Practica Weka - SentiWordNet - Python

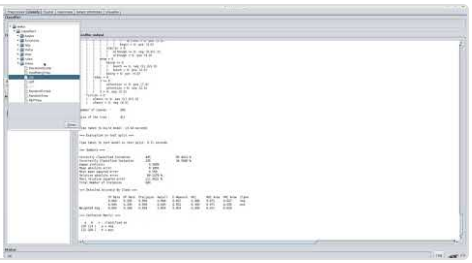
WEKA
SentiWordNet
Ejemplo
Python detección de emociones con Bayes Naive

WEKA

PLN práctica WEKA IMDB

En este vídeo muestro como usar Weka, para realizar un análisis de sentimiento sobre el conjunto de datos de criticas positivas y negativas de IMDB

https://www.youtube.com/watch?v=SuM-ZM4d1DM



Weka 3 - Data Mining with Open Source Machine Learning Software in Java

Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. Found only on the islands of New Zealand, the Weka is a flightless bird with an inquisitive nature. The name is pronounced like this, and the

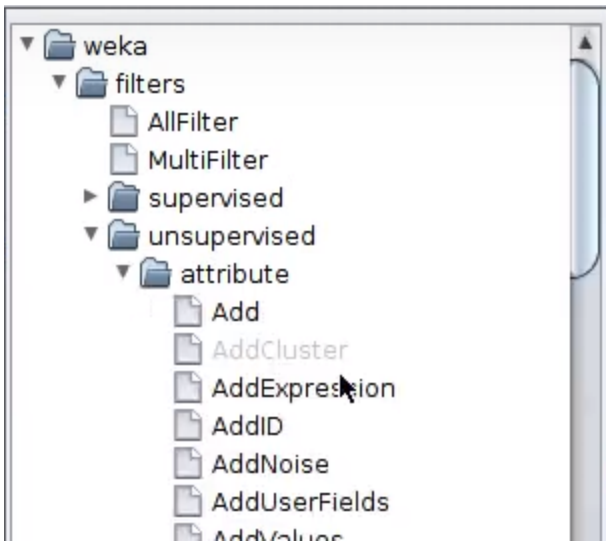
https://www.cs.waikato.ac.nz/ml/weka/

```
java -jar weka.jar
```

abrimos directorio donde estan los archivos

Le decimos que tipo de archivo es → TextDirectoryLoader

Tenemos que tokenizar los archivos de texto



Seleccionamos StringToWordVector → tokenizacion utilizando una red

Tocamos apply y aparecen todas las palabras y la cantidad de veces que aparecen cada una

Segunda solapa: Classifier

Vamos a elegir un metodo de clasificacion → Bayes

Elegimos NaiveBayes

Le podemos decir que queremos usar Cross Validation utilizando 10 folds o que queremos usar una division de 2/3 y 1/3 para entrenamiento y prueba

Otro metodo: Naive Bayes Multinomial

Cambia la distribucion, la primera tiene una distribucion de tipo Gaussiana y la segunda va a asumir que para cada atributo hay una distribucion de tipo Multinomial

Los resultados son similares.

Si elegimos otro metodo como:

- Arboles

  J48 → es el C4.5

Los resultados no son tan buenos

- SVM

  SMO

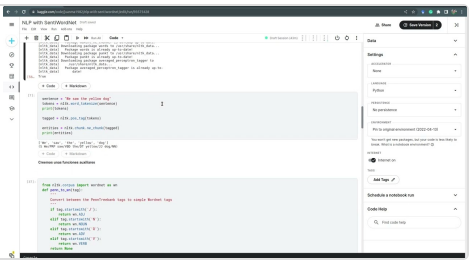
  Mejor que el arbol, pero debajo de Bayes

SentiWordNet

PLN práctica SentiWordNet

En este vídeo explico como utilizar el diccionario o Lexicón de sentimientos: SentiWordNet desde código Python, y en un maquina Jupyter

https://www.youtube.com/watch?v=LIKhzTCF4kM



Diccionario de palabras que tienen sentimientos

Las palabras estan separadas segun su categoria gramatical

Las letras indican de qué forma es usada la palabra:
N: Sustantivo (noun)
A: Adjetivo
R: Adverbio
V: Verbo
O mostrará una S para indicar un sinónimo. El número indica la acepción, ya que una misma palabra aún utilizada en la misma categoría gramatical (sustantivo, por ejemplo) podría tener diversas acepciones.

