Extracción y análisis de tweets

- Servicio de Microblogging
- Mensajes cortos llamados tweets (máximo de 140 caracteres)
- Cualquiera nos puede seguir y podemos seguir a cualquiera
- Mantener Conversaciones públicas
- Enterados de tópicos de conversación

donald trump

Top | Latest | People | Photos | Videos

John 'Murder Hornet' Cardillo @johncardillo · 23m

Judge Sullivan is sending a very loud and clear message on behalf of the Deep State,

"If you help **@realDonaldTrump** get re-elected and/or work in his admin, we'll never stop coming and you'll never get a fair hearing in a DC federal courtroom."

91 | 728 | 1.3K

People

Donald J. Trump
@realDonaldTrump
45th President of the United States of America🇺🇸

Follow

Trump War Room - Text TRUMP to 88022 & get the APP
@TrumpWarRoom
Highlighting @realDonaldTrump's #PromisesKept, fighting #FakeNews. This account punches back 10x harder. Managed by the #TeamTrump 2020 campaign. #MAGA

Follow

Trump Baby
@TrumpBabyUK
I am a six metre high inflatable orange baby - support groups fighting the far right and I'll fly again during Trump's state visit trumpbabyuk@gmail.com

Follow

View all

Search filters

People

From anyone
People you follow

Location

Anywhere
Near you

Advanced search

Trends for you

Trending in Peru
TikTok
561K Tweets

#JuntosLaPasamosMejor
¡Con DIRECTV GO, cada uno mira lo que quiere!
Promoted by DIRECTV Perú | #QuedateEnCasa

Pop · Trending
#WatermelonSugar
175K Tweets

Politics · Trending
Martín Vizcarra

Cómo acceder a los datos?

- Ingresar y crear una cuenta https://apps.twitter.com/.
- Crear una app
- Obtener los siguientes datos:
  - API Key
  - API secret Key
  - Access token
  - Access token secret

Librería tweepy

- https://github.com/tweepy/tweepy
- Descargar e instalar

```
pip install tweepy
```

Autenticación

```python
import tweepy
from tweepy import OAuthHandler

#Credenciales del Twitter API
access_key = "XXXXXX"
access_secret = "XXXXXX"
consumer_key = "XXXXXX"
consumer_secret = "XXXXXX"

'''Método de Autenticación para conectarse a twitter'''
def autenticacion():
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)
    return api
```

Timeline

```
'''leer el timeline de nuestra cuenta de Twitter '''
def get_my_tweets():
    api = autenticacion()
    public_tweets = api.home_timeline(10)
    for tweet in public_tweets:
        print (tweet.text)

get_my_tweets()
```

```
#Opinión21
Juan José García: De interés nacional
Lee y comenta 'Opina.21' https://t.co/0Ld1HK3qJl https://t.co/OBHDYmbne9
Natalia Lafourcade casi lista para lanzar su nuevo disco https://t.co/yIfPyYHKSk
Raheem Sterling: “Miembros de mi familia murieron por coronavirus”
https://t.co/JEse4RO3iN
#Día59
Hombre llena de saliva carrito de supermercado e intenta escupir a un agente de seguridad en
https://t.co/yReyy2DqQx
#Día59
Hombre con síntomas de #COVID_19 muere en puerta del Hospital Angamos https://t.co/EvVyw7f1cc
YGG7fCI4JH
Estación Espacial Internacional se pudo ver desde el cielo de Lima
https://t.co/EkGY2fFxEZ
```

Obtener información de un Usuario

```
'''obtener información de un usuario'''
def get_last_tweets_x_user(screen_name):
    api = autenticacion()
    tweetCount = 10

    print("Getting data for " + screen_name)
    item = api.get_user(screen_name)
    print("name: " + item.name)
    print("screen_name: " + item.screen_name)
    print("description: " + item.description)
    print("friends_count: " + str(item.friends_count))
    print("followers_count: " + str(item.followers_count))

    resultado = api.user_timeline(id=screen_name, count=tweetCount)
    for tweet in resultado:
        print(tweet.text)

get_last_tweets_x_user("@realDonaldTrump")
```

