

Universidad Internacional de La Rioja (UNIR) - Máster Universitario en Inteligencia Artificial - Procesamiento del Lenguaje Natural

Datos del alumno: **JOSUE DAVID PINTO NIETO**

Fecha: Lunes, 02 de mayo del 2022

Trabajo: Etiquetado morfosintáctico

Objetivos

Con esta actividad se tratará de que el alumno consiga aplicar un método basado en modelos ocultos de Markov (HMM) para realizar el etiquetado morfosintáctico de una oración.

Descripción

En esta actividad debes implementar en Python un etiquetador morfosintáctico basado en modelos ocultos de Markov (HMM) y realizar el etiquetado morfosintáctico de la oración:

Habla con el enfermo grave de trasplantes.

Implementando también en Python el algoritmo de Viterbi.

Parte 1: Construir el etiquetador morfosintáctico

En esta primera parte de la actividad tienes que implementar en Python el etiquetador morfosintáctico basado en un HMM bigrama a partir de un corpus etiquetado.

Para ello debes utilizar el corpus mia07_t3_tra_Corpus-tagged, que se encuentra disponible en el aula virtual.

El corpus se compone de frases en español etiquetadas con conocimiento sobre las partes de la oración (categorías gramaticales o POS tags). Estas frases etiquetadas han sido extraídas de algunos documentos que forman parte de Wikicorpus, un corpus trilingüe (español, catalán e inglés) compuesto por más de 750 millones de palabras. Wikicorpus fue creado por investigadores de la Universitat Politècnica de Catalunya a partir de documentos de la Wikipedia que fueron anotados con la librería opensource FreeLing.

La tabla 1 muestra en formato de texto plano y sin etiquetar algunos ejemplos de frases que componen el corpus. De hecho, también se indica el identificador del documento del cual han sido extraídas las frases etiquetadas.

La versión anotada la conforma el corpus anotado proporcionado para realizar esta actividad. El formato del fichero de texto que contiene el corpus es el mismo que el utilizado en Wikicorpus.

Por lo tanto, cada uno de los documentos de Wikipedia se identifica con el tag XML donde se indica el identificador del documento (id).

Además, cada una de las frases en el documento viene separada por una línea en blanco. La información relativa a cada palabra de la frase se representa en una nueva línea del fichero. Para cada palabra, es decir, en cada línea del fichero, se proporciona —además del token que representa a la propia palabra— su lema, la etiqueta gramatical (POS tag) asociada a la palabra y el sentido de esta.

La figura 1 muestra una captura del corpus anotado, donde se observa la frase «Tristana es una película del director español nacionalizado mexicano Luis Buñuel.» perteneciente al documento de Wikicorpus con identificador 27315 y titulado Tristana.

Si se analizan las anotaciones para la palabra «es», se observa que su lema es «ser», que la categoría gramatical a la que pertenece esa palabra es la identificada por la etiqueta gramatical «VSIP3S0» y que el sentido de la palabra es el identificado por el código «01775973175».

También se observa que la palabra «del» en la frase se representa en dos líneas y se anota con dos tokens, el primero «de» y el segundo «el». Esto se debe a que la palabra «del» es la contracción de la preposición «de» y el artículo «el». Por el contrario, el nombre propio «Luis Buñuel», que está formado por dos palabras (el nombre «Luis» y el apellido «Buñuel»), se anota como un único token «luis_buñuel». Además, se observa que el punto final de la frase también viene anotado como un token «.».

Aunque el corpus anotado proporciona más información (ver figura 1), es importante tener en cuenta de que para realizar esta actividad solo será necesario el token y la etiqueta gramatical (POS tag) de cada palabra; es decir, la información contenida en la primera y la tercera cadena de cada línea que representa una palabra en el corpus anotado.

Las etiquetas gramaticales (POS tags) utilizadas para anotar la información morfosintáctica del corpus son las definidas en FreeLing y se basan en EAGLES, una recomendación para la anotación de la mayoría de las lenguas europeas. La definición del conjunto de etiquetas gramaticales (POS tags) utilizadas por FreeLing en el etiquetado de un corpus en español se puede consultar en la web.

