Predictor de Texto

Moroni Silverio Flores
Sistemas Inteligentes

Junio 2016

1 Introducción

Los predictores de texto no son algo nuevo para nosotros, los vimos antes de que aparecieran los teléfonos inteligentes en los teclados T9.

En este proyecto hice dos predictores de texto: uno para QWERTY y otro para T9, cada uno presenta sus dificultades(en especial el T9), aunque ambos se alimentan de la misma estructura de datos(Árbol de Prefijos la manera en la que buscan e insertan datos en el es diferente).

2 Árbol de Prefijos

Un poco de notación primero: Sea $V = \{1, 2, \ldots, n\}$ un conjunto de vertices(nodos) y sea $E \subseteq V^2$

Sea $N(u) = \{v|(u, v) \in E(G)\}$ los hijos de u.

Y sea $P(u) = \{v|(v, u) \in E(G)\}$ los padres de u. Una hoja es un vértice v tal que $|N(v)| = 0$

Un árbol dirigido T(de aquí en adelante árbol) es una gráfica $G(V, E)$ conectada que tiene $|V|-1$ aristas, y donde $\forall v \in V \ |P(v)| = 1$ a excepción de la raíz r que no tiene padres. Decimos que un árbol dirigido T esta conectado si $\exists$ un trayectoria (r, v) $\forall v \in V$

Usaremos un T como estructura de datos para el texto predictivo. Ahora debemos encontrar la manera de representarlo y agregar algunas mejoras.

Esta estructura de datos es utilizada comúnmente para completar palabras. El árbol que construí tiene algunas modificaciones, ya que sigue aprendiendo con las palabras nuevas que el usuario ingresa. Cada nodo del árbol es identificado por un $id \in V$, cuyo uso se explicará más adelante. El árbol se comporta de manera similar a un autómata "finito" (sigue creciendo), en cuanto a determinista o no determinista lo veremos al explicar los dos tipos de teclados.

Los estados son los nodos del árbol, el alfabeto Σ son las letras $a \ldots z$ en minúsculas(en inglés).

A diferencia de otros arboles de prefijos, cada nodo cuenta cuantas veces se ha repetido cierta palabra (palabra que termina en ese nodo), si ninguna palabra termina en cierto nodo, entonces Repeticiones(nodo)=0. .También cada nodo v guarda el índice u tal que

$$IndiceDelMax(v) = u :$$

$$Repeticiones(u) = max\{Repeticiones(u)|u \in N(v) \cup \{v\}\}$$

Cada arista esta etiquetada de $a \ldots z$ y cada nodo tiene a lo mas 26 vecinos, por lo que el numero de vértices en el árbol T tiende a

$$26^{|P|}$$

donde P es la palabra mas grande guardada en el árbol. Esto es posible teóricamente (y crea problemas de memoria en las computadoras), pero afortunadamente en el Español no existen palabras tales como 'aaaaaa' o 'asjdhfiasdc' que son aceptadas por el árbol.

Cada nodo también cuenta con $Palabra(v)$, que es la cadena formada por los aristas en la $trayectoria(r, v)$, pero es almacenada en $Palabra(v)$ para hacerlo de manera constante sin necesidad de reconstruir el camino.

