In [5]:

```python
import numpy as np

class HMM:
    def __init__(self,
                 estados,           # lista de nombres de estados, e.g. ["Ll
                 observaciones,     # lista de posibles observaciones, e.g.
                 probabilidadInicio,       # distribución inicial P(R_0)
                 probabilidadTransicion,  # matriz de transición P(R_t | R_t
                 probabilidadEmision):   # matriz de emisión P(U_t | R_t)

        self.estados = estados
        self.observaciones = observaciones

        # Convertimos todo a arreglos de numpy
        self.probabilidadInicio = np.array(probabilidadInicio)           # s
        self.probabilidadTransicion = np.array(probabilidadTransicion) # sha
        self.probabilidadEmision = np.array(probabilidadEmision)     # shape

        # Diccionarios para mapear nombres -> índices
        self.indiceEstados = { estado: i for i, estado in enumerate(estados)
        self.indiceObservaciones = { observacion: i for i, observacion in er

    def Adelante(self, secuenciaObservada):
        """
        Dada una secuencia de observaciones (e.g. ["Paraguas", "Paraguas", "
        calcula alpha_t(i) = P(U_1, U_2, ..., U_t, R_t = i)
        y retorna la distribución normalizada P(R_t | U_1,...,U_t) en cada p
        """
        T = len(secuenciaObservada)
        N = len(self.estados)

        # alpha[t, i] = P(U_1,...,U_t, R_t = i)
        alpha = np.zeros((T, N))

        # Inicialización (t = 0)
        primeraObservacion = secuenciaObservada[0]
        indicePrimeraObservacion = self.indiceObservaciones[primeraObservaci

        for i in range(N):
            alpha[0, i] = self.probabilidadInicio[i] * self.probabilidadEmis

        # Normalización para el paso t=0
        alpha[0, :] = alpha[0, :] / np.sum(alpha[0, :])

        # Iteración para t > 0
        for t in range(1, T):
            observacion = secuenciaObservada[t]
            indiceObservaciones = self.indiceObservaciones[observacion]

            for j in range(N):
                # Suma sobre todos los estados posibles en t-1
                alpha[t, j] = np.sum(alpha[t-1, :] * self.probabilidadTransi

            # Normalizamos para mantener valores numéricamente estables
```

```
            alpha[t, :] = alpha[t, :] / np.sum(alpha[t, :])

        return alpha

    def estadosMasProbablesViterbi(self, secuenciaObservada):
        """
        Implementación del algoritmo de Viterbi para hallar la secuencia
        de estados ocultos más probable dada la evidencia.
        Retorna la lista de estados (strings) que maximizan P(R_1,...,R_T |
        """
        T = len(secuenciaObservada)
        N = len(self.estados)

        # delta[t, i] = max_{r1..r_{t-1}} P(r1..r_{t-1}, R_t = i, U_1..U_t)
        delta = np.zeros((T, N))
        # psi[t, i] = argmax_{r_{t-1}} (delta[t-1,r_{t-1}] * trans_prob[r_{t
        psi = np.zeros((T, N), dtype=int)

        # Inicialización
        indicePrimeraObservacion = self.indiceObservaciones[secuenciaObserva
        for i in range(N):
            delta[0, i] = self.probabilidadInicio[i] * self.probabilidadEmis
            psi[0, i] = 0

        # Recursión
        for t in range(1, T):
            indiceObservaciones = self.indiceObservaciones[secuenciaObservac
            for j in range(N):
                # Calculamos la mejor transición desde cada i
                probIJ = delta[t-1, :] * self.probabilidadTransicion[:, j]
                psi[t, j] = np.argmax(probIJ)
                delta[t, j] = np.max(probIJ) * self.probabilidadEmision[j, i

        # Retroceso (backtracking) para encontrar la mejor secuencia
        secuenciaEstados = np.zeros(T, dtype=int)
        secuenciaEstados[T-1] = np.argmax(delta[T-1, :])

        for t in reversed(range(T-1)):
            secuenciaEstados[t] = psi[t+1, secuenciaEstados[t+1]]

        # Convertimos índices a nombres de estados
        mejorCamino = [self.estados[indice] for indice in secuenciaEstados]
        return mejorCamino


if __name__ == "__main__":
    # Definimos el conjunto de estados y observaciones
    estados = ["Lluvia", "NoLluvia"]
    observaciones = ["Paraguas", "NoParaguas"]

    # Probabilidades iniciales P(R_0)
    pInicial = [0.5, 0.5]

    # Matriz de transición P(R_t | R_{t-1})
