Tarea 10

NLP

Descripción: Dados 3 textos construir la matriz término-documento a partir de bigramas de palabras y trigramas de caracteres y determinar la similitud coseno.

Alumno: Miguel Angel Soto Hernandez

```
import nltk
import urllib.request as urllib2
from bs4 import BeautifulSoup
nltk.download('book')
from nltk.book import *
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.express as px
```

```
# Moby Dick
texto_1 = text1

# Sense and Sensibility
texto_2 = text2

# The book of genesis
texto_3 = text3
```

```
import re
def normalizar_texto(texto, quitar_numeros, incluir_puntuacion, convertir_a_minusculas, incluir_s
  if quitar_numeros == True:
    texto = [re.sub('\d', ' ', palabra) for palabra in texto]

  if incluir_puntuacion == True:
    texto = [palabra for palabra in texto if re.search('\w+', palabra)]

  if convertir_a_minusculas == True:
    texto = [palabra.lower() for palabra in texto]

  if incluir_stopwords == True:
    stopwords = sorted(nltk.corpus.stopwords.words('english'))
    texto = [palabra for palabra in texto if palabra not in stopwords]

  return texto
```

```
texto_1 = normalizar_texto(texto_1, quitar_numeros=True,
                           incluir_puntuacion=True,
                           convertir_a_minusculas= True,
                           incluir_stopwords=True)
print(texto_1[:10])
```

```
['moby', 'dick', 'herman', 'melville', 'etymology', 'supplied', 'late', 'consumptive', 'usher'
```

```
texto_2 = normalizar_texto(texto_2, quitar_numeros=True,
                           incluir_puntuacion=True,
                           convertir_a_minusculas= True,
                           incluir_stopwords=True)
print(texto_2[:10])
```

```
['sense', 'sensibility', 'jane', 'austen', 'chapter', 'family', 'dashwood', 'long', 'settled',
```

```
texto_3 = normalizar_texto(texto_3, quitar_numeros=True,
                           incluir_puntuacion=True,
                           convertir_a_minusculas= True,
                           incluir_stopwords=True)
print(texto_3[:10])
```

```
['beginning', 'god', 'created', 'heaven', 'earth', 'earth', 'without', 'form', 'void', 'darkne
```

```
def obtener_bigramas(texto):
  bigramas = list(nltk.bigrams(texto))
  return bigramas


def contar_bigramas(bigramas):
  frecuencia_bigramas = FreqDist(bigramas)
  return frecuencia_bigramas
```

```
bigramas_texto_1 = obtener_bigramas(texto_1)
print(bigramas_texto_1[:10])

frecuencia_bigramas_texto_1 = contar_bigramas(bigramas_texto_1)
print(frecuencia_bigramas_texto_1[:10])
```

```
[('moby', 'dick'), ('dick', 'herman'), ('herman', 'melville'), ('melville', 'etymology'), ('ety
[(('sperm', 'whale'), 182), (('white', 'whale'), 106), (('moby', 'dick'), 84), (('old', 'man')
```

```
bigramas_texto_2 = obtener_bigramas(texto_2)
print(bigramas_texto_2[:10])

frecuencia_bigramas_texto_2 = contar_bigramas(bigramas_texto_2)
print(frecuencia_bigramas_texto_2[:10])
```

```
[('sense', 'sensibility'), ('sensibility', 'jane'), ('jane', 'austen'), ('austen', 'chapter'),
[(('mrs', 'jennings'), 230), (('colonel', 'brandon'), 132), (('mrs', 'dashwood'), 121), (('sir
```

```
