# Spoken and Written Language Processing - POE

## Assignment 4. Speech Recognition using Dynamic Time Warping


## Gerard Martín Pey

## Jose Àngel Mola Audí


## 21st May 2023

## Assignment 3. Language identification (Sentence Classification)

### Introduction

**In this homework assignment, you will work on a simple speech recognition task: digit recognition using dynamic time warping.**

### Task description

**Modify the wer function to evaluate the speaker error rate (SER), i.e., the performance of the DTW algorithm for text-dependent speaker identification. The wer function calculates the Word Error Rate (WER), the probability of error in the word classification task. By default, it compares the 'text' of the correct answer with the 'text' of the labeled recording that is close to each of the test wav files (using DTW distance). Identify the line that counts errors and change 'text' with 'speaker' to increment the number of errors if the speaker of the closest recording is not equal to the speaker of the test signal.**

Si executem el cas que està al codi, on s'intenta detectar el nombre amb la veu, per a diferents parlant i igual parlants, veiem com el conjunt de dades del mateix parlant, té un error molt baix (6.9%). Això es deu a que com tenim el mateix parlant dient diferents paraules, les característiques vocals úniques de la persona fa que tinguem més probabilitats de poder detectar-ho bé ja que les característiques de referència i de prova són similars, i per tant tenim un WER inferior.

Ara bé, si usem diferents parlants, hi ha una probabilitat més alta que les característiques acústiques de les mostres de prova i referència siguin diferents, i per tant s'hi afegeix un error que abans no teníem amb el mateix parlant, per això el WER és superior (28.9%).

Ara, modifiquem el codi i mirem el SER, on intentem detectar el parlant. Veiem com per a un mateix parlant, té un error relativament baix (4.4%), mentre que per a diferents parlants té un error del 100%.

**For the rest of the tasks of this assignment keep the original wer function to evaluate the Word Error Rate of the different variants of the DTW-based speech recognition system.**

**Compare the WER with respect to the number of cepstral coefficients Change the default number of cepstral coefficients computed for each speech frame, and check how this number affects the WER of the speech recognizer. Use the obtained best value for the rest of the tasks.**

Per cada nombre de coeficients, hem comprovat el WER i ho hem guardat en la següent taula:

## Assignment 3. Language identification (Sentence Classification)

| N_MFCC | WER SAME SPEAKER | WER DIFFERENT SPEAKER |
|---|---|---|
| **2** | 37.5% | 50% |
| **3** | 25% | 41.7% |
| **4** | 16.9% | 30.2% |
| **5** | 16.2% | 27.5% |
| **6** | 12.5% | 27.6% |
| **7** | 8.8% | 27.8% |
| **8** | 8.1% | 26.3% |
| **9** | 7.5% | 24.3% |
| **10** | 6.2% | 26.0% |
| **11** | 5.6% | 27.5% |
| **12** | 6.9% | 28.9% |
| **13** | 7.5% | 29.4% |
| **14** | 6.9% | 30.7% |

Veiem com el WER disminueix de manera dràstica quan estem amb el mateix speaker. Aquest descens s'atura quan arribem a 11 coeficients. Quan tenim diferents speakers, l'error baixa i s'encalla entre 20 i 30, a partir de 9 coeficients és quan l'error deixa de disminuir. En general, si veiem la comparació entre baixades d'error, podríem considerar un valor de coeficients de 9 com a òptim ja que és l'últim valor de nombre de coeficients que fa disminuir tots dos errors conjuntament i és el que té mitjana de tots dos WERS més baixa. El següent nombre de coeficients fa augmentar aquest error en el cas de diferents speakers.

**Analyze the influence on the recognition accuracy of each of the cepstral normalization steps (mean and variance) with and without littering. Use the default littering parameter (sinus window size) of 22.**

Tenint en compte que hem pres com a referència N_MFCC amb un valor de 9, hem fet les diferents combinacions fent ús de la normalització de la mitjana i variància, normalització de la mitjana i littering.

Les proves les hem recollit en la següent taula:

| LITTERING | MEAN NORM | VAR NORM | WER SAME SPEAKER | WER DIFFERENT SPEAKER |
|---|---|---|---|---|
| **SÍ** | **SÍ** | **SÍ** | 7.5% | 38.2% |
| **SÍ** | **SÍ** | **NO** | 5.6% | 28.2% |
| **SÍ** | **NO** | **NO** | 2.5% | 41.4% |
| **NO** | **SÍ** | **SÍ** | 7.5% | 24.3% |
| **NO** | **SÍ** | **NO** | 9.4% | 39.8% |
| **NO** | **NO** | **NO** | 6.9% | 54.6% |

