

Datos del alumno: Jose Miguel Chacón

Fecha: 02 de Mayo de 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 todas las 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 continen 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.

```
1 class Palabra:
2     '''
3     Clase para guardar el token y la etiqueta de una palabra de un corpus
4     '''
5     '''
```

```

6 def __init__(self, token: str, tag: str):
7     '''
8     Constructor de la clase
9
10    token : str
11        Token de la palabra
12
13    tag : str
14        Etiqueta de la palabra
15    '''
16    self._token = token
17    self._tag = tag
18
19    def Token(self):
20        '''
21        Método para acceder al token de la palabra
22        '''
23        return self._token
24
25    def Tag(self):
26        '''
27        Método par acceder a la etiqueta de la palabra
28        '''
29        return self._tag
30

```

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.

```

1 archivo = open('mia07_t3_tra_Corpus-tagged.txt', "r")
2
3 corpus = list()
4 oracion_actual = list()
5
6 for entrada in archivo.readlines():
7     entrada = entrada.split()
8     if len(entrada) == 0:
9         # Puede ser la primera oración del documento
10        # 0 que termina la oración
11        if len(oracion_actual) > 0:
12            # Fin de la oración
13            corpus.append(oracion_actual)
14            oracion_actual = list()
15            continue
16
17    elif entrada[0] == '<doc':
18        # Inicio de documento. No se hace nada
19        continue
20
21    elif entrada[0] == '</doc>':
22        # Fin del documento. No se hace nada
23        continue
24
25    oracion_actual.append(Palabra(token=entrada[0], tag=entrada[2]))
26
27 archivo.close()

```

```

1 corpus
~_main_.Palabra at 0x17d1d033c50:

```

```

<__main__.Palabra at 0x7f7dfa035cd0>,
<__main__.Palabra at 0x7f7dfa035dd0>,
<__main__.Palabra at 0x7f7dfa035e50>,
<__main__.Palabra at 0x7f7dfa035f10>,
<__main__.Palabra at 0x7f7dfa035f90>,
<__main__.Palabra at 0x7f7dfa037090>,
<__main__.Palabra at 0x7f7dfa0370d0>],
[<__main__.Palabra at 0x7f7dfa037210>, <__main__.Palabra at 0x7f7dfa0371d0>],
[<__main__.Palabra at 0x7f7dfa037310>,
<__main__.Palabra at 0x7f7dfa0373d0>,
<__main__.Palabra at 0x7f7dfa0374d0>,
<__main__.Palabra at 0x7f7dfa037490>],
[<__main__.Palabra at 0x7f7dfa0375d0>,
<__main__.Palabra at 0x7f7dfa037690>,
<__main__.Palabra at 0x7f7dfa037790>,
<__main__.Palabra at 0x7f7dfa037810>,
<__main__.Palabra at 0x7f7dfa0378d0>,
<__main__.Palabra at 0x7f7dfa037950>,
<__main__.Palabra at 0x7f7dfa037a10>,
<__main__.Palabra at 0x7f7dfa037a50>,
<__main__.Palabra at 0x7f7dfa037ad0>,
<__main__.Palabra at 0x7f7dfa037b90>,
<__main__.Palabra at 0x7f7dfa037bd0>,
<__main__.Palabra at 0x7f7dfa037cd0>,
<__main__.Palabra at 0x7f7dfa037c90>,
<__main__.Palabra at 0x7f7dfa037d90>,
<__main__.Palabra at 0x7f7dfa037dd0>,
<__main__.Palabra at 0x7f7dfa037e50>,
<__main__.Palabra at 0x7f7dfa037f10>,
<__main__.Palabra at 0x7f7dfa037fd0>,
<__main__.Palabra at 0x7f7dfa0380d0>,
<__main__.Palabra at 0x7f7dfa038110>,
<__main__.Palabra at 0x7f7dfa0381d0>,
<__main__.Palabra at 0x7f7dfa038210>,
<__main__.Palabra at 0x7f7dfa038290>,
<__main__.Palabra at 0x7f7dfa0383d0>,
<__main__.Palabra at 0x7f7dfa038450>,
<__main__.Palabra at 0x7f7dfa038550>,
<__main__.Palabra at 0x7f7dfa038510>,
<__main__.Palabra at 0x7f7dfa038650>,
<__main__.Palabra at 0x7f7dfa038750>,
<__main__.Palabra at 0x7f7dfa038710>,
<__main__.Palabra at 0x7f7dfa038850>,
<__main__.Palabra at 0x7f7dfa038950>,
<__main__.Palabra at 0x7f7dfa0389d0>,
<__main__.Palabra at 0x7f7dfa038a90>,
<__main__.Palabra at 0x7f7dfa038b50>,
<__main__.Palabra at 0x7f7dfa038c50>,
<__main__.Palabra at 0x7f7dfa038d10>,
<__main__.Palabra at 0x7f7dfa038cd0>,
<__main__.Palabra at 0x7f7dfa038e10>,
<__main__.Palabra at 0x7f7dfa038e50>,
<__main__.Palabra at 0x7f7dfa038f10>,
<__main__.Palabra at 0x7f7dfa038f50>,
<__main__.Palabra at 0x7f7dfa038fd0>,
<__main__.Palabra at 0x7f7dfa03a150>,
<__main__.Palabra at 0x7f7dfa03a1d0>,
<__main__.Palabra at 0x7f7dfa03a290>,

```

El siguiente código te permite imprimir el corpus:

```

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

