Medidas_de_informacion_Emmanuel_Santos_Rodriguez

August 25, 2023

1 Introducción

Actualmente contamos con una cantidad ingente de información, de la cual gran parte se encuentra en forma de texto. Para las personas es fácil leerla y encontrar relaciones entre las palabras que conforman un texto, sin embargo para las computadoras es una tarea compleja. Debido a esto surgen áreas de las ciencias de la computación que se encargan de la extracción de información en texto mediante diversas técnicas. En este sentido, la información mutua es una técnica se puede aplicar en el descubrimiento de las asociaciones de palabras.

El propósito de este documento es realizar el descubrimiento de asociaciones de palabras para los conjuntos de datos proporcionados utilizando la medida de información mutua.

2 Desarrollo

Para la solución de este problema se utilizó Python.

[1]:
```
!pip install unidecode
```

```
Requirement already satisfied: unidecode in /usr/local/lib/python3.10/dist-
packages (1.3.6)
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[3]:
```
from nltk.corpus import stopwords
import pandas as pd
from nltk.tokenize import word_tokenize
import nltk
import string
import unidecode
import re
import itertools
from nltk.util import ngrams
import math
```

[4]:
```
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[4]:
```
True
```

[5]:
```
es_stop_words = set(stopwords.words('spanish'))
```

2.0.1 Descripción de los datos

Para realizar esta tarea se tienen 3 conjuntos de datos que están contienen tweets de México, España y Venezuela del mes de marzo de 2020, con 1000000, 1000000 y 300000 filas respectivamente. Cada conjunto está formado por 3 columnas: created_at, id y text. En este caso, la columna que nos interesa es text.

[6]:
```
mx_tweets = pd.read_json("/content/drive/MyDrive/ProcesamientoInformacion/MX_1M.
 ↪json", encoding="utf-8", lines = True)
es_tweets = pd.read_json("/content/drive/MyDrive/ProcesamientoInformacion/ES_1M.
 ↪json", encoding="utf-8", lines = True)
vz_tweets = pd.read_json("/content/drive/MyDrive/ProcesamientoInformacion/
 ↪VE_300K.json", encoding="utf-8", lines = True)
```

[7]:
```
mx_tweets.head(10)
```

[7]:

	created_at	id	text
0	2020-03-08 04:54:06+00:00	1236515511900360704	@Reporte_Indigo Ups!!! A ver que dicen
1	2020-03-21 19:51:35+00:00	1241452415754223618	@Zuanna_Sol Si mi amor dame a tomar tus jugos …
2	2020-03-31 22:58:08+00:00	1245123242651570176	@Oscar62776751 Jajaja que muevas rapido perra!…
3	2020-03-19 15:11:43+00:00	1240657208561545218	Lic. @navejadaltonico en Plaza de la Tecnol…
4	2020-03-27 16:59:02+00:00	1243583318479626240	Estoy bien tentada ahhhhh ayudaaaaAAAAAA.
5	2020-03-26 22:21:42+00:00	1243302133245440000	._. Es en serio?\n\nMe identifico con el gusan…
6	2020-03-27 23:56:16+00:00	1243688319730569223	@AdultModels2018 Wuuuaaaooooooo Asus pies q …
7	2020-03-18 01:08:00+00:00	1240082491056283649	
8	2020-03-08 02:47:45+00:00	1236483717641777155	
9	2020-03-27 15:23:44+00:00	1243559334686179330	

```
7  Te pasamos esta información sobre el #Coronavi…
8  Cuando veo a alguien feliz, mi corazón se pone…
9  @FondesoCDMX buen dia, quiero saber si es posi…
```

En la celda anterior podemos ver las primeras 10 filas del conjunto de datos de tweets de México.