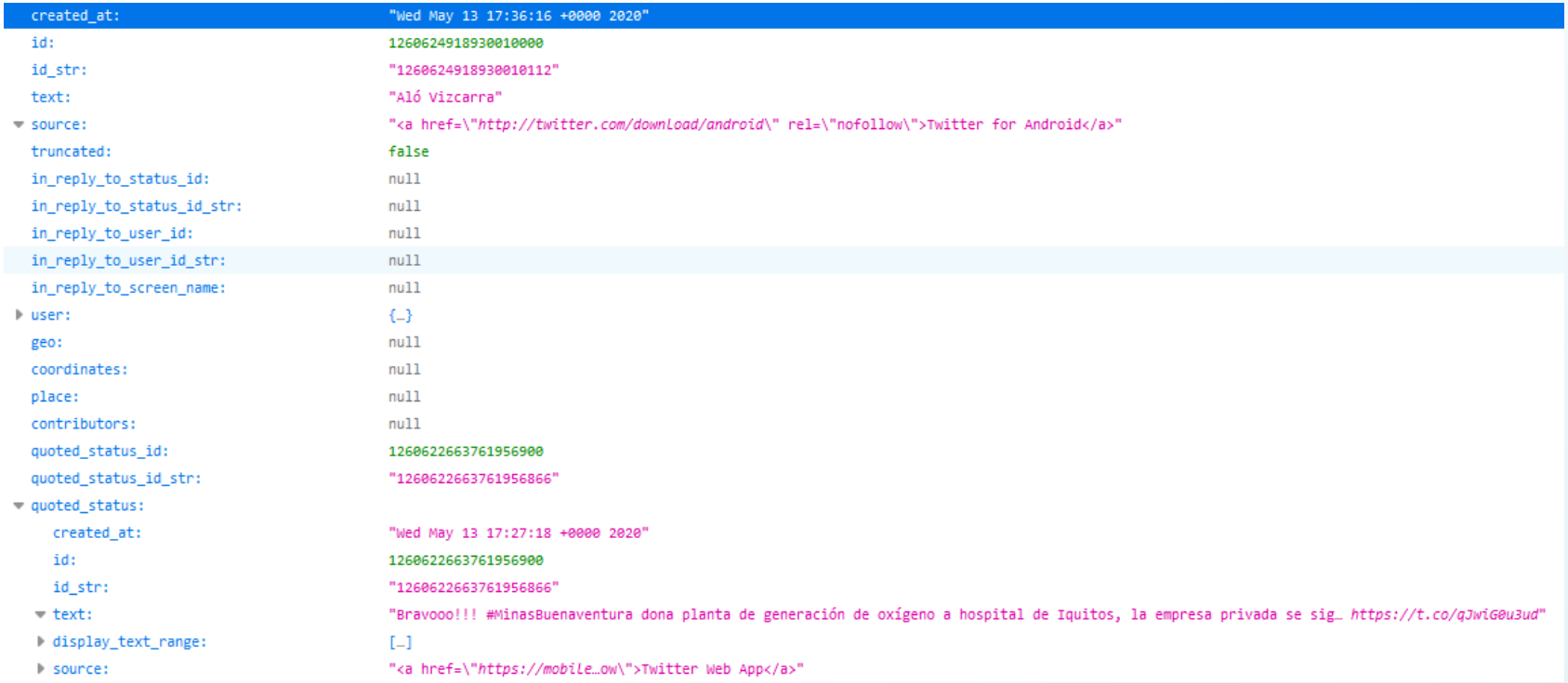
Obtener información de un Usuario

```
Getting data for @realDonaldTrump
name: Donald J. Trump
screen_name: realDonaldTrump
description: 45th President of the United States of AmericaUS
friends_count: 46
followers_count: 79804712
THANK YOU, WORKING HARD! #MAGA https://t.co/NytV20gopi
As I have said for a long time, dealing with China is a very expensive thing to do. We just
made a great Trade Deal… https://t.co/aVzJ0Lr1OS
When the so-called “rich guys” speak negatively about the market, you must always remember
that some are betting bi… https://t.co/d2GPNzjaAU
RT @dcexaminer: "It was the worst journalistic fiasco of now my more than 50 some years in
journalism."

@BritHume called the media's cover…
RT @dcexaminer: MORE:

https://t.co/34C4HQoyNG
RT @alexsalvinews: Fox News’ Brit Hume calls the media’s coverage of alleged collusion with
the Trump campaign and Russia the "worst journa…
RT @dbongino: Brit Hume: Mueller report coverage 'worst journalistic fiasco' he's seen in
50-year news caree https://t.co/bYiMt30Kar
Obama was always wrong! https://t.co/xRU4kJtExs
Does anybody believe this man? Caught! https://t.co/WfnIqs6BsE
“Newly released documents show Schiff knew all along there was no proof of Russia-Trump
collusion.” Wall Street Journal
```