```

cancion NCRS000

```
aparece VMIP3S0
en SPS00
el DA0MS0
álbum NCMS000
Murder_Metal NP00000
. Fp
Referencias NP00000
. Fp
---- Fz
Enlaces NP00000
externos AQOMP0
. Fp
---- Fz
Crimelibrary NP00000
habla VMIP3S0
de SPS00
Young NP00000
( Fpa
Inglés NCMS000
) Fpt
; Fx
Vida_de_Young NP00000
y CC
crímenes NCMP000
( Fpa
Inglés NCMS000
) Fpt
; Fx
Artículo NP00000
sobre SPS00
Young NP00000
( Fpa
Inglés NCMS000
) Fpt
; Fx
Adolescente NP00000
japonesa AQ0FS0
envenena VMIP3S0
a SPS00
su DP3CS0
madre NCFS000
y CC
lo DA0NS0
cuenta NCFS000
en SPS00
un DI0MS0
blog NCMN000
haciendo VMG0000
honor NCMS000
a SPS00
Young NP00000
( Fpa
Inglés NCMS000
) Fpt
; Fx
Perfil NP00000
de SPS00
la DA0FS0
```

## ▼ 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.

```

1 #Se usa pandas para crear las tablas.
2 import pandas as pd
3
4 class HMMBigrama:
5     '''
6     Clase para obtener las matrices de probabilidad HMM Bigrama a partir de un corpus
7     '''
8
9     def __init__(self, corpus: [[Palabra]]):
10         '''
11         Constructor de la clase para calcular el Modelo Oculto de Markov Bigrama
12         '''
13         self._corpus = corpus
14         self._estados = dict()
15         self._tokens = dict()
16         self._q0 = 'q0'
17         self._qF = 'qF'
18
19         self._prob_trans = pd.DataFrame()
20         self._prob_obs = pd.DataFrame()
21
22     def Corpus(self):
23         return self._corpus.copy()
24
25     def EstadoInicial(self):
26         return self._q0
27
28     def EstadoFinal(self):
29         return self._qF
30
31     def _ProcesarCorpus(self):
32         '''
33         Método para contar el número de ocurrencias de estados y tokens
34         '''
35         for oracion in self._corpus:
36             for palabra in oracion:
37
38                 # Se recorren todas las palabras de todas las oraciones del corpus recuperando las etiquetas (estados)
39                 estado = palabra.Tag()
40                 estados = self._estados
41                 estados[estado] = estados[estado] + 1 if estado in estados else 1
42
43                 # Se recorren todas las palabras de todas las oraciones del corpus recuperando los tokens
44                 token = palabra.Token()
45                 tokens = self._tokens
46                 tokens[token] = tokens[token] + 1 if token in tokens else 1
47
48
49     def Estados(self, incluir_inicial: bool = False, incluir_final: bool = False):
50         '''
51         Devuelve los estados del bigrama en base al corpus proporcionado al constructor
52
53         incluir_inicial : bool (False)
54             Flag para indicar si se quiere recuperar el estado inicial
55
56         incluir_final : bool (False)
57             Flag para indicar si se quiere recuperar el estado final
58
59         return
60             Diccionario de estados con el número de ocurrencias de cada estado en el corpus
61         '''

```

```

62
63     if len(self._estados) == 0:
64         self._ProcesarCorpus()
65
66     copia_estados = dict()
67     if incluir_inicial:
68         # Hay tantos estados como oraciones en el corpus
69         copia_estados[self._q0] = len(self._corpus)
70
71     copia_estados.update(self._estados)
72
73     if incluir_final:
74         # Hay tantos estados como oraciones en el corpus
75         copia_estados[self._qF] = len(self._corpus)
76
77     return copia_estados
78
79 def Tokens(self):
80     '''
81     Devuelve los tokens del bigrama en base al corpus proporcionado al constructor
82
83     return
84     Diccionario de tokens con el número de ocurrencias de cada token en el corpus
85     '''
86
87     if len(self._tokens) == 0:
88         self._ProcesarCorpus()
89
90     return self._tokens.copy()
91
92
93 def ProbabilidadesDeTransicion(self):
94     '''
95     Método para calcular las probabilidades de transición bigrama
96     a partir del corpus proporcionado a la clase
97     '''
98
99     # Si ya se ha calculado se devuelve
100    if len(self._prob_trans) != 0:
101        return self._prob_trans.copy()
102
103    '''
104    En esta parte del código se calcula el número de
105    transiciones bigrama, es decir, en el diccionario
106    'contador_transiciones' se almacenarán los contadores
107    de las transiciones t-1 -> t
108
109    Las claves del diccionario serán los estados de partida
110    mientras que los valores de cada clave serán los estados
111    de destino y el número de veces que transitan a cada estado
112    '''
113    q0 = self._q0
114    qF = self._qF
115    contador_transiciones = {q0: dict()}
116
117    for oracion in self._corpus:
118        # Contador de transición q0 a estado q1
119        q1 = oracion[0].Tag()
120        if q1 not in contador_transiciones[q0]:
121            contador_transiciones[q0][q1] = 0
122            contador_transiciones[q0][q1] += 1

```

```

123 # Contador de transiciones entre palabras de la oración
124 for it in range(0, len(oracion) - 1):
125     #####
126     ##### 1. INICIO código añadido #####
127     #####
128     # Se completa el contador de transiciones
129
130
131     # Sí la entrada para el diccionario no existe, es necesario crearla
132     if oracion[it].Tag() not in contador_transiciones:
133         contador_transiciones[oracion[it].Tag()] = dict()
134
135     # Sí no existen valores para el bigrama (tags oracion[it], oracion[it+1]) en el diccionario, se inicializa
136     if oracion[it + 1].Tag() not in contador_transiciones[oracion[it].Tag()]:
137         contador_transiciones[oracion[it].Tag()][oracion[it + 1].Tag()] = 0
138
139     # Después de las validaciones anteriores, finalmente se incrementa en 1 el conteo para el bigrama (oracion[it], oracion[it+1])
140     contador_transiciones[oracion[it].Tag()][oracion[it+1].Tag()] += 1
141
142     #####
143     ##### FIN código añadido #####
144     #####
145
146 # Contador de transición qF_1 a qF
147 qF_1 = oracion[-1].Tag()
148
149 if qF_1 not in contador_transiciones:
150     contador_transiciones[qF_1] = dict()
151 if qF not in contador_transiciones[qF_1]:
152     contador_transiciones[qF_1][qF] = 0
153
154 contador_transiciones[qF_1][qF] += 1
155
156 '''