Accede al recurso a través del aula virtual o desde la siguiente dirección web: <https://freeling-user-manual.readthedocs.io/en/v4.1/tagsets/tagset-es/>

Las etiquetas gramaticales de EAGLES utilizadas por FreeLing son de longitud variable, donde cada carácter corresponde a una característica morfosintáctica. El primer carácter en la etiqueta es siempre la categoría gramatical o parte de la oración. Esa categoría gramatical determina la longitud de la etiqueta y la interpretación de cada uno del resto de caracteres en la misma.

La definición de la etiqueta para la categoría gramatical «verbo» se muestra en la tabla 2. Entonces, la etiqueta «VSIP3S0», con la que ha sido etiquetada la palabra «es» en la frase que se

presentó anteriormente, se interpreta de la siguiente forma: se refiere a un verbo (V) de tipo semiauxiliar (S) en modo indicativo (I) y en tiempo presente (P) para la tercera persona (3) de (número) singular (S). Asimismo, el carácter «0» al final de la etiqueta indica que esta forma verbal no tiene género.

Es importante destacar que para realizar la actividad se deben utilizar las etiquetas con las que se anota el corpus en formato EAGLES; por ejemplo, «VSIP3S0».

Importante: Si se utilizan otras etiquetas la actividad será considerada incorrecta y puntuada con cero puntos.

Para construir el etiquetador morfosintáctico a partir del corpus etiquetado con los datos de entrenamiento, deberás seguir los siguientes pasos:

- Cargar el corpus para extraer la primera y tercera columna de cada registro.
- Calcular las probabilidades que rigen el HMM bigrama, es decir:
 - Calcular las probabilidades de emisión del HMM a partir del corpus etiquetado.
 - Calcular las probabilidades de transición del HMM a partir del corpus etiquetado.

Nota: Presenta en el envío de la actividad la tabla (guardada en formato de hoja de cálculo de Microsoft Excel (.xlsx) o equivalente) con las probabilidades de emisión y las de transición, calculadas para todos los etiquetas y tokens (palabras) que aparecen en el corpus.

▼ Cargar el corpus para extraer la primera y tercera columna de cada registro

En primer lugar se va a cargar el corpus leyendo el archivo y recuperando la información de la *primera y tercera* columna de cada registro que contienen el *token* de la palabra y la *etiqueta*, respectivamente.

Estos valores se almacenarán en objetos de la clase `Palabra`.

Esta clase permitirá recuperar el `Token()` y el `Tag()` fácilmente para cada registro.

```
class Palabra:
    """
    Clase para guardar el token y la etiqueta de una palabra de un corpus
    """

    def __init__(self, token: str, tag: str):
        """
        Constructor de la clase

        token : str
            Token de la palabra

        tag : str
            Etiqueta de la palabra
        """
        self._token = token
        self._tag = tag
```

```
def Token(self):
    '''
    Método para acceder al token de la palabra
    '''
    return self._token

def Tag(self):
    '''
    Método par acceder a la etiqueta de la palabra
    '''
    return self._tag
```

El corpus se guardará como una lista que a su vez contiene una serie de listas de objetos del tipo `Palabra`. Cada una de las listas de objetos del tipo `Palabra` guarda una oración.

```
archivo = open('mia07_t3_tra_Corpus-tagged.txt', "r")

corpus = list()
oracion_actual = list()

for entrada in archivo.readlines():
    entrada = entrada.split()
    if len(entrada) == 0:
        # Puede ser la primera oración del documento
        # O que termina la oración
        if len(oracion_actual) > 0:
            # Fin de la oración
            corpus.append(oracion_actual)
            oracion_actual = list()
            continue

    elif entrada[0] == '<doc':
        # Inicio de documento. No se hace nada
        continue

    elif entrada[0] == '</doc>':
        # Fin del documento. No se hace nada
        continue

    oracion_actual.append(Palabra(token=entrada[0], tag=entrada[2]))

archivo.close()
```

