Multiclass Text Classification with

Logistic Regression Implemented with PyTorch and CE Loss

Este código prepara un entorno para entrenar un modelo de clasificación de texto multiclase utilizando regresión logística implementada en PyTorch con una función de pérdida de entropía cruzada (CE Loss).

First, we will do some initialization.

In [1]:
```
# Importación de librerías
import random
import torch
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm

# Habilita tqdm en pandas
tqdm.pandas()  # Añade una barra de progreso a las operaciones de pandas, útil para monitorear procesos largos

# Establece si se va a usar la GPU (si está disponible)
use_gpu = True  # Si es True, intentará usar la GPU, si no, usará la CPU

# Selección del dispositivo
device = torch.device('cuda' if use_gpu and torch.cuda.is_available() else 'cpu')
print(f'device: {device.type}')

# Semilla aleatoria
seed = 1234  # Establece una semilla aleatoria para asegurar la reproducibilidad de los resultados

# Configuración de la semilla aleatoria
# La semilla asegura que las operaciones aleatorias, como la inicialización de pesos y el particionamiento de datos,
if seed is not None:
    print(f'random seed: {seed}')
    random.seed(seed)
```

```
    np.random.seed(seed)
    torch.manual_seed(seed)
```

```
device: cuda
random seed: 1234
```

El código detectó una GPU compatible con CUDA y la está utilizando para las operaciones de PyTorch. CUDA es una plataforma de computación paralela desarrollada por NVIDIA que permite el uso de GPUs para acelerar cómputos intensivos, como el entrenamiento de modelos de machine learning.

Se configuró una semilla aleatoria de valor 1234. Esto es importante para garantizar la reproducibilidad de los resultados. Usar una semilla fija asegura que las operaciones aleatorias (como la inicialización de pesos del modelo, el particionamiento de datos, etc.) produzcan los mismos resultados en ejecuciones posteriores del código.

We will be using the AG's News Topic Classification Dataset. It is stored in two CSV files: `train.csv` and `test.csv`, as well as a `classes.txt` that stores the labels of the classes to predict.

First, we will load the training dataset using pandas and take a quick look at how the data.

In [13]:
```
# Carga el archivo CSV
train_df = pd.read_csv('/kaggle/input/ag-news-classification-dataset/train.csv', header=None)

# Toma una muestra aleatoria del 80% de los datos de entrenamiento
train_df = train_df.sample(frac=0.8, random_state=42)

# Asigna nombres a las columnas del DataFrame
# 'class index' es la categoría de la noticia
# 'title' es el título de la noticia
# 'description' es el resumen o descripción de la noticia
train_df.columns = ['class index', 'title', 'description']

# Mostrar dataframe
train_df
```

Out[13]:

	class index	title	description
71788	3	BBC set for major shake-up, claims newspaper	London - The British Broadcasting Corporation,...

67218	4	Taking Microsoft for a spin?	The software juggernaut that conquered the des...
54066	3	September sales at Target stores beat retail a...	MINNEAPOLIS - While other retailers struggled ...
7168	4	Macromedia launches Flex Builder	Macromedia this week will ship Flex Builder, w...
29618	1	Rocket lands near Afghan school as President K...	AFP - A rocket landed near a school in southea...
...	...	...	...
59228	4	Technical Problems Subside at PayPal	Most members of the online payment service Pay...
61417	3	Shoppers Spring Back to Life in September	Shoppers got their buying groove back last mon...
20703	3	UPDATE 1-Yellow Roadway raises 3rd-qtr profit ...	Yellow Roadway Corp. (YELL.O: Quote, Profile, ...
40626	3	Next to digital IDs, passwords look lame	How big is your key ring? There are the house ...
25059	2	Prime-time Eagles	They opened their season Sept. 2 in the smalle...

96001 rows × 3 columns

The dataset consists of 120,000 examples, each consisting of a class index, a title, and a description. The class labels are distributed in a separated file. We will add the labels to the dataset so that we can interpret the data more easily. Note that the label indexes are one-based, so we need to subtract one to retrieve them from the list.

Convertimos la columna `'class index'` a valores numéricos, reemplazamos los valores no numéricos por `NaN`, eliminamos las filas con `NaN`, y finalmente convertimos la columna a enteros para asegurar que todos los valores sean numéricos válidos.

