almacenamiento. Finalmente se decide hacer uso de la estructura Hashmap debido a la naturaleza de los datos contenidos, el volumen y la forma de acceso. Debido a la volatilidad de esta estructura se toma la decisión de realizar un volcado al comienzo de la aplicación archivo de texto a los Hashmap mediante un método creado a tal efecto.

Una vez realizado este proceso se pasó a la creación del diccionario que contenga el vocabulario aceptado de nuestra herramienta. Para ello se hace uso de tres fuentes distintas con cierto grado de fiabilidad como son el diccionario Hunspell usado por OpenOffice, la terminología médica de SNOMED-CT y la lista de medicamentos mostrada en la página de referencia farmacológica vademécum. Una vez obtenido todo el vocabulario se ha de decidir la estructura que albergará las palabras. Tras un proceso de documentación bastante arduo se decide la utilización de la estructura TST – Ternary Search Tree. El TST permite el almacenamiento de cada uno de los caracteres de una palabra en un nodo distinto del árbol, lo cual ofrece resultados de búsqueda e inserción realmente eficientes.

El siguiente paso es el de la creación del módulo de creación de candidatas que se basa en crear todas las palabras posibles a través de los métodos anteriormente explicados basados en distancias de edición. Una vez creadas esta lista de posibles palabras se ha de realizar una comprobación de su existencia en el diccionario, desechando evidentemente aquellas que no se encuentren en la estructura.

Cuando ambas partes, diccionario y ngramas, se encontraron desarrolladas se pasó al proceso de creación del módulo de cálculo probabilístico. Este módulo ofrece un resultado numérico de la probabilidad de una palabra dentro de un contexto. Es decir se realiza un cálculo de la probabilidad de que un término en función de las palabras que lo rodean. Para saber que

método ofrece un mejor resultado se realizó un proceso de evaluación y comparación entre los resultados de todos los métodos utilizados (modelo de lenguaje basado en unigramas, bigramas y regla de la cadena).

De manera que el proceso de funcionamiento de la herramienta es:

1. Separación de cada palabra del texto.
2. Detección de palabras erróneas mediante uso de diccionario.
3. Creación de posibles palabras correctas.
4. Cálculo probabilístico de cada una de las sugerencias usando el modelo de lenguaje.

Por último, se pasó a la creación de las interfaces de usuario. 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. 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.

5. CONCLUSIONES Y TRABAJO FUTURO

5.1. 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.

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.

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.

5.2. 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.