    #   [ P(Lluvia|Lluvia), P(NoLluvia|Lluvia) ]
    #   [ P(Lluvia|NoLluvia), P(NoLluvia|NoLluvia) ]
```

```
    tProb = [
        [0.7, 0.3],
        [0.3, 0.7]
    ]

    # Matriz de emisión P(U_t | R_t)
    # Filas: R_t = Lluvia, R_t = NoLluvia
    # Columnas: U_t = Paraguas, U_t = NoParaguas
    eProb = [
        [0.9, 0.1],
        [0.2, 0.8]
    ]

    # Creamos el HMM
    hmm = HMM(estados, observaciones, pInicial, tProb, eProb)

    # Supongamos que observamos una secuencia de 4 días:
    # Día 1: director con paraguas
    # Día 2: director con paraguas
    # Día 3: director sin paraguas
    # Día 4: director con paraguas
    evidencia = ["Paraguas", "Paraguas", "NoParaguas", "Paraguas"]


    for i in range(4):
        # 1. Filtración (Forward): P(R_t | U_{1:t})
        alpha = hmm.Adelante(evidencia)

        #print("Distribución filtrada (Forward) en cada día:")
        #for t, probs in enumerate(alpha):
            #print(f"Día {t+1} -> P(Lluvia|obs)={probs[0]:.4f}, P(NoLluvia|o

        # 2. Decodificación (Viterbi): la secuencia de estados más probable
        caminoViterbi = hmm.estadosMasProbablesViterbi(evidencia)
        #print("\nSecuencia más probable:", caminoViterbi)

        for i in range(len(caminoViterbi)):
            if caminoViterbi[i] == 'Lluvia':
                evidencia.append("Paraguas")
            else:
                evidencia.append("NoParaguas")

        print(f'Nueva evidencia: {evidencia}')
print("Distribución filtrada (Forward) en cada día:")
for t, probs in enumerate(alpha):
    print(f"Día {t+1} -> P(Lluvia|obs)={probs[0]:.4f}, P(NoLluvia|obs)={prob
print("\nSecuencia más probable:", caminoViterbi)
```

```
Nueva evidencia: ['Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas']
Nueva evidencia: ['Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas']
Nueva evidencia: ['Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas']
Nueva evidencia: ['Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paragua
s', 'Paraguas', 'NoParaguas', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParagua
s', 'Paraguas', 'Paraguas', 'Paraguas', 'NoParaguas', 'Paraguas']
Distribución filtrada (Forward) en cada día:
Día 1 -> P(Lluvia|obs)=0.8182, P(NoLluvia|obs)=0.1818
Día 2 -> P(Lluvia|obs)=0.8834, P(NoLluvia|obs)=0.1166
Día 3 -> P(Lluvia|obs)=0.1907, P(NoLluvia|obs)=0.8093
Día 4 -> P(Lluvia|obs)=0.7308, P(NoLluvia|obs)=0.2692
Día 5 -> P(Lluvia|obs)=0.8673, P(NoLluvia|obs)=0.1327
Día 6 -> P(Lluvia|obs)=0.8918, P(NoLluvia|obs)=0.1082
Día 7 -> P(Lluvia|obs)=0.1930, P(NoLluvia|obs)=0.8070
Día 8 -> P(Lluvia|obs)=0.7316, P(NoLluvia|obs)=0.2684
Día 9 -> P(Lluvia|obs)=0.8675, P(NoLluvia|obs)=0.1325
Día 10 -> P(Lluvia|obs)=0.8919, P(NoLluvia|obs)=0.1081
Día 11 -> P(Lluvia|obs)=0.1930, P(NoLluvia|obs)=0.8070
Día 12 -> P(Lluvia|obs)=0.7316, P(NoLluvia|obs)=0.2684
Día 13 -> P(Lluvia|obs)=0.8675, P(NoLluvia|obs)=0.1325
Día 14 -> P(Lluvia|obs)=0.8919, P(NoLluvia|obs)=0.1081
Día 15 -> P(Lluvia|obs)=0.1930, P(NoLluvia|obs)=0.8070
Día 16 -> P(Lluvia|obs)=0.7316, P(NoLluvia|obs)=0.2684
Día 17 -> P(Lluvia|obs)=0.8675, P(NoLluvia|obs)=0.1325
Día 18 -> P(Lluvia|obs)=0.8919, P(NoLluvia|obs)=0.1081
Día 19 -> P(Lluvia|obs)=0.1930, P(NoLluvia|obs)=0.8070
Día 20 -> P(Lluvia|obs)=0.7316, P(NoLluvia|obs)=0.2684
Día 21 -> P(Lluvia|obs)=0.8675, P(NoLluvia|obs)=0.1325
Día 22 -> P(Lluvia|obs)=0.8919, P(NoLluvia|obs)=0.1081
Día 23 -> P(Lluvia|obs)=0.1930, P(NoLluvia|obs)=0.8070
Día 24 -> P(Lluvia|obs)=0.7316, P(NoLluvia|obs)=0.2684
Día 25 -> P(Lluvia|obs)=0.8675, P(NoLluvia|obs)=0.1325
Día 26 -> P(Lluvia|obs)=0.8919, P(NoLluvia|obs)=0.1081
Día 27 -> P(Lluvia|obs)=0.1930, P(NoLluvia|obs)=0.8070
Día 28 -> P(Lluvia|obs)=0.7316, P(NoLluvia|obs)=0.2684
Día 29 -> P(Lluvia|obs)=0.8675, P(NoLluvia|obs)=0.1325
Día 30 -> P(Lluvia|obs)=0.8919, P(NoLluvia|obs)=0.1081
Día 31 -> P(Lluvia|obs)=0.1930, P(NoLluvia|obs)=0.8070
Día 32 -> P(Lluvia|obs)=0.7316, P(NoLluvia|obs)=0.2684
```

```
Secuencia más probable: ['Lluvia', 'Lluvia', 'NoLluvia', 'Lluvia', 'Lluvia',
'Lluvia', 'NoLluvia', 'Lluvia', 'Lluvia', 'Lluvia', 'NoLluvia', 'Lluvia', 'L
luvia', 'Lluvia', 'NoLluvia', 'Lluvia', 'Lluvia', 'Lluvia', 'NoLluvia', 'Llu
via', 'Lluvia', 'Lluvia', 'NoLluvia', 'Lluvia', 'Lluvia', 'Lluvia', 'NoLluvi
a', 'Lluvia', 'Lluvia', 'Lluvia', 'NoLluvia', 'Lluvia']
```