In [14]:
```
# Convertir la columna 'class index' a tipo numérico, por si hay valores no numéricos
# El parámetro errors='coerce' convierte cualquier valor no numérico en NaN
train_df['class index'] = pd.to_numeric(train_df['class index'], errors='coerce')

# Eliminar filas con valores NaN que hayan sido generados por la conversión
train_df = train_df.dropna(subset=['class index'])
```

```
# Convertir los valores de 'class index' a enteros
train_df['class index'] = train_df['class index'].astype(int)
```

```
/tmp/ipykernel_30/1371604693.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  train_df['class index'] = train_df['class index'].astype(int)
```

Aquí cargamos las etiquetas de `'classes.txt'`, se asigna el nombre de cada clase a la columna `'class index'` del DataFrame `train_df`, ajustamos los índices y añadimos una nueva columna `'class'` con los nombres de las clases correspondientes.

In [15]:
```
# Se lee el archivo 'classes.txt' y se dividen las líneas para obtener una lista de etiquetas (una para cada clase)
labels = open('/kaggle/input/classes/classes.txt').read().splitlines()

# Asignar los nombres de las clases a las filas del DataFrame
# Se mapea cada valor de 'class index' (que es un índice) al nombre de la clase correspondiente en 'labels'
# Dado que los índices en el archivo son 1-based (empiezan en 1), se resta 1 al índice para hacer el mapeo correcto
classes = train_df['class index'].map(lambda i: labels[i-1])

# Insertar una nueva columna 'class' en la posición 1 del DataFrame que contiene los nombres de las clases correspor
train_df.insert(1, 'class', classes)

# Mostrar el DataFrame
train_df
```

Out[15]:

	class index	class	title	description
71788	3	Business	BBC set for major shake-up, claims newspaper	London - The British Broadcasting Corporation,...
67218	4	Sci/Tech	Taking Microsoft for a spin?	The software juggernaut that conquered the des...
54066	3	Business	September sales at Target stores beat retail a...	MINNEAPOLIS - While other retailers struggled ...
7168	4	Sci/Tech	Macromedia launches Flex Builder	Macromedia this week will ship Flex Builder, w...
29618	1	World	Rocket lands near Afghan school as President K...	AFP - A rocket landed near a school in southea...

...	...	...	...	...
59228	4	Sci/Tech	Technical Problems Subside at PayPal	Most members of the online payment service Pay...
61417	3	Business	Shoppers Spring Back to Life in September	Shoppers got their buying groove back last mon...
20703	3	Business	UPDATE 1-Yellow Roadway raises 3rd-qtr profit ...	Yellow Roadway Corp. (YELL.O: Quote, Profile, ...
40626	3	Business	Next to digital IDs, passwords look lame	How big is your key ring? There are the house ...
25059	2	Sports	Prime-time Eagles	They opened their season Sept. 2 in the smalle...