corpus

```
[[<__main__.Palabra at 0x7f8b17c07390>,
  <__main__.Palabra at 0x7f8b17c07490>,
  <__main__.Palabra at 0x7f8b17c07510>,
  <__main__.Palabra at 0x7f8b17c07590>,
  <__main__.Palabra at 0x7f8b17c07650>,
  <__main__.Palabra at 0x7f8b17c07750>,
```

```

<__main__.Palabra at 0x7f8b17c07850>,
<__main__.Palabra at 0x7f8b17c07890>,
<__main__.Palabra at 0x7f8b17c07990>,
<__main__.Palabra at 0x7f8b17c07a10>,
<__main__.Palabra at 0x7f8b17c079d0>,
<__main__.Palabra at 0x7f8b17c07a90>],
[<__main__.Palabra at 0x7f8b17c07b90>,
<__main__.Palabra at 0x7f8b17c07c50>,
<__main__.Palabra at 0x7f8b17c07cd0>,
<__main__.Palabra at 0x7f8b17c07d90>,
<__main__.Palabra at 0x7f8b17c07e90>,
<__main__.Palabra at 0x7f8b17c07f50>,
<__main__.Palabra at 0x7f8b17c09050>,
<__main__.Palabra at 0x7f8b17c09150>,
<__main__.Palabra at 0x7f8b17c09210>,
<__main__.Palabra at 0x7f8b17c09250>,
<__main__.Palabra at 0x7f8b17c09290>,
<__main__.Palabra at 0x7f8b17c09310>],
[<__main__.Palabra at 0x7f8b17c09450>,
<__main__.Palabra at 0x7f8b17c09510>,
<__main__.Palabra at 0x7f8b17c094d0>,
<__main__.Palabra at 0x7f8b17c09610>,
<__main__.Palabra at 0x7f8b17c096d0>,
<__main__.Palabra at 0x7f8b17c09750>,
<__main__.Palabra at 0x7f8b17c09850>,
<__main__.Palabra at 0x7f8b17c09950>,
<__main__.Palabra at 0x7f8b17c09990>,
<__main__.Palabra at 0x7f8b17c09a10>,
<__main__.Palabra at 0x7f8b17c09b10>,
<__main__.Palabra at 0x7f8b17c09b90>,
<__main__.Palabra at 0x7f8b17c09c50>,
<__main__.Palabra at 0x7f8b17c09d10>,
<__main__.Palabra at 0x7f8b17c09d90>,
<__main__.Palabra at 0x7f8b17c09dd0>],
[<__main__.Palabra at 0x7f8b17c09ed0>,
<__main__.Palabra at 0x7f8b17c09f10>,
<__main__.Palabra at 0x7f8b17c09f90>,
<__main__.Palabra at 0x7f8b17c0b110>,
<__main__.Palabra at 0x7f8b17c0b190>,
<__main__.Palabra at 0x7f8b17c0b210>,
<__main__.Palabra at 0x7f8b17c0b350>,
<__main__.Palabra at 0x7f8b17c0b3d0>,
<__main__.Palabra at 0x7f8b17c0b410>,
<__main__.Palabra at 0x7f8b17c0b4d0>,
<__main__.Palabra at 0x7f8b17c0b510>,
<__main__.Palabra at 0x7f8b17c0b610>,
<__main__.Palabra at 0x7f8b17c0b5d0>,
<__main__.Palabra at 0x7f8b17c0b710>,
<__main__.Palabra at 0x7f8b17c0b810>,
<__main__.Palabra at 0x7f8b17c0b7d0>],
[<__main__.Palabra at 0x7f8b17c0b910>,
<__main__.Palabra at 0x7f8b17c0b990>,

```

El siguiente código te permite imprimir el corpus:

```

for oracion in corpus:
    for palabra in oracion:
        print(palabra.Token(), palabra.Tag())

```

Tristana NP00000
es VSIP3S0
una DI0FS0
película NCFS000
de SPS00
el DA0MS0
director NCMS000
español AQ0MS0
nacionalizado VMP00SM
mexicano AQ0MS0
Luis_Buñuel NP00000
. Fp
Está VAIP3S0
basada VMP00SF
en SPS00
la DA0FS0
novela NCFS000
de SPS00
el DA0MS0
mismo AQ0MS0
nombre NCMS000
de SPS00
Benito_Pérez_Galdós NP00000
. Fp
Fue VSIS3S0
nominada VMP00SF
a SPS00
el DA0MS0
Oscar NP00000
a SPS00
la DA0FS0
mejor AQ0CS0
película NCFS000
de SPS00
habla NCFS000
no RN
inglesa AQ0FS0
en SPS00
1970 Z
. Fp
Tristana NP00000
y CC
Nazarín NP00000
son VSIP3P0
las DA0FP0
dos Z
novelas NCFP000
de SPS00
Benito_Pérez_Galdós NP00000
que CS
Buñuel NP00000
adaptó VMIS3S0
a SPS00
el DA0MS0
cine NCMS000
. Fp
La DA0FS0
película NCFS000

▼ Calcular las probabilidades que rigen el HMM bigrama

Una vez se dispone del `corpus` correctamente cargado se creará un objeto, `hmmbigrama` de la clase `HMMBigrama`.