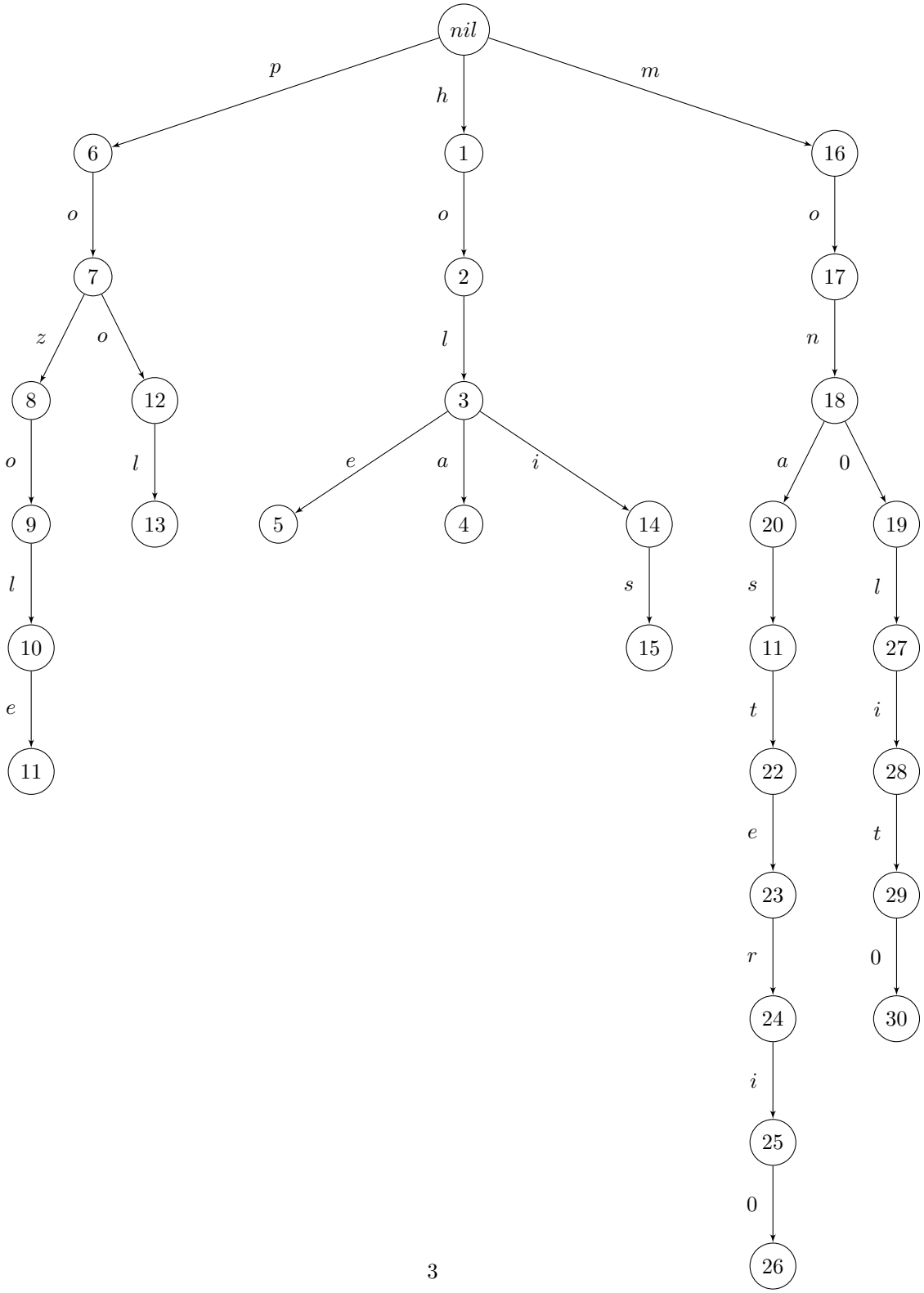
Después de insertar las palabras

```
hola    pool    holis   monolito
hole    pozole  hola    mono
hola    pozole  hol     monasterio
pozole  pool    mono
pool    pozol   mona
pool    hola    monasterio
```

El árbol quedaría de la siguiente manera:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Repeticiones	0	0	1	4	1	0	0	0	0	1	3	0	4	0	1
$IndDelMax$	4	4	4	4	5	13	13	11	11	11	11	13	13	15	15

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Repeticiones	0	0	0	2	1	0	0	0	0	0	2	0	0	0	1
$IndDelMax$	19	19	19	19	26	26	26	26	26	26	26	30	30	30	30


nil
p
h
m
6
1
16
o
o
o
7
2
17
z
o
l
n
8
12
3
18
o
l
e
a
i
a
0
9
13
5
4
14
20
19
l
s
l
s
10
15
11
27
e
t
i
11
22
28
e
t
23
29
r
0
24
30
i
25
0
26

2.1 Insertar

Cuando insertamos una Palabra, ocurre de la siguiente manera:

Pila $\leftarrow \emptyset$
v=nil
para cada *letra* $\in$ *Palabra* **hacer**
  si $\exists\ u\ |etiqueta(v,u) = letra$ **entonces**
    v=u
  en otro caso
    $u = |V| + 1$
    $V = V \cup \{u\}$
    $E = E \cup \{(v,u)\}$
    etiqueta (v,u)=letra
  fin
  $Pila = Pila \cup \{u\}$
  $v = u$
fin
w=v
actualizar(Pila, Palabra, w)

Algoritmo 1: Insertar Palabra

Puede que la palabra exista o no, y en el algoritmo no se ve reflejado eso, hasta que utilizamos la función actualizar?

¿Para qué sirve insertar una palabra que ya existe? Una vez terminado el proceso anterior podemos decir que la palabra ya existe, entonces debemos actualizar nuestros datos.

2.2 Actualizar

Después de insertar una Palabra en el árbol estamos en un nodo hoja, sin importar si existía antes o no ,entonces hacemos lo siguiente:

$Repeticiones(w) += 1$
Palabra(w)=Palabra
para cada $v \in Pila$ **hacer**
  si $Repeticiones(IndDelMax(v)) < Repeticiones(w)$ **entonces**
    $IndDelMax(v) = w$
  en otro caso
  fin
fin

Algoritmo 2: Actualizar Datos

2.3 Buscar

Es similar a insertar, la única diferencia es que cuando no se encuentra el arista deseado se detiene el algoritmo.