2.0.2 Implementación

A continuación se describen las funciones usadas y se proporcionan ejemplos de su funcionamiento

[8]:
```
def preprocess(text):
    no_urls = re.sub(r"http\S+", "", text)
    #remover @usuarios y #hashtags
    without_users = re.sub('@[\w]+','',no_urls)
    without_users_and_hashtags = re.sub('#[\w]+','',without_users)
    #remover acentos
    unaccented_string = unidecode.unidecode(without_users_and_hashtags)
    #convertir a minuscula y remover signos de puntuacion
    text_lower = unaccented_string.lower().translate(str.maketrans('', '',␣
 ↪string.punctuation))
    #limitar los characteres repetidos consecutivos
    #no_dups_text = limit_dups(text_lower)
    #obtenemos los tokens
    text_tokens = word_tokenize(text_lower)
    #filtramos las stopwords de los tokens obtenidos
    tokens_without_sw = [word for word in text_tokens if not word in␣
 ↪es_stop_words]
    tokens_without_dups = [limit_dups(token) for token in tokens_without_sw]
    return tokens_without_dups

def limit_dups(s, limit=2):
    max_vs = (''.join(itertools.islice(g, limit)) for k, g in itertools.
 ↪groupby(s))
    components = ([s[:l + 1] for l in range(len(s))] for s in max_vs)
    return [''.join(letters) for letters in itertools.product(*components)][-1]
```

La función preprocess recibe una cadena de texto y se encarga de realizar el preprocesamiento en el siguiente orden: remover urls, usuarios y hashtags, después se quitan los acentos, se convierte el texto a minúsculas y se remueven los signos de puntuación, se convierte el texto a tokens para posteriormente removerle las stop-words y finalmente se limitan los carácteres redundantes.

En la siguiente celda veremos el resultado de la función de preprocesamiento aplicada al texto: "Felizzzzzzzz jueves!!! en Torre Reforma https://t.co/EV46yOUOEx"

[9]:
```
preprocess("Felizzzzzzzz jueves!!! en Torre Reforma https://t.co/EV46yOUOEx")
```

[9]:
```
['felizz', 'jueves', 'torre', 'reforma']
```

[10]:
```
def fill_dict(text_array):
    my_dictionary = {}
    for word in text_array:
        for item in word:
            if item in my_dictionary:
                my_dictionary[item] = my_dictionary[item] + 1
            else:
                my_dictionary[item] = 1
    return my_dictionary

filter_dict = lambda my_dict, limit = 5: {k: v for k, v in my_dict.items() if v␣
 ↪> limit}

def calculate_mutual_information(data_dict, data_bigram_dict, limit = 50):
  mx_mutual_information = {}
  for item in data_bigram_dict:
      mx_mutual_information[item] = mutual_information(item, item[0], item[1],␣
 ↪data_bigram_dict, data_dict)
  return sorted(mx_mutual_information.items(), key=lambda x:x[1],␣
 ↪reverse=True)[:limit]
```

La función fill_dict crea un diccionario a partir de un arreglo de strings usando cada palabra como llave. Si la llave ya está, se aumenta su cuenta, sino se le asigna un 1. Esta función se usa para contar la frecuencia de los tokens.

La función filter_dict nos permite filtrar un diccionario por valor. Esta función se usa para filtrar los tokens que tengan una frecuencia menor a cierto valor. En este caso, 5 es un valor predeterminado.

La función calculate_mutual_information obtiene las medidas de información mutua para todos los bigramas de un conjunto. Devuelve, de forma predeterminada, los 50 primeros ordenados según su índice de información mutua de mayor a menor.

[11]:
```
def mutual_information(xy, x, y, data1, data2):
    pxy = data1[xy]/len(data2)
    px = data2[x]/len(data2)
    py = data2[y]/len(data2)
    return math.log2(pxy/((px)*(py)))
```

La función mutual_information nos permite calcular la información mutua de acuerdo a la siguiente fórmula: $I(x_i; y_i) = \log_2 \frac{p(x_i, y_i)}{p(x_i)p(y_i)}$, donde $p(x_i)$ y $p(y_i)$ se obtienen de las frecuencias relativas de los datos

[12]:
```
mx_tweets['processed_text']= mx_tweets['text'].map(preprocess)
es_tweets['processed_text']= es_tweets['text'].map(preprocess)
vz_tweets['processed_text']= vz_tweets['text'].map(preprocess)
```

2.0.3 Resultados

Primero aplicamos el preprocesamiento al campo text de cada uno de los conjuntos y lo asignamos a una nueva columna llamada processed_text

