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

Autenticación

```
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")
```

api.get_user: función para obtener información del usuario

api.user_timeline: función para obtener los tweets del usuario (se indica la cantidad de tweets)

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'] )
```

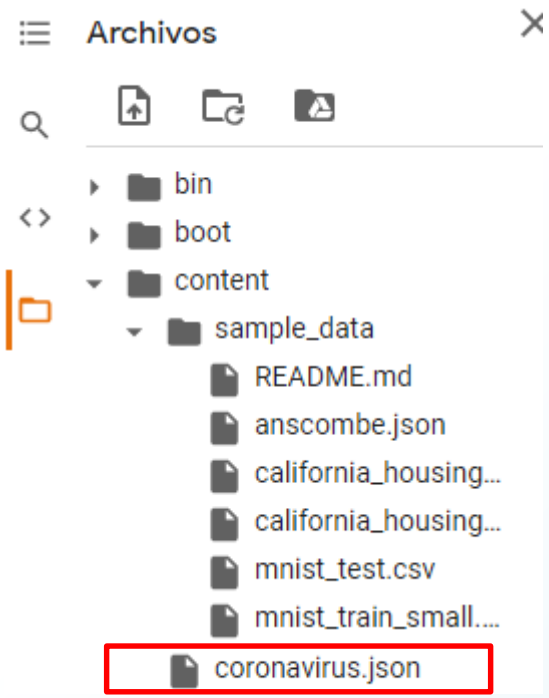
StreamListener: Establece una sesión streaming

Nombre del archivo donde se grabarán los tweets

Invocación de la clase streaming

Palabra clave y lenguaje

Obtener tweets en Tiempo Real



Obtener tweets en Tiempo Real


```
created_at:                  "Wed May 13 17:36:16 +0000 2020"
id:                          1260624918930010000
id_str:                      "1260624918930010112"
text:                        "Aló Vizcarra"
▼ source:                    "<a href=\"http://twitter.com/download/android\" rel=\"nofollow\">Twitter for Android</a>"
truncated:                   false
in_reply_to_status_id:       null
in_reply_to_status_id_str:   null
in_reply_to_user_id:         null
in_reply_to_user_id_str:     null
in_reply_to_screen_name:     null
▶ user:                      {…}
geo:                         null
coordinates:                 null
place:                       null
contributors:                null
quoted_status_id:            1260622663761956900
quoted_status_id_str:        "1260622663761956866"
▼ quoted_status:
    created_at:              "Wed May 13 17:27:18 +0000 2020"
    id:                      1260622663761956900
    id_str:                  "1260622663761956866"
  ▼ text:                    "Bravooo!!! #MinasBuenaventura dona planta de generación de oxígeno a hospital de Iquitos, la empresa privada se sig… https://t.co/qJwiG0u3ud"
  ▶ display_text_range:      […]
  ▶ source:                  "<a href=\"https://mobile…ow\">Twitter Web App</a>"
```

Obtener tweets con fecha

```
def obtener_tweets(screen_name):
    search_words = screen_name
    date_since = "2020-08-14"
    new_search = search_words + " -filter:retweets"
    api = autenticacion()
    outtweets = []

    for tweet in tweepy.Cursor(api.search,
              q=new_search,
              lang="es",
              since=date_since).items(100):
      full_text = tweet.text
      print(full_text)
      outtweets.append((tweet.id_str, tweet.created_at, strip_undesired_chars(full_text),
                        tweet.retweet_count,str(tweet.favorite_count)+''))