```

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.

```

165 '''
166 tags_estados_iniciales = list(
167     self.Estados(incluir_inicial=True).keys())
168 tags_estados_finales = list(self.Estados(incluir_final=True).keys())
169 estados_totales = self.Estados(
170     incluir_inicial=True, incluir_final=True)
171
172 prob_trans = {qt_1: {qt: 0 for qt in tags_estados_finales}
173               for qt_1 in tags_estados_iniciales}
174 for qt_1 in tags_estados_iniciales:
175     for qt in tags_estados_finales:
176         prob = 0
177         if qt_1 in contador_transiciones and qt in contador_transiciones[qt_1]:
178             #####
179             ##### 2. INICIO código añadido #####
180             #####
181
182         # Sí la condición se cumple, entonces se calcula la probabilidad de transición aplicando la fórmula  $p(T_i|T_{i-1}) = c(T_{i-1}, T_i) / c(T_{i-1})$ . [Página 13, tema 3]
183         prob = contador_transiciones[qt_1][qt] / estados_totales[qt_1]

```



```

184 #####
185 ##### FIN código añadido #####
186 #####
187 #####
188
189     prob_trans[qt_1][qt] = prob
190
191     self._prob_trans = pd.DataFrame.from_dict(prob_trans, orient='index')
192
193     return self._prob_trans.copy()
194
195 def ProbabilidadesDeEmision(self):
196     '''
197     Método para calcular las probabilidades de emisión
198     a partir del corpus proporcionado a la clase
199     '''
200
201     if len(self._prob_obs) != 0:
202         return self._prob_obs.copy()
203
204     '''
205     En esta parte del código se calculan el número de
206     ocurrencias de la palabra Wi para la etiqueta Ti
207     '''
208     estados = self.Estados()
209     contador_observaciones = {key: dict() for key in estados.keys()}
210
211     for oracion in self._corpus:
212         for palabra in oracion:
213             token = palabra.Token()
214             etiqueta = palabra.Tag()
215             if token not in contador_observaciones[etiqueta]:
216                 #####
217                 ##### 3. INICIO código añadido #####
218                 #####
219
220                 # Si no existe un valor para la variable contador_observaciones en los índices [etiqueta][token], entonces se inicializa con cero
221                 contador_observaciones[etiqueta][token] = 0
222
223                 #####
224                 ##### FIN código añadido #####
225                 #####
226
227                 contador_observaciones[etiqueta][token] += 1
228
229     '''
230     Cálculo de la tabla de probabilidades de emisión.
231
232     Se calculan ahora las probabilidades de emisión
233     siguiendo la relación:  $P(W_i|T_i) = C(T_i, W_i) / C(T_i)$ .
234
235     En 'contador_observaciones' se han acumulado la coincidencias C(Ti, Wi)
236     y en 'estados' se tiene disponible C(Ti) por lo que es posible
237     calcular la tabla de probabilidad de emisión con estos elementos.
238     '''
239     tokens = self.Tokens()
240     prob_obs = {Ti: {Wi: 0 for Wi in tokens} for Ti in estados}
241     for Ti in estados:
242         for Wi in tokens:
243             prob = 0
244             if Ti in contador_observaciones and Wi in contador_observaciones[Ti]:

```

```

245 #####
246 ##### 4. INICIO código añadido #####
247 #####
248
249 # Sí la condición se cumple, entonces, se realiza el cálculo la probabilidad de emisión siguiendo la fórmula  $p(W_i|T_i) = c(T_i, W_i) / c(T_i)$ . [Página 13, ten
250 prob = contador_observaciones[Ti][Wi] / estados[Ti]
251
252 #####
253 ##### FIN código añadido #####
254 #####
255
256 prob_obs[Ti][Wi] = prob
257
258 self._prob_obs = pd.DataFrame.from_dict(prob_obs, orient='index')
259 return self._prob_obs
260

```

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

```

1 hmmbigrama = HMMBigrama(corpus)

```

```

1 hmmbigrama.Tokens()
   emocionamente : 1,
   'pacientes': 1,
   'grado': 1,
   'romper': 1,
   'reglas': 1,
   'Historia': 1,
   'Antes_de': 1,
   'temporada': 5,
   'niña': 2,
   'era': 5,
   'pobre': 1,
   'vivía': 2,
   'camping': 1,
   'caravanas': 1,
   'alrededor_de': 2,
   'Chehalis': 1,
   'Washington': 1,
   'quiso': 1,
   'doctora': 1,
   'así': 3,
   'ahorró': 1,
   'pudo': 2,
   'madre': 2,
   'gastó': 1,
   'dinero': 3,
   'fármacos': 1,
   'Por': 2,
   'ello': 1,
   'tuvo': 3,
   'modelo': 2,
   'ropa': 2,
   'interior': 2,
   'ganar': 2,
   '16': 1,
   'quedó': 2,
   'embarazada': 1,
   'dió': 1,
   'adopción': 1,
   'hija': 1,
   'Primera Temporada': 1.