[13]:
```
mx_tweets["bigrams"] = mx_tweets["processed_text"].apply(lambda x:␣
 ↪list(ngrams(x, 2)))
es_tweets["bigrams"] = es_tweets["processed_text"].apply(lambda x:␣
 ↪list(ngrams(x, 2)))
vz_tweets["bigrams"] = vz_tweets["processed_text"].apply(lambda x:␣
 ↪list(ngrams(x, 2)))
```

Luego obtenemos los bigramas a partir del texto procesado

[14]:
```
mx_dictionary = fill_dict(mx_tweets['processed_text'])
es_dictionary = fill_dict(es_tweets['processed_text'])
vz_dictionary = fill_dict(vz_tweets['processed_text'])
```

[15]:
```
mx_bigram_dictionary = fill_dict(mx_tweets['bigrams'])
es_bigram_dictionary = fill_dict(es_tweets['bigrams'])
vz_bigram_dictionary = fill_dict(vz_tweets['bigrams'])
```

Después se llenan los diccionarios con las frecuencias de las palabras y bigramas

[16]:
```
mx_dictionary_freq_filtered = filter_dict(mx_dictionary)
es_dictionary_freq_filtered = filter_dict(es_dictionary)
vz_dictionary_freq_filtered = filter_dict(vz_dictionary)
```

[17]:
```
mx_bigrams_dictionary_freq_filtered = filter_dict(mx_bigram_dictionary)
es_bigrams_dictionary_freq_filtered = filter_dict(es_bigram_dictionary)
vz_bigrams_dictionary_freq_filtered = filter_dict(vz_bigram_dictionary)
```

Posteriormente se filtran los diccionarios para quedarnos solo con aquellos que tengan una frecuencia mayor a 5

Finalmente, llamamos a la función que calcula la información mutua para los datos de cada conjunto y obtenemos una lista de tuplas de las 50 asociaciones más importantes junto a su índice de información mutua.

[18]:
```
mx_mutual_information =␣
 ↪calculate_mutual_information(mx_dictionary_freq_filtered,␣
 ↪mx_bigrams_dictionary_freq_filtered)
mx_mutual_information
```

[18]:
```
[(('borquez', 'schwarzbeck'), 12.935533412950747),
 (('nisidominus', 'aksenti'), 12.935533412950747),
 (('duane', 'cochran'), 12.935533412950747),
 (('57819356', '52336966'), 12.935533412950747),
 (('523328144198', '584145535261'), 12.7131409916143),
 (('cambiario', 'rehobot'), 12.7131409916143),
 (('yves', 'rocher'), 12.7131409916143),
 (('kendo', 'iaido'), 12.7131409916143),
 (('5511', '8383'), 12.7131409916143),
 (('geiq', 'aviation'), 12.7131409916143),
```

```
(('charlize', 'theron'), 12.7131409916143),
(('masayoshi', 'ohira'), 12.7131409916143),
(('xdia', '5habitaciones'), 12.7131409916143),
(('lindsay', 'lohan'), 12.7131409916143),
(('eyed', 'peas'), 12.520495913671903),
(('4885', '0736'), 12.520495913671903),
(('bolora', 'bolora'), 12.490748570277852),
(('crecencio', 'rejon'), 12.35057091222959),
(('prismas', 'basalticos'), 12.35057091222959),
(('jicotes', 'gotcha'), 12.35057091222959),
(('festivaladverso', 'ackpromote'), 12.35057091222959),
(('avelina', 'lesper'), 12.298103492335455),
(('krav', 'maga'), 12.19856781878454),
(('dimes', 'diretes'), 12.19856781878454),
(('snoop', 'dogg'), 12.19856781878454),
(('weriink', 'tatoos'), 12.19856781878454),
(('9811547314', 'kathy'), 12.19856781878454),
(('hakuna', 'matata'), 12.19856781878454),
(('viktor', 'frankl'), 12.19856781878454),
(('brock', 'lesnar'), 12.157925834287195),
(('representaciones', 'albis'), 12.128178490893143),
(('argot', 'beisbolero'), 12.105458414393059),
(('iaido', 'jodo'), 12.061064295034605),
(('leonora', 'carrington'), 12.061064295034605),
(('barniz', 'policromado'), 11.976175397448092),
(('kenneth', 'wapnick'), 11.935533412950747),
(('krispy', 'kreme'), 11.935533412950747),
(('bampb', 'photography'), 11.935533412950747),
(('parleys', 'doki'), 11.92356077128467),
(('peaky', 'blinders'), 11.891139293592293),
(('bout', 'aztlan'), 11.835997739399831),
(('kimetsu', 'yaiba'), 11.82005619553081),
(('perfumeria', 'martel'), 11.82005619553081),
(('aldous', 'huxley'), 11.82005619553081),
(('chundo', 'maleado'), 11.810002530866887),
(('moros', 'tranchete'), 11.765608411508435),
(('nahim', 'lujan'), 11.7131409916143),
(('waths', '5566898049'), 11.7131409916143),
(('molleras', 'sumidas'), 11.7131409916143),
(('wt', '5525227606'), 11.7131409916143)]
```