    with open('%s_search.csv' % screen_name, "w", encoding="utf-8", newline='') as f:
        writer = csv.writer(f, quoting=csv.QUOTE_ALL)
        writer.writerow(['id','created_at','text','retweet_count','favorite_count'''])
        writer.writerows(outtweets)
    pass
```

Cursor: Función para manejar la paginación
api.search: la función del api para la búsqueda
q=new_search: cadena a buscar
lang: idioma (ejemplo: “es” para español
since: fecha desde donde se extraerán los tweets

Función para guardar los datos en archivo csv

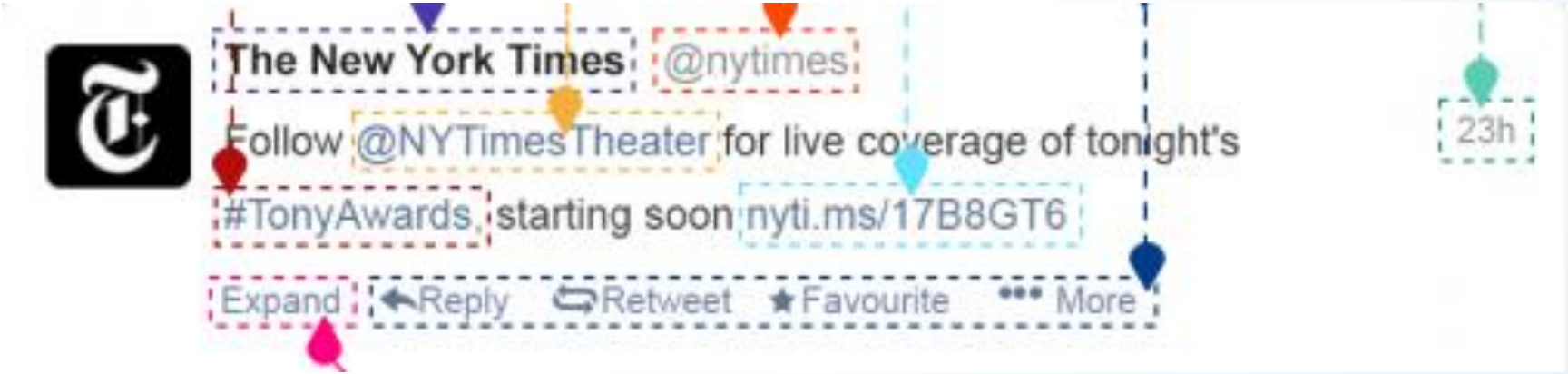
Leer tweets de archivos csv

```
import pandas as pd

def leer_tweets(name):
    data = pd.read_csv(name, encoding = "UTF-8")
    tweets = data['text']
    for t in tweets:
        print(t)

leer_tweets('#pzifer_search.csv')
```

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Tokenizar tweets



Tokenizar tweets

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

Tweet

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

Tokenización normal

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

Tokenización de tweet

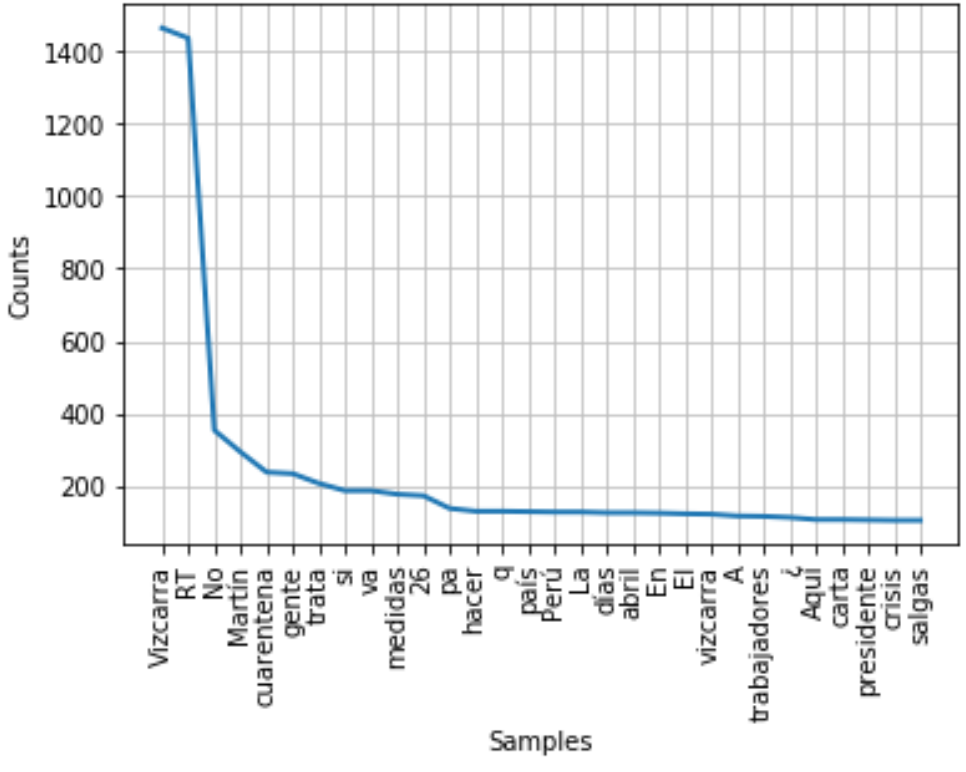
Tokenizar tweets