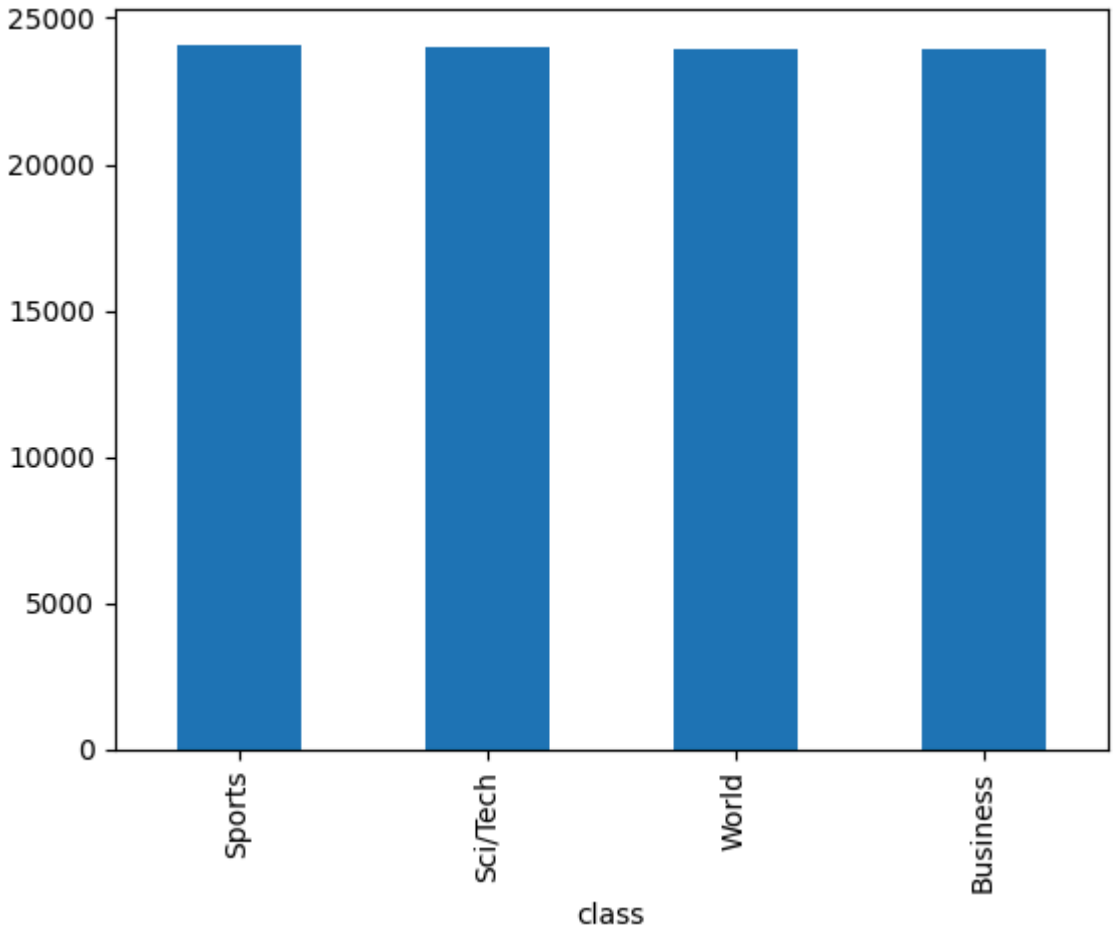
96000 rows × 4 columns

Let's inspect how balanced our examples are by using a bar plot.

In [16]:
```
# Contar el número de ejemplos en cada clase y crear un gráfico de barras
pd.value_counts(train_df['class']).plot.bar()
```

```
/tmp/ipykernel_30/1245903889.py:1: FutureWarning: pandas.value_counts is deprecated and will be removed in a future version. Use pd.Series(obj).value_counts() instead.
  pd.value_counts(train_df['class']).plot.bar()
```

Out[16]:
```
<Axes: xlabel='class'>
```



The classes are evenly distributed. That's great!

However, the text contains some spurious backslashes in some parts of the text. They are meant to represent newlines in the original text. An example can be seen below, between the words "dwindling" and "band".

Lo siguiente nos permitirá inspeccionar la descripción que contiene los backslashes.

In [17]:
```
# Imprimir la descripción de la fila con índice 1 para visualizar el texto
print(train_df.loc[1, 'description'])
```

```
Reuters - Short-sellers, Wall Street's dwindling\band of ultra-cynics, are seeing green again.
```

We will replace the backslashes with spaces on the whole column using pandas replace method.

En esta sección se reemplazan las barras invertidas (backslashes) por espacios en la columna combinada de texto (que incluye título y descripción).

In [18]:
```
# Convertir el título a minúsculas
title = train_df['title'].str.lower()

# Convertir la descripción a minúsculas
descr = train_df['description'].str.lower()

# Concatenar título y descripción en una nueva columna de texto
text = title + " " + descr

# Reemplazar cualquier backslash '\' por un espacio en la columna 'text'
train_df['text'] = text.str.replace('\\', ' ', regex=False)

# Mostrar el DataFrame
train_df
```

```
/tmp/ipykernel_30/889621432.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df['text'] = text.str.replace('\\', ' ', regex=False)
```

Out[18]:

	class index	class	title	description	text
71788	3	Business	BBC set for major shake-up, claims newspaper	London - The British Broadcasting Corporation,...	bbc set for major shake-up, claims newspaper l...
67218	4	Sci/Tech	Taking Microsoft for a spin?	The software juggernaut that conquered the des...	taking microsoft for a spin? the software jugg...

