

Consideraciones

Sobre el manejo de archivos:

- Es más eficiente trabajar con el archivo binario representativo de un archivo de texto, sin embargo, no es el adecuado para el analizador, ya que se tendría que interpretar la secuencia de unos y ceros en lugar de trabajar con los símbolos propios del texto.

Si se fuera a estudiar patrones en números se consideraría trabajar con el binario del archivo.

- No se crearán nuevos archivos de texto, se utilizarán estructuras y apuntadores para el analizador.

Inicio del analizador:

- Se preprocesara el texto a analizar según los requerimientos. Se tendrá opción de eliminar o no comas y puntos.

Algoritmos

Artículos que se revisan para la solución de los problemas de analizar un texto:

1. Boyer, Moore 1977, *A fast string searching algorithm*
2. Noriyoshi Uratani Masayuki 1993 *A fast string-searching algorithms for multiple patterns*

Sobre el Algoritmo de Knuth

Es de los primeros algoritmos para la búsqueda de patrones en un texto.

En el peor caso el algoritmo tiene complejidad cuadrada.

Este algoritmo se puede usar cuando definamos un alfabeto, y a cada palabra le asignamos un símbolo del alfabeto.

Tiene como ventaja con respecto al algoritmo de Noriyoshi que solo se necesita implementar una función auxiliar, a comparación de las varias que hay que implementarse para el algoritmo de Noriyoshi.

Ejemplar:

```
numWord: 126, tamPatron: 8, numSimbolos: 719
El patron tiene tam: 8
a f e r r a d a
El numero de palabras 126
Tabla Auxiliar del algoritmo Knuth
tablaKunth[0] = 0
tablaKunth[1] = 0
tablaKunth[2] = 0
tablaKunth[3] = 0
tablaKunth[4] = 4
tablaKunth[5] = 1
tablaKunth[6] = 0
tablaKunth[7] = 1
```

Algoritmo de Knuth. Primero se calculó la tabla auxiliar del algoritmo Knuth, sin embargo, había que emparejar el arreglo de los índices y el patrón para utilizarlo en el algoritmo.

Para facilitar los cálculos se emparejo los índices del patrón y la tabla de Knuth iniciando desde la posición 1 el patrón.

```
  a f e r r a d a
0 0 0 0 0 4 1 0 1
```

Una vez emparejado estos dos arreglos se prosiguió a la implementación del algoritmo de Knuth utilizando esta tabla auxiliar, que fue manipulada con apuntadores.

A continuación se presenta un ejemplar.

```
ummyers@ummyers:~/Documentos/Git/ComputoParaleloTesis/C/Analizador$ gcc KnuthAlgorithm.c
ummyers@ummyers:~/Documentos/Git/ComputoParaleloTesis/C/Analizador$ ./a.out
numWord: 126, tamPatron: 8, numSimbolos: 718
El patron tiene tam: 8
a f e r r a d a
El numero de palabras 126
Tabla Auxiliar del algoritmo Knuth
a f e r r a d a
0 0 0 0 0 4 1 0 1

----Inicia Algoritmo de Knuth----
El patrón se encuentra en el intervalo [214,222] del texto
Valor de numSimbolos 718
Valor de i_Indu 222
Valor de j_Indu 8
```

El programa imprime en pantalla el intervalo donde se encuentra el patrón en el texto, respecto a la posición que tiene.

Al final simplemente se imprimieron los valores de las variables utilizadas en el algoritmo para afinar detalles y comprender que estaba sucediendo cuando no había sido implementado de manera correcta el algoritmo. Simplemente el incremento en la variable que itera sobre el texto estaba incrementandose en el momento incorrecto.

Esto se encuentra implementado en archivo KnuthAlgorithm.c que ya se encuentra en el GitLab del servidor @132.248.150.207

Sobre el Algoritmo de Moore-Boyer

Busca un solo patrón en el texto a la vez.

Puede ser utilizado cuando solo se requiere buscar una palabra en el texto.

Sobre el Algoritmo de Noriyoshi

Presenta la ventaja de que es eficiente para buscar múltiples patrones en un texto a la vez. Entre mayor longitud del patrón menos tiempo le toma al algoritmo encontrar el patrón en el texto, sumando que tienen que buscarse varios patrones.