```
""" 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))
```

Estadísticas de tweets

```
def leer_tweets(name):
    data = pd.read_csv(name, encoding = "UTF-8")
    print(data)
    tweets = data['text']
    tweets_tokens_all=[]
    for t in tweets:
        print(t)
        terms_all = [term for term in preprocess(t)
                          if term not in stopwords]
        tweets_tokens_all.extend(terms_all)

    terms_hash = [term for term in tweets_tokens_all if term.startswith('#')]
    fdist = nltk.FreqDist(terms_hash)
    print('50 palabras mas frequentes',fdist.most_common(50))
    fdist.plot(30, cumulative=False)

    terms_only = [term for term in tweets_tokens_all if term not in stopwords
                  and not term.startswith(('#', '@'))]
    fdist_todos = nltk.FreqDist(terms_only)
    print('50 palabras mas frequentes',fdist_todos.most_common(50))
    fdist_todos.plot(30, cumulative=False)

leer_tweets('#pzifer_search.csv')
```

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)]
```

Análisis de sentimientos - Twitter

TASS

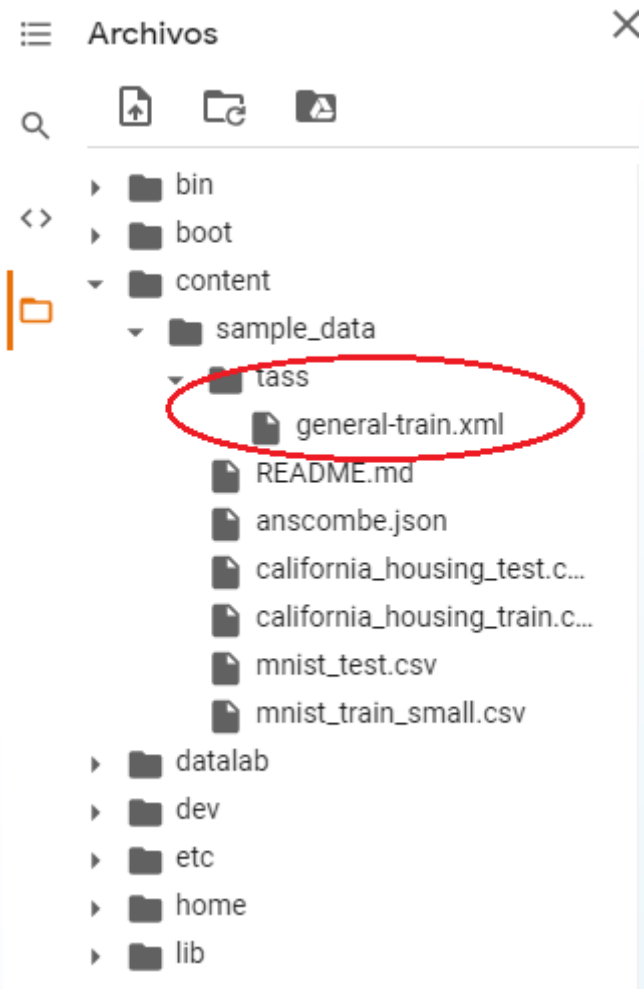
Taller de Análisis Semántico en la SEPLN