Podem veure com el littering fa que l'error per a un mateix parlant sigui inferior si fem ús d'aquesta tècnica. Podem veure també com normalitzar la mitjana i la variància ajuda en alguns casos a fer baixar l'error, de fet podem veure el cas en el qual no apliquem cap tipus de tècnica de les 3 obtenim l'error més alt per a parlants diferents (54.6%). Les combinacions que proporcionen una millor mitjana de tots dos errors són aquelles que almenys inclouen una normalització de la mitjana i els que donen un error més gran per a diferents parlants no normalitzen la mitjana per tant considerem aquesta tècnica com a molt útil per al nostre cas d'estudi. Podem veure com normalitzar la variància amb la mitjana fa que millorem l'accuracy tant per al mateix parlant com per a parlants diferents. Si tenim en compte la mitjana entre tots dos errors, les millors combinacions serien NO LITTERING + MEAN NORM + VAR NORM amb un error mitjà de 15.9% i LITTERING + MEAN NORM amb un error de 16.9%.

**Extend the MFCC parameters with the first-order derivatives, and check that the WER is reduced.**

Per cada combinació anterior, calcularem l'error fent ús de la primera i la segona derivada:

Posem el codi que hem utilitzat per calcular la derivada:

```
def first_dev(M):
    M_D = []
    n = M.shape
    for i in range(0, n[0]):
        aux = []
        M_C = np.insert(M[i], 0, 0)
        M_C = np.insert(M_C, len(M_C), 0)
        for j in range(1, n[1] - 1):
            aux.append((M_C[j + 1] - M_C[j - 1]) / 2)
        M_D.append(aux)
    return np.array(M_D)
```

I a la funció get_mfcc hem afegit això:

```
op = 'second'
DM = first_dev(M)
if op == 'first':
    M = np.hstack((M, DM))
elif op == 'second':
    DM2 = first_dev(DM)
    M = np.hstack((M, DM, DM2))
```

| LITTERING | MEAN NORM | VAR NORM | DERIVADA | WER SAME SPEAKER | WER DIFFERENT SPEAKER |
|---|---|---|---|---|---|
| **SÍ** | **SÍ** | **SÍ** | 1st | 6.2% | 33% |
| **SÍ** | **SÍ** | **SÍ** | 1st - 2nd | 6.9% | 31% |
| **SÍ** | **SÍ** | **NO** | 1st | 7.5% | 26.1% |
| **SÍ** | **SÍ** | **NO** | 1st - 2nd | 7.5% | 26.3% |
| **SÍ** | **NO** | **NO** | 1st | 2.5% | 39.3% |
| **SÍ** | **NO** | **NO** | 1st - 2nd | 2.5% | 39.9% |
| **NO** | **SÍ** | **SÍ** | 1st | 6.9% | 23.5% |
| **NO** | **SÍ** | **SÍ** | 1st - 2nd | 7.5% | 23.7% |
| **NO** | **SÍ** | **NO** | 1st | 8.1% | 37% |
| **NO** | **SÍ** | **NO** | 1st - 2nd | 8.8% | 36.5% |
| **NO** | **NO** | **NO** | 1st | 6.9% | 54.6% |
| **NO** | **NO** | **NO** | 1st - 2nd | 6.9% | 55% |

Podem veure com generalment fer ús de les derivades de primer ordre ha fet disminuir el WER. En alguns casos, aquest descens s'ha notat considerablement en el cas de LITTERING – MEAN NORM – VAR NORM, on l'error ha baixat un 7% per a diferents parlants. Veiem que és més complicat que l'error per a un mateix parlant baixi, això suposem a que es deu que aquest ja és prou baix, mentre que per a diferents parlants, com era prou alt, usar les derivades fa que l'error baixi considerablement. Destaquem el cas on no usem cap tècnica i usem primera i segona derivades on l'error augmenta un 0.4% però veient en general com actua i mirant les mitjanes dels errors, fer ús de les derivades fa que l'error baixi considerablement.

**Complete the DTW algorithm to compute the alignment (backtracking). Plot one example of the alignment with the closest reference when the test signal is different from the reference signal (different digit) or equal (same digit).**

Pe realitzar l'alignement, hem dissenyat el següent codi:

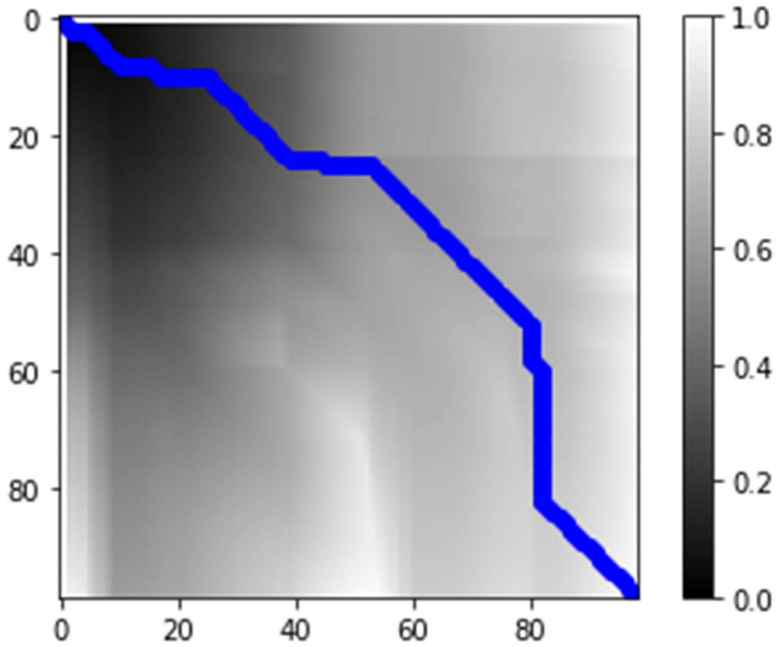
```
def _traceback(D):
    N, M = D.shape[0], D.shape[1]

    i = N - 1
    j = M - 1

    #First step
    path = []
    path.append((i, j))

    while i > 0 and j > 0:
        #If we reach the first row finish
        if i == 0:
            p = (0, j - 1)
        #If we reach the first column finish
        elif j == 0:
            p = (i - 1, 0)
        else:
            neigb = min(D[i - 1, j], D[i - 1, j - 1], D[i, j - 1])
            if neigb == D[i - 1, j]:
                p = (i - 1, j)
            elif neigb == D[i - 1, j - 1]:
                p = (i - 1, j - 1)
            else:
                p = (i, j - 1)
        path.append(p)
        (i, j) = p
    path.reverse()
    return np.array(path)
```

```
def plot_alignment(x, y):
    dtw_ = dtw(x, y)
    cost = dtw_[0]
    path = dtw_[1]
    D = dtw_[2]

    supp = D
    supp[np.isinf(supp)] = np.nan
    maxval = np.nanmax(supp)

    D /= maxval
    D[np.isnan(D)] = np.inf

    plot = plt.imshow(D, cmap = 'gray')
    for p in path:
        plt.scatter(p[1], p[0], color = 'blue', marker = 'o')
    plt.colorbar(plot)
    plt.show()
```

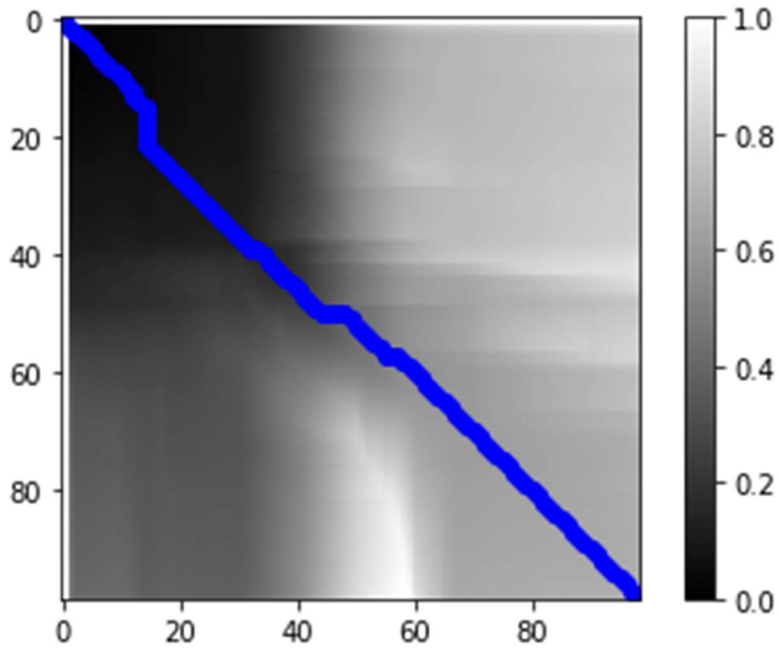
I hem canviat el que retorna la funció dtw per tal de poder tenir guardada la matriu també. (return D[-1, -1], path, D)

Amb això, hem calculat el DTW i alignement per a dígits iguals i diferents. Veiem els resultats:

- Nombre 2 i 8



- Nombre 2 i 2




Es pot apreciar com més proper a la diagonal és aquest alignement, voldrà dir que en teoria més s'assemblen les dos seqüències i que, per tant, més probable és que estiguem tractant un nombre igual. En el cas dels dos nombres iguals veiem com això passa, mentre que en els diferents, aquesta diagonal no es veu tan clara.