54066	3	Business	September sales at Target stores beat retail a...	MINNEAPOLIS - While other retailers struggled ...	september sales at target stores beat retail a...
7168	4	Sci/Tech	Macromedia launches Flex Builder	Macromedia this week will ship Flex Builder, w...	macromedia launches flex builder macromedia th...
29618	1	World	Rocket lands near Afghan school as President K...	AFP - A rocket landed near a school in southea...	rocket lands near afghan school as president k...
...	...	...	...	...	...
59228	4	Sci/Tech	Technical Problems Subside at PayPal	Most members of the online payment service Pay...	technical problems subside at paypal most memb...
61417	3	Business	Shoppers Spring Back to Life in September	Shoppers got their buying groove back last mon...	shoppers spring back to life in september shop...
20703	3	Business	UPDATE 1-Yellow Roadway raises 3rd-qtr profit ...	Yellow Roadway Corp. (YELL.O: Quote, Profile, ...	update 1-yellow roadway raises 3rd-qtr profit ...
40626	3	Business	Next to digital IDs, passwords look lame	How big is your key ring? There are the house ...	next to digital ids, passwords look lame how b...
25059	2	Sports	Prime-time Eagles	They opened their season Sept. 2 in the smalle...	prime-time eagles they opened their season sep...

96000 rows × 5 columns

Now we will proceed to tokenize the title and description columns using NLTK's word_tokenize(). We will add a new column to our dataframe with the list of tokens.

Utilizamos la función `word_tokenize()` de NLTK para dividir el texto en palabras individuales (tokens) en cada fila de la columna `'text'` del DataFrame `train_df`. El resultado de la tokenización se almacena en una nueva columna llamada `'tokens'`.

In [19]:
```
# Importar la función de tokenización de palabras de NLTK
from nltk.tokenize import word_tokenize
```

```
# Tokenizar el texto de la columna 'text' usando word_tokenize() y agregarlo a una nueva columna 'tokens'
# Se usa progress_map() para mostrar una barra de progreso mientras se procesan los datos
train_df['tokens'] = train_df['text'].progress_map(word_tokenize)

# Mostrar el DataFrame
train_df
```

```
  0%|          | 0/96000 [00:00<?, ?it/s]
```

```
/tmp/ipykernel_30/1907274149.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df['tokens'] = train_df['text'].progress_map(word_tokenize)
```

Out[19]:

	class index	class	title	description	text	tokens
71788	3	Business	BBC set for major shake-up, claims newspaper	London - The British Broadcasting Corporation,...	bbc set for major shake-up, claims newspaper l...	[bbc, set, for, major, shake-up, ,, claims, ne...
67218	4	Sci/Tech	Taking Microsoft for a spin?	The software juggernaut that conquered the des...	taking microsoft for a spin? the software jugg...	[taking, microsoft, for, a, spin, ?, the, soft...
54066	3	Business	September sales at Target stores beat retail a...	MINNEAPOLIS - While other retailers struggled ...	september sales at target stores beat retail a...	[september, sales, at, target, stores, beat, r...
7168	4	Sci/Tech	Macromedia launches Flex Builder	Macromedia this week will ship Flex Builder, w...	macromedia launches flex builder macromedia th...	[macromedia, launches, flex, builder, macromed...
29618	1	World	Rocket lands near Afghan school as President K...	AFP - A rocket landed near a school in southea...	rocket lands near afghan school as president k...	[rocket, lands, near, afghan, school, as, pres...
...	...	...	...	...	...	...
59228	4	Sci/Tech	Technical Problems Subside at PayPal	Most members of the online payment service Pay...	technical problems subside at paypal most memb...	[technical, problems, subside, at, paypal, mos...

61417	3	Business	Shoppers Spring Back to Life in September	Shoppers got their buying groove back last mon...	shoppers spring back to life in september shop...	[shoppers, spring, back, to, life, in, septemb...
20703	3	Business	UPDATE 1-Yellow Roadway raises 3rd-qtr profit ...	Yellow Roadway Corp. (YELL.O: Quote, Profile, ...	update 1-yellow roadway raises 3rd-qtr profit ...	[update, 1-yellow, roadway, raises, 3rd-qtr, p...
40626	3	Business	Next to digital IDs, passwords look lame	How big is your key ring? There are the house ...	next to digital ids, passwords look lame how b...	[next, to, digital, ids, ,, passwords, look, l...
25059	2	Sports	Prime-time Eagles	They opened their season Sept. 2 in the smalle...	prime-time eagles they opened their season sep...	[prime-time, eagles, they, opened, their, seas...

96000 rows × 6 columns

Now we will create a vocabulary from the training data. We will only keep the terms that repeat beyond some threshold established below.

Se crea un vocabulario de palabras que aparecen más de 10 veces en el conjunto de entrenamiento, lo que ayuda a reducir la complejidad del modelo y a ignorar palabras poco frecuentes o ruido en los datos.