- 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

```
<tweet>
  <tweetid>0000000000</tweetid>
  <user>usuario0</user>
  <content>
    <![CDATA[¡Conozco a alguien q es adicto al drama! Ja ja ja te suena d algo!]]>
  </content>
  <date>2011-12-02T02:59:03</date>
  <lang>es</lang>
  <sentiments>
    <polarity>
      <value>P+</value>
      <type>AGREEMENT</type>
    </polarity>
  </sentiments>
  <topics>
    <topic>entretenimiento</topic>
  </topics>
</tweet>
<tweet>
  <tweetid>0000000001</tweetid>
  <user>usuario1</user>
  <content>
    <![CDATA[¡UPyD contará casi seguro con grupo gracias al Foro Asturias.]]>
  </content>
  <date>2011-12-02T00:21:01</date>
  <lang>es</lang>
  <sentiments>
    <polarity>
      <value>P</value>
      <type>AGREEMENT</type>
    </polarity>
    <polarity>
      <entity>UPyD</entity>
      <value>P</value>
      <type>AGREEMENT</type>
    </polarity>
    <polarity>
      <entity>Foro_Asturias</entity>
      <value>P</value>
      <type>AGREEMENT</type>
    </polarity>
  </sentiments>
  <topics>
    <topic>política</topic>
  </topics>
</tweet>
```

TASS

Taller de Análisis Semántico en la SEPLN



Extraer datos de TASS

```
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 obtain_tokens(tweet):
    tweet_token = [term for term in preprocess(tweet) if term not in stopwords]
```

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
```

BeautifulSoup: Librería para leer documentos tipo html, xml
soup.find_all: la función para buscar ‘tweet’ en el xml
tweet.sentiments.polarity.value.text: la ruta donde se encuentra la etiqueta en el xml
neg_reviews: listado de comentarios negativos
pos_reviews: listado de comentarios positivos

Leer archivos TASS

```
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:]
```

Pos_reviews_set: listado de BOW negativos
Neg_reviews_set: listado de BOW positivos
testSet: listado de BOW para prueba
trainSet: listado de BOW para entrenamiento

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)))  )
```

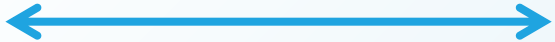
```
@dw_espanol: Lo más triste de la #pandemia del #coronavirus son la cantidad de fallecidos N
@dw_espanol: Todos los adultos mayores al fin vacunados!!! P
```

Predicciones

```
def leer_tweets_final(name, clasificador):
    data = pd.read_csv(name, encoding = "UTF-8")
    tweets = data['text']
    tweets_tokens_all=[]
    for t in tweets:
        print(t)
        print(preprocess(t))
        terms_all = [term for term in preprocess(t)
                          if term not in stopwords]
        tweets_tokens_all.extend(terms_all)
        newTexto = clasificador.classify(bag_of_words(obtain_tokens(t)))
        print("Resultado Clasificador Naive Bayes", newTexto)

    terms_hash = [term for term in tweets_tokens_all if term.startswith('#')]
    fdist = nltk.FreqDist(terms_hash)
    print('50 palabras mas frequentes',fdist.most_common(50))
    fdist.plot(30, cumulative=False)

    terms_only = [term for term in tweets_tokens_all if term not in stopwords
                  and not term.startswith(('#', '@'))]
    fdist_todos = nltk.FreqDist(terms_only)
    print('50 palabras mas frequentes',fdist_todos.most_common(50))
    fdist_todos.plot(30, cumulative=False)

locCorpusTass1 = '/content/sample_data/tass/'
clasificador = clasificadorSentimientos(locCorpusTass1)
leer_tweets_final('#pzifer_search.csv',clasificador)
```

Librería sentiment-spanish

```
#!pip install sentiment-analysis-spanish
from sentiment_analysis_spanish import sentiment_analysis

sentiment = sentiment_analysis.SentimentAnalysisSpanish()
print(sentiment.sentiment("me gusta la tombola es genial"))
print(sentiment.sentiment("me parece terrible esto que me estás diciendo"))
```