```

```
'Entra': 1,  
'hospital': 5,  
'Grace_de_Seattle': 1,  
'todos': 2,  
'piensan': 1,  
'sería': 1,  
'enfermera': 2,  
'convirtió': 1,  
'rápidamente': 1,  
'amiga': 1,  
'George': 7,  
'Meredith': 3,  
'Cristina': 2,  
'enemiga': 1,  
'Alex_Karev': 1,  
'instala': 1,  
'junto_con': 1,  
'físico': 1,  
'soñadora': 1,
```

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

```
1501
```

```
1 hmmbigrama.Estados()
```

```
'PROCS000': 1,  
'PROFS000': 1,  
'PROMS000': 1,  
'PT0CN000': 2,  
'PT0CP000': 1,  
'PT0CS000': 1,  
'RG': 127,  
'RN': 18,  
'SPS00': 683,  
'VAIC1S0': 2,  
'VAII1S0': 16,  
'VAII3P0': 1,  
'VAIP3P0': 3,  
'VAIP3S0': 15,  
'VAIS3S0': 2,  
'VAN0000': 4,  
'VAP00SM': 2,  
'VASI1S0': 1,  
'VASP1S0': 1,  
'VMG0000': 32,  
'VMIC1S0': 4,  
'VMIC3P0': 1,  
'VMIF3P0': 2,  
'VMIF3S0': 2,  
'VMII1S0': 30,  
'VMII3P0': 4,  
'VMIP1S0': 1,  
'VMIP3P0': 41,  
'VMIP3S0': 134,  
'VMIS1S0': 1,  
'VMIS2S0': 1,  
'VMIS3P0': 4,  
'VMIS3S0': 69,  
'VMM02S0': 4,  
'VMM03P0': 1,  
'VMN0000': 88,  
'VMP00PF': 4,  
'VMP00PM': 13,  
'VMP00SF': 31,  
'VMP00SM': 76,
```

```
'VMSI1S0': 1,  
'VMSI3P0': 1,  
'VMSP1S0': 7,  
'VMSP2S0': 1,  
'VMSP3P0': 4,  
'VSIC1S0': 1,  
'VSIF3S0': 1,  
'VSII1S0': 5,  
'VSIP3P0': 3,  
'VSIP3S0': 31,  
'VSIS3P0': 5,  
'VSIS3S0': 21,  
'VSN0000': 12,  
'VSP00SM': 2,  
'VSSP1S0': 2,  
'W': 11,  
'Z': 55,  
'Zd': 1,  
'Zu': 1}
```

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

```
134
```

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

```
1 def non_zero_green(val):  
2     '''  
3     Función para resaltar en verde las probabilidades que no sean 0  
4     '''  
5     return 'background-color: Aquamarine' if val > 0 else ''
```

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

	NP00000	VSIP3S0	DIOFS0	NCFS000	SPS00	DAOMS0	NCMS000	AQOMS0	VMP00SM	Fp	VAIP3S0	VMP00SF	DAOFS0	VSIS3S0	AQ0CS0	RN	AQ0FS0	Z
q0	0.196532	0.005780	0.017341	0.000000	0.277457	0.057803	0.005780	0.000000	0.005780	0.000000	0.005780	0.005780	0.040462	0.011561	0.000000	0.000000	0.000000	0.023121
NP00000	0.000000	0.021875	0.000000	0.000000	0.078125	0.003125	0.009375	0.000000	0.003125	0.143750	0.003125	0.003125	0.006250	0.018750	0.006250	0.006250	0.003125	0.000000
VSIP3S0	0.000000	0.000000	0.161290	0.000000	0.064516	0.064516	0.000000	0.000000	0.129032	0.000000	0.000000	0.129032	0.000000	0.000000	0.096774	0.000000	0.032258	0.000000
DIOFS0	0.000000	0.000000	0.000000	0.739130	0.065217	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.065217	0.000000	0.065217	0.000000
NCFS000	0.025830	0.011070	0.000000	0.000000	0.361624	0.000000	0.003690	0.003690	0.003690	0.095941	0.000000	0.022140	0.000000	0.014760	0.066421	0.007380	0.062731	0.000000
SPS00	0.149341	0.000000	0.019034	0.058565	0.001464	0.149341	0.045388	0.001464	0.000000	0.000000	0.000000	0.000000	0.136164	0.000000	0.001464	0.000000	0.000000	0.038067
DAOMS0	0.019108	0.000000	0.000000	0.019108	0.012739	0.000000	0.694268	0.019108	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.019108	0.000000	0.000000	0.006369
NCMS000	0.040293	0.007326	0.000000	0.000000	0.322344	0.007326	0.010989	0.040293	0.018315	0.073260	0.007326	0.003663	0.000000	0.010989	0.054945	0.003663	0.000000	0.000000
AQOMS0	0.048780	0.000000	0.000000	0.000000	0.341463	0.000000	0.219512	0.000000	0.024390	0.024390	0.000000	0.000000	0.000000	0.024390	0.000000	0.000000	0.000000	0.000000
VMP00SM	0.000000	0.000000	0.000000	0.000000	0.421053	0.026316	0.039474	0.026316	0.000000	0.039474	0.000000	0.000000	0.000000	0.000000	0.013158	0.000000	0.000000	0.013158
Fp	0.000000	0.000000	0.000000	0.000000	0.005618	0.000000	0.000000	0.000000	0.000000	0.005618	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VAIP3S0	0.000000	0.000000	0.000000	0.000000	0.066667	0.000000	0.000000	0.000000	0.666667	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMP00SF	0.000000	0.000000	0.032258	0.032258	0.580645	0.000000	0.000000	0.000000	0.000000	0.129032	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.161290
DAOFS0	0.056338	0.000000	0.000000	0.711268	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.007042	0.000000	0.000000	0.028169	0.000000	0.021127	0.000000
VSIS3S0	0.047619	0.000000	0.047619	0.000000	0.142857	0.000000	0.000000	0.000000	0.380952	0.000000	0.000000	0.047619	0.095238	0.000000	0.000000	0.000000	0.000000	0.000000
AQ0CS0	0.027397	0.000000	0.000000	0.136986	0.246575	0.000000	0.095890	0.000000	0.013699	0.095890	0.000000	0.013699	0.000000	0.000000	0.000000	0.027397	0.000000	0.082270
RN	0.000000	0.055556	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.055556	0.000000	0.000000	0.000000	0.000000	0.000000	0.055556	0.000000
AQ0FS0	0.000000	0.000000	0.000000	0.142857	0.171429	0.000000	0.000000	0.000000	0.000000	0.085714	0.000000	0.085714	0.000000	0.000000	0.000000	0.000000	0.000000	0.200000
Z	0.072727	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.054545	0.000000	0.018182	0.000000	0.000000	0.000000	0.000000	0.000000	0.018182
CC	0.105263	0.006579	0.013158	0.019737	0.125000	0.006579	0.032895	0.013158	0.013158	0.000000	0.000000	0.006579	0.032895	0.000000	0.013158	0.000000	0.019737	0.000000
VSIP3P0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.333333
DAOFP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.055556	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.055556
NCFP000	0.040816	0.000000	0.000000	0.000000	0.387755	0.000000	0.000000	0.000000	0.020408	0.102041	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.142857
CS	0.141414	0.000000	0.010101	0.010101	0.070707	0.050505	0.030303	0.000000	0.000000	0.000000	0.000000	0.000000	0.030303	0.000000	0.000000	0.030303	0.000000	0.000000
VMIS3S0	0.000000	0.000000	0.043478	0.014493	0.231884	0.057971	0.014493	0.000000	0.043478	0.057971	0.000000	0.014493	0.043478	0.000000	0.014493	0.000000	0.000000	0.028986
VSN0000	0.000000	0.000000	0.000000	0.083333	0.000000	0.000000	0.083333	0.000000	0.416667	0.000000	0.000000	0.083333	0.000000	0.000000	0.083333	0.000000	0.000000	0.000000
PIOMS000	0.000000	0.000000	0.000000	0.000000	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DD0MP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.200000
NCMP000	0.007812	0.000000	0.000000	0.000000	0.343750	0.000000	0.000000	0.000000	0.000000	0.085938	0.000000	0.000000	0.000000	0.000000	0.000000	0.007812	0.000000	0.046875
RG	0.015748	0.015748	0.000000	0.000000	0.173228	0.000000	0.000000	0.023622	0.031496	0.047244	0.023622	0.000000	0.000000	0.000000	0.039370	0.015748	0.031496	0.000000
VMP00PM	0.000000	0.000000	0.000000	0.000000	0.615385	0.000000	0.000000	0.000000	0.000000	0.153846	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DIOFP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.125000
VMN0000	0.000000	0.000000	0.045455	0.022727	0.227273	0.056818	0.034091	0.011364	0.000000	0.034091	0.000000	0.000000	0.045455	0.000000	0.000000	0.000000	0.000000	0.034091
PP3FSA00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fd	0.529412	0.058824	0.117647	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.058824	0.000000	0.000000	0.000000	0.000000	0.000000
Fc	0.155340	0.019417	0.019417	0.014563	0.169903	0.038835	0.014563	0.000000	0.014563	0.000000	0.000000	0.009709	0.033981	0.004854	0.004854	0.009709	0.000000	0.106383