3 Predecir

Con este marco teórico veamos los teclados.

3.1 QWERTY

Este teclado en particular trabaja bajo el supuesto que únicamente se ingresaran caracteres validos, se usa el espacio para indicar que ya se termino la palabra y que es momento de actualizar. Mientras no se haya escrito el espacio, se sigue insertando en el árbol.

Cada vez que se ingresa una palabra se usa el algoritmo de insertar. Actualizar datos toma $O(|Palabra|)$, que es el tamaño de la palabra, el dar una sugerencia lo podemos obtener en $O(1)$, ya que solamente es

$$sugerencia = Palabra(IndDelMax(v))$$

donde v es el nodo actual.

Si alguien toma la sugerencia entonces se termina de ingresar la palabra sugerida, de la misma manera como si lo hubiera hecho el usuario manualmente.

Cuando se buscan datos funciona de manera similar a un autómata determinista, la diferencia es que cuando se en la entrada se recibe un arista 'no valido' entonces se agrega un nuevo nodo y entonces se vuelve valido. Vemos que las palabras aceptadas cada vez serán mas, de alguna manera podemos decir que aceptará cualquier palabra generada por el alfabeto.

3.2 T9

Este teclado tiene complicaciones mayores, si alguien presiono la tecla 2, dos veces, quiso decir aa, ab, ac, ba, bb, bc, ca, cb, cc? ¿Cómo lo sabremos ? El teclado QWERTY supone que todo lo escrito es valido, este no puede hacer eso. En un caso feo si alguien presiono n teclas, se pueden formar 3^n palabras distintas. Esto es horrible.

Por fortuna haciendo una búsqueda acotamos esto, similarmente trabajamos ahora con un autómata no determinista.

```
mientras Se presione una tecla hacer
    ListadeIndices ← nil
    para cada letra ∈ TeclaPresionada hacer
        ListaCopia ← ∅
        para cada v ∈ ListadeIndices hacer
            si ∃ u |etiqueta(v, u) = letra entonces
                ListaCopia=ListaCopia∪{u}
            en otro caso
            fin
        fin
    fin
    ListaIndices=CopiaListaIndices
fin
```

Algoritmo 3: T9 Buscar

La sugerencia son los tres Repeticiones máximas de los IndDelMax() de los vértices en la Lista de Índices, cuando se escoge una sugerencia entonces se utiliza el algoritmo de

$$insertar(Palabra = Sugerencia)$$

La ventaja de esto es que no crece exponencialmente, ya que si no existe la cadena en el árbol entonces se elimina de la lista de indices.

3.3 Programando

El reto aqui es luchar memoria vs tiempo de ejecución:

Podemos asignarle a cada nodo memoria para sus 26 posibles aristas que quizá nunca ocupe, al final ocupamos $26 * |V|$ espacios de memoria, lo cual nos permite en $O(1)$ saber si el arista existe,podemos por otro lado ir asignando la memoria conforme se necesite, el problema es que(para Visual Basic) el reasignar memoria es O(n) y buscar un posible elemento seria igual $O(n)$ lo cual no es practico. Se opto por la primera opción suponiendo que no desbordaremos la memoria.

3.4 Entrenamiento

El entrenamiento es únicamente insertar una gran cantidad de palabras al árbol, simulando que lo hace el usuario, nada mas. Se explica mas adelante cómo se hace.

4 Instrucciones

El programa trabaja bajo el supuesto que siempre se ingresan palabras correctas

Primero se tiene que dar clic en el botón teclado para cargar la configuración inicial.

4.1 QWERTY

El botón de entrenar carga un archivo de texto lleno de palabras, este archivo de texto esta lleno de palabras, después de cada palabra hay un espacio. Lo que hace es simular que el usuario escribió todas esas palabras, ese es el entrenamiento. Cada vez que se presiona el espacio se da por terminada la palabra y se actualizan los datos.

4.2 T9

Cada vez que se presiona un botón se da la sugerencia de tres ramas distintas del árbol cuya repeticiones de palabras sean máximas. Si solo queda una rama se sugiere una palabra. Si no quedan palabras el usuario debe dar clic en el textbox de abajo e ingresar la palabra que quería y después dar clic en go.

Una vez que se tiene la sugerencia deseada, se da clic en la sugerencia y se continua escribiendo

5 Extra Game of Life

Para cargar un archivo con alguna forma básica(spaceship, puffer, oscilator) se da clic en abrir y se abre un documento de texto con el siguiente formato:

Dos números(r, c) indicando renglones y columnas, después vienen r grupos de c números cada uno, con ceros y unos indicando si la célula esta viva, debería verse en forma matricial, sin embargo el archivo es una sola linea donde separa cada numero por un espacio. Los archivos txt están en este formato, los que no tienen formato están en forma matricial y el programa *salida.cpp* los pasa al formato mencionado anteriormente.