0
Negativo

1
Positivo

```
0.9304396176531412
2.1830853580533075e-06
```

https://github.com/sentiment-analysis-spanish/sentiment-spanish

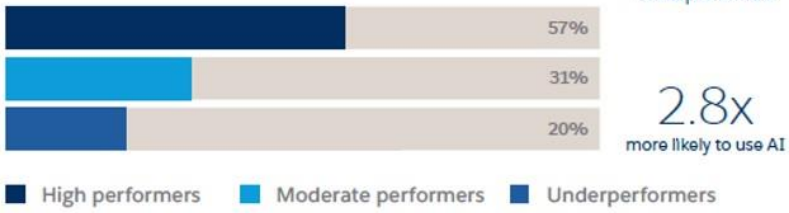
Material

AI Reaches Critical Mass with Top Performers

Sales Organizations Reporting AI Use

76% increase

AI Adoption by Sales Performance Level



High performers vs. Underperformers

2.8x more likely to use AI

Base: Sales leaders

A Surge in AI Adoption Makes Its Mark

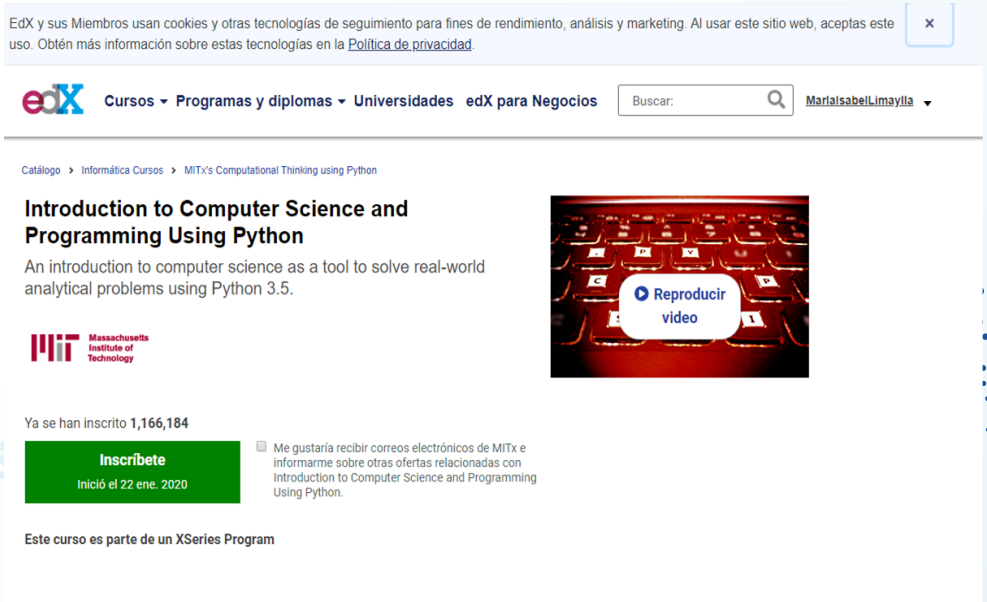
Ranking of AI Impacts

1. Understanding customer needs
2. Forecasting
3. Visibility into rep activity
4. Competitive intelligence
5. Lead prioritization
6. Use of reps' time
7. Personalization for customers

Base: Sales ops and sales leadership at companies using AI.
Ranked by percentage who say the improvements have been "major."

MOOCS

- Udemy – Curso Básico de Machine Learning
- EDX - The Analytics Edge
- EDX - Introduction to Computer Science and Programming Using Python



MOOCS

- Coursera - Machine Learning
- Coursera - Natural Language Processing in TensorFlow
- AWS Machine Learning Accelerator - Natural Language Processing (youtube)

LINKS

- https://www.nltk.org/book/
- https://www.aprendemachinelearning.com/guia-de-aprendizaje/
- https://www.datasciencecentral.com/profiles/blogs/comprehensive-repository-of-data-science-and-ml-resources

Datasets

- UCI Machine Learning Repositorios (http://archive.ics.uci.edu/ml/index.php)
- Kaggle Datasets (https://www.kaggle.com/datasets)
- Google Datasets Search (https://datasetsearch.research.google.com)
- Airbnb (http://insideairbnb.com/get-the-data.html)
- Lists of DataSets (https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research)
- En español (https://lionbridge.ai/datasets/22-best-spanish-language-datasets-for-machine-learning/)

Gracias!

Alguna pregunta?

- Maria.limaylla@gmail.com