PI0FS000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VASI1S0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VAP00SM	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.500000	0.000000	0.000000	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AO0FS0	0.000000	0.000000	0.000000	0.750000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.083333	0.000000	0.000000	0.000000	0.000000	0.166667
DP3CP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DI0MS0	0.092308	0.000000	0.000000	0.015385	0.000000	0.030769	0.600000	0.061538	0.015385	0.000000	0.000000	0.000000	0.000000	0.000000	0.046154	0.000000	0.000000	0.000000	0.000000
PR0FS000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DP3CS0	0.000000	0.000000	0.000000	0.524590	0.016393	0.000000	0.311475	0.049180	0.016393	0.000000	0.000000	0.000000	0.000000	0.000000	0.016393	0.000000	0.000000	0.000000	0.000000
PD0MS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMIP3S0	0.022388	0.007463	0.029851	0.014925	0.283582	0.037313	0.014925	0.007463	0.014925	0.022388	0.000000	0.007463	0.059701	0.000000	0.000000	0.000000	0.014925	0.000000	0.029851
VMG0000	0.000000	0.000000	0.000000	0.000000	0.312500	0.031250	0.031250	0.000000	0.000000	0.000000	0.000000	0.000000	0.031250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3CNA00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.066667	0.000000	0.066667	0.066667	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VSIF3S0	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DD0MS0	0.076923	0.000000	0.000000	0.000000	0.000000	0.000000	0.846154	0.000000	0.076923	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
P0000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
W	0.181818	0.000000	0.000000	0.000000	0.181818	0.000000	0.000000	0.000000	0.000000	0.181818	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NCMN000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.076923	0.000000	0.076923	0.000000	0.000000	0.000000	0.000000	0.384615	0.000000	0.000000	0.000000	0.153846
DA0MP0	0.022727	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.045455	0.000000
NCCP000	0.071429	0.000000	0.000000	0.000000	0.214286	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.214286
VMIP3P0	0.024390	0.000000	0.024390	0.000000	0.292683	0.024390	0.000000	0.000000	0.000000	0.024390	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.024390	0.024390
AQ0FP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.090909	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.181818
PR0CN000	0.025641	0.051282	0.000000	0.000000	0.038462	0.012821	0.000000	0.000000	0.000000	0.000000	0.038462	0.000000	0.000000	0.000000	0.000000	0.025641	0.000000	0.000000	0.000000
PP3MPA00	0.000000	0.000000	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
DA0NS0	0.000000	0.000000	0.000000	0.066667	0.000000	0.000000	0.000000	0.133333	0.066667	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AQ0CP0	0.000000	0.000000	0.000000	0.100000	0.300000	0.000000	0.000000	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AQ0MP0	0.000000	0.000000	0.000000	0.000000	0.066667	0.000000	0.000000	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.133333
PR0CS000	0.117647	0.000000	0.000000	0.000000	0.000000	0.058824	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.058824	0.000000	0.000000	0.000000	0.000000	0.000000
DI0MP0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fe	0.148148	0.018519	0.000000	0.037037	0.111111	0.018519	0.055556	0.000000	0.000000	0.129630	0.000000	0.000000	0.018519	0.000000	0.037037	0.000000	0.000000	0.018519	0.000000
VMSP3P0	0.000000	0.000000	0.250000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.250000	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PP3CN000	0.000000	0.000000	0.000000	0.000000	0.625000	0.000000	0.125000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.125000
Fpa	0.387097	0.000000	0.000000	0.064516	0.193548	0.000000	0.129032	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.032258	0.000000
Fpt	0.000000	0.032258	0.000000	0.000000	0.064516	0.000000	0.000000	0.000000	0.000000	0.161290	0.000000	0.000000	0.000000	0.032258	0.000000	0.000000	0.000000	0.000000	0.064516
VMIS3P0	0.000000	0.000000	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.250000	0.000000
VMM03P0	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
PR000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMII3P0	0.000000	0.000000	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fx	0.720000	0.000000	0.000000	0.000000	0.120000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.080000



