Estructuras de Datos (Gestión), curso 2008/09

Primera Práctica – Corrector Ortográfico (I)

INTRODUCCIÓN

Hoy en día prácticamente todos los editores de texto llevan incorporado un módulo de corrección ortográfica, cuya función es detectar palabras que posiblemente estén mal escritas y presentar al usuario una serie de alternativas. El objetivo de ésta práctica es desarrollar la parte de un corrector ortográfico encargada de generar la lista de palabras alternativas más probable.

Para saber si una palabra esta correctamente escrita o no, el enfoque más sencillo (el que vamos a seguir aquí) consiste en disponer de un fichero con la lista de (casi) todas las palabras válidas del lenguaje utilizado, que llamaremos **corpus**. Una palabra es correcta si pertenece al corpus, e incorrecta si no pertenece a él. Para obtener la lista de alternativas, obtendremos del corpus aquellas palabras que, de acuerdo a un cierto criterio, sean **compatibles** con la palabra incorrecta.

La idea que vamos a seguir es que una palabra incorrecta se corresponde con una palabra que pertenece al corpus pero en la que el usuario, ya sea por desconocimiento de la ortografía o por errores al teclearla, ha introducido una serie de **fallos** que supondremos que pueden ser de los tipos siguientes:

- **Cambio** de una letra por otra (por ejemplo *avierto* en lugar de *abierto*)
- **Omisión** de una letra (por ejemplo *acer* en lugar de *hacer*)
- **Inserción** de una letra extra (por ejemplo *adicción* en lugar de *adición*)

Definiremos un parámetro, el **número máximo de fallos**, que nos va a acotar las palabras del corpus que son compatibles con una determinada palabra errónea, y por lo tanto deben presentarse en el listado de alternativas:

Diremos que la palabra S_1 es **compatible** con la palabra S_2 si es posible **transformar** S_1 en S_2 mediante un número menor o igual de operaciones de cambio, omisión e inserción que el número máximo de fallos permitidos.

Por ejemplo, si el número máximo de fallos es dos, entonces la palabra "atis" es compatible con la palabra "datos" (cambiando la *i* por *o* e insertando la *d*), pero "diasot" no lo es (requiere como mínimo tres operaciones: borrado de la *i*, cambio de *s* por *t*, cambio de *t* por *s*).

Un corrector ortográfico debe proporcionar las alternativas en un tiempo muy breve para ser utilizable (menos de un segundo). Dado que el corpus suele ser bastante grande (usaremos uno de alrededor de 70.000 palabras) y que el número de posibles transformaciones de una palabra puede ser enorme, es vital el uso de algoritmos y/o estructuras de datos eficientes.

DESCRIPCIÓN DETALLADA

El objetivo va a ser crear una aplicación que se comporte de la siguiente forma: Inicialmente pedirá el nombre del fichero que va a servir de corpus y el valor del número máximo de fallos. La aplicación leerá y procesará el fichero del corpus, y a partir de ese momento entrará en un bucle en el que pedirá una palabra y mostrará la lista de palabras alternativas, en el formato que se describirá posteriormente. El programa terminará al introducir una palabra vacía.

El fichero del corpus está disponible en la página de la asignatura. Está basado en un corpus del español con licencia GNU, y su formato es un fichero de texto en el que la primera línea es un número, *n*, que indica el número de palabras, y al que siguen *n* líneas cada una de las cuales contiene una palabra. Las palabras están formadas por los siguientes caracteres: 'A'..'Z', 'a'..'z', 'ñ', 'Á', 'á', 'é', 'í', 'ó', 'ú', 'ü' y '.' **Nota**: No debe suponerse que el corpus esté ordenado.

Cada vez que se introduce una palabra, independientemente de que se encuentre o no en el diccionario el programa mostrará el listado de alternativas (palabras del corpus que se pueden transformar en la dada con un número de operaciones menor o igual que el máximo), una por línea e indicando entre paréntesis el número de transformaciones. Al final se añadirá una línea que indique cuántas palabras se han encontrado y el tiempo (en segundos) que ha tardado el cálculo. El orden en que se presenten las alternativas no es relevante. Ejemplo:

```
Fichero del corpus: corpus.txt
Número máximo de fallos: 2
Palabra: pazcato
acato (2)
azoato (2)
pacato (1)
pacto (2)
pazca (2)
pazco (2)
pazguato (2)
7 palabras encontradas, 8.71 segundos
Palabra: _
```

ALGORITMO UTILIZADO

En esta primera práctica se debe resolver usando la técnica *backtracking* para obtener un subprograma que, a partir de una palabra dada, obtenga todas las posibles palabras transformadas mediante operaciones de cambio, omisión e inserción siempre que el número de esas operaciones no supere el número máximo de fallos. Hay que tener en cuenta que en las operaciones de cambio e inserción no importa el carácter que se inserta o por el que se sustituye, y por lo tanto estas operaciones insertan o sustituyen por un carácter comodín (representado aquí por '?') que a la hora de compararlo es igual a cualquier carácter.