bigramas_texto_3 = obtener_bigramas(texto_3)
print(bigramas_texto_3[:10])

frecuencia_bigramas_texto_3 = contar_bigramas(bigramas_texto_3)
print(frecuencia_bigramas_texto_3[:10])
```

```
[('beginning', 'god'), ('god', 'created'), ('created', 'heaven'), ('heaven', 'earth'), ('earth
[(('said', 'unto'), 180), (('came', 'pass'), 63), (('thou', 'shalt'), 50), (('thou', 'hast'), 4
```

```
# Convertir palabras (tokens) en ngramas de caracteres
def ngramas_caracteres(texto, ngramas):
  caracter_ngramas = [''.join(caracteres) for palabra in texto
    for caracteres in zip(*[palabra[i:] for i in range(ngramas)])]
  return caracter_ngramas


def frecuencia_ngramas_caracteres(ngramas_caracteres):
  frecuencia_ngramas = FreqDist(ngramas_caracteres)
  return frecuencia_ngramas
```

```
ngramas_caracteres_texto_1 = ngramas_caracteres(texto_1, 3)
print(ngramas_caracteres_texto_1[:10])

frecuencia_ngramas_caracteres_texto_1 = frecuencia_ngramas_caracteres(ngramas_caracteres_texto_1)
print(frecuencia_ngramas_caracteres_texto_1.most_common(10))
```

```
['mob', 'oby', 'dic', 'ick', 'her', 'erm', 'rma', 'man', 'mel', 'elv']
[('ing', 7733), ('ter', 2292), ('ent', 2193), ('hal', 2135), ('wha', 1914), ('ale', 1903), ('or
```

```
ngramas_caracteres_texto_2 = ngramas_caracteres(texto_2, 3)
print(ngramas_caracteres_texto_2[:10])

frecuencia_ngramas_caracteres_texto_2 = frecuencia_ngramas_caracteres(ngramas_caracteres_texto_2)
print(frecuencia_ngramas_caracteres_texto_2.most_common(10))
```

```
['sen', 'ens', 'nse', 'sen', 'ens', 'nsi', 'sib', 'ibi', 'bil', 'ili']
[('ing', 3355), ('ion', 1794), ('ent', 1726), ('eve', 1464), ('tio', 1418), ('ter', 1304), ('ve
```

```
ngramas_caracteres_texto_3 = ngramas_caracteres(texto_3, 3)
print(ngramas_caracteres_texto_3[:10])

frecuencia_ngramas_caracteres_texto_3 = frecuencia_ngramas_caracteres(ngramas_caracteres_texto_3)
print(frecuencia_ngramas_caracteres_texto_3.most_common(10))
```

```
['beg', 'egi', 'gin', 'inn', 'nni', 'nin', 'ing', 'god', 'cre', 'rea']
[('the', 795), ('unt', 653), ('nto', 598), ('her', 590), ('aid', 533), ('sai', 488), ('ing', 48
```

```
print(f'Longitud del texto 1: {len(texto_1)}')
print(f'Longitud de bigramas del texto 1: {len(bigramas_texto_1)}')
print(f'Longitud de trigramas de caracteres del texto 1: {len(ngramas_caracteres_texto_1)}')
```

```
Longitud del texto 1: 110482
Longitud de bigramas del texto 1: 110481
Longitud de trigramas de caracteres del texto 1: 425131
```

```
print(f'Longitud del texto 2: {len(texto_2)}')
print(f'Longitud de bigramas del texto 2: {len(bigramas_texto_2)}')
print(f'Longitud de trigramas de caracteres del texto 2: {len(ngramas_caracteres_texto_2)}')
```

```
Longitud del texto 2: 53987
Longitud de bigramas del texto 2: 53986
Longitud de trigramas de caracteres del texto 2: 228150
```

```
print(f'Longitud del texto 3: {len(texto_3)}')
print(f'Longitud de bigramas del texto 3: {len(bigramas_texto_3)}')
print(f'Longitud de trigramas de caracteres del texto 3: {len(ngramas_caracteres_texto_3)}')
```

```
Longitud del texto 3: 18335
Longitud de bigramas del texto 3: 18334
Longitud de trigramas de caracteres del texto 3: 56086
```