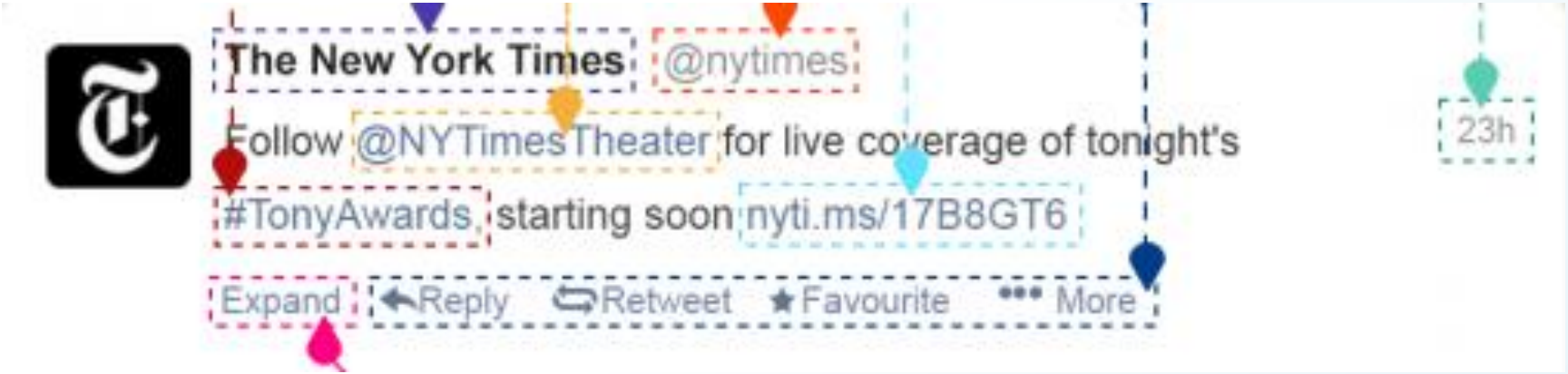
Obtener tweets en Tiempo Real

```
""" Obtener tweets en línea"""
class MyListener(StreamListener):
    def on_data(self,data):
        print("")
        try:
            with open("vizcarra13052020.json","a") as f:
                f.write(data)
                return True
        except BaseException as e:
            print(e)
            return True

    def on_error(self,status):
        print(status)
        return True

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
twitter_stream = Stream(auth,MyListener())
twitter_stream.filter(track=["vizcarra"], languages=['es'] )
```

Obtener tweets en Tiempo Real



Tokenizar tweets



Tokenizar tweets

```
""" tokenizar tweets"""
def preprocess(s):
    emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""

    regex_str =[emoticons_str,
                r'<[^>]+>' , #HTML tags
                r'(?:@[\w_]+)' , #@-Mención
                r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)" , #Hash-tags
                r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&amp;+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+', #URLs
                r'(?:[\w_]+)' , #Otras Palabras
                r'(?:\S)' #Otras Palabras
                ]

    tokens_re = re.compile (r'('+'|'.join(regex_str)+')' ,re.VERBOSE | re.IGNORECASE)
    tokens = tokens_re.findall(s)
    return tokens

tweet = ' RT @amla: sólo un ejemplo! :D http://example.com #NLP en Perú :-)'
print(word_tokenize(tweet))
print(preprocess(tweet))
```