En la tabla siguiente se muestran las transformaciones de la cadena 'abc' si el número máximo de fallos es 2 (el carácter '?' representa un carácter modificado o insertado en la cadena anterior)

??abc	?ac	a??bc	ab?	b
??bc	?b	a??c	ab??	b?
??c	?b?	a?b	ab??c	b?c
?a?bc	?b?c	a?b?	ab?c	bc
?a?c	?bc	a?b?c	ab?c?	bc?
?ab	?bc?	a?bc	abc	c
?ab?	?c	a?bc?	abc?	
?ab?c	a	a?c	abc??	
?abc	a?	a?c?	ac	
?abc?	a??	ab	ac?	

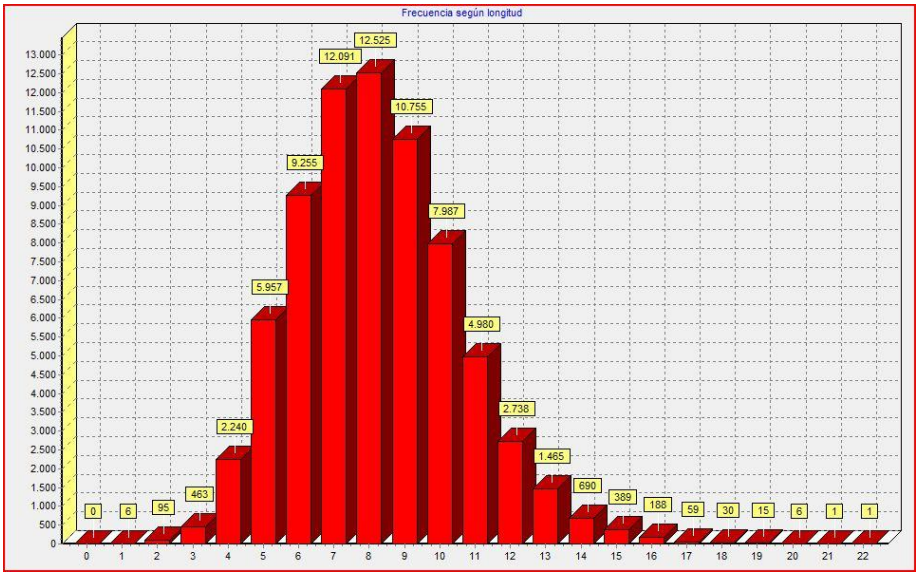
El subprograma anterior puede usarse para resolver el problema siguiendo dos enfoques distintos: En el primero se recorren todas las palabras del corpus, para cada una de ellas se generan todas sus posibles transformaciones y se comprueba si alguna de ellas es igual a la palabra del usuario: En ese caso se incorpora la palabra del corpus a la lista de alternativas.

En el segundo enfoque se generan todas las transformaciones de la palabra del usuario y cada transformación se **busca** en el corpus: Las palabras del corpus que sean iguales se incorporan a la lista de alternativas (hay que tener cuidado de eliminar las posibles alternativas duplicadas).

ANÁLISIS

Es posible analizar cuál de los enfoques es mejor con la siguiente información:

- Si llamamos α al número máximo de fallos, el número de transformaciones de una palabra de longitud m es $4.5\ m^{\alpha} + O(m^{\alpha-1})$
- La distribución de palabras según su longitud, en el corpus que vamos a utilizar, es la siguiente:



DOCUMENTACIÓN A ENTREGAR

Se deberá entregar el código fuente del programa (se permiten los lenguajes Eiffel, Java, Free-Pascal o Delphi/Kylix) y un documento (en formato Word, Látex (DVI) o Acrobat PDF) donde se indique lo siguiente:

- Nombre de los alumnos que han realizado la práctica.
- Una descripción del algoritmo backtracking utilizado y de la alternativa escogida y su justificación.

Estos datos se enviarán como documentos adjuntos en un **correo electrónico** enviado a la dirección cvaca@infor.uva.es, donde el título del mensaje será **PRAC-ED-1** y el contenido el nombre de los alumno(s) que la han realizado, el día **19 de diciembre de 2008** antes de las **23:59**.