`hmmbigrama` permitirá hacer el cálculo de las tablas de probabilidades de transición y de emisión.

```
#Se usa pandas para crear las tablas.
import pandas as pd

class HMMBigrama:
    '''
    Clase para obtener las matrices de probabilidad HMM Bigrama a partir de un corpus
    '''

    def __init__(self, corpus: [[Palabra]]):
        '''
        Constructor de la clase para calcular el Modelo Oculto de Markov Bigrama
        '''
        self._corpus = corpus
        self._estados = dict()
        self._tokens = dict()
        self._q0 = 'q0'
        self._qF = 'qF'

        self._prob_trans = pd.DataFrame()
        self._prob_obs = pd.DataFrame()

    def Corpus(self):
        return self._corpus.copy()

    def EstadoInicial(self):
        return self._q0

    def EstadoFinal(self):
        return self._qF

    def _ProcesarCorpus(self):
        '''
        Método para contar el número de ocurrencias de estados y tokens
        '''
        for oracion in self._corpus:
            for palabra in oracion:

                # Se recorren todas las palabras de todas las oraciones del corpus recuper
                estado = palabra.Tag()
                estados = self._estados
                estados[estado] = estados[estado] + 1 if estado in estados else 1

                # Se recorren todas las palabras de todas las oraciones del corpus recuper
                token = palabra.Token()
```

```
        tokens = self._tokens
        tokens[token] = tokens[token] + 1 if token in tokens else 1

def Estados(self, incluir_inicial: bool = False, incluir_final: bool = False):
    """
    Devuelve los estados del bigrama en base al corpus proporcionado al constructor

    incluir_inicial : bool (False)
        Flag para indicar si se quiere recuperar el estado inicial

    incluir_final : bool (False)
        Flag para indicar si se quiere recuperar el estado final

    return
        Diccionario de estados con el número de ocurrencias de cada estado en el corpus
    """

    if len(self._estados) == 0:
        self._ProcesarCorpus()

    copia_estados = dict()
    if incluir_inicial:
        # Hay tantos estados como oraciones en el corpus
        copia_estados[self._q0] = len(self._corpus)

    copia_estados.update(self._estados)

    if incluir_final:
        # Hay tantos estados como oraciones en el corpus
        copia_estados[self._qF] = len(self._corpus)

    return copia_estados

def Tokens(self):
    """
    Devuelve los tokens del bigrama en base al corpus proporcionado al constructor

    return
        Diccionario de tokens con el número de ocurrencias de cada token en el corpus
    """

    if len(self._tokens) == 0:
        self._ProcesarCorpus()

    return self._tokens.copy()

def ProbabilidadesDeTransicion(self):
    """
    Método para calcular las probabilidades de transición bigrama
    a partir del corpus proporcionado a la clase
    """

    # Si ya se ha calculado se devuelve
```



```

if len(self._prob_trans) != 0:
    return self._prob_trans.copy()

'''
En esta parte del código se calcula el número de
transiciones bigrama, es decir, en el diccionario
'contador_transiciones' se almacenarán los contadores
de las transiciones t-1 -> t

Las claves del diccionario serán los estados de partida
mientras que los valores de cada clave serán los estados
de destino y el número de veces que transitan a cada estado
'''

q0 = self._q0
qF = self._qF
contador_transiciones = {q0: dict()}

for oracion in self._corpus:
    # Contador de transición q0 a estado q1
    q1 = oracion[0].Tag()
    if q1 not in contador_transiciones[q0]:
        contador_transiciones[q0][q1] = 0
    contador_transiciones[q0][q1] += 1

#####
##### mi código #####
#####

for it in range(0, len(oracion)-1): # recorre cada palabra de la oracion hasta 1
    if q0 in contador_transiciones: # comprueba si la etiqueta de la palabra an
        if oracion[it].Tag() in contador_transiciones[q0]: # busca si la etiqueta
            contador_transiciones[q0][oracion[it].Tag()] += 1 # incrementa el contad
        else:
            contador_transiciones[q0][oracion[it].Tag()] = 1 # crea el contador de 1
        else: # si no esta la etiqueta en el contador de transiciones, entonces la a
            contador_transiciones[q0]=dict() # se agrega la etiqueta q0 como un diccio
            q0 = oracion[it].Tag() # la palabra actual se convierte en la palabra anteri
            q0 = self._q0 # Se resetea la variable q0 a su parametro de entrada por defecto

    # Contador de transición qF_1 a qF
    qF_1 = oracion[-1].Tag()

    if qF_1 not in contador_transiciones:
        contador_transiciones[qF_1] = dict()
    if qF not in contador_transiciones[qF_1]:
        contador_transiciones[qF_1][qF] = 0

    contador_transiciones[qF_1][qF] += 1

'''
Cálculo de la tabla de probabilidades de transición.

Se calculan ahora las probabilidades de transición
siguiendo la relación:  $P(T|T-1) = C(T-1, T) / C(T-1).$ 