In [20]:
```
# Definir un umbral para conservar solo las palabras que aparecen más que el threshold
threshold = 10

# 'explode' convierte listas en filas individuales, luego 'value_counts' cuenta cuántas veces aparece cada token en
tokens = train_df['tokens'].explode().value_counts()

# Filtrar los tokens que aparecen más veces que el threshold establecido
tokens = tokens[tokens > threshold]

# Crear una lista con los tokens que pasan el umbral, añadiendo '[UNK]' para palabras desconocidas
id_to_token = ['[UNK]'] + tokens.index.tolist()  # '[UNK]' es para términos desconocidos (out-of-vocabulary)

# Crear un diccionario que mapea cada token a un ID (índice) en el vocabulario
token_to_id = {w: i for i, w in enumerate(id_to_token)}

# Calcular el tamaño del vocabulario
```

```
vocabulary_size = len(id_to_token)

# Imprimir el tamaño del vocabulario generado
print(f'vocabulary size: {vocabulary_size:,}')
```

```
vocabulary size: 17,436
```

Se crea un diccionario vector que usa enteros con valor inicial 0 por defecto, útil para contar la frecuencia de tokens. Para cada token, se obtiene su índice del diccionario token_to_id. Si el token es desconocido, se le asigna el índice de [UNK] (por defecto, 0). Finalmente, la columna train_df['features'] almacena el vector de frecuencias generado para cada fila, aplicando make_feature_vector() sobre cada lista de tokens con progress_map() para ver el avance.

In [21]:
```
# Importar defaultdict para crear un diccionario con valores por defecto
from collections import defaultdict

# Función para crear el vector de características
def make_feature_vector(tokens, unk_id=0):
    # Crear un diccionario con valores por defecto de tipo entero (cuenta de frecuencias)
    vector = defaultdict(int)

    # Iterar sobre cada token en la lista de tokens
    for t in tokens:
        # Obtener el índice del token del diccionario 'token_to_id'; si no está, usar 'unk_id'
        i = token_to_id.get(t, unk_id)

        # Incrementar el conteo de la posición correspondiente al índice del token
        vector[i] += 1

    # Retornar el diccionario como vector de características
    return vector

# Aplicar 'make_feature_vector' a cada fila de 'tokens' y almacenar el resultado en una nueva columna 'features'
train_df['features'] = train_df['tokens'].progress_map(make_feature_vector)

# Mostrar el DataFrame
train_df
```

```
  0%|          | 0/96000 [00:00<?, ?it/s]
/tmp/ipykernel_30/4055419782.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df['features'] = train_df['tokens'].progress_map(make_feature_vector)
```

Out[21]:

	class index	class	title	description	text	tokens	features
71788	3	Business	BBC set for major shake-up, claims newspaper	London - The British Broadcasting Corporation,...	bbc set for major shake-up, claims newspaper l...	[bbc, set, for, major, shake-up, ,, claims, ne...	{2729: 1, 168: 1, 11: 1, 204: 1, 7015: 2, 2: 5...
67218	4	Sci/Tech	Taking Microsoft for a spin?	The software juggernaut that conquered the des...	taking microsoft for a spin? the software jugg...	[taking, microsoft, for, a, spin, ?, the, soft...	{612: 1, 84: 1, 11: 1, 5: 1, 4586: 1, 88: 1, 1...
54066	3	Business	September sales at Target stores beat retail a...	MINNEAPOLIS - While other retailers struggled ...	september sales at target stores beat retail a...	[september, sales, at, target, stores, beat, r...	{446: 1, 131: 2, 22: 1, 782: 2, 599: 1, 377: 1...
7168	4	Sci/Tech	Macromedia launches Flex Builder	Macromedia this week will ship Flex Builder, w...	macromedia launches flex builder macromedia th...	[macromedia, launches, flex, builder, macromed...	{5419: 2, 965: 1, 8376: 3, 7550: 2, 59: 1, 93:...
29618	1	World	Rocket lands near Afghan school as President K...	AFP - A rocket landed near a school in southea...	rocket lands near afghan school as president k...	[rocket, lands, near, afghan, school, as, pres...	{1129: 2, 3801: 1, 365: 2, 704: 1, 535: 2, 21:...
...	...	...	...	...	...	...	...
59228	4	Sci/Tech	Technical Problems Subside at PayPal	Most members of the online payment service Pay...	technical problems subside at paypal most memb...	[technical, problems, subside, at, paypal, mos...	{2445: 1, 911: 1, 0: 1, 22: 1, 4009: 2, 147: 1...
61417	3	Business	Shoppers Spring Back to Life in September	Shoppers got their buying groove back last	shoppers spring back to life in september	[shoppers, spring, back, to, life, in,	{2762: 2, 2649: 1, 119: 2, 4: 1, 486:

