Analizador de Opiniones PcComponentes

Rafael Ángel Ruiz Lucena

Resumen

El presente trabajo consiste en un analizador de opiniones de la web www.pccomponentes.com, usando la herramienta Lex, la cual permite extraer expresiones regulares de un texto.

Memoria

1 Archivo script

Para ejecutar el programa lo único que debemos hacer es ejecutar en un terminal ./script en el directorio en el que se encuentra el archivo y abrir el archivo final2.txt que se genera en el mismo directorio, e interpretar los resultados.

```
#!/bin/bash

mkdir datos

cd datos

# Con el comando wget descargamos la web en la que aparece el producto, si queremos aplicarlo sobre otro producto debemos cambiar la url por la del producto.

wget -i http://www.pccomponentes.com/leotec_l_pad_meteor_dcx_9__8gb.html >> datos

# Con la ayuda del comando cat volcamos todas las opiniones de lo usuario (que se encuentra en los archivos pagina_opiniones_usuario.php\?idc\=*) en el archivo solucion.txt

cat pagina_opiniones_usuario.php\?idc\=* >> solucion.txt

# Los siguientes comandos hacen ejecutable el programa en cualquier entorno Linux, independientemente del sistema y lugar en el que situemos el programa.

DIRECTORIOANTES=$(pwd)

cd ..

DIRECTORIODESPUES=$(pwd)

cp $DIRECTORIOANTES/solucion.txt $DIRECTORIODESPUES

# Compilación de los dos programas lex que analizan los archivos correspondientes

lex practica3.l

g++ lex.yy.c -o prog -lfl
```

```
./prog datos/solucion.txt >> final.txt

lex practica3a.l

g++ lex.yy.c -o prog1 -lfl

./prog1 final.txt >> final2.txt

# Limpieza de los datos que no sirven

rm -r -f datos

rm prog

rm prog1

rm solucion.txt

rm final.txt

rm lex.yy.c
```

2 Archivo practica3.l

```
/*----- Sección de Declaraciones --------------*/
%{
#include <stdio.h>
#include <string>
#include <iostream>
#include <vector>

using namespace std;


%}

%%
        /*----- Sección de Reglas ----------------*/

// La siguiente regla extrae del archivo todas las opiniones de los usuarios eliminando todos los comentarios html del archivo y limpiando información de elementos no relevantes para evaluar una opinión.

"margin-left:100px;".*"\n"*[A-Z]*.*"\n"*[a-z]*.*"\n"*[A-Z]*.*"\n"*[a-z]*.*"\n"*"<br/>"          {

                cout << yytext << endl;

        }


\n                                      {}
.                                       {}

%%

        /*----- Sección de Procedimientos --------*/
int main (int argc, char *argv[]) {
```

```
	if (argc == 2) {
		yyin = fopen (argv[1], "rt");
		if (yyin == NULL) {
		printf ("El fichero %s no se puede abrir\n", argv[1]);
		exit (-1);
		}
	}else yyin = stdin;

	opiniones = 0;

	yylex ();



	return 0;
}
```