Esta ventaja se puede usar para buscar el singular y plural de una palabra en el texto, pues al ser prefijo el singular del plural será eficiente la búsqueda.

Tiene como desventaja la complejidad en memoria.

Tabla de similitud

Al estar comparando palabra con palabra y representado en una matriz de n por n , donde n es el número de palabras, tendremos que la mitad diagonal inferior es igual a la mitad diagonal superior. Por lo cual, limitaremos hacer las comparaciones para calcular la mitad diagonal inferior. Realizando así en lugar de $n \times n$ operaciones realizaremos $(n(n-1))/2$ comparaciones. (No muy significativo)

Dos palabras que no tienen ningún símbolo en común tendrán valor igual a cero,

Seleccione la implementación del archivo *analizador2.c* ya que utiliza struct word y es fácil hacer la comparación entre las palabras.

Cuatro personajes inmersos en la apocalíptica modernidad de una gran urbe verán cómo se cruzan sus destinos.

```
--Inicia toda la matriz--
-1
0 -1
1 0 -1
0 0 1 -1
0 0 0 1 -1
0 0 0 0 0 -1
1 2 2 0 0 0 -1
0 1 0 1 1 0 0 -1
1 0 1 1 0 0 0 0 -1
1 0 0 0 0 0 0 0 1 -1
0 0 1 0 0 0 1 0 1 2 -1
1 3 0 0 0 0 1 1 0 0 0 -1
0 1 0 0 0 0 0 0 0 0 0 0 -1
0 1 0 1 1 0 0 2 0 0 0 1 0 -1
1 1 0 0 0 1 1 0 0 1 1 2 1 0 -1
1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 -1
1 2 3 0 0 0 1 2 0 0 0 2 0 1 1 1 -1
```

Medidas de similitud

Medidas de similitud

$$C_1: S(u, v) = \begin{cases} 0 & \text{si } u \text{ y } v \text{ son "totalmente disímiles"} \\ 1 & \text{si } u = v \end{cases}$$

$$C_2: S(u, v) = S(v, u) \quad \forall u, v$$

C_3 : Reversibilidad

$$S(u, v) = S(u^R, v^R) \quad \forall u, v$$

pera
|||
manzana

anaznana
á r é p

Medida

$$S(u, v) = \frac{T(u, v)}{N(u, v)} \quad \begin{array}{l} \text{Número de ocurrencias idénticas} \\ \text{Función de normalización } (C_1) \end{array}$$

T y N deben satisfacer C_1, C_2 y C_3

Propuestas de medidas de similitud

- Comparar el número de sílabas que tienen entre un par de palabras.

Ejemplo; manzana cuenta con tres sílabas, tanzania cuenta con tres sílabas, tendrían una similitud de uno. No respeta Reversibilidad.

- Contar con cuántas sílabas idénticas cuentan dos palabras.

Ejemplo; manzana cuenta con 3 sílabas, man-za-na pero la palabra manzanas, cuenta con el mismo número de sílabas, man-za-nas, solo no coinciden en la última sílaba, observando que tenemos una “s” al final. Esto se podría usar para identificar palabras plurales.

- Contar el número de vocales de las dos palabras.

Ejemplo; tanzania cuenta con 3 a's y manzana igual, sin embargo, tanzania cuenta con una i adicional a las tres a's. (Esto respeta C2, C3) Se tendría que “ponderar” para C1

- Contar el número de consonantes que tienen ambas palabras. No contabilizar el tipo de consonante por separado, sino solo el número de consonates.

Que las medidas de similitud no respeten los 3 criterios no es problema. Los algoritmos de clustering no pueden respetar los 3 axiomas a la vez.

1. Usar un K-medida para separar palabras singulares con plurales, tendríamos un 2-media.
2. Usar varias medidas de similitud para crear un vector en un plano cartesiano y que la distancia euclidiana determine la similitud de las palabras.

Conclusiones


Por lo expuesto anteriormente, no es que se decida usar un algoritmo sobre otro, sino usarlo para solucionar el problema que mejor convenga con el algoritmo. No dejamos a un lado ninguno de ellos, por ejemplo, el algoritmo de Knuth lo podremos usar cuando nos interese buscar un patrón en el texto, o cuando se vaya a determinar el tema del texto, esto será gracias a la relación que le daremos a la palabra con un símbolo.