```

```

En 'contador_transiciones' se han acumulado la coincidencias  $C(T-1, T)$ 
y en 'estados' se tiene disponible  $C(T-1)$  por lo que es posible
calcular la tabla de probabilidades de transiciones con estos elementos.
'''

tags_estados_iniciales = list(
    self.Estados(incluir_inicial=True).keys())
tags_estados_finales = list(self.Estados(incluir_final=True).keys())
estados_totales = self.Estados(
    incluir_inicial=True, incluir_final=True)

prob_trans = {qt_1: {qt: 0 for qt in tags_estados_finales}
               for qt_1 in tags_estados_iniciales}
for qt_1 in tags_estados_iniciales:
    for qt in tags_estados_finales:
        prob = 0
        if qt_1 in contador_transiciones and qt in contador_transiciones[qt_1]:
            #####
            ##### Mi código #####
            #####
            prob = contador_transiciones[qt_1][qt]/estados_totales[qt_1] # se aplica la

        prob_trans[qt_1][qt] = prob

self._prob_trans = pd.DataFrame.from_dict(prob_trans, orient='index')

return self._prob_trans.copy()

def ProbabilidadesDeEmision(self):
    '''
    Método para calcular las probabilidades de emisión
    a partir del corpus proporcionado a la clase
    '''

    if len(self._prob_obs) != 0:
        return self._prob_obs.copy()

    '''
    En esta parte del código se calculan el número de
    ocurrencias de la palabra  $W_i$  para la etiqueta  $T_i$ 
    '''

    estados = self.Estados()
    contador_observaciones = {key: dict() for key in estados.keys()}

    for oracion in self._corpus:
        for palabra in oracion:
            token = palabra.Token()
            etiqueta = palabra.Tag()
            if token not in contador_observaciones[etiqueta]:
                #####
                ##### Mi código #####
                #####
                contador_observaciones[etiqueta][token] = 0 #el contador de observaciones s

            contador_observaciones[etiqueta][token] += 1

```

```

'''
Cálculo de la tabla de probabilidades de emisión.

Se calculan ahora las probabilidades de emisión
siguiendo la relación:  $P(W_i|T_i) = C(T_i, W_i) / C(T_i)$ .

En 'contador_observaciones' se han acumulado la coincidencias  $C(T_i, W_i)$ 
y en 'estados' se tiene disponible  $C(T_i)$  por lo que es posible
calcular la tabla de probabilidad de emisión con estos elementos.
'''

tokens = self.Tokens()
prob_obs = {Ti: {Wi: 0 for Wi in tokens} for Ti in estados}
for Ti in estados:
    for Wi in tokens:
        prob = 0
        if Ti in contador_observaciones and Wi in contador_observaciones[Ti]:

            #####
            ##### mi código #####
            #####
            prob = contador_observaciones[Ti][Wi]/estados[Ti] # aplica la relacion

        prob_obs[Ti][Wi] = prob

self._prob_obs = pd.DataFrame.from_dict(prob_obs, orient='index')

return self._prob_obs

```

El siguiente código te permite crear el HMM Bigrama y obtener información relevante:

```
hmmbigrama = HMMBigrama(corpus)
```

```
hmmbigrama.Tokens()
```

```

{'Tristana': 5,
 'es': 28,
 'una': 39,
 'película': 27,
 'de': 236,
 'el': 147,
 'director': 3,
 'español': 1,
 'nacionalizado': 1,
 'mexicano': 1,
 'Luis_Buñuel': 2,
 '.': 178,
 'Está': 1,
 'basada': 2,
 'en': 128,
 'la': 137,
 'novela': 2,

