INFORME PROYECTO ALM

December 26, 2020

Victor Callejas Fuentes	David Alarcón	Jose Mira
4CO11	4CO11	4CO11
viccalfu@inf.upv.es	viccalfu@inf.upv.es	jomigar4@inf.upv.es

Abstract

En este documento se recoge el código desarrollado así como los resultados obtenidos y decisiones tomadas durante la realización de las práctias de laboratorio.

Contents

1 Tarea 1 - Distancias de edición

Implementaión de distancias de edición entre cadenas de forma iterativa y mediante programación dinámica

1.1 Distancia de Levensthtein

Considerando las operaciones de inserción, borrado y sustitución con coste = 1.

Listing 1: Algoritmo distancia de levenshtein

```
    mat = matriz(x, y)
    res = np.zeros(shape=(len(x)+1,len(y)+1))
    for i in range(0,len(x)+1):
      for j in range(0,len(y)+1):
            if i==0 or j==0:
               res[i,j] = res[i,j] + i + j
            else:
               res[i,j] =min(
                  mat[i-1,j-1] + res[i-1,j-1],
                  1 + res[i-1,j],
                  1 + res[i,j-1]
               )

    return res[len(x),len(y)]
```

1.2 Distancia de Damerau-Levensthtein restringida

Se añade la operación de trasposición. En esta versión una vez intercambiados dos símbolos, éstos no se pueden utilizar en otras operaciones de edición.

Listing 2: Sample Python code – Damerau-Levensthtein restringido

```
    INF = len(x) + len(y)

    mat = matriz(x, y)
    res = np.zeros(shape=(len(x)+1,len(y)+1))

    for i in range(0,len(x)+1):
      for j in range(0,len(y)+1):
            if i==0 or j==0:
               res[i,j] = res[i,j] + i + j
            elif i == 1 or j == 1:
               res[i,j] =min(
                  mat[i-1,j-1] + res[i-1,j-1],
                  1 + res[i-1,j],
                  1 + res[i,j-1],
               )
            else:
               res[i,j] =min(
                  mat[i-1,j-1] + res[i-1,j-1],
                  1 + res[i-1,j],
                  1 + res[i,j-1],
                  1 + res[i-2,j-2] + (mat[i-2,j-1] + mat[i-1,j-2]) * INF
               )
    return res[len(x),len(y)]
```

1.3 Distancia de Damerau-Levensthtein intermedia

Considerando las operaciones de trasposición cuando:

$$|u| + |v| \leqslant cte \Leftarrow cte = 1 \tag{1}$$

Listing 3: Damerau-Levensthtein intermedio

```
    M = np.zeros((len(x) + 1, len(y) + 1))
    for i in range(1, len(x) + 1):
      M[i, 0] = i
    for j in range(1, len(y) + 1):
      M[0, j] = j
    for i in range(1, len(x) + 1):
        for j in range(1, len(y) + 1):
            minInit = 0
            if x[i - 1] == y[j - 1]:
                minInit = min(M[i-1, j] + 1, M[i, j-1] + 1, M[i-1][j-1])
            else:
                minInit = min(M[i-1, j] + 1, M[i, j-1] + 1, M[i-1][j-1] + 1)

            if j > 1 and i > 1 and x[i - 2] == y[j - 1] and x[i - 1] == y[j - 2]:
                M[i,j] = min(minInit, M[i-2][j-2] + 1)
            elif j > 2 and i > 1 and x[i-2] == y[j-1] and x[i-1] == y[j-3]:
                M[i,j] = min(minInit, M[i-2][j-3] + 2)
            elif i > 2 and j > 1 and x[i - 3] == y[j-1] and x[i-1] == y[j-2]:
                M[i,j] = min(minInit, M[i-3][j-2] + 2)
            else:
                M[i,j] = minInit
    return M[len(x), len(y)]
```

1.4 Opcional - Distancia de Damerau-Levensthtein general

Esta por hacer

Listing 4: Damerau-Levensthtein general

```
        M = np.zeros((len(x) + 1, len(y) + 1))
        for i in range(1, len(x) + 1):
          M[i, 0] = i
        for j in range(1, len(y) + 1):
          M[0, j] = j
        for i in range(1, len(x) + 1):
            for j in range(1, len(y) + 1):
                minInit = 0
                if x[i - 1] == y[j - 1]:
                    minInit = min(M[i-1, j] + 1, M[i, j-1] + 1, M[i-1][j-1])
                else:
                    minInit = min(M[i-1, j] + 1, M[i, j-1] + 1, M[i-1][j-1] + 1)

                if j > 1 and i > 1 and x[i - 2] == y[j - 1] and x[i - 1] == y[j - 2]:
                    M[i,j] = min(minInit, M[i-2][j-2] + 1)
                elif j > 2 and i > 1 and x[i-2] == y[j-1] and x[i-1] == y[j-3]:
                    M[i,j] = min(minInit, M[i-2][j-3] + 2)
                elif i > 2 and j > 1 and x[i - 3] == y[j-1] and x[i-1] == y[j-2]:
                    M[i,j] = min(minInit, M[i-3][j-2] + 2)
                else:
                    M[i,j] = minInit
        return M[len(x), len(y)]
```

2 Distancias de edición con thresholds

Listing 5: Sample Bash code.

```
#! /bin/bash
python stage1.py
echo "Stage I done!"
python stage2.py
echo "Stage II done!"
python stage3.py
echo "Stage III done!"
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3 Tarea 3 - Metodo Suggest

3.1 TO BE CONTINUED

wertewrt

4 Tarea 4 - Implementación mediante Tries

4.1 TO BE CONTINUED

erwter

5 Tarea 5 - Estudio experimental

5.1 TO BE CONTINUED

wert

6 Tarea 6 - Implementación proyecto SAR

6.1 TO BE CONTINUED

wqrteqwe