Práctica2

Descripción datos y métodos

Tipo de datos → diccionario

Permite almacenar pares de elementos (clave única,definición)

Métodos:

-Inserción de elementos

-Consulta de un elemento por clave

-Consulta del elemento con mayor valor en la definición

Asociado al diccionario tendremos los tipos:

diccionario::entrada(first(clave),second(definición)

diccionario::size_type()

La clave → string con la palabra

El segundo campo es un entero que hace referencia a la frecuencia de ocurrencia de la palabra.

Nuestra base de datos será elquijote.txt

-Procesamos el fichero

-Para cada palabra contaremos la frecuencia de aparación en el fichero, que se almacenarán en un diccionario.

-A la hora de corregir una palabra usaremos la que mas se asemeje y sea mas frecuente en el lenguaje.

-Un segundo diccionario en el que se almacenan todas las palabras del primer diccionario que podrian ser una sugerencia valida. De entre todas estas sugeriremos la mas probable.

¿Cómo corregir por ejemplo “holo”?

-Borrado de un carácter → “olo hlo hoo hol”

-Transposición de dos caracteres → obtenemos “ohlo hloo hool”

-Alteraciones en un carácter → por ejemplo la o la d, “aolo halo hoao hola,dolo hdlo hodo” …

-Inserciones de un carácter: por ejemplo la a o la t, “aholo haolo hoalo holao holoa tholo htolo hotlo holto holot” …

Generariamos 241 resultados, consultamos los que están en el aprendido del quijote. Si es cierto(pertenece al quijote) lo introducimos en un segundo diccionario donde se almacenan las palabras correctas que se pueden ofrecer como alternativa para la corrección. Finalmente de entre todas las candidatas se selecciona la mas común.

Objetivos

Entregas:

- documentación.pdf
- dox_diccionario Contiene la configuración de doxigen necesario para generar la documentacion del proyecto (html y pdf) (doxygen dox_diccionario)
- diccionario.h Especificación del TDA diccionario.
- diccionarioV1.hxx Fichero donde debemos implementar la primera versión del diccionario.
- diccionarioV2.hxx Fichero donde debemos implementar la segunda versión del diccionario.
- corrector.h Clase corrector, que es la que se encarga de toda la lógica del algoritmo de corrección ortográfica.
- principal.cpp Fichero donde se incluye el main del programa. En este caso se toma como entrada el fichero de datos “quijote.txt” ya utilizado en la práctica anterior.

Se debe usar el vector de la stl.

Para compilar debemos usar:

g++ -D DICC_V1 -o correctorV1 principal.cpp

g++ -D DICC_V2 -o correctorV2 principal.cpp

Además se debe hacer un análisis comparando eficiencia de ambas versiones, AnalisisComparativo.pdf
En la primera versión no debe estar ordenado, en la segunda si.

Primera Representacion:

Funcion de Abstraccion :

Función de Abstracción: AF: Rep => Abs
dado D=(vector<entrada> dic, int mayor) ==> Diccionario Dic;
Un objeto abstracto, Dic, representando una colección de pares (string,int) se instancia en la clase diccionario como un vector de entradas, definidas como diccionario::entrada. Dada una entrada, x, en D, x.first representa a una palabra válida (clave) y x.second representa el número de veces que ocurre x (definición).
D.dic[D.mayor] hace referencia a la entrada más frecuente en el diccionario.

Invariante de la Representacion:

Propiedades que debe cumplir cualquier objeto
Dic.size() == D.dic.size();
Para todo i, 0 <= i < D.dic.size() se cumple
    D.dic[D.mayor] >= D.dic[i].second;
    D.dic[i].second > 1;
    D.dic[i].first != "";
    D.dic[i].fisrt != D.dic[j].first, para todo j!=i.

Segunda Representacion:

En este caso, la representación que se utiliza es un vector ordenado de entradas, teniendo en cuenta el valor de la clave.

Funcion de Abstraccion :

Función de Abstracción: AF: Rep => Abs
dado D=(vector<entrada> dic, int mayor) ==> Diccionario Dic;
Un objeto abstracto, Dic, representando una colección de pares (string,int) se instancia en el diccionario como un vector ordenado de entradas, diccionario::entrada. Dada una entrada, x, en D, x.first representa a una palabra válida (clave) y x.second representa el número de veces que ocurre x (definición).
D.dic[D.mayor] hace referencia a la entrada más frecuente en el diccionario.

Invariante de la Representacion:

Propiedades que debe cumplir cualquier objeto
Dic.size() == D.dic.size();
Para todo i, 0 <= i < D.dic.size() se cumple
    D.dic[D.mayor].second >= D.dic[i].second;
    D.dic[i].second > 1;
    D.dic[i].first != "";pq.size() == pq.V.size();
Para todo i, 0 <= i < D.dic.size()-1 se cumple
    D.dic[i].first< D.dic[i+1].first

Lista de tareas pendientes

Miembro diccionario::cheq_rep () const
    implementa esta función

Miembro diccionario::diccionario ()
    implementa esta función

Miembro diccionario::diccionario (const entrada &nula)
    implementa esta función

Miembro diccionario::diccionario (const diccionario &d)
implementa esta función

Miembro diccionario::empty () const
implementa esta función

Miembro diccionario::find (const string &s) const
implementa esta función

Miembro diccionario::max_element () const
implementa esta función

Miembro diccionario::operator= (const diccionario &org)
implementa esta función

Miembro diccionario::operator[] (const string &s)
implementa esta función

Miembro diccionario::operator[] (const string &s) const
implementa esta función

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:
corrector 11
diccionario
Clase diccionario 12

Referencia de la Clase corrector

Métodos publicos

std::string correct (const std::string &word)
determina la palabra corregida

void load (const std::string &filename)
lectura del fichero de entrenamiento

Métodos privados

void edits (const std::string &word, std::vector< std::string > &result)
Genera todas las posibles modificaciones tras una "edicion" de la cadena word.

void known (std::vector< std::string > &results, diccionario &candidates)
busca ocurrencias en un diccionario

Atributos privados

diccionario dictionary

Documentacion de la funciones miembro

std::string corrector::correct (const std::string & word) [inline]
determina la palabra corregida
Acepta palabra origen, devuelve sugerencia para corrección.

void corrector::edits (const std::string & word, std::vector< std::string > & result) [inline, private]
Genera todas las posibles modificaciones tras una "edicion" de la cadena word.

