notebookf5bc456b00-1

October 31, 2024

1 Multiclass Text Classification with

2 Logistic Regression Implemented with PyTorch and CE Loss

María Fernanda Pérez Ruiz A01742102

First, we will do some initialization.

En este primer chunk de código preparamos el entorno para el pipeline de procesamiento y entrenamiento en NLP, primeramente importamos la librerias claves necesarias: random, torch, numpy (np) y pandas, van a ser necesarias para poder trabajar y manipular los datos y poder entrenar el modelo en PyTorch, se incluye tambien tqdm que nos va a ayudar a facilitar el seguimiento visual del progreso en las operaciones de procesamiento con pandas.

Despues en el código se verifica si es que hay un GPU disponible paara el entrenamiento, si es que encuentra una le asigna el dispositivo cuda. Despues establecemos nuestra random seed en 1234 para que al volver correr el ejercicio obtengamos los mismos resultados.

Finalmente se imprimen el dispositivo seleccionado y la semilla que ya preestablecimos para confirmar la configuración del entorno de entrenamiento.

[1]:
```
#fer
import random
import torch
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm

# enable tqdm in pandas
tqdm.pandas()

# set to True to use the gpu (if there is one available)
use_gpu = True

# select device
device = torch.device('cuda' if use_gpu and torch.cuda.is_available() else␣
 ↪'cpu')
print(f'device: {device.type}')

# random seed
```

```
seed = 1234

# set random seed
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
```

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.

En este chunk de código cargamos con pandas el conjunto de datos de entrenamiento para la clasificación. El primer archivo que lee es el de train al cual le asignamos el nombre a las columnas: 'class index', 'title', 'description'. Se toma una muestra aleatoria de 70% de los datos para usarlo para el entrenamiento (frac=0.7) y establecemos el random_state en 42 (random_state=42) para que sea repoducible el ejercicio.

[2]:
```
train_df = pd.read_csv('/kaggle/input/train-csv/train.csv', header=None)
train_df.columns = ['class index', 'title', 'description']
train_df

train_df = train_df.sample(frac=0.7,random_state=42)
```

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.

En este chunk de código se cargan las etiquetas de las clases tomadas del archivo classes.txt, el cual son los nombres de las categorías de noticias correspondientes a cada índice de clase , despues el dataframe train_df se reindexa para reiniciar los índices después de haber tomado una muestra en el paso anterior. La columna 'class index' la vamos a hacer entero para asegurarnos del formato. Con la función de map vamos a asociar cada índice de clase con su respectiva etiqueta textual desde labels, se ocupan ajustar los índices porque los valores en el archivos empiezan desde 1. Por ultimo insertamos la nueva columna llamada 'class' que incluye los nombres de las clases.

[3]:
```
labels = open('/kaggle/input/classes-txt/classes.txt').read().splitlines()
train_df = train_df.drop(0).reset_index(drop=True)
train_df['class index'] = train_df['class index'].astype(int)
train_df['class index'] = train_df['class index'].astype(int)
classes = train_df['class index'].map(lambda i: labels[i-1])
train_df.insert(1, 'class', classes)
train_df
```

[3]:

	class index	class	title	description
0	3	Business	BBC set for major shake-up, claims newspaper	London - The British Broadcasting Corporation,…
1	4	Sci/Tech	Taking Microsoft for a spin?	The software juggernaut that conquered the des…
2	3	Business	September sales at Target stores beat retail a…	MINNEAPOLIS - While other retailers struggled …
3	4	Sci/Tech	Macromedia launches Flex Builder	Macromedia this week will ship Flex Builder, w…
4	1	World	Rocket lands near Afghan school as President K…	AFP - A rocket landed near a school in southea…
…	…	…	…	…
83995	3	Business	Boston techies envision TV's on-demand future	At the end of a boisterous dinner, after sever…
83996	2	Sports	Levy is the glue that holds Wake together	We can talk all you want about Chris Paul, Jus…
83997	1	World	Three tested for bird flu discharged from Mala…	KUALA LUMPUR : Three Malaysians who fell ill i…
83998	4	Sci/Tech	Lexmark Recalls Laser Printers	Lexmark is recalling 39,400 laser printers bec…
83999	2	Sports	Hopkins leaves chink in the Golden Boy #39;s a…	On paper, it looked like a matchup of youth ag…

[84000 rows x 4 columns]