<b>VSIS3P0</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>PP3MS000</b>	0.000000	0.000000	0.000000	0.000000	0.166667	0.000000	0.000000	0.083333	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.083333	0.000000	0.000000	0.083333
<b>PI0MP000</b>	0.000000	0.000000	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>VMII1S0</b>	0.033333	0.000000	0.000000	0.000000	0.200000	0.033333	0.000000	0.000000	0.000000	0.100000	0.000000	0.000000	0.033333	0.000000	0.000000	0.000000	0.000000	0.200000	0.000000
<b>VAII1S0</b>	0.000000	0.000000	0.000000	0.000000	0.062500	0.000000	0.000000	0.000000	0.687500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>PD0FS000</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>AQ0CN0</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>I</b>	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>VSII1S0</b>	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>PP3NS000</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>VSIC1S0</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000

```
1 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.

```
1 prob_emision = hmmbigrama.ProbabilidadesDeEmision()
2 prob_emision.style.applymap(non_zero_green)
```



[illegible]

[illegible]

[illegible]

<b>PP3MS000</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>PI0MP000</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>VMII1S0</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>VAII1S0</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

```

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

```

<b>Δ0N0C0N0</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
-----------------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

## 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 quedarte con los estados relevantes. Además, debes tener en cuenta la transición al estado final representado por el punto al final de la oración a analizar.
- Obtener la ruta con máxima probabilidad, es decir, traza la ruta inversa para obtener la mejor secuencia de etiquetas.
- Mostrar la oración etiquetada. Debes indicar claramente el resultado obtenido del etiquetado morfosintáctico de la oración estudiada.

**Nota:** Presenta en el envío de la actividad la tabla (guardada en formato de hoja de cálculo de Microsoft Excel (.xlsx) o equivalente) con la matriz de probabilidades de la ruta Viterbi para el etiquetado morfosintáctico de la oración «Habla con el enfermo grave de trasplantes. ».

<b>NCFN000</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
----------------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

## Calcular la matriz de probabilidades de la ruta de Viterbi

La clase `Viterbi` permitirá realizar el cálculo de la matriz de probabilidades de la ruta de Viterbi y la posterior decodificación de la secuencia óptima de etiquetado para una oración a analizar.

El etiquetado morfosintáctico creado en la Parte 1, es decir el objeto `hmmbigrama` de la clase `HMMBigrama`, será proporcionado al objeto `viterbi` de la clase `Viterbi` para poder aplicar el Algoritmo de Viterbi.

El cálculo de los valores de Viterbi se realiza en el método `Probabilidades()` de la clase `Viterbi`.

## ▼ Obtener la ruta con máxima probabilidad

El método `DecodificacionSecuenciaOptima()` de la clase `Viterbi` permite obtener la secuencia de etiquetas más probables para la oración a analizar.

```

1 class Viterbi:
2     '''
3     Algoritmo de Viterbi para obtener las mejores
4     etiquetas de las palabras de una oración
5     '''
6
7     def __init__(self, hmmbigrama: HMMBigrama, oracion: str):
8         self._hmmbigrama = hmmbigrama

```

```

9 self._oracion = oracion
10
11 self._estados_relevantes = None
12 self._prob_viterbi = pd.DataFrame()
13 self._estado_max_anterior = None
14
15 def _CalculoEstadosRelevantes(self):
16     self._estados_relevantes = set()
17     for palabra_analizar in [x.lower() for x in self._oracion.split()]:
18         # Búsqueda de estados
19         for oracion in self._hmmbigrama.Corpus():
20             for palabra_corpus in oracion:
21                 if palabra_corpus.Token() == palabra_analizar:
22                     self._estados_relevantes.add(palabra_corpus.Tag())
23
24 def Probabilidades(self):
25     if len(self._prob_viterbi) != 0:
26         return self._prob_viterbi.copy()
27
28     if not self._estados_relevantes:
29         self._CalculoEstadosRelevantes()
30
31     estados_relevantes = self._estados_relevantes
32
33     '''
34     Matriz en la que se guardan los valores de Viterbi
35     '''
36     matriz_viterbi = {q: dict() for q in estados_relevantes}
37
38     '''
39     Matriz asociada a la matriz de Viterbi en la que se almacena
40     el estado de origen que maximiza cada probabilidad
41     '''
42     self._estado_max_anterior = {q: dict() for q in estados_relevantes}
43
44     q0 = self._hmmbigrama.EstadoInicial()
45     prob_trans = self._hmmbigrama.ProbabilidadesDeTransicion()
46     prob_obs = self._hmmbigrama.ProbabilidadesDeEmision()
47
48     token_anterior = None
49     for token in [x.lower() for x in self._oracion.split()]:
50         for qDestino in estados_relevantes:
51
52             prob_max = 0
53             if not token_anterior:
54                 # Estado q0
55                 prob_max = prob_trans[qDestino][q0]
56             else:
57                 # Resto de estados
58                 for qOrigen in estados_relevantes:
59                     #####
60                     ##### 5. INICIO código añadido #####
61                     #####
62
63                 # Para cada estado relevante se obtiene la probabilidad de la matriz de proabilidades de transición anteriormente calculada, para esto se declara y as
64                 prob_qOrigen = prob_trans[qDestino][qOrigen]
65
66                 #####
67                 ##### FIN código añadido #####
68                 #####
69

```