				mon...	shop...	septemb...	1, 7: 1,...
20703	3	Business	UPDATE 1-Yellow Roadway raises 3rd-qtr profit ...	Yellow Roadway Corp. (YELL.O: Quote, Profile, ...	update 1-yellow roadway raises 3rd-qtr profit ...	[update, 1-yellow, roadway, raises, 3rd-qtr, p...	{347: 1, 0: 2, 12962: 2, 1453: 1, 11057: 1, 16...
40626	3	Business	Next to digital IDs, passwords look lame	How big is your key ring? There are the house ...	next to digital ids, passwords look lame how b...	[next, to, digital, ids, ,, passwords, look, l...	{118: 1, 4: 3, 449: 1, 0: 4, 2: 5, 6026: 1, 60...
25059	2	Sports	Prime-time Eagles	They opened their season Sept. 2 in the smalle...	prime-time eagles they opened their season sep...	[prime-time, eagles, they, opened, their, seas...	{10794: 1, 1360: 1, 74: 1, 1214: 1, 47: 1, 126...

96000 rows × 7 columns

La columna features contiene, para cada fila, un diccionario que representa un vector de características con las frecuencias de cada token en esa fila. Este vector se puede utilizar para tareas de modelado de texto, como clasificación o análisis de similitud, representando cada texto como un conjunto de frecuencias de tokens.

El siguiente paso es convertir los vectores de características dispersos en vectores densos y luego transformarlos a tensores de PyTorch.

In [22]:
```python
# Definir una función para convertir vectores de características dispersos en vectores densos
def make_dense(feats):
    # Crear un arreglo de ceros con tamaño igual al vocabulario
    x = np.zeros(vocabulary_size)

    # Asignar los valores de los tokens en 'feats' a sus respectivas posiciones en el arreglo denso
    for k, v in feats.items():
        x[k] = v

    # Retornar el vector denso
    return x

# Aplicar la función make_dense a cada vector de características en 'features' y convertirlo en un array de numpy
X_train = np.stack(train_df['features'].progress_map(make_dense))
```

```
# Convertir la columna 'class index' a un array de etiquetas, restando 1 para hacerla 0-based
y_train = train_df['class index'].to_numpy() - 1

# Convertir X_train y y_train a tensores de PyTorch para su uso en modelos
X_train = torch.tensor(X_train, dtype=torch.float32)
y_train = torch.tensor(y_train)
```

```
  0%|          | 0/96000 [00:00<?, ?it/s]
```

La función make_dense toma cada vector disperso de características (un diccionario de índices y frecuencias) y lo convierte en un arreglo denso de tamaño igual al vocabulario, con las frecuencias de los tokens ubicadas en sus respectivas posiciones. Luego, X_train se crea al aplicar make_dense a cada fila en la columna features de train_df, consolidando los vectores resultantes en una matriz densa de numpy. Las etiquetas y_train se convierten a una matriz de numpy y se ajustan a un formato de 0-based restando 1. Finalmente, X_train y y_train se convierten en tensores de PyTorch, completando así la preparación de los datos para el entrenamiento en un modelo de PyTorch.

En la siguiente parte, se implementa el entrenamiento de un modelo de clasificación de texto lineal en PyTorch utilizando descenso de gradiente estocástico (SGD) y pérdida de entropía cruzada.