Tokenizar tweets

```
['RT', '@', 'amla', ':', 'sólo', 'un', 'ejemplo', '!', ':', 'D', 'http', ':', '//example.com', '#', 'NLP', 'en', 'Perú', ':', '-',
')']
['RT', '@amla', ':', 'sólo', 'un', 'ejemplo', '!', ':D', 'http://example.com', '#NLP', 'en', 'Perú', ':-)']
```

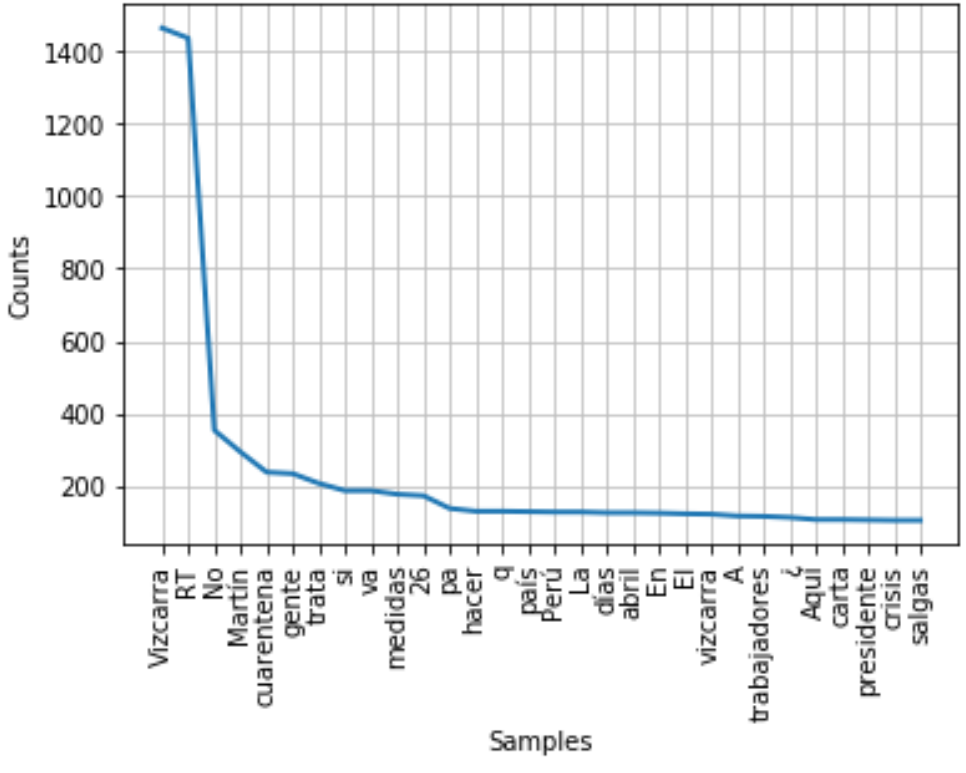
Obtener token de los tweets

```
'''Obtener tokens de los tweets'''
def tokens_text(arc):
    count=0
    tweets_tokens_all=[]
    with open (arc ,"r") as f:
        count_all = Counter()
        for line in f:
            count= count + 1
            if not line.isspace():
                tweet = json.loads(line)
                full_text = tweet["text"]

                if "quoted_status" in tweet:
                    tweet_quoted= tweet["quoted_status"]
                    if "extended_tweet" in tweet_quoted:
                        tweet_extended = tweet_quoted["extended_tweet"]
                        if "full_text" in tweet_extended:
                            full_text = tweet_extended["full_text"]

                "#Crea una lista con todos los términos sin stop"
                terms_all = [term for term in preprocess(full_text)
                             if term not in stopwords]

                "#Actualiza el contador"
                count_all.update(terms_all)
                tweets_tokens_all.extend(terms_all)

    print("total de tweets", count)
```