**Optional: perform a comparative study of different variants of the DTW algorithm in terms of speed and accuracy.**

En aquest cas, considerarem fer un canvi en el DTW tal com hem vist a classe de teoria. Per a la pràctica hem usat la variant tipus 2:

$$D(i,j) = \|t(i), r(j)\| + \min\begin{Bmatrix} D(i,j-1) \\ D(i-1,j-1) \\ D(i-1,j) \end{Bmatrix}$$

Doncs ara procedirem a usar la variant tipus 1:

DTW recurrence:

$$D(i,j) = \|t(i) - r(j)\| + \min\begin{Bmatrix} D(i-1,j-2) \\ D(i-1,j-1) \\ D(i-2,j-1) \end{Bmatrix}$$

I farem alguns experiments per veure com afecta això al WER.

## Assignment 3. Language identification (Sentence Classification)

Primer mostrem el codi de la funció tant de DTW com d'alignement.

```
from numba import jit
@jit
def dtw_type1(x, y, metric='sqeuclidean'):
  """
  Computes Dynamic Time Warping (DTW) of two sequences.
  :param array x: N1*M array
  :param array y: N2*M array
  :param func dist: distance used as cost measure
  """
  r, c = len(x), len(y)

  D = np.zeros((r + 1, c + 1))
  D[0, 1:] = np.inf
  D[1:, 0] = np.inf

  # Initialize the matrix with dist(x[i], y[j])
  D[1:, 1:] = scipy.spatial.distance.cdist(x, y, metric)

  for i in range(r):
    for j in range(c):
      min_prev = min(D[i, j - 1], D[i, j], D[i - 1, j])
      # D[i+1, j+1] = dist(x[i], y[j]) + min_prev
      D[i+1, j+1] += min_prev


  if len(x) == 1:
        path = zeros(len(y)), range(len(y))
  elif len(y) == 1:
        path = range(len(x)), zeros(len(x))
  else:
        path = _traceback_type1(D)

  return D[-1, -1], path, D
```

```
def _traceback_type1(D):
    N, M = D.shape[0], D.shape[1]

    i = N - 1
    j = M - 1

    #First step
    path = []
    path.append((i, j))

    while i > 0 and j > 0:
        #If we reach the first row finish
        if i == 0:
            p = (0, j - 1)
        #If we reach the first column finish
        elif j == 0:
            p = (i - 1, 0)
        else:
            neigb = min(D[i - 1, j - 1], D[i - 2, j - 1], D[i - 1, j - 2])
            if neigb == D[i - 1, j - 1]:
                p = (i - 1, j - 1)
            elif neigb == D[i - 2, j - 1]:
                p = (i - 2, j - 1)
            else:
                p = (i - 1, j - 2)
        path.append(p)
        (i, j) = p
    path.reverse()
    return np.array(path)
```

## Assignment 3. Language identification (Sentence Classification)

Amb això farem alguns experiments que abans hem fet i veiem com afecta el fet que haguem canviat l'algorisme. Com ja hem vist que fer ús de les derivades és una bona tècnica, compararem els WER fent ús de les derivades:

| LITTERING | MEAN NORM | VAR NORM | DERIVADA | WER SAME SPEAKER | WER DIFFERENT SPEAKER |
|---|---|---|---|---|---|
| **SÍ** | **SÍ** | **SÍ** | 1st | 5.6% | 36.8% |
| **SÍ** | **SÍ** | **SÍ** | 1st - 2nd | 5.6% | 35.2% |
| **SÍ** | **SÍ** | **NO** | 1st | 4.4% | 29.9% |
| **SÍ** | **SÍ** | **NO** | 1st - 2nd | 4.4% | 29.5% |
| **SÍ** | **NO** | **NO** | 1st | 1.2% | 42.6% |
| **SÍ** | **NO** | **NO** | 1st - 2nd | 1.2% | 42% |
| **NO** | **SÍ** | **SÍ** | 1st | 5.6% | 26.7% |
| **NO** | **SÍ** | **SÍ** | 1st - 2nd | 5.6% | 25.6% |
| **NO** | **SÍ** | **NO** | 1st | 8.1% | 40.1% |
| **NO** | **SÍ** | **NO** | 1st - 2nd | 8.1% | 40.4% |
| **NO** | **NO** | **NO** | 1st | 3.1% | 55.2% |
| **NO** | **NO** | **NO** | 1st - 2nd | 1.9% | 54.9% |

Un cop comparats tots dos tipus de DTW, podem veure com el que hem usat originalment actua millor quan tenim diferents parlants, mentre que el que hem usat ara que mira diferents veïns, té un error molt més baix per al mateix parlant.