Apoyo

<https://wordcounter.net/>

Documentación

Me di a la tarea de buscar que tipo de analizador de texto se encuentran en línea, la siguiente URL <https://www.online-utility.org/text/analyzer.jsp> tiene un analizador de texto, como parámetro introduje este texto y las siguientes imágenes son la información que proporciona el analizador.

**Online-Utility.org**
Utilities for Online Operating System

Online Utility ▾ English Language ▾ Text ▾ Math ▾ Other ▾

Text Analyzer

[Tweet](#)

Free software utility which allows you to find the most frequent phrases and frequencies of words. Non-English language texts are supported. It also counts number of words, characters, sentences and syllables. Also calculates lexical density.

Number of characters (including spaces): 471
Number of characters (without spaces): 345
Number of words: 96
Number of sentences: 9
Number of syllables: 126

Some top phrases containing 3 words (without punctuation marks)	Occurrences
spanning trees of	2

Some top phrases containing 2 words (without punctuation marks)	Occurrences
of g	4
is a	3
trees of	2
g is	2
spanning tree	2
spanning trees	2
a tree	2
graph g	2
in fig	2

Order	Unfiltered word count	Occurrences	Percentage
1.	a	7	7.2917
2.	g	6	6.2500
3.	of	5	5.2083
4.	is	4	4.1667
5.	tree	4	4.1667
6.	spanning	4	4.1667
7.	graph	4	4.1667
8.	=	3	3.1250
9.	in	3	3.1250
10.	the	3	3.1250
11.	8	2	2.0833
12.	trees	2	2.0833
13.	10	2	2.0833
14.	g'	2	2.0833
15.	v'	2	2.0833
16.	connected	2	2.0833
17.	undirected	2	2.0833
18.	weighted	2	2.0833
19.	and	2	2.0833
20.	are	2	2.0833
21.	fig	2	2.0833
22.	e	1	1.0417
23.	i	1	1.0417
24.	k	1	1.0417
25.	q	1	1.0417
26.	such	1	1.0417
27.	an	1	1.0417
28.	e'	1	1.0417
29.	if	1	1.0417
30.	ii	1	1.0417
31.	no	1	1.0417
32.	among	1	1.0417
33.	cycles	1	1.0417
34.	that	1	1.0417
35.	then	1	1.0417

Texto proporcionado, el analizador tiene soporte específico para el idioma Inglés. Es la primera desventaja que se puede observar con respecto a nuestro lenguaje.