Parámetros

·in *word* cadena de entrada

·out *result* vector con las posibles palabras que se obtienen al realizar borrados, transposiciones, alteraciones o inserciones en la cadena word.

void corrector::known (std::vector< std::string > & results, diccionario & candidates) [inline, private]

busca ocurrencias en un diccionario

Parámetros

·in *results* conjunto de palabras a buscar

·in,out *candidates* conjunto de palabras en results cuyas entradas tambien se encuentra en el diccionario

void corrector::load (const std::string & filename) [inline]

lectura del fichero de entrenamiento

Parámetros

·in *filename* nombre del fichero a leer

La documentación para esta clase fue generada a partir del siguiente fichero:

corrector.h

Referencia de la Clase diccionario

Clase diccionario

#include <diccionario.h>

Tipos públicos

-typedef pair<string,int> entrada

-typedef unsigned int size_type

Métodos públicos

diccionario ()

constructor primitivo.

diccionario (const entrada &nula)

constructor primitivo.

diccionario (const diccionario &d)

constructor de copia

bool empty () const

vacia Chequea si el priority_queue esta vacio (size()==0)

const entrada & find (const string &s) const

busca una cadena en el diccionario

const string & max_element () const

devuelve una referencia constante a la entrada con un mayor numero de ocurrencias en el diccionario.

const entrada & null () const

entrada nula del diccionario

diccionario & operator= (const diccionario &org)
operador de asignación

int & operator[ ] (const string &s)
Consulta/Inserta una entrada en el diccionario.

const int & operator[ ] (const string &s) const
Consulta una entrada en el diccionario.

size_type size () const
numero de entradas en el diccionario

Métodos privados
bool cheq_rep () const
Chequea el Invariante de la representacion.

Atributos privados
- ·vector< entrada > dic
- ·int pos_max

Amigas
ostream & operator<<(ostream &, const diccionario &)
imprime todas las entradas del diccionario.

Clase diccionario.
diccionario:: diccionario, end, find, operator[], size, max_element
Tipos
diccionario::entrada, diccionario::size_type

Documentacion del constructor y destructor
diccionario::diccionario ( )
constructor primitivo.

Postcondicion
define la entrada nula como el par ("",-1)

Referencia de la Clase diccionario
diccionario::diccionario (const entrada & nula)
constructor primitivo.
Parámetros
in *nula* representa a la entrada nula para el diccionario
Postcondicion:
define la entrada nula

diccionario::diccionario (const diccionario & d)
constructor de copia
Parámetros
in *d* diccionario a copiar

Documentacion de las funciones miembro
bool diccionario::cheq rep ( ) const [private]
Chequea el Invariante de la representacion.
Devuelve
true si el invariante es correcto, falso en caso contrario

bool diccionario::empty ( ) const
vacia Chequea si el priority_queue esta vacio (size()==0)

const diccionario::entrada & diccionario::find (const string & s) const
busca una cadena en el diccionario
Parámetros
s cadena a buscar
Devuelve
una copia de la entrada en el diccionario. Si la palabra s no se encuentra devuelve end()
Postcondicion
no modifica el diccionario.
Uso
```
if (D.fin("hola")!=D.end())
    cout << "Esta" ;
else
    cout << "No esta";
```

const string & diccionario::max element ( ) const
devuelve una referencia constante a la entrada con un mayor numero de ocurrencias en el diccionario.
Postcondicion
No se modifica el diccionario.

const diccionario::entrada & diccionario::null ( ) const
entrada nula del diccionario
Devuelve
Devuelve la entrada nula del diccionario.
Postcondicion
no modifica el diccionario

diccionario & diccionario::operator= (const diccionario & org)
operador de asignación
Parámetros
in *org* diccionario a copiar. Crea un diccionario duplicado exacto de org.

int & diccionario::operator[ ] (const string & s)
Consulta/Inserta una entrada en el diccionario.
Busca la cadena s en el diccionario, si la encuentra devuelve una referencia al numero de ocurrencias de la misma en caso contrario la inserta, con frecuencia cero, devolviendo una referencia a este valor.
Parámetros
in s cadena a insertar
out int & referencia a la definicion asociada a la entrada
Postcondicion
Si s no esta en el diccionario, el size() sera incrementado en 1.

const int & diccionario::operator[ ] (const string & s) const
Consulta una entrada en el diccionario.
Busca la cadena s en el diccionario, si la encuentra devuelve una referencia constante al numero de ocurrencias de la misma, si no la encuentra da un mensaje de error.
Parámetros
in s cadena a insertar

out int & referencia constante a la definicion asociada a la entrada

Postcondicion

No se modifica el diccionario.

diccionario::size type diccionario::size ( ) const

numero de entradas en el diccionario

Postcondicion

No se modifica el diccionario.

Documentacion de las funciones relacionadas y clases amigas

ostream& operator<< (ostream & sal, const diccionario & D) [friend]

imprime todas las entradas del diccionario

Postcondicion

No se modifica el diccionario.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

diccionario.h

diccionarioV1.hxx

diccionarioV2.hxx