Procesamiento del lenguaje natural en tweets

Para la realización de este ejercicio, partimos de un dataset que encontré en internet de tweets clasificados por el tema de cada tweet.
(fuente: https://huggingface.co/datasets/cardiffnlp/tweet_topic_multi).

Este dataset contiene más de 5000 tweets para entrenamiento y otros 5000 para test. He tenido que realizar algo de preprocesamiento ya que los tweets están en inglés y a la hora de hacer pruebas quería hacerlas en español, pero no ha supuesto problema gracias a la librería ***deep_translator***.

Además he tenido que borrar de cada tweet las menciones y los hashtags, así como los enlaces.

Importación y preprocesamiento de los datos:

```
X_train =
pd.read_csv('C:/Users/Pablo/Desktop/IA-BD/MIA/Twitter/Twitter/datasets/x_train.csv')
Y_train =
pd.read_csv('C:/Users/Pablo/Desktop/IA-BD/MIA/Twitter/Twitter/datasets/y_train.csv')
X_test =
pd.read_csv('C:/Users/Pablo/Desktop/IA-BD/MIA/Twitter/Twitter/datasets/x_test.csv')
Y_test =
pd.read_csv('C:/Users/Pablo/Desktop/IA-BD/MIA/Twitter/Twitter/datasets/y_test.csv')


X_train['text'] = X_train['text'].apply(lambda x: re.sub("@[A-Za-z0-9_]+","", x))
X_train['text'] = X_train['text'].apply(lambda x: re.sub(r"\s*{.*}\s*", " ", x))
X_train['text'] = X_train['text'].apply(lambda x: re.sub("#[A-Za-z0-9_]+","", x))
X_train['text'] = X_train['text'].apply(lambda x: re.sub(r'https\S+', '', x))
X_train['text'] = X_train['text'].apply(lambda x: re.sub(r'http\S+', '', x))
```

El siguiente paso ha sido vectorizar los tweets haciendo uso de ***TfidVectorizer().***

Esto lo hacemos para el posterior entrenamiento en el que usaremos una máquina de soporte vectorial.

Vectorización de los datos

```
vectorizer = TfidfVectorizer()
#cálculo de los TF-IDF para todos los términos de la colección
vectorizer.fit(X_train['text'])
vectorizer.fit(X_test['text'])
#convertir a valores numéricos TF-IDF los datos de entrenamiento
X_train_Tfidf = vectorizer.transform(X_train['text'])
```

Una vez vectorizados los datos, podemos proceder al entrenamiento. En este caso vamos a usar una máquina de soporte vectorial como ya hemos dicho. El kernel que mejores resultados ha dado y el que he elegido ha sido el ***kernel linear.***

```
SVM = svm.SVC(kernel='linear', gamma=0.0001, random_state=42, degree=12) #
Definir el modelo SVM
SVM.fit(X_train_Tfidf,Y_train['topic']) # Fase de entrenamiento del modelo
X_test_Tfidf = vectorizer.transform(X_test['text']) #convertir a valores
numéricos TF-IDF los datos de test. Fase de uso
resultado = SVM.predict(X_test_Tfidf) #uso
```

Una vez entrenado el modelo, ya podemos probarlo y medir su rendimiento.

```
from deep_translator import GoogleTranslator

def predecir(tweet):
    tweet_eng = GoogleTranslator(source='auto', target='en').translate(tweet)
    prediction = SVM.predict(vectorizer.transform([tweet_eng]))
    if prediction[0] == 'gaming':
        return (tweet, prediction[0])
    return (tweet, GoogleTranslator(source='auto',
target='es').translate(prediction[0].replace('_', ' ')))

print(predecir("No sé a qué partido político debería votar"))
print(predecir("El pádel en sin duda mi deporte favorito"))
print(predecir("Este álbum me parece un antes y un después, recomendadísimo"))
print(predecir("Me encanta jugar a la xbox con mis amigos"))
```

En el código de arriba, vemos cómo traducimos los tweets de prueba, ya que el modelo está entrenado con tweets en inglés, y luego traducimos de vuelta el resultado. A continuación las predicciones de nuestro modelo:

```
('No sé a qué partido político debería votar', 'noticias y preocupación social')
('El pádel en sin duda mi deporte favorito', 'Deportes')
('Este álbum me parece un antes y un después, recomendadísimo', 'música')
('Me encanta jugar a la xbox con mis amigos', 'gaming')
```

En cuanto a rendimiento, este modelo tiene una tasa de aciertos de un 47%. Aquí abajo dejo las métricas detalladas y la Matriz de Confusión:

classification_report

	precision	recall	f1-score	support
arts_&_culture	0.53	0.20	0.29	267
business_&_entrepreneurs	0.50	0.14	0.22	258
celebrity_&_pop_culture	0.36	0.48	0.41	754
diaries_&_daily_life	0.34	0.34	0.34	574
family	0.57	0.06	0.11	64
fashion_&_style	0.59	0.31	0.41	74
film_tv_&_video	0.41	0.31	0.35	474
fitness_&_health	0.66	0.12	0.21	185
food_&_dining	0.57	0.13	0.21	61
gaming	0.70	0.09	0.16	158
learning_&_educational	1.00	0.01	0.03	69
music	0.59	0.39	0.47	363
news_&_social_concern	0.45	0.76	0.57	1078
other_hobbies	0.00	0.00	0.00	72
relationships	0.00	0.00	0.00	23
science_&_technology	0.00	0.00	0.00	55
sports	0.60	0.75	0.67	991
travel_&_adventure	0.00	0.00	0.00	13
youth_&_student_life	0.00	0.00	0.00	2
accuracy			0.47	5535
macro avg	0.41	0.22	0.23	5535
weighted avg	0.48	0.47	0.43	5535

Matriz de confusión