3 Archivo practica3a.l

```
/*----- Sección de Declaraciones ----------*/
%{
#include <stdio.h>
#include <string>
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

// Vector de pair en el que almaceno una palabra y su impotancia con un valor entero.

vector<pair<string,int> > dic;

// Vectores que contienen palabras que se clasifican según diferentes critererios.

string adj_in[20] =
{"un","una","algunas","ningún","pocos","muchas","escasos","demasiadas","bastantes","otras","ta
ntos","todos","varias","demás","ambos","cada","demasiados","ciertos","tales","tantas"};

string adj_indefinidos[20] =
{"mi","sus","nuestra","su","mis","tu","tus","tuyo","nuestras","vuestros","tuya","mía","suyas","nuest
ros","sus","su","míos","vuestras","mi","tus"};

string adj_demostrativos[20] =
{"estas","aquellos","estos","este","esta","aquella","aquel","estas","estos","esta","aquel","aquella",
"aquellos","esas","esos","ese","esa","aquel","aquellas","estas"};

string calificativos_buenos[27] =
{"calidad","barata","ligera","fuerte","fantastico","fantástico","rapido","agil","ágil","util","útil","rápido
", "enorme","suave","amable", "bueno", "bonito", "impresionante","perfecto", "buen", "versátil",
"versatil","aceptable","excelente","completo","insuperable","sobresaliente"};

string calificativos_malo[19] = {"es","lento","espantoso","horrible","frágil","fragil","diminuto",
"sucio", "oxidado", "vacío", "triste", "feo", "demasiado", "malo", "débil", "debil", "difícil",
"dificil","falta"};

string conjunciones[20] = {"a", "ante", "cabe", "con", "contra", "de", "desde", "en", "entre",
"hacia", "hasta", "para", "por", "según", "si", "so", "sobre", "tras", "durante", "mediante"};
```

```
// Función para ordenar por el segundo campo un vector de pair.
bool myfunction( const pair<string, int>& i, const pair<string, int>& j ) {
   if( i.second < j.second ) return false;
   if( j.second < i.second ) return true;
   return j.second < i.second;
}

// Función que devuelve una heurística para un vector de palabras.
int heuristica( vector<pair<string,int> > diccionario)
{

	int valor_heuristica = 0;


	for(int i = 0; i < diccionario.size(); i++)
	{
		bool encontrado = false;

		for(int j = 0 ; j < 19 && !encontrado; j++)
		{
			if(diccionario[i].first == calificativos_malo[j])
			{
				encontrado = true;

				diccionario[i].second = diccionario[i].second - 3; // Resto porque
es un calificativo malo!!!
			}
		}



		for(int j = 0 ; j < 27 && !encontrado; j++)
		{
			if(diccionario[i].first == calificativos_buenos[j])
			{
				encontrado = true;

				diccionario[i].second = diccionario[i].second + 1; // Sumo porque
es algo positivo!!!
			}
		}



		for(int j = 0 ; j < 20 && !encontrado; j++)
		{
			if(diccionario[i].first == adj_demostrativos[j])
			{
				encontrado = true;

				diccionario[i].second = 0; // Elimino algo inservible
			}
		}



		for(int j = 0 ; j < 20 && !encontrado; j++)
		{
			if(diccionario[i].first == adj_indefinidos[j])
			{
```

```
                                encontrado = true;

                                diccionario[i].second = 0; // Elimino algo inservible
                        }
                }



                for(int j = 0 ; j < 20 && !encontrado; j++)
                {
                        if(diccionario[i].first == adj_in[j])
                        {
                                encontrado = true;

                                diccionario[i].second = 0; // Elimino algo inservible
                        }
                }



                for(int j = 0 ; j < 20 && !encontrado; j++)
                {
                        if(diccionario[i].first == conjunciones[j])
                        {
                                encontrado = true;
                                diccionario[i].second = 0; // Elimino algo inservible
                        }
                }

                encontrado = false;

        }



        for(int i = 0; i < diccionario.size(); i++)
        {
                if(diccionario[i].second > 0)
                valor_heuristica = valor_heuristica + diccionario[i].second; // Sumo la heurística
        }

        return valor_heuristica;

}

// Función que inserta un carácter en un diccionario de forma binaria para mayor eficiencia
void insert( string & cadena){

        bool enc = false;
        int inf=0; int sup = dic.size()-1;
        int centro = 0;

        while(inf<=sup && !enc){

                centro=(sup+inf)/2;

                if(dic[centro].first==cadena){

                        enc = true;
                }
```

```
		else if(cadena < dic[centro].first ){

			sup=centro-1;
		}
		else
		{
			inf=centro+1;
		}
	}

	if (!enc) {

		pair<string,int> aux;

		aux.first = cadena;

		aux.second = 0;

		dic.insert(dic.begin()+inf,aux);

		}
	else{
		dic[inf].second = dic[inf].second + 1; // Si ya existe la palabra aumento su valor
	}



}

%}

%%
	/*----- Sección de Reglas ----------------*/
// La siguiente regla extrae palabras mayores o iguales a cuatro caracteres para evitar introducir palabra insevibles

[a-z][a-z][a-z][a-z][a-z]*		{
					string cad = yytext;
					if (cad != "strong" &&
cad != "entajas" && cad != "esventajas" && cad != "sventajas" && cad != "para"){
						insert(cad);
					}
				}
// La siguiente regla extrae la palabra especial que se encuentra en el campo de si recomendaría el producto
[A-Z][a-z]		{
					string cad = yytext;
					if(cad == "Si" || cad ==
"No" || cad == "si" || cad == "no"){
						insert(cad);
					}
				}

\n			{}
.			{}
%%

	/*----- Sección de Procedimientos --------*/
int main (int argc, char *argv[]) {
```

```
	if (argc == 2) {
		yyin = fopen (argv[1], "rt");
		if (yyin == NULL) {
		printf ("El fichero %s no se puede abrir\n", argv[1]);
		exit (-1);
		}
	}else yyin = stdin;

	opiniones = 0;

	yylex ();

	cout << "Diccionario " << endl;

	int mayor = 0;

	int indice = 0;

	// Busco la palabra más repetida en el texto

	for(int i = 0 ; i < dic.size()-1 ; i++)
	{
		if(dic[i].second >= mayor)
		{
			mayor = dic[i].second;

			indice = i;
		}
	}


	// Extraigo la heurística que ronda entre 16000 para un producto muy bueno y 3000
para un producto de peor relevancia.
	cout << "La heuristica es " << heuristica(dic) << endl;

	cout << "La palabra más repetida es " << dic[indice].first << endl;

	// Ordeno el diccionario de palabras según la importancia de cada palabra.
	sort(dic.begin(), dic.end(),myfunction);

	cout << "Las palabras más usadas son las siguientes " << endl;

	for(int i = 0 ; i < dic.size()-1 ; i++)
	{
		cout << dic[i].first << " " << dic[i].second << endl;
	}


	return 0;
}
```

4 Referencias

http://flex.sourceforge.net/manual/