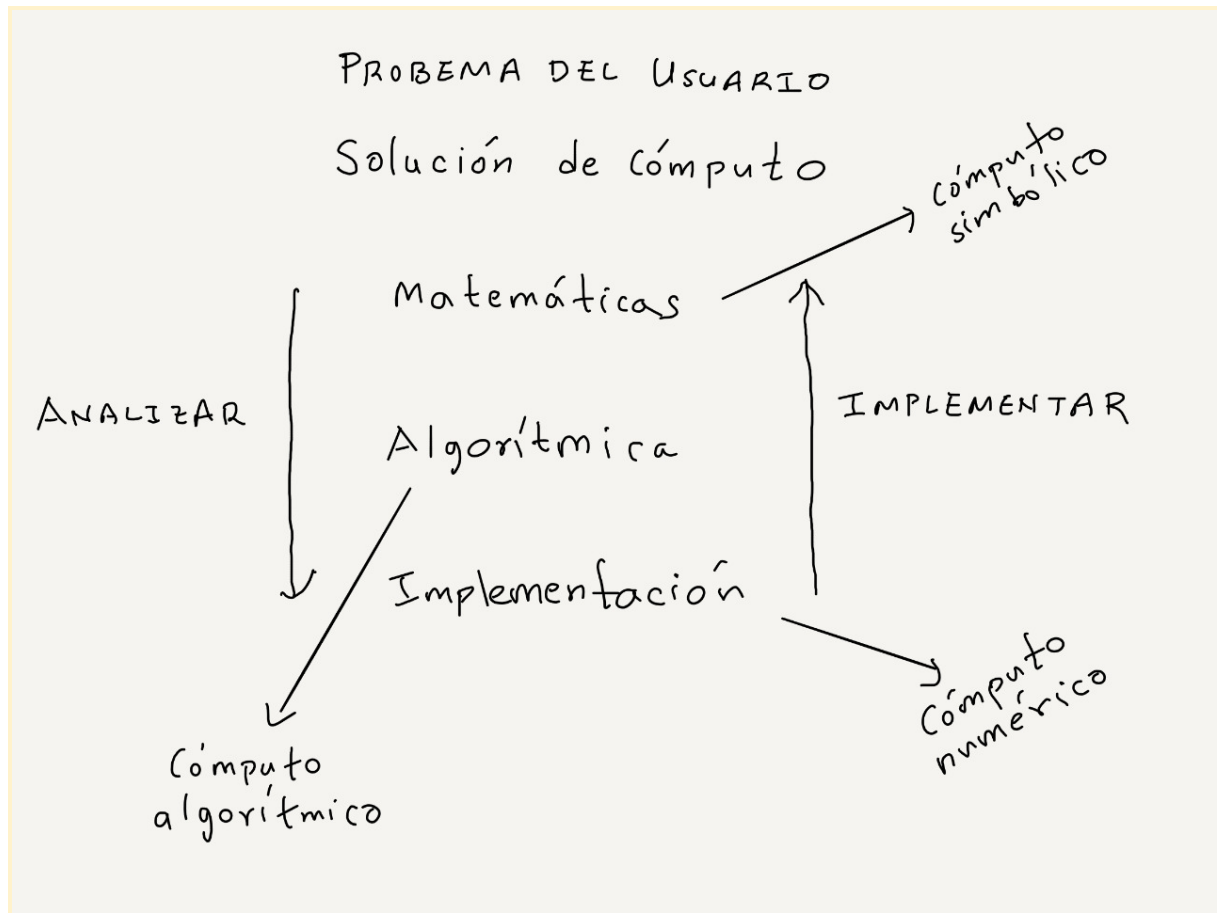
A tree is a connected (undirected) graph with no cycles. Given an undirected and connected graph $G = (q, E)$, a spanning tree of G is a subgraph $G' = (V', E')$ of G such that

(i) G' is a tree, and

(ii) $V' = k$!

If the graph G is weighted, then a minimum spanning tree (MST) of G has the smallest edge-weight sum among all spanning trees of G . These definitions are illustrated in Fig. 10.8. Three spanning trees of the weighted graph in Fig. 10.8(a) are shown in Figs.

Lo primero que podemos evidenciar es que el analizador ya puede proporcionar algunos de los resultados presentados por el analizador de texto. Ya contabilizamos palabras, símbolos, caracteres y contabilizar líneas se implementa con las funciones que se encuentran en el analizador.



Lista de problemas

- Pre procesamiento de texto.
 1. Eliminar comas, puntos entre otros símbolos
 2. Contabilizar las palabras de un texto
 3. Contabilizar el número de caracteres
- Análisis de texto
 1. Encontrar un patrón en el texto.

Propuesta de formalización de tareas.

Definición del problema: Un escritor quiere saber si utiliza de forma desproporcionada palabras singulares.

Objetivo: Determinar si se usaron más palabras singulares o plurales.

Propuesta de una aproximación a la solución

- Separar las palabras en dos conjuntos, el conjunto **S** de singular y el conjunto **P** de plural. (Se utilizan conjuntos para no tener repeticiones)
- Obtener la cardinalidad de ambos conjuntos, si $|\mathbf{S}| > |\mathbf{P}|$ entonces se usaron más palabras singulares, de lo contrario se determina que se usaron más palabras plurales.

En la implementación:

- Se puede agregar un parámetro llamado clase a struct word que se encuentra y cambiar ese parámetro conforme la siguiente regla:

clase = 1 si termina en s la palabra
clase = 0 si no termina en s la palabra

- Crear una lista llamada singular y agregar en ella todas las palabras con terminación en s, mediante el algoritmo insertion sort, así tendríamos ordenadas las palabras por orden alfabético y para tener un orden total, bastaría con utilizar el mismo algoritmo para fusionar las dos listas. Esto último no es indispensable ni resuelve alguno de nuestros problemas por ahora.

(word: String) -> Pertenece a la clase 1 || Pertenece a la clase 0