```

```
'mismo': 2,
'nombre': 3,
'Benito_Pérez_Galdós': 2,
'Fue': 2,
'nominada': 1,
'a': 119,
'Oscar': 3,
'mejor': 5,
'habla': 7,
'no': 18,
'inglesa': 1,
'1970': 1,
'y': 112,
'Nazarín': 1,
'son': 3,
'las': 15,
'dos': 9,
'novelas': 1,
'que': 133,
'Buñuel': 2,
'adaptó': 1,
'cine': 4,
'La': 9,
'pasó': 2,
'ser': 12,
'uno': 1,
'esos': 1,
'proyectos': 1,
'largamente': 1,
'acariciados': 1,
'por': 37,
'constantemente': 1,
'aplazados': 1,
'Hubo': 1,
'otras': 1,
'tentativas': 1,
'realizar': 2,
':': 17,
'México': 1,
'1952': 1,
...

```

```
len(hmmbigrama.Tokens())
```

```
1501
```

```
hmmbigrama.Estados()
```

```
{ 'AQ0FS0': 12,
  'AQ0MP0': 1,
  'AQ0MS0': 4,
  'AQ0CN0': 2,
  'AQ0CP0': 10,
  'AQ0CS0': 73,
  'AQ0FP0': 11,
  'AQ0FS0': 35,
  'AQ0MP0': 15,
  'AQ0MS0': 41,
  'CC': 152,

```

```

'CS': 99,
'DA0FP0': 18,
'DA0FS0': 142,
'DA0MP0': 44,
'DA0MS0': 157,
'DA0NS0': 15,
'DD0CP0': 1,
'DD0CS0': 2,
'DD0FP0': 1,
'DD0FS0': 7,
'DD0MP0': 5,
'DD0MS0': 13,
'DI0CS0': 7,
'DI0FP0': 8,
'DI0FS0': 46,
'DI0MP0': 18,
'DI0MS0': 65,
'DP3CP0': 18,
'DP3CS0': 61,
'Fc': 206,
'Fd': 17,
'Fe': 54,
'Fg': 2,
'Fp': 178,
'Fpa': 31,
'Fpt': 31,
'Fs': 3,
'Fx': 25,
'Fz': 2,
'I': 1,
'NC00000': 2,
'NCCP000': 14,
'NCCS000': 12,
'NCFN000': 1,
'NCFP000': 49,
'NCFS000': 271,
'NCMN000': 13,
'NCMP000': 128,
'NCMS000': 273,
'NCMS00D': 2,
'NP00000': 320,
'P0000000': 69,
'PD0FS000': 1,
'PD0MP000': 1,
'PD0MS000': 4,
'PD0NS000': 7,
'PD0CS000': 5

```

```
len(hmmbigrama.Estados())
```

```
134
```

El método `ProbabilidadesDeTransición()` de la clase `HMMBigrama` devuelve la tabla de probabilidades de transición.

```
def non_zero_green(val):
    '''
```

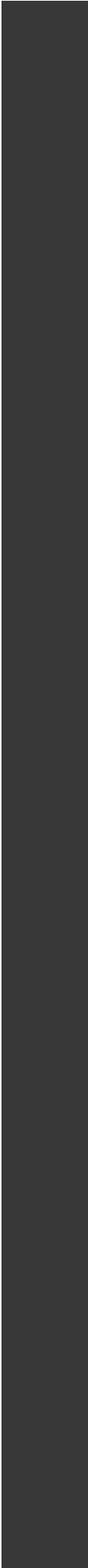
```
Función para resaltar en verde las probabilidades que no sean 0  
'''  
return 'background-color: Aquamarine' if val > 0 else ''
```

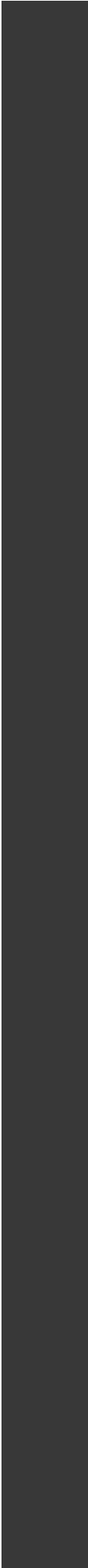
```
prob_transicion = hmmbigrama.ProbabilidadesDeTransicion()  
prob_transicion.style.applymap(non_zero_green)
```