POS: part of the speech (categoria gramatical)

Score que puede ser positivo o negativo

1. Instalar el paquete si no existe:
   ```
   !pip install nltk #! Permite ejecutar comandos del sistema operativo
   ```
2. Importar la librería y descargar el diccionario
   ```
   import nltk
   nltk.download('sentiwordnet')
   nltk.download('wordnet')
   from nltk.corpus import sentiwordnet as swn
   ```

```
from nltk.corpus import sentiwordnet as swn
good = swn.senti_synsets('good')
list(good)
```

```
[SentiSynset('good.n.01'),
 SentiSynset('good.n.02'),
 SentiSynset('good.n.03'),
 SentiSynset('commodity.n.01'),
 SentiSynset('good.a.01'),
 SentiSynset('full.s.06'),
 SentiSynset('good.a.03'),
 SentiSynset('estimable.s.02'),
 SentiSynset('beneficial.s.01'),
 SentiSynset('good.s.06'),
 SentiSynset('good.s.07'),
 SentiSynset('adept.s.01'),
 SentiSynset('good.s.09'),
 SentiSynset('dear.s.02'),
 SentiSynset('dependable.s.04'),
 SentiSynset('good.s.12'),
 SentiSynset('good.s.13'),
 SentiSynset('effective.s.04'),
 SentiSynset('good.s.15'),
 SentiSynset('good.s.16'),
 SentiSynset('good.s.17'),
 SentiSynset('good.s.18'),
 SentiSynset('good.s.19'),
 SentiSynset('good.s.20'),
 SentiSynset('good.s.21'),
 SentiSynset('well.r.01'),
 SentiSynset('thoroughly.r.02')]
```

Tambien muestra sinonimos

Hay varias entradas para una misma palabra → lo que representa un problema

3. Ejemplo, para ver las posibilidades de la palabra “good” (bueno en inglés)

```
good = swn.senti_synsets('good', 'n')
```

```
list(good)
```

```
[SentiSynset('good.a.01'),
 SentiSynset('full.s.06'),
 SentiSynset('good.a.03'),
 SentiSynset('estimable.s.02'),
 SentiSynset('beneficial.s.01'),
 SentiSynset('good.s.06'),
 SentiSynset('good.s.07'),
 SentiSynset('adept.s.01'),
 SentiSynset('good.s.09'),
 SentiSynset('dear.s.02'),
 SentiSynset('dependable.s.04'),
 SentiSynset('good.s.12'),
 SentiSynset('good.s.13'),
 SentiSynset('effective.s.04'),
 SentiSynset('good.s.15'),
 SentiSynset('good.s.16'),
 SentiSynset('good.s.17'),
 SentiSynset('good.s.18'),
 SentiSynset('good.s.19'),
 SentiSynset('good.s.20'),
 SentiSynset('good.s.21')]
```

```
good = swn.senti_synsets('good', 'a')
posscore=0
negscore=0
for synst in good:

    posscore=posscore+synst.pos_score()
    negscore=negscore+synst.neg_score()


print(posscore)
print(negscore)
```

```
13.0
0.125
```

Recorrer todos los synst y obtener todos los puntajes positivos e ir sumadolos en una variable general (lo mismo con los negativos).

8. Utilizar un parser superficial, para detectar la categoría gramatical de cada palabra. Y mejorar el código anterior usando solo el puntaje de la palabra para la categoría gramatical detectada. (Es decir solo A, R, N o V según se haya detectado).
   Código de ayuda:
   a.
   ```
   nltk.download('maxent_ne_chunker')
   nltk.download('words')
   nltk.download('punkt')
   nltk.download('averaged_perceptron_tagger')
   ```
   b.
   ```
   sentence = "We saw the yellow dog"
   tokens = nltk.word_tokenize(sentence)
   print(tokens)
   tagged = nltk.pos_tag(tokens)
   entities = nltk.chunk.ne_chunk(tagged)
   print(entities)
   ```

```
sentence = "We saw the yellow dog"
tokens = nltk.word_tokenize(sentence)
print(tokens)

tagged = nltk.pos_tag(tokens)

entities = nltk.chunk.ne_chunk(tagged)
print(entities)
```

```
['We', 'saw', 'the', 'yellow', 'dog']
(S We/PRP saw/VBD the/DT yellow/JJ dog/NN)
```

Creamos funciones auxiliares

```python
from nltk.corpus import wordnet as wn
def penn_to_wn(tag):
    """
    Convert between the PennTreebank tags to simple Wordnet tags
    """
    if tag.startswith('J'):
        return wn.ADJ
    elif tag.startswith('N'):
        return wn.NOUN
    elif tag.startswith('R'):
        return wn.ADV
    elif tag.startswith('V'):
        return wn.VERB
    return None

from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

def get_sentiment(word,tag):
    """ returns list of pos neg and objective score. But returns empty list if not present in senti wordnet. """

    wn_tag = penn_to_wn(tag)
    if wn_tag not in (wn.NOUN, wn.ADJ, wn.ADV):
        return []

    lemma = lemmatizer.lemmatize(word, pos=wn_tag)
    if not lemma:
        return []

    #print(f"word: {word}")
    synsets = wn.synsets(word, pos=wn_tag)
    if not synsets:
        return []
    #print(f"synsets: {synsets[0]}")

    # Take the first sense, the most common
    synset = synsets[0]
    swn_synset = swn.senti_synset(synset.name())

    return [swn_synset.pos_score(),swn_synset.neg_score(),swn_synset.obj_score()]
```

PennTreebank: conjunto de oraciones etiquetadas a mano en idioma ingles para construir herramientas de parsing y de etiquetado gramatical

Cuando estamos trabajando con las categorias gramaticales que se obtuvieron de ese original conjunto de datos, estamos hablando de las categorias de tipo PennTreebank.

Algunos parses la utilizan

Wordnet utiliza otro tipo de tageo

lemmatizar: convertir la palabra a su expresion minima o expresion raiz

Ejemplo: si tenemos gato, gata, gatitos, gatos ⇒ raiz de la palabra es gat

A veces no se puede entonces devuelve vacio.

```
def get_sentiment_binary(word,tag):
    aux = get_sentiment(word,tag)
    if not aux:
        return 0

    return aux[0]-aux[1]

def check_sentence(sentence):
    rr = 0
    tokens = nltk.word_tokenize(sentence)
    pos_val = nltk.pos_tag(tokens)
    for e in pos_val:
        rr += get_sentiment_binary(e[0],e[1])
    return rr
```

En `get_sentiment_binary` si el score positivo es mas grande ⇒ va a ser positivo.

Si el negativo era mas grande ⇒ va a ser negativo

Asi el resultado es mas facil de leer

Si es positivo ⇒ es una palabra con carga positiva

Si es negativo ⇒ es una palabra con carga negativa

```
check_sentence("this movie is wonderful")
```

```
word: movie
synsets: Synset('movie.n.01')
word: wonderful
synsets: Synset('fantastic.s.02')
0.75
```

Ejemplo

```
import pandas as pd
emotions = pd.read_csv('/kaggle/input/emotion-detection-from-text/tweet_emotions.csv')

emotions.head(10)
```

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...
5	1956968477	worry	Re-pinging @ghostridah14: why didn't you go to...
6	1956968487	sadness	I should be sleep, but im not! thinking about ...
7	1956968636	worry	Hmmm. http://www.djhero.com/ is down
8	1956969035	sadness	@charviray Charlene my love. I miss you
9	1956969172	sadness	@kelcouch I'm sorry at least it's Friday?

```
print(emotions['sentiment'].unique())
```

```
['empty' 'sadness' 'enthusiasm' 'neutral' 'worry' 'surprise' 'love' 'fun'
 'hate' 'happiness' 'boredom' 'relief' 'anger']
```

Usando SentiWordNet tengo dos posibilidades → una sentencia puede ser positiva o negativa

Entonces no me va a servir para detectar estas emociones

⇒ voy a tratar de simplificar el problema

Englobar todas las emociones positivas en una categoria y lo mismo con las negativas

Cuantas emociones hay en el dataset original:

```
print('Row count is:', len(emotions))
```

```
Row count is: 40000
```

```
negative=emotions.query("sentiment in ('sadness','worry', 'hate', 'boredom','anger')")
print(negative)
```

```
        tweet_id sentiment                                          content
1      1956967666   sadness  Layin n bed with a headache  ughhhh...waitin o...
2      1956967696   sadness                Funeral ceremony...gloomy friday...
5      1956968477     worry  Re-pinging @ghostridah14: why didn't you go to...
6      1956968487   sadness  I should be sleep, but im not! thinking about ...
7      1956968636     worry              Hmmm. http://www.djhero.com/ is down
...           ...       ...                                                ...
39939  1753903509   sadness  @watermelon39 haha! And Twitter! Hard though i...
39941  1753903578     worry  @PH7S sure. But be careful also of making stat...
39956  1753903987     worry                How Do You Sleep - Jesse McCartney
39965  1753904398   sadness                          is heading off to the fair
39978  1753904868     worry  @givemestrength bloody Feds, they lost last st...

[15236 rows x 3 columns]
```

```
positive=emotions.query("sentiment in ('enthusiasm','love', 'fun', 'happiness')")
print(positive)
```

```
        tweet_id   sentiment  \
3      1956967789  enthusiasm
16     1956971170        love
21     1956972097         fun
40     1956977084   happiness
41     1956977187         fun
...           ...         ...
39994  1753918900   happiness
39996  1753919001        love
39997  1753919005        love
39998  1753919043   happiness
39999  1753919049        love

                                           content
3                  wants to hang out with friends SOON!
16                               @annarosekerr agreed
21     Wondering why I'm awake at 7am,writing a new s...
40     mmm much better day... so far! it's still quit...
41     @DavidArchie &lt;3 your gonna be the first  tw...
...                                              ...
39994                  Succesfully following Tayla!!
39996                Happy Mothers Day  All my love
39997  Happy Mother's Day to all the mommies out ther...
39998  @niariley WASSUP BEAUTIFUL!!! FOLLOW ME!!  PEE...
39999  @mopedronin bullet train from tokyo    the gf ...

[11586 rows x 3 columns]
```

```
print('Row count is:', len(positive))
print('Row count is:', len(negative))
```

```
Row count is: 11586
Row count is: 15236
```

Ver que tan bien desempeña SensitWordNet

```
check_sentence('@tiffanylue i know  i was listenin to bad habit earlier and i started freakin at his part =[')
```

```
word: tiffanylue
word: i
synsets: Synset('iodine.n.01')
word: i
synsets: Synset('iodine.n.01')
word: bad
synsets: Synset('bad.a.01')
word: habit
synsets: Synset('habit.n.01')
word: earlier
synsets: Synset('earlier.r.01')
word: freakin
word: part
synsets: Synset('part.n.01')
word: [
-0.625
```

Devolvio negativo

```
total_true_neg = 0
total_false_neg = 0
for index, row in negative.iterrows():
    if(check_sentence(row['content']) < 0 ):
        total_true_neg+=1
    elif(check_sentence(row['content']) > 0 ):
        total_false_neg+=1
```

```
print('TN:', total_true_neg)
print('FN:', total_false_neg)
```

```
TN: 5979
FN: 4553
```

```
total_true_pos = 0
total_false_pos = 0
for index, row in positive.iterrows():
    if(check_sentence(row['content']) > 0 ):
        total_true_pos+=1
    elif(check_sentence(row['content']) < 0 ):
        total_false_pos+=1
```

```
print('TP:', total_true_pos)
print('FP:', total_false_pos)
```

```
TP: 6316
FP: 1961
```

```
print('Precision clase positiva:', total_true_pos / (total_true_pos+total_false_pos))
print('Precision clase negativa:', total_true_neg / (total_true_neg+total_false_neg))
```

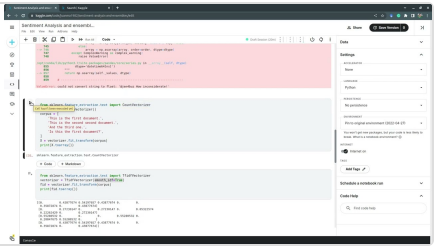
```
Precision clase positiva: 0.7630784100519512
Precision clase negativa: 0.5676984428408659
```

Python detección de emociones con Bayes Naive

PLN práctica Python detección de emociones con Bayes Naive

En este vídeo explico como utilizar Bayes Naive en código Python desde una maquina Jupiter.

https://www.youtube.com/watch?v=M9L2Ze2uDbs



```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/emotion-final-fixed/Emotion_final.csv
/kaggle/input/emotion-prediction-with-semi-supervised-learning/Emotion_Prediction_with_Semi_Supervised_Learning_of_Machine_Learning_Software_with_RC_Algorithm_By_Emirhan_BULUT.ipynb
/kaggle/input/emotion-prediction-with-semi-supervised-learning/LICENSE
/kaggle/input/emotion-prediction-with-semi-supervised-learning/.gitignore
/kaggle/input/emotion-prediction-with-semi-supervised-learning/README.md
/kaggle/input/emotion-prediction-with-semi-supervised-learning/Emotion Prediction with Semi Supervised Learning of Machine Learning Software with RC Algorithm - By Emirhan BULUT.png
/kaggle/input/emotion-prediction-with-semi-supervised-learning/tweet_emotions.csv
```

```
#Import to ML (scikit-learn), Data (Pandas) and Math (NumPy) Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

#Data Loading from 'Kaggle'
df = pd.read_csv('../input/emotion-prediction-with-semi-supervised-learning/tweet_emotions.csv')

df.head(10)
```

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...
5	1956968477	worry	Re-pinging @ghostridah14: why didn't you go to...
6	1956968487	sadness	I should be sleep, but im not! thinking about ...
7	1956968636	worry	Hmmm. http://www.djhero.com/ is down
8	1956969035	sadness	@charviray Charlene my love. I miss you
9	1956969172	sadness	@kelcouch I'm sorry at least it's Friday?

```
#Data Preprocessing
X_train, X_test, y_train, y_test = train_test_split(df.content,
                                                    df.sentiment,
                                                    test_size=0.30,
                                                    random_state=25,
                                                    shuffle=True)
```

```
from sklearn.naive_bayes import MultinomialNB

bayes = MultinomialNB()
bayes.fit(X_train, y_train)
```

ⓘ You have categorical data, but your model needs something numerical. See our one hot encoding tutorial for a solution.

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_17/3218511017.py in <module>
      2
      3 bayes = MultinomialNB()
----> 4 bayes.fit(X_train, y_train)
```

Falla

Necesitamos un modelo para conertir palabras a vectores

Vamos a utilizar el Bag of Words

Count Vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
corpus = [
    'This is the first document.',
    'This is the second second document.',
    'And the third one.',
    'Is this the first document?',
]
X = vectorizer.fit_transform(corpus)
print(X.toarray())
```

```
[[0 1 1 1 0 0 1 0 1]
 [0 1 0 1 0 2 1 0 1]
 [1 0 0 0 1 0 1 1 0]
 [0 1 1 1 0 0 1 0 1]]
```

Dado un corpus que es un array de strings, va a contar las ocurrencias

TfidVectorizer

Vectorizer, pero de indice de frecuencia invertido

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(smooth_idf=True)
fid = vectorizer.fit_transform(corpus)
print(fid.toarray())
```

```
[[0.         0.43877674 0.54197657 0.43877674 0.         0.
  0.35872874 0.         0.43877674]
 [0.         0.27230147 0.         0.27230147 0.         0.85322574
  0.22262429 0.         0.27230147]
 [0.55280532 0.         0.         0.         0.55280532 0.
  0.28847675 0.55280532 0.        ]
 [0.         0.43877674 0.54197657 0.43877674 0.         0.
  0.35872874 0.         0.43877674]]
```

Ahora lo hacemos con el conjunto de datos de emociones

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix, accuracy_score

# Build the model
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model2 = make_pipeline(CountVectorizer(), MultinomialNB())
# Train the model using the training data
model.fit(X_train, y_train)
model2.fit(X_train, y_train)
# Predict the categories of the test data
predicted_categories = model.predict(X_test)
predicted_categories_2 = model2.predict(X_test)
```

pipeline donde pongo una entrada y la salida de esa entrada pasa al segundo modelo

Con un pipeline puedo construir un modelo que primero haga una conversion

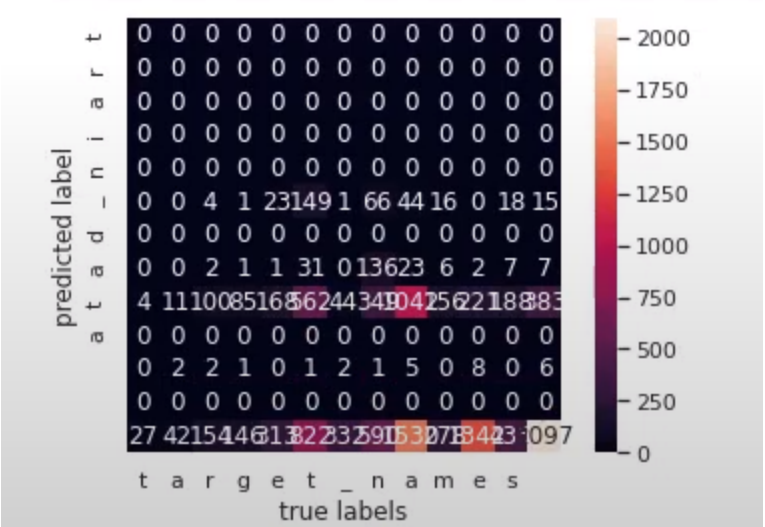
```
print(predicted_categories)
```

```
['worry' 'neutral' 'neutral' ... 'worry' 'worry' 'worry']
```

Ahora tengo que visualizar y graficar que tan bien le fue a estos dos modelos

```
import numpy as np, pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score
sns.set() # use seaborn plotting style

# plot the confusion matrix
mat = confusion_matrix(y_test, predicted_categories)
sns.heatmap(mat.T, square = True, annot=True, fmt = "d", xticklabels="target_names",yticklabels="train_data")
plt.xlabel("true labels")
plt.ylabel("predicted label")
plt.show()
print("The accuracy using TfidfVectorizer  is {}".format(accuracy_score(y_test, predicted_categories)))
print("The precision is TfidfVectorizer {}".format(precision_score(y_test, predicted_categories, average='micro',)))
print("The accuracy count is  {}".format(accuracy_score(y_test, predicted_categories_2)))
print("The precision count is {}".format(precision_score(y_test, predicted_categories_2, average='micro',)))
```




```
The accuracy using TfidfVectorizer  is 0.286
The precision is TfidfVectorizer 0.286
The accuracy count is  0.3139166666666667
The precision count is 0.3139166666666667
```

El de indice de frecuencia invertido dio peores resultados que el count

Pero ambos no fueron demasiado buenos