> Nicole Vera

```
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import  seaborn as sns
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

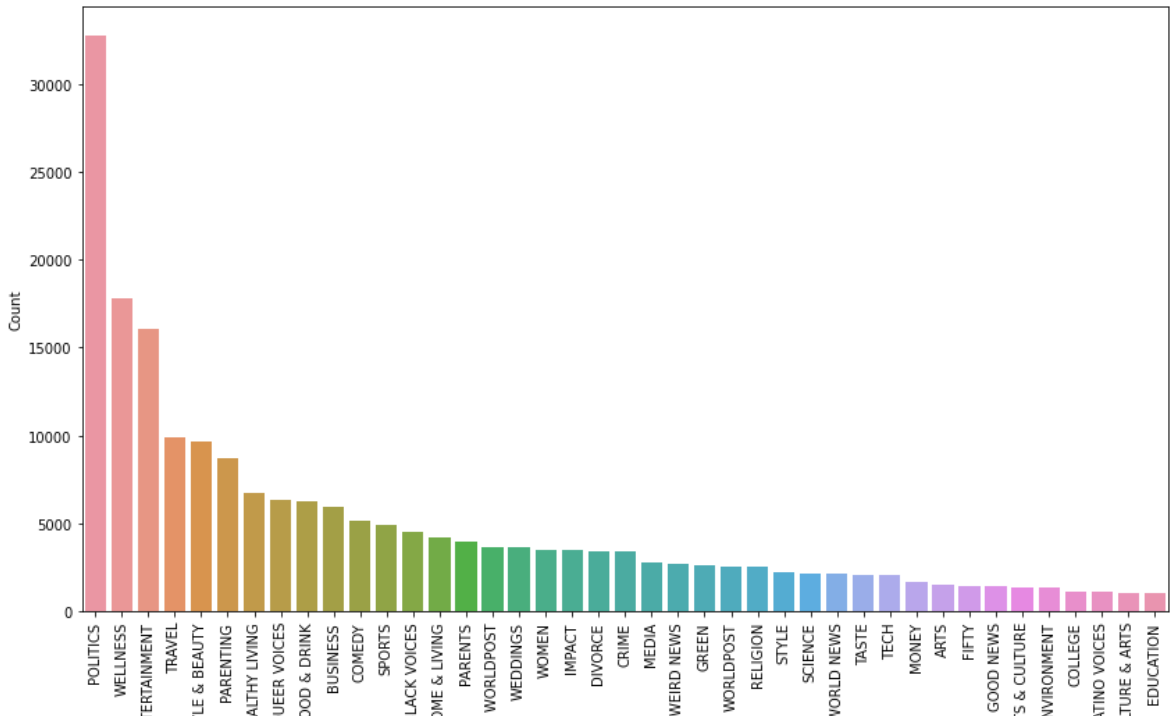
```
df = pd.read_json("/content/drive/MyDrive/datasets/data/News_Category_Dataset_v2.json",lines=True)

palabras_vacias = [linea.rstrip('\n') for linea in open('/content/drive/MyDrive/datasets/data/palabras_vacias.txt')]
len(palabras_vacias)
```

```
290
```

```
def remove_sw(t):
  tokens = t.strip().split(' ')
  tokens = [token for token in tokens if token not in palabras_vacias]
  tout = ' '.join(tokens).strip()
  return tout

def clean_text(t):
  t = str(t)
  t = re.sub(r'\d+',' ',t)
  t = re.sub(r'[^\w\n]',' ',t)
  t = re.sub(r'\s\s+',' ',t)
  t = t.lower()
  t = t.strip()
  return t

from wordcloud import WordCloud

def crear_nubes(col_cluster):
  gc = 2
  gr = int(nc/gc)
  wcwidth  = 1600
  wcheight = 800
  fig = plt.figure(figsize = (25,60))
  fig.suptitle('Análisis de clusters con {}'.format(col_cluster), fontsize=14)

  for i in range(nc):
    dfs = df[df[col_cluster] == i]['texto']
    texto = ' '.join(map(str,dfs.tolist()))
    wc = WordCloud(width=wcwidth, height=wcheight).generate(texto)
    ax = fig.add_subplot(gr,gc,i+1)
    ax.imshow(wc,interpolation='bilinear')
    ax.axis('off')
    ax.set_title('{} cluster # {}'.format(col_cluster,str(i+1)))
    ax.title.set_size(20)
  return fig
```

Analisís exploratorio

```
plt.figure(figsize=(14,8))
count = df.category.value_counts()
sns.barplot(x=count.index, y=count)
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=90);
```



LDA

```
from sklearn.decomposition import LatentDirichletAllocation
import numpy as np
```

```
corpus = list(df['headline'])
print(len(corpus))
```

```
200853
```

```
def remove_sw(t):
  tokens = t.strip().split(' ')
  tokens = [token for token in tokens if token not in palabras_vacias]
  tout = ' '.join(tokens).strip()
  return tout
def clean_text(t):
  t = str(t)
  t = re.sub(r'\d+',' ',t)
  t = re.sub(r'[^\w\n]',' ',t)
  t = re.sub(r'\s\s+',' ',t)
  t = t.lower()
  t = t.strip()
  return t
from wordcloud import WordCloud
def crear_nubes(col_cluster):
  gc = 2
  gr = int(nc/gc)
  wcwidth  = 1600
  wcheight = 800
  fig = plt.figure(figsize = (25,60))
  fig.suptitle('Análisis de clusters con {}'.format(col_cluster), fontsize=14)
  for i in range(nc):
    dfs = df[df[col_cluster] == i]['headline']
    texto = ' '.join(map(str,dfs.tolist()))
    wc = WordCloud(width=wcwidth, height=wcheight).generate(texto)
    ax = fig.add_subplot(gr,gc,i+1)
    ax.imshow(wc,interpolation='bilinear')
    ax.axis('off')
    ax.set_title('{} cluster # {}'.format(col_cluster,str(i+1)))
    ax.title.set_size(20)
  return fig
```

```
corpus = [clean_text(t) for t in corpus]
corpus = [remove_sw(t) for t in corpus]
#corpus
```

```
vectorizer = TfidfVectorizer(stop_words='english',sublinear_tf=True)
X = vectorizer.fit_transform(corpus)
vocab = vectorizer.get_feature_names_out()
print(X.shape)
```

```
(200853, 54288)
```

```
nc = 20
df['texto'] = corpus
```

LDA

```
from sklearn.decomposition import LatentDirichletAllocation
import numpy as np
```

```
lda_model = LatentDirichletAllocation(n_components=nc,random_state=10)
lda = lda_model.fit_transform(X)
```

```
tp_palabra = {}
n_top_pal = 15
for topico, comp in enumerate(lda_model.components_):
  pal_ind = np.argsort(comp)[::-1][:n_top_pal]
  tp_palabra[topico] = [vocab[i] for i in pal_ind]
```

```
for topico,palabras in tp_palabra.items():
  print("Tópico {} : {}".format(topico,' ,'.join(palabras)))
```

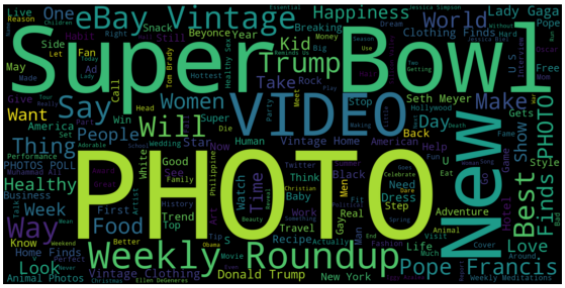
```
Tópico 0 : photos ,super ,bowl ,happiness ,pope ,vintage ,healthy ,hotels ,weekly ,lady ,francis ,finds ,ebay ,seth ,animal
Tópico 1 : health ,study ,cancer ,care ,sleep ,heart ,risk ,women ,change ,mental ,tweets ,brain ,kids ,parents ,week
Tópico 2 : trump ,meditation ,questions ,tax ,daily ,ask ,noah ,donald ,immigration ,myths ,trevor ,plan ,obama ,los ,letting
Tópico 3 : trump ,saudi ,killed ,dead ,video ,taught ,dreams ,car ,walking ,game ,season ,mcdonald ,yemen ,police ,revolution
Tópico 4 : north ,korea ,photos ,red ,carpet ,middle ,workout ,secrets ,east ,friends ,south ,kitchen ,carolina ,video ,mommy
Tópico 5 : trump ,donald ,clinton ,hillary ,gop ,paul ,democrats ,house ,russia ,president ,election ,ryan ,obama ,obamacare ,
Tópico 6 : huffpost ,need ,divorce ,rise ,married ,mistakes ,couples ,weddings ,wall ,financial ,working ,words ,world ,make ,
Tópico 7 : court ,trump ,supreme ,obama ,gop ,rights ,cruz ,marriage ,ted ,republican ,donald ,presidential ,rubio ,gay ,democ
Tópico 8 : photos ,fashion ,week ,kardashian ,year ,kim ,style ,video ,wedding ,jenner ,look ,photo ,justin ,wear ,fall
Tópico 9 : jimmy ,star ,black ,trends ,trump ,wars ,harvey ,kimmel ,songs ,fallon ,photos ,video ,weinstein ,grief ,hurricane
Tópico 10 : roundup ,taylor ,swift ,sunday ,matter ,news ,violence ,black ,week ,photos ,world ,domestic ,video ,watch ,youtub
Tópico 11 : morning ,email ,trump ,ferguson ,tuesday ,drag ,clinton ,travel ,monday ,silence ,donald ,zen ,october ,shades ,sc
Tópico 12 : day ,valentine ,recipes ,recipe ,father ,chocolate ,photos ,planned ,make ,parenthood ,video ,cheese ,bars ,lose ,
Tópico 13 : letter ,trump ,colbert ,stephen ,open ,mom ,golden ,single ,john ,bush ,mother ,donald ,jeb ,sean ,resolution
Tópico 14 : bernie ,sanders ,ve ,trump ,learned ,amy ,seen ,schumer ,traveling ,george ,heard ,day ,ebola ,planning ,zika
Tópico 15 : photos ,life ,ideas ,love ,ways ,kate ,learning ,talking ,home ,diy ,middleton ,make ,day ,easy ,video
Tópico 16 : holiday ,thanksgiving ,stress ,halloween ,day ,ways ,tips ,gift ,photos ,holidays ,free ,guide ,talk ,vacation ,mc
Tópico 17 : police ,road ,shooting ,trip ,man ,photos ,technology ,dead ,suspect ,year ,school ,video ,world ,shot ,officer
Tópico 18 : video ,places ,fearless ,baby ,photos ,names ,understand ,unexpected ,joe ,san ,biden ,francisco ,life ,lies ,norm
Tópico 19 : sexual ,assault ,wish ,miley ,photos ,cyrus ,facts ,trust ,harassment ,foods ,word ,girls ,independence ,selena ,t
```

```
topicos = []
for n in range(lda.shape[0]):
  topicos.append(lda[n].argmax())
df['lda'] = topicos
```

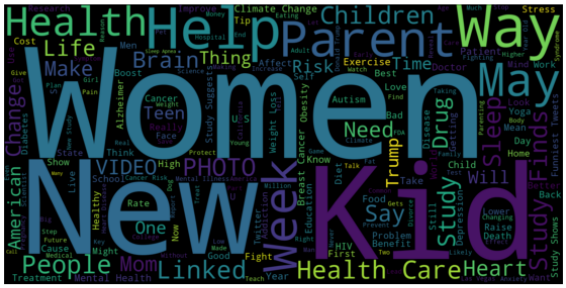
```
crear_nubes('lda')
```

Análisis de clusters con lda

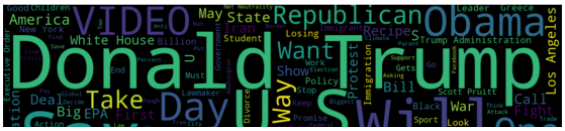
lda cluster # 1



lda cluster # 2



lda cluster # 3



lda cluster # 4

NMF

```
from sklearn.decomposition import NMF
```



```
nmf_model = NMF(n_components=nc,random_state=10,max_iter=400)
nmf = nmf_model.fit_transform(X)
```

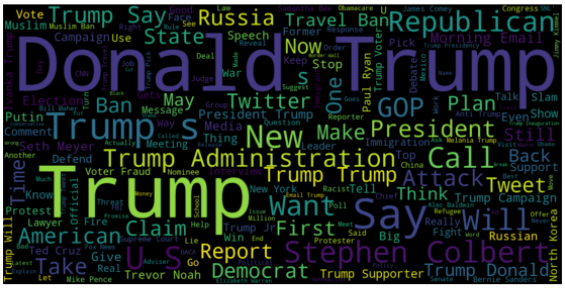
```
/usr/local/lib/python3.8/dist-packages/sklearn/decomposition/_nmf.py:289: FutureWarning: The 'init' value, when 'init=None' ar
  warnings.warn(
```

```
tp_palabras = {}
for topico, comp in enumerate(nmf_model.components_):
  pal_ind = np.argsort(comp)[::-1][:n_top_pal]
  tp_palabras[topico] = [vocab[i] for i in pal_ind]


topicos = []
for n in range(nmf.shape[0]):
  topicos.append(nmf[n].argmax())
df['nmf'] = topicos
```

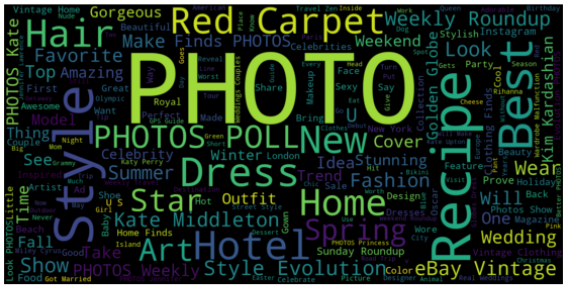
```
crear_nubes('nmf')
```

Análisis de clusters con nmf

nmf cluster # 1

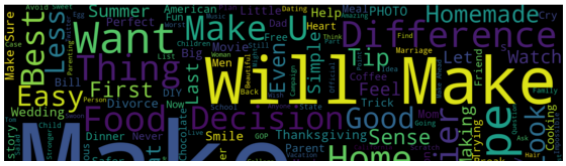


nmf cluster # 2



nmf cluster # 3

nmf cluster # 4




```
from sklearn.decomposition import LatentDirichletAllocation, NMF
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
def clean_text(t):
  t = str(t)
  t = re.sub(r'\d+',' ',t)
  t = re.sub(r'[^\w\n]',' ',t)
  t = re.sub(r'\s\s+',' ',t)
  t = t.lower()
  t = t.strip()
  return t
```

```
cv = CountVectorizer(stop_words='english')
tv = TfidfVectorizer(stop_words='english')
lda = LatentDirichletAllocation(n_components=10, random_state=42)
nmf = NMF(n_components=10, random_state=42)
```

```
lda_scores = []
nmf_scores = []
categories = df['category'].unique().tolist()
```

```
import numpy as np
from scipy import spatial
from sklearn.metrics import pairwise
```

```
cv = CountVectorizer(stop_words='english')
tv = TfidfVectorizer(stop_words='english')
lda = LatentDirichletAllocation(n_components=5)
nmf = NMF(n_components=5)
categories = df['category'].unique().tolist()
```

```
cv_vectors = cv.fit_transform(df['category'])
tv_vectors = tv.fit_transform(df['category'])
lda.fit(cv_vectors)
nmf.fit(tv_vectors)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/decomposition/_nmf.py:289: FutureWarning: The 'init' value, when 'init=None' ar
  warnings.warn(
NMF(n_components=5)
```

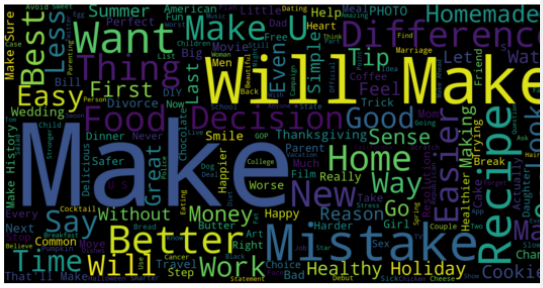
```
lda_scores = []
nmf_scores = []
for category in set(df['category']):
    df_categoria = df[df['category'] == category]
    lda_category_scores = []
    nmf_category_scores = []
    for i in range(len(df_categoria)):
        cv_vector = cv.transform([df_categoria.iloc[i]['headline']])
        tv_vector = tv.transform([df_categoria.iloc[i]['headline']])
        lda_similarity = np.max(cv_vector.dot(lda.components_.T))
        nmf_similarity = np.max(tv_vector.dot(nmf.components_.T))
        lda_category_scores.append(lda_similarity)
        nmf_category_scores.append(nmf_similarity)
    lda_scores.append((category, np.mean(lda_category_scores)))
    nmf_scores.append((category, np.mean(nmf_category_scores)))

for i in range(len(lda_scores)):
    category = lda_scores[i][0]
    lda_score = lda_scores[i][1]
    nmf_score = nmf_scores[i][1]
    print(f"Categoría: {category}")
    print(f"Puntaje promedio LDA: {lda_score:.2f}")
    print(f"Puntaje promedio NMF: {nmf_score:.2f}")
```

```
 Categoría: POLITICS
 Puntaje promedio LDA: 515.63
 Puntaje promedio NMF: 0.12
 Categoría: CULTURE & ARTS
 Puntaje promedio LDA: 638.56
 Puntaje promedio NMF: 0.10
 Categoría: WORLD NEWS
 Puntaje promedio LDA: 450.16
 Puntaje promedio NMF: 0.07
 Categoría: BLACK VOICES
 Puntaje promedio LDA: 1497.93
 Puntaje promedio NMF: 0.17
 Categoría: STYLE & BEAUTY
 Puntaje promedio LDA: 1797.14
 Puntaje promedio NMF: 0.98
 Categoría: WELLNESS
 Puntaje promedio LDA: 622.24
 Puntaje promedio NMF: 0.06
 Categoría: QUEER VOICES
 Puntaje promedio LDA: 953.02
 Puntaje promedio NMF: 0.10
 Categoría: TASTE
 Puntaje promedio LDA: 854.90
 Puntaje promedio NMF: 0.04
 Categoría: THE WORLDPOST
 Puntaje promedio LDA: 390.50
 Puntaje promedio NMF: 0.06
 Categoría: ENTERTAINMENT
 Puntaje promedio LDA: 368.70
 Puntaje promedio NMF: 0.06
 Categoría: FIFTY
 Puntaje promedio LDA: 548.84
 Puntaje promedio NMF: 0.09
 Categoría: ARTS
 Puntaje promedio LDA: 502.44
 Puntaje promedio NMF: 0.07
 Categoría: COMEDY
 Puntaje promedio LDA: 448.16
 Puntaje promedio NMF: 0.05
 Categoría: DIVORCE
 Puntaje promedio LDA: 1862.37
 Puntaje promedio NMF: 0.02
 Categoría: ENVIRONMENT
 Puntaje promedio LDA: 550.07
 Puntaje promedio NMF: 0.06
 Categoría: ARTS & CULTURE
 Puntaje promedio LDA: 757.66
 Puntaje promedio NMF: 0.15
 Categoría: PARENTS
 Puntaje promedio LDA: 1041.54
 Puntaje promedio NMF: 0.05
 Categoría: TRAVEL
 Puntaje promedio LDA: 1402.09
 Puntaje promedio NMF: 0.96
```

```
Categoría: HEALTHY LIVING
Puntaje promedio LDA: 563.84
Puntaje promedio NMF: 0.06
Categoría: WEIRD NEWS
```

nmf cluster # 3

nmf cluster # 4



nmf cluster # 5

nmf cluster # 6