In [23]:
```
from torch import nn  # Importar módulos de redes neuronales de PyTorch
from torch import optim  # Importar optimizadores de PyTorch

# Definición de hiperparámetros
lr = 1.0  # Tasa de aprendizaje
n_epochs = 5  # Número de épocas de entrenamiento
n_examples = X_train.shape[0]  # Número de ejemplos en el conjunto de entrenamiento
n_feats = X_train.shape[1]  # Número de características (dimensión de entrada)
n_classes = len(labels)  # Número de clases en el conjunto de datos

# Inicializar el modelo, la función de pérdida y el optimizador
model = nn.Linear(n_feats, n_classes).to(device)  # Modelo lineal con salida de tamaño igual al número de clases
loss_func = nn.CrossEntropyLoss()  # Función de pérdida de entropía cruzada para clasificación multiclase
optimizer = optim.SGD(model.parameters(), lr=lr)  # Optimizador SGD con la tasa de aprendizaje especificada

# Entrenar el modelo
indices = np.arange(n_examples)  # Crear un array de índices para realizar el muestreo aleatorio en cada época
for epoch in range(n_epochs):  # Iterar sobre cada época
    np.random.shuffle(indices)  # Barajar los índices para garantizar la aleatoriedad en cada época
    for i in tqdm(indices, desc=f'epoch {epoch+1}'):  # Iterar sobre cada ejemplo en la época
        # Reiniciar los gradientes del modelo
```

```
        model.zero_grad()

        # Enviar el ejemplo y su etiqueta al dispositivo adecuado (GPU o CPU)
        x = X_train[i].unsqueeze(0).to(device)  # Redimensionar x para que sea de tamaño batch (1, n_feats) y enviar
        y_true = y_train[i].unsqueeze(0).to(device)  # Redimensionar y_true y enviarlo al dispositivo

        # Predecir las probabilidades de clase
        y_pred = model(x)

        # Calcular la pérdida entre las predicciones y la etiqueta verdadera
        loss = loss_func(y_pred, y_true)

        # Realizar la retropropagación para calcular los gradientes
        loss.backward()

        # Actualizar los parámetros del modelo utilizando el optimizador
        optimizer.step()
```

```
epoch 1:   0%|          | 0/96000 [00:00<?, ?it/s]
epoch 2:   0%|          | 0/96000 [00:00<?, ?it/s]
epoch 3:   0%|          | 0/96000 [00:00<?, ?it/s]
epoch 4:   0%|          | 0/96000 [00:00<?, ?it/s]
epoch 5:   0%|          | 0/96000 [00:00<?, ?it/s]
```

Primero, se definieron los hiperparámetros, como la tasa de aprendizaje, el número de épocas, y las dimensiones de los datos y clases. Luego, se inicializa un modelo lineal (`nn.Linear`), junto con la función de pérdida `CrossEntropyLoss()` y el optimizador SGD. Durante el entrenamiento, se itera por épocas, reorganizando aleatoriamente los índices para el muestreo estocástico. En cada iteración, las características y etiquetas se envían al dispositivo (GPU o CPU), se obtiene la predicción, se calcula la pérdida y se realiza la retropropagación para actualizar los parámetros del modelo. Este ciclo ajusta el modelo para minimizar la pérdida en los datos de entrenamiento.

Next, we evaluate on the test dataset

Para evaluar en el dataset de prueba, cargamos el archivo de prueba y seleccionamos una muestra aleatoria del 70% de los datos. Se asignan nombres descriptivos a las columnas para facilitar su uso y, luego, convierte la columna de índice de clase a valores numéricos, eliminando filas con valores no válidos y ajustándola a enteros. En el preprocesamiento de texto, combina y estandariza el título y la descripción en minúsculas, eliminando caracteres no deseados como `\` . A continuación, tokeniza el texto y crea un vector de

características denso para cada fila. Finalmente, convierte los datos de prueba (`X_test` y `y_test`) a tensores de PyTorch para ser evaluados en el modelo entrenado.