Similitud coseno

```
def crear_tabla_basica(nombre_texto, bigramas, trigramas):
  df = pd.DataFrame()
  bigramas_trigramas = bigramas + trigramas
  df['texto'] = [nombre_texto for i in bigramas_trigramas]
  df['bigramas y trigramas'] = [ngrama for ngrama in bigramas_trigramas]
  return df
```

```
tabla_1 = crear_tabla_basica('texto_1', bigramas_texto_1,
                             ngramas_caracteres_texto_1)
tabla_1.head(10)
```

	texto	bigramas y trigramas
0	texto_1	(moby, dick)
1	texto_1	(dick, herman)
2	texto_1	(herman, melville)
3	texto_1	(melville, etymology)
4	texto_1	(etymology, supplied)
5	texto_1	(supplied, late)
6	texto_1	(late, consumptive)
7	texto_1	(consumptive, usher)
8	texto_1	(usher, grammar)
9	texto_1	(grammar, school)

```
tabla_2 = crear_tabla_basica('texto_2', bigramas_texto_2,
                             ngramas_caracteres_texto_2)
tabla_2.head(10)
```

	texto	bigramas y trigramas
0	texto_2	(sense, sensibility)
1	texto_2	(sensibility, jane)
2	texto_2	(jane, austen)
3	texto_2	(austen, chapter)

```
tabla_3 = crear_tabla_basica('texto_3', bigramas_texto_3,
                             ngramas_caracteres_texto_3)
tabla_3.head(10)
```

	texto	bigramas y trigramas
0	texto_3	(beginning, god)
1	texto_3	(god, created)
2	texto_3	(created, heaven)
3	texto_3	(heaven, earth)
4	texto_3	(earth, earth)
5	texto_3	(earth, without)
6	texto_3	(without, form)
7	texto_3	(form, void)
8	texto_3	(void, darkness)
9	texto_3	(darkness, upon)

```
tabla_final = pd.concat([tabla_1, tabla_2, tabla_3], ignore_index=True)
tabla_final.tail(10)
```

	texto	bigramas y trigramas
892158	texto_3	lme
892159	texto_3	med
892160	texto_3	put
892161	texto_3	cof
892162	texto_3	off
892163	texto_3	ffi
892164	texto_3	fin
892165	texto_3	egy
892166	texto_3	gyp
892167	texto_3	ypt

```
tabla_pivoteada = tabla_final.groupby(['texto', 'bigramas y trigramas'])['bigramas y trigramas']\
                  .agg(['count']).reset_index()\
                  .pivot(index='bigramas y trigramas', columns='texto', values='count')

```

```
tabla_pivoteada.columns.name = None
tabla_pivoteada = tabla_pivoteada.fillna(0)
tabla_pivoteada[153477:153497]
```

bigramas y trigramas	texto_1	texto_2	texto_3
(zone, zone)	1.0	0.0	0.0
(zoned, letters)	1.0	0.0	0.0
(zoned, quest)	1.0	0.0	0.0
(zones, accounts)	1.0	0.0	0.0
(zones, surface)	1.0	0.0	0.0
(zones, world)	1.0	0.0	0.0
(zoology, anatomy)	1.0	0.0	0.0
(zoology, much)	1.0	0.0	0.0
(zoroaster, died)	1.0	0.0	0.0
(zuzims, ham)	0.0	0.0	1.0
	8.0	2.0	0.0
l	0.0	1.0	0.0
s	2.0	0.0	0.0
t	8.0	0.0	0.0
nd	1.0	0.0	0.0
st	4.0	0.0	0.0
th	10.0	0.0	0.0
___	22.0	0.0	0.0
aac	1.0	0.0	78.0
aah	0.0	0.0	1.0

```
from scipy.spatial.distance import cosine

def similitud_coseno(a,b):
  distancia = cosine(a,b)
  return 1-distancia

tabla_pivoteada.corr(method=similitud_coseno)
```

	texto_1	texto_2	texto_3
texto_1	1.000000	0.856819	0.644977
texto_2	0.856819	1.000000	0.600518
texto_3	0.644977	0.600518	1.000000

```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
```

```
from colab_pdf import colab_pdf
colab_pdf('Tarea 10 - NLP.ipynb')
```

```
File ‘colab_pdf.py’ already there; not retrieving.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

[NbConvertApp] Converting notebook /content/drive/My Drive/Colab Notebooks/Tarea 10 - NLP.ipynb
[NbConvertApp] Writing 50717 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: [u'xelatex', u'./notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: [u'bibtex', u'./notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 43911 bytes to /content/drive/My Drive/Tarea 10 - NLP.pdf
'File ready to be Downloaded and Saved to Drive'
```

```
1
```