5/2/22, 2:25 PM

mia07_t3_tra_Estandar.ipynb - Colaboratory

PP1CSN00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fs	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMM02S0	0.000000	0.000000	0.000000	0.250000	0.500000	0.000000	0.000000	0.000000	0.000000
VAIS3S0	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMSP2S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Zd	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NC00000	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NCMS00D	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VSIS3P0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3MS000	0.000000	0.000000	0.000000	0.000000	0.166667	0.000000	0.000000	0.000000	0.000000
PI0MP000	0.000000	0.000000	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000
VMII1S0	0.033333	0.000000	0.000000	0.000000	0.200000	0.033333	0.000000	0.000000	0.000000
VAII1S0	0.000000	0.000000	0.000000	0.000000	0.062500	0.000000	0.000000	0.000000	0.000000
PD0FS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AQ0CN0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.500000	0.000000	0.000000
I	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VSII1S0	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3NS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VSIC1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMIP1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Zu	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PT0CS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fg	0.000000	0.000000	0.000000	0.000000	0.500000	0.000000	0.000000	0.000000	0.000000
AO0MP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PT0CN000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMSI1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DD0CP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DD0FP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3FP000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NCFN000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PT0CP000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fz	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000





```
prob_transicion.to_excel('mia07_t3_tra_resultados_trans.xlsx', sheet_name='prob_trans')
```

El método `ProbabilidadesDeEmision()` de la clase `HMMBigrama` devuelve la tabla de probabilidades de emisión.

```
prob_emision = hmmbigrama.ProbabilidadesDeEmision()  
prob_emision.style.applymap(non_zero_green)  
#prob_emision
```



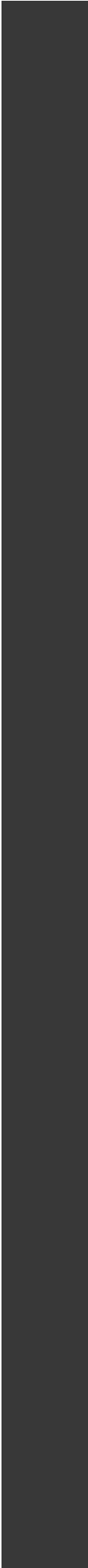
5/2/22, 2:25 PM

mia07_t3_tra_Estandar.ipynb - Colaboratory

PP1CSN00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fs	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMM02S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VAIS3S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMSP2S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Zd	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NC00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NCMS00D	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VSIS3P0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3MS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PI0MP000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMII1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VAII1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PD0FS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AQ0CN0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
I	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VSII1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3NS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VSIC1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMIP1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Zu	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PT0CS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fg	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AO0MP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PT0CN000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMSI1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DD0CP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DD0FP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3FP000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NCFN000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PT0CP000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fz	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

https://colab.research.google.com/drive/1lwW-s5kpRqPSysGxcn5vl6dlwRtlr-k4?authuser=1#scrollTo=ZWH2_PACb6i1&printMode=true

19/22





```
prob_emision.to_excel('mia07_t3_tra_resultados_emision.xlsx', sheet_name='prob_emision')
```

Parte 2: Etiquetar morfosintácticamente una oración

En esta segunda parte de la actividad tienes que implementar en Python un programa que permita calcular la mejor secuencia de etiquetas para una oración, dicho de otro modo, realizar el etiquetado morfosintáctico de la oración: «Habla con el enfermo grave de trasplantes. ».

Para ello debes utilizar el etiquetador que has construido en la parte 1 de esta actividad, es decir las tablas de probabilidades calculadas, y aplicar el algoritmo de Viterbi.

Para aplicar el algoritmo de Viterbi, se deben seguir los siguientes pasos:

- Calcular la matriz de probabilidades de la ruta se Viterbi (matriz con los valores de Viterbi) donde se representen claramente las observaciones y los estados de la máquina de estados finitos. Calcula el valor de Viterbi para cada celda de la matriz e indica claramente los valores obtenidos. Nota: Para simplificar, puedes eliminar todos aquellos estados asociados a etiquetas que no aparezcan en el posible análisis de la oración y sólo