La lista anterior es la lista de las 50 asociaciones más frecuentes para el conjunto de datos de México. Podemos hallar asociaciones como Duane Cochran, prismas basálticos, Lindsay Lohan o Hakuna Matata, etc.

[19]:

```
es_mutual_information = calculate_mutual_information(es_dictionary_freq_filtered, es_bigrams_dictionary_freq_filtered)
es_mutual_information
```

[19]:
```
[(('17500km', 'aproximados'), 13.005098178239706),
 (('villajos', 'tolin'), 13.005098178239706),
 (('hakuna', 'matata'), 13.005098178239706),
 (('cartulina15x17', 'cm2018'), 12.782705756903258),
 (('bed', 'townhouse'), 12.782705756903258),
 (('lindsay', 'lohan'), 12.782705756903258),
 (('635484716', '980532590'), 12.782705756903258),
 (('c11ii', 'quadrotor'), 12.782705756903258),
 (('appena', 'pubblicata'), 12.590060678960862),
 (('signal', 'iduna'), 12.590060678960862),
 (('emakumeen', 'eguneko'), 12.590060678960862),
 (('frany', 'dejota'), 12.590060678960862),
 (('grandesseremos', 'inmateriales'), 12.590060678960862),
 (('bioeasy', 'biotechnology'), 12.590060678960862),
 (('childish', 'gambino'), 12.590060678960862),
 (('catastroficas', 'desdichas'), 12.590060678960862),
 (('hyaluron', 'pen250eur'), 12.56031333556681),
 (('34952321277', 'mobile34632167403'), 12.42013567751855),
 (('videntes', '981070108806499756'), 12.42013567751855),
 (('ludovico', 'einaudi'), 12.42013567751855),
 (('lolits', 'diary'), 12.42013567751855),
 (('f12', 'berlinetta'), 12.42013567751855),
 (('berlinetta', 'pertenecio'), 12.42013567751855),
 (('7977', '7698'), 12.42013567751855),
 (('50th', 'anniversary'), 12.42013567751855),
 (('donovan', 'mitchell'), 12.397415601018466),
 (('gravity', 'falls'), 12.397415601018466),
 (('sustrato', 'preexistente'), 12.397415601018466),
 (('avril', 'lavigne'), 12.367668257624414),
 (('alitea', 'artemania'), 12.2681325840735),
 (('facciamo', 'finta'), 12.2681325840735),
 (('fbc', 'fortuny'), 12.2681325840735),
 (('bong', 'joonho'), 12.2681325840735),
 (('germans', 'trias'), 12.2681325840735),
 (('solarium', 'sunlife'), 12.197743256182102),
 (('anosmia', 'ageusia'), 12.197743256182102),
 (('citric', 'madness'), 12.197743256182102),
 (('rarezas', 'meteorologicas'), 12.197743256182102),
 (('elon', 'musk'), 12.197743256182102),
 (('correduria', 'segurosslu'), 12.130629060323566),
 (('hugh', 'jackman'), 12.130629060323566),
 (('004', '7977'), 12.130629060323566),
```

```
 (('javielico', 'pulpi'), 12.130629060323566),
 (('donaras', '1eurmes'), 12.130629060323566),
 (('j16', 'lasesarre'), 12.11612949062845),
 (('plores', 'plores'), 12.11612949062845),
 (('ortiga', 'smint'), 12.045740162737053),
 (('nicki', 'minaj'), 12.045740162737053),
 (('webco', 'mediadores'), 12.005098178239706),
 (('greys', 'anatomy'), 12.005098178239706)]
```

La lista anterior es la lista de las 50 asociaciones más frecuentes para el conjunto de datos de España. Podemos encontrar asociaciones como Hugh Jackman, Greys Anatomy, Childish Gambino, etc.