```

70         if prob_qOrigien > prob_max:
71             #####
72             ##### 6. INICIO código añadido #####
73             #####
74
75             # Sí prob_qOrigien (anteriormente asignada) es mayor a prob_max, se asigna el valor de prob_qOrigien como la máxima probabilidad
76             prob_max = prob_qOrigien
77
78             #####
79             ##### FIN código añadido #####
80             #####
81
82             matriz_viterbi[qDestino][token] = prob_max * prob_obs[token][qDestino]
83
84             token_anterior = token
85
86             self._prob_viterbi = pd.DataFrame.from_dict(matriz_viterbi, orient='index')
87
88             return self._prob_viterbi.copy()
89
90     def DecodificacionSecuenciaOptima(self):
91         # Decodificación de la secuencia óptima
92         oracion_invertida = [x.lower() for x in self._oracion.split()]
93         oracion_invertida.reverse()
94
95         prob_viterbi = self.Probabilidades()
96
97         oracion_etiquetada = []
98         # Se busca la probabilidad máxima de Viterbi asociada a la última palabra de la oración
99         palabra = oracion_invertida[0]
100        etiqueta = prob_viterbi[palabra].idxmax()
101        oracion_etiquetada.append({'token': palabra, 'tag': etiqueta, 'prob': prob_viterbi[palabra].max()})
102
103        # Ahora se usa la tabla auxiliar de Viterbi que contiene
104        # el estado de origen que maximiza cada probabilidad Viterbi
105        palabra_anterior = palabra
106        for palabra in oracion_invertida[1:]:
107            #####
108            ##### 7. INICIO código añadido #####
109            #####
110
111            # Para cada iteración dentro de la oración se valida la máxima probabilidad de Viterbi para seleccionar la ruta con máxima probabilidad
112            etiqueta = prob_viterbi[palabra].idxmax()
113            oracion_etiquetada.append(
114                {'token': palabra, 'tag': etiqueta, 'prob': prob_viterbi[palabra].max()})
115
116            #####
117            ##### FIN código añadido #####
118            #####
119
120        # Se recupera el orden de la oración con las palabras ya etiquetadas
121        oracion_etiquetada.reverse()
122
123        return oracion_etiquetada

```

El siguiente código te permite realizar el análisis de la oración: "Habla con el enfermo grave de trasplantes."

```

1 viterbi = Viterbi(hmmbigrama=hmmbigrama, oracion='Habla con el enfermo grave de trasplantes .')

```

El siguiente código te permite mostrar la matriz de probabilidades de la ruta de Viterbi (solo se presentan aquellas etiquetas que tienen algún valor no nulo para alguna de las palabras de la oración analizada).

```
1 matriz_prob_viterbi = viterbi.Probabilidades()
2 matriz_prob_viterbi.style.applymap(non_zero_green)
```

	habla	con	el	enfermo	grave	de	trasplantes	.
Fp	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.095941
AQ0MS0	0.000000	0.000000	0.000000	0.000983	0.000000	0.000000	0.000000	0.000000
DA0MS0	0.000000	0.000000	0.139829	0.000000	0.000000	0.000000	0.000000	0.000000
NCFS000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
VMIP3S0	0.001294	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
AQ0CS0	0.000000	0.000000	0.000000	0.000000	0.002730	0.000000	0.000000	0.000000
SPS00	0.000000	0.019590	0.000000	0.000000	0.000000	0.124953	0.000000	0.000000
NCMS000	0.000000	0.000000	0.000000	0.010172	0.000000	0.000000	0.000000	0.000000
NCMP000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000755	0.000000

```
1 matriz_prob_viterbi.to_excel('mia07_t3_tra_resultados_viterbi.xlsx', sheet_name='viterbi')
```

El siguiente código te permite mostrar la ruta de Viterbi con máxima probabilidad

```
1 oracion_etiquetada = viterbi.DecodificacionSecuenciaOptima()
```

```
1 oracion_etiquetada
```

```
[{'prob': 0.0012941074971961003, 'tag': 'VMIP3S0', 'token': 'habla'},
{'prob': 0.019590151977654475, 'tag': 'SPS00', 'token': 'con'},
{'prob': 0.1398289673695107, 'tag': 'DA0MS0', 'token': 'el'},
{'prob': 0.010172417815729917, 'tag': 'NCMS000', 'token': 'enfermo'},
{'prob': 0.002729616337259263, 'tag': 'AQ0CS0', 'token': 'grave'},
{'prob': 0.12495340180341774, 'tag': 'SPS00', 'token': 'de'},
{'prob': 0.0007549414348462665, 'tag': 'NCMP000', 'token': 'trasplantes'},
{'prob': 0.0959409594095941, 'tag': 'Fp', 'token': '.'}]
```

▼ Mostrar la oración etiquetada

El siguiente código te permite mostrar la oración etiquetada

```
1 for palabra in oracion_etiquetada:
2     print('{} / {}'.format(palabra['token'], palabra['tag']))
```