Estadísticas de tweets

```
'''Obtener estadísticas de los tweets'''
def get_estadisticas(tweets_tokens_all):

    # Cuenta las palabras unicamente, sin #hashtags ni @-menciones
    terms_only = [term for term in tweets_tokens_all if term not in stopwords
                  and not term.startswith(('#', '@'))]
    print('Distribución de Frequencia de Todas las Palabras')
    fdist_todos = nltk.FreqDist(terms_only)
    print('50 palabras mas frequentes',fdist_todos.most_common(50))
    fdist_todos.plot(30, cumulative=False)

    # Cuenta los terminos solo una vez en cada tweet.
    terms_single = set(tweets_tokens_all)
    print(len(terms_single))

    terms_bigram = bigrams(tweets_tokens_all)
    print("BIGRAMAS",terms_bigram)

get_estadisticas(tokens_text("vizcarra08042020.json"))
```

Estadísticas de tweets




```
Distribución de Frequencia de Todas las Palabras
50 palabras mas frequentes [('Vizcarra', 1465), ('RT', 1437), ('No',
353), ('Martín', 293), ('cuarentena', 237), ('gente', 233), ('trata',
206), ('si', 186), ('va', 186), ('medidas', 176), ('26', 172), ('pa',
137), ('hacer', 129), ('q', 129), ('país', 128), ('Perú', 127), ('La',
127), ('días', 125), ('abril', 125), ('En', 124), ('El', 122),
('vizcarra', 121), ('A', 116), ('trabajadores', 115), ('¿', 112),
('Aquí', 106), ('carta', 106), ('presidente', 105), ('crisis', 104),
('salgas', 104), ('masivos', 103), ('conchatumare', 103), ('Ministro',
103), ('plena', 102), ('despidos', 102), ('Salud', 102), ('ministra',
100), ('laborales', 100), ('Una', 100), ('empresas', 100),
('Presidente', 99), ('solo', 99), ('solicitado', 98), ('aplicar', 98),
('paquete', 98), ('puedan', 98), ('https://t.co/uHomt1wfdy', 98),
('haciendo', 98), ('Lima', 97), ('ver', 92)]
```

TASS

Taller de Análisis Semántico en la SEPLN

tweet ID

user ID

content

creation date

language (always 'es')