[20]:
```
vz_mutual_information =␣
 ↪calculate_mutual_information(vz_dictionary_freq_filtered,␣
 ↪vz_bigrams_dictionary_freq_filtered)
vz_mutual_information
```

[20]:
```
[(('truene', 'relampaguee'), 12.199416742716586),
 (('details', 'huntinggt'), 12.199416742716586),
 (('960', 'pixels'), 12.199416742716586),
 (('marcadahiatsu', 'modeloterios'), 12.199416742716586),
 (('bohemian', 'rhapsody'), 12.199416742716586),
 (('cgabriel', 'lanusse'), 11.977024321380139),
 (('produjera', 'arrollam'), 11.977024321380139),
 (('lenny', 'tavarez'), 11.977024321380139),
 (('escuacada', 'putrfacta'), 11.977024321380139),
 (('pedaling', 'paddling'), 11.977024321380139),
 (('licorestamos', 'pantanos'), 11.977024321380139),
 (('peaky', 'blinders'), 11.784379243437742),
 (('coronil', 'hartmann'), 11.784379243437742),
 (('stylo', 'xauxas'), 11.784379243437742),
 (('efectivosin', 'gasolinasin'), 11.784379243437742),
 (('0414420059', '02416181097'), 11.754631900043691),
 (('oponerme', 'boicotear'), 11.61445424199543),
 (('oraren', 'buscaren'), 11.61445424199543),
 (('viktor', 'frankl'), 11.561986822101295),
 (('plot', 'twist'), 11.561986822101295),
 (('feng', 'shui'), 11.46245114855038),
 (('carmina', 'burana'), 11.46245114855038),
 (('bombos', 'platillos'), 11.46245114855038),
 (('audiencias', 'diferida'), 11.46245114855038),
 (('stephany', 'molero'), 11.46245114855038),
 (('cimarron', 'erlin'), 11.421809164053034),
 (('alcasa', 'agroislena'), 11.392061820658983),
 (('recibirlas', 'callesel'), 11.324947624800446),
 (('pixels', 'procesador'), 11.324947624800446),
 (('invocado', 'oraren'), 11.324947624800446),
 (('mare', 'nostrum'), 11.324947624800446),
```

```
 (('hong', 'kong'), 11.292526147108068),
 (('opereta', 'bufa'), 11.251884162610722),
 (('bin', 'laden'), 11.240058727213933),
 (('tang', 'villanueva'), 11.240058727213933),
 (('modus', 'operandi'), 11.199416742716586),
 (('craig', 'faller'), 11.199416742716586),
 (('vivito', 'coleando'), 11.199416742716586),
 (('raimundo', 'andueza'), 11.155022623358134),
 (('gasolinasin', 'seguri'), 11.13230254685805),
 (('muay', 'thai'), 11.083939525296651),
 (('perolito', 'escarlata'), 11.047413649271537),
 (('kylie', 'jenner'), 11.047413649271537),
 (('design', 'thinking'), 11.047413649271537),
 (('johantattooart', 'johantattooart'), 11.029491741274274),
 (('kenny', 'rogers'), 11.003019529913082),
 (('5tas', 'columnas'), 10.977024321380139),
 (('cristianas', 'evangel'), 10.977024321380139),
 (('riko', 'malt'), 10.968462307876715),
 (('hipertensa', 'diabetica'), 10.909910125521602)]
```

La lista anterior es la lista de las 50 asociaciones más frecuentes para el conjunto de datos de Venezuela. Podemos hallar asociaciones como Peaky Blinders, hipertensa diabética, vivito coleando, feng shui, etc.

Como último experimento, filtramos los tokens y bigramas que tengan una frecuencia mayor a 100 y nuevamente calculamos el índice de información mutua

[23]:
```
mx_dictionary_freq_filtered_100 = filter_dict(mx_dictionary, 100)
mx_bigrams_dictionary_freq_filtered_100 = filter_dict(mx_bigram_dictionary, 100)
mx_mutual_information_100 =␣
 ↪calculate_mutual_information(mx_dictionary_freq_filtered_100,␣
 ↪mx_bigrams_dictionary_freq_filtered_100)
mx_mutual_information_100
```

[23]:
```
[(('alteza', 'serenisima'), 5.712765071111492),
 (('dua', 'lipa'), 5.2972426160175035),
 (('harry', 'potter'), 5.0483753752957545),
 (('venustiano', 'carranza'), 4.919961813199911),
 (('siglo', 'xxi'), 4.528536471861586),
 (('barbara', 'regil'), 4.410322280719463),
 (('neumonia', 'atipica'), 4.351043469524749),
 (('lazaro', 'cardenas'), 4.1332707830298485),
 (('echa', 'vistazo'), 4.096595784803501),
 (('bla', 'bla'), 4.036766741036545),
 (('tuxtla', 'gutierrez'), 3.950350304719116),
 (('alvaro', 'obregon'), 3.917340132553805),
 (('carnita', 'asada'), 3.833260363298052),
 (('salinas', 'pliego'), 3.764353689557467),
```

```
(('seguirme', 'tuiteros'), 3.743980786058472),
(('anillo', 'periferico'), 3.7242192239881993),
(('milagro', 'celda'), 3.6824102190301287),
(('uber', 'eats'), 3.6624333955613477),
(('pronta', 'recuperacion'), 3.5373736745009086),
(('locura', 'religiosa'), 3.4297931026756516),
(('cubre', 'bocas'), 3.3767890369332823),
(('recuerdas', 'uniste'), 3.358145230180255),
(('j', 'balvin'), 3.295702501946557),
(('fake', 'news'), 3.2849908312975433),
(('ignacio', 'zaragoza'), 3.201708642460351),
(('firma', 'peticion'), 3.1353219164040764),
(('nicolas', 'garza'), 3.06846790648225),
(('tik', 'tok'), 2.953320564240348),
(('rueda', 'prensa'), 2.9107254549194406),
(('quintana', 'roo'), 2.8597984442702953),
(('circuito', 'interior'), 2.854665855187755),
(('juegos', 'olimpicos'), 2.8178318506356352),
(('bad', 'bunny'), 2.7645651556123307),
(('tik', 'toks'), 2.689439115791639),
(('seres', 'queridos'), 2.6685856225429423),
(('barbacoa', 'res'), 2.5911215300444255),
(('eventos', 'masivos'), 2.5662665860847564),
(('puerto', 'vallarta'), 2.4935137964667917),
(('sistema', 'inmunologico'), 2.4632883125973293),
(('villahermosa', 'tabasco'), 2.4517526628629094),
(('adultos', 'mayores'), 2.419928465515994),
(('gel', 'antibacterial'), 2.399243413357013),
(('carne', 'asada'), 2.3279676969538725),
(('tren', 'maya'), 2.3129318153508587),
(('tuiteros', 'red'), 2.265933489253828),
(('luis', 'potosi'), 2.15607877426178),
(('home', 'office'), 2.1542311982539255),
(('distrito', 'federal'), 2.1333840878670665),
(('papel', 'higienico'), 2.1246222312885603),
(('pizza', 'pina'), 2.060713346967012)]
```

Es interesante ver que hay asociaciones como ('neumonia', 'atipica'), ('pronta', 'recuperacion'), ('cubre', 'bocas'), ('gel', 'antibacterial'). Cabe destacar que en marzo de 2020 se declaró que (COVID-19) se podía caracterizar como una pandemia.

3 Conclusiones

La medida de información mutua es muy útil para hallar asociaciones entre palabras en un texto. Sin embargo, su cálculo y las asociaciones que encontremos estarán directamente relacionadas con el preprocesamiento que le apliquemos al texto. Por otra parte, debe reconocerse su importancia es otras áreas como procesamiento de lenguaje natural o recuperación de información.