```
habla / VMIP3S0
con / SPS00
el / DA0MS0
enfermo / NCMS000
grave / AQ0CS0
de / SPS00
```

### Parte 3: Analizar el etiquetador morfosintáctico

Una vez hayas creado el etiquetador morfosintáctico y lo hayas utilizado para etiquetar la oración «Habla con el enfermo grave de trasplantes.», reflexiona sobre los resultados obtenidos, interprétalos y analiza el rendimiento del etiquetador creado y sus limitaciones. Para ello responde de forma razonada a las siguientes preguntas:

- ¿Es correcto el etiquetado morfosintáctico que has obtenido? Indica por qué.

Sí es correcto el etiquetado morfosintáctico que se obtiene con la ejecución de este notebook, podemos verlo al analizar cada una de las palabras:

- **habla -> VMIP3S0** : Se refiere a un verbo (V) de tipo principal (M:main) en modo indicativo (I) en tiempo presente (P) para la tercera persona (3) de (número) singular (S) y (0) sin género. - *CORRECTO*
- **con -> SPS00** : Se refiere a una adposición (S) de tipo preposición (P). - *CORRECTO*
- **el -> DA0MS0** : Se refiere a un determinante (D) de tipo artículo (A) de género masculino (M) de (número) singular (S). - *CORRECTO*
- **enfermo -> NCMS000** : Se refiere a un sustantivo (N) de género masculino (M). C:common; S:person; no nesubclas; no degree. - *CORRECTO*
- **grave -> AQ0CS0** : Se refiere a un adjetivo (A) cualificativo (Q) de (número) singular (S). No degree; C:common; No possessorpers. - *CORRECTO*
- **de -> SPS00** : Se refiere a una adposición (S) de tipo preposición (P). - *CORRECTO*
- **trasplantes -> NCMP000** : Se refiere a un sustantivo (N) de género masculino (M) de (número) plural (P). C:common; No necclass; no nesubclass; no degree. - *CORRECTO*

Acá podemos notar que para otro contexto la palabra *enfermo* podría ser un adjetivo, sin embargo el etiquetador ha realizado correctamente su labor al etiquetarlo como sustantivo.

- Indica el resultado de etiquetar la oración «El enfermo grave habla de trasplantes.» utilizando el etiquetador morfosintáctico. ¿Es correcto el etiquetado morfosintáctico que has obtenido? Indica por qué.

```
1 viterbi = Viterbi(hmmbigrama = hmmbigrama,
2                   oracion = 'El enfermo grave habla de trasplantes .')
3 matriz_prob_viterbi = viterbi.Probabilidades()
4 matriz_prob_viterbi.style.applymap(non_zero_green)
5
```



	el	enfermo	grave	habla	de	trasplantes	.
<b>Fp</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.095941
<b>AQ0MS0</b>	0.000000	0.000983	0.000000	0.000000	0.000000	0.000000	0.000000
<b>DA0MS0</b>	0.054122	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

```
1 oracion_etiquetada = viterbi.DecodificacionSecuenciaOptima()
2 for palabra in oracion_etiquetada:
3     print('{} / {}'.format(palabra['token'], palabra['tag']))
4
```

el / DA0MS0  
enfermo / NCMS000  
grave / AQ0CS0  
habla / VMIP3S0  
de / SPS00  
trasplantes / NCMP000  
. / Fp

Aplicando el etiquetador a la oración "El enfermo grave habla de trasplantes", podemos ver los siguientes resultados:

- **el -> DA0MS0** : Se refiere a un determinante (D) de tipo artículo (A) de género masculino (M) de (número) singular (S). - *CORRECTO*
- **enfermo -> NCMS000** : Se refiere a un sustantivo (N) de género masculino (M). C:common; S:person; no nesubclas; no degree. - *CORRECTO*
- **grave -> AQ0CS0** : Se refiere a un adjetivo (A) cualificativo (Q) de (número) singular (S). No degree; C:common; No possessorpers. A:adjective; Q:qualificative; No degree; C:common; S:singular; No possessorpers. - *CORRECTO*
- **habla -> VMIP3S0** : Se refiere a un verbo (V) de tipo principal (M:main) en modo indicativo (I) en tiempo presente (P) para la tercera persona (3) de (número) singular (S) y (0) sin género. - *CORRECTO*
- **de -> SPS00** : Se refiere a una adposición (S) de tipo preposición (P). - *CORRECTO*
- **trasplantes -> NCMP000** : Se refiere a un sustantivo (N) de género masculino (M) de (número) plural (P). C:common; No necclass; no nesubclass; no degree. - *CORRECTO*

Por tanto, decimos que es correcto. El orden en este caso no afecta el contexto.

- ¿Cuáles son las limitaciones del analizador morfosintáctico que has creado?

Podemos decir los siguiente acerca de las limitaciones:

1. El analizador no funcionaría con palabras que no están en el corpus (debido a que no tendría valores en las tablas de probabilidades), y debido a que el modelo es pequeño este puede estar limitado.
2. La limitación anterior también aplica a los casos en que existan faltas de ortografía, ya que el analizador busca tokens exactos.
3. Al usar un corpus más grande sacrificaríamos procesamiento debido al aumento del tamaño de las matrices de probabilidades a construir y la relación de los bigramas.
4. Este analizador por su naturaleza depende mucho de un corpus debidamente construido y completo.

- ¿Qué posibles mejoras se podrían aplicar para mejorar el rendimiento del etiquetador morfosintáctico creado?

Para mejorar el desempeño del etiquetador podríamos:

1. Implementar un algoritmo que permita el cálculo paralelo de las probabilidades de transición dividiendo en partes del corpus y manejando un diccionario compartido entre los distintos hilos. Aunque esto aceleraría el tiempo de procesamiento de las matrices, no

resolvería el problema del alto uso de memoria.

2. Usar un corpus más grande y variado que nos permita un rango más amplio de vocabulario.
3. Usar trigramas para mejorar la precisión del analizador, a pesar de que se aumentaría la complejidad del modelo.
4. El estado del arte consiste en modelos basados en un tipo redes neuronales recurrentes: Long Short Term Memory (LSTM) bidireccionales que dan notables resultados. Por esto, sería válido experimentar con el desarrollo de un modelo mixto de ensamble que mejore el desempeño.