In [31]:
```
# Cargar el dataset de prueba y tomar una muestra aleatoria del 70%
test_df = pd.read_csv('/kaggle/input/ag-news-classification-dataset/test.csv', header=None)
test_df = test_df.sample(frac=0.7, random_state=42)  # Utiliza random_state para asegurar reproducibilidad

# Asignar nombres de columnas para facilitar la interpretación
test_df.columns = ['class index', 'title', 'description']

# Convertir la columna 'class index' a tipo numérico, en caso de que contenga valores no numéricos
test_df['class index'] = pd.to_numeric(test_df['class index'], errors='coerce')

# Eliminar filas con valores NaN en 'class index' generados por la conversión
test_df = test_df.dropna(subset=['class index'])

# Convertir los valores de 'class index' a enteros para que sean índices enteros
test_df['class index'] = test_df['class index'].astype(int)

# Preprocesar el texto: convertir a minúsculas y combinar título y descripción en una sola columna
test_df['text'] = test_df['title'].str.lower() + " " + test_df['description'].str.lower()

# Reemplazar cualquier barra invertida '\' en el texto por un espacio
test_df['text'] = test_df['text'].str.replace('\\', ' ', regex=False)

# Tokenizar el texto de cada fila en 'text' y almacenar los tokens en una nueva columna 'tokens'
test_df['tokens'] = test_df['text'].progress_map(word_tokenize)

# Aplicar la función 'make_feature_vector' para convertir los tokens en un vector de características disperso
test_df['features'] = test_df['tokens'].progress_map(make_feature_vector)

# Convertir los vectores dispersos a vectores densos y apilarlos en un array de numpy
X_test = np.stack(test_df['features'].progress_map(make_dense))

# Convertir las etiquetas de clase a un array numpy y restar 1 para hacerlas cero-basadas
y_test = test_df['class index'].to_numpy() - 1

# Convertir los arrays de numpy en tensores de PyTorch
X_test = torch.tensor(X_test, dtype=torch.float32)  # Características como float32
y_test = torch.tensor(y_test)  # Etiquetas como enteros
```

```
# Imprimir las dimensiones de X_test y y_test para verificar la forma de los datos
print(X_test.shape, y_test.shape)
```

```
  0%|          | 0/5320 [00:00<?, ?it/s]
  0%|          | 0/5320 [00:00<?, ?it/s]
  0%|          | 0/5320 [00:00<?, ?it/s]
torch.Size([5320, 17436]) torch.Size([5320])
```

Este código permite evaluar el desempeño del modelo en términos de precisión, exhaustividad y F1-score, proporcionando una visión del rendimiento por clase en el conjunto de prueba

In [34]:
```
from sklearn.metrics import classification_report  # Importar el reporte de clasificación de sklearn

# Poner el modelo en modo de evaluación
model.eval()

# Desactivar el cálculo de gradientes (ahorra memoria y mejora rendimiento)
with torch.no_grad():
    # Enviar los datos de prueba al dispositivo adecuado (GPU o CPU)
    X_test = X_test.to(device)

    # Realizar predicciones sobre el conjunto de prueba
    y_pred = torch.argmax(model(X_test), dim=1)  # Obtener las etiquetas predichas con mayor probabilidad

    # Convertir las predicciones a un array numpy y moverlas a la CPU
    y_pred = y_pred.cpu().numpy()

    # Imprimir el reporte de clasificación que incluye métricas de precisión, exhaustividad y F1-score
    print(classification_report(y_test, y_pred, target_names=labels))
```

```
              precision    recall  f1-score   support

       World       0.96      0.77      0.86      1344
      Sports       0.91      0.98      0.94      1325
    Business       0.74      0.90      0.81      1325
    Sci/Tech       0.87      0.79      0.83      1326

    accuracy                           0.86      5320
   macro avg       0.87      0.86      0.86      5320
```

```
weighted avg       0.87      0.86      0.86      5320
```

El modelo muestra un buen desempeño general con un 86% de exactitud, destacándose en la clase **Sports** con altos valores de precisión (91%) y exhaustividad (98%), mientras que su rendimiento es menos efectivo en **Business** (precisión de 74% y F1 de 0.81), indicando cierta confusión al clasificar otras clases como "Business". La clase **World** tiene una precisión alta (96%) pero menor exhaustividad (77%), sugiriendo que algunos ejemplos de esta clase se clasifican incorrectamente en otras. En general, los promedios macro y ponderado (86% en precisión, exhaustividad y F1) reflejan un rendimiento consistente del modelo a pesar de algunas diferencias entre las clases.

Conclusión: El pipeline del código aborda la preparación, entrenamiento y evaluación de un modelo de clasificación de texto multiclase utilizando PyTorch y embeddings de palabras. Primero, realiza una limpieza y normalización del texto, incluyendo la combinación de títulos y descripciones y la eliminación de caracteres no deseados. Luego, aplica tokenización y construye un vocabulario para convertir el texto en vectores de características, necesarios para el modelo. Posteriormente, el modelo lineal de PyTorch se entrena en este conjunto de datos usando descenso de gradiente estocástico (SGD) y se ajusta para minimizar la pérdida de entropía cruzada. Finalmente, el modelo se evalúa en un conjunto de prueba, y el rendimiento se mide en términos de precisión, exhaustividad y F1-score, mostrando un buen rendimiento general con variaciones según la clase. Este enfoque permite que el modelo clasifique noticias en categorías como "World", "Sports", "Business" y "Sci/Tech" con una precisión global del 86%, aunque con algunas áreas para mejorar en ciertas clases.