global polarity, in 5 levels: P+, P, NEU, N, N+ plus NONE

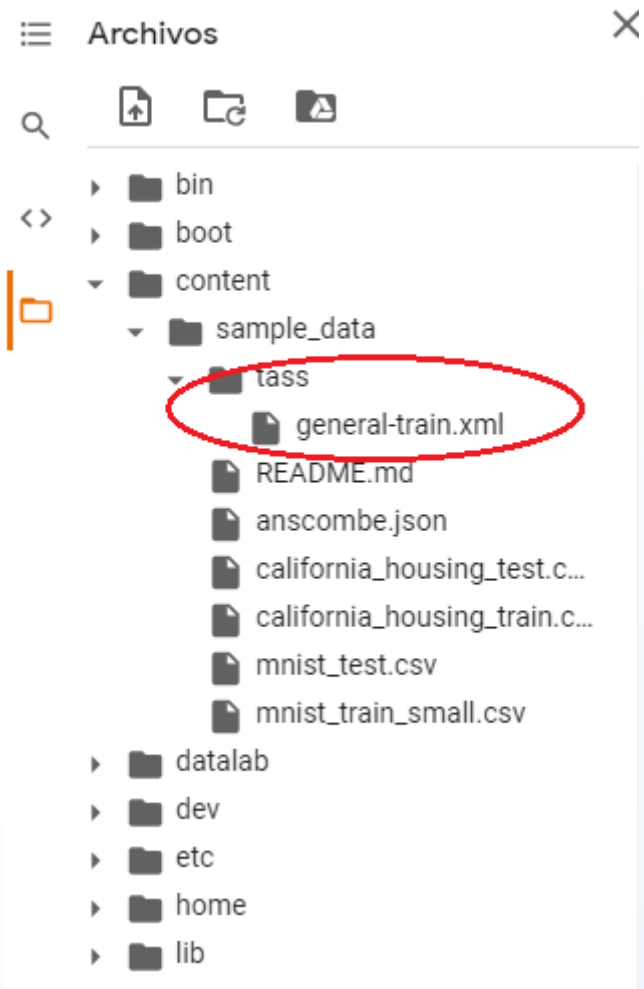
agreement level: AGR, DISAGR

when applicable, **polarity** and **agreement level** related to each entity

topics

TASS

Taller de Análisis Semántico en la SEPLN



Extraer datos de TASS

```
from nltk.stem import SnowballStemmer
stopwords = nltk.corpus.stopwords.words('spanish')

def bag_of_words(words):
    words_dictionary = dict([word, True] for word in words)
    #print('dictionario',words_dictionary)
    return words_dictionary

def stem_tokens(tokens, stemmer):
    stemmed = []
    for item in tokens:
        stemmed.append(stemmer.stem(item))
    return stemmed

def obtain_tokens(tweet):
    stemmer = SnowballStemmer('spanish')
    features  = {}
    #primero se realiza la identificación de tokens y se quitan los stopwords
    tweet_token = [term for term in preprocess(tweet) if term not in stopwords]
    total_words = []
    #segundo se obtienen los stemm
    for word in stem_tokens(tweet_token,stemmer):
        total_words.append(word)
    return total_words
```

Leer archivos TASS

```
for fileName in listFiles:

      soup = BeautifulSoup(open(fileName,'r',encoding='utf8'),features="xml")
      for tweet in soup.find_all("tweet"):
          words = obtain_tokens(tweet.content.text)
          label = tweet.sentiments.polarity.value.text
          if (label=='NONE'):
              #etiqueta='X'
              continue
          if (label=='NEU'):
              etiqueta='Y'
          if (label in ('N','P')):
              etiqueta=label
              if (label=='N'):
                  neg_reviews.append(words)
                  count1= count1+1
              if (label=='P'):
                  pos_reviews.append(words)
                  count2= count2+1
          count= count+1
```

```
for words in pos_reviews:
      pos_reviews_set.append((bag_of_words(words), 'pos'))
for words in neg_reviews:
      neg_reviews_set.append((bag_of_words(words), 'neg'))

size = int(len(pos_reviews_set) * 0.1)
testSet = pos_reviews_set[:size] + neg_reviews_set[:size]
trainSet = pos_reviews_set[size:] + neg_reviews_set[size:]
```

Ejecutar clasificador

```
def clasificadorSentimientos(loc):
    (trainSet, testSet) = lee_datos(loc)

    #Naive Bayes classifier
    classifier1 = nltk.NaiveBayesClassifier.train(trainSet)
    print('Naive Bayes classifier',nltk.classify.accuracy(classifier1,  testSet))

    #Predicting on the test set.
    X_test = [f for (f,pos) in testSet]
    y_test = [pos for (f,pos) in testSet]
    predSet=[]
    for xtest in X_test:
      y_pred = classifier1.classify(xtest)
      predSet.append(y_pred)
    f1_score2 = flat_f1_score(y_test, predSet, average = 'weighted')
    print('f1_score',f1_score2)
    report = flat_classification_report(y_test, predSet)
    print(report)

    return classifier1

locCorpusTass1 = '/content/sample_data/tass/'
clasificadorSentimientos(locCorpusTass1)
```

Ejecutar clasificador

```
Naive Bayes classifier 0.6829268292682927
f1_score 0.6781400966183575
              precision    recall  f1-score   support

           N       0.65      0.80      0.72       123
           P       0.74      0.56      0.64       123

    accuracy                           0.68       246
   macro avg       0.69      0.68      0.68       246
weighted avg       0.69      0.68      0.68       246
```

Predicciones

```
tweet1="@dw_espanol: Lo más triste de la #pandemia del #coronavirus son la cantidad de fallecidos"
tweet2="@dw_espanol: Todos los adultos mayores al fin vacunados!!!"
print(tweet1, clas.classify(bag_of_words(obtain_tokens(tweet1)))  )
print(tweet2, clas.classify(bag_of_words(obtain_tokens(tweet2)))  )
```

```
@dw_espanol: Lo más triste de la #pandemia del #coronavirus son la cantidad de fallecidos N
@dw_espanol: Todos los adultos mayores al fin vacunados!!! P
```

Gracias!

Alguna pregunta?

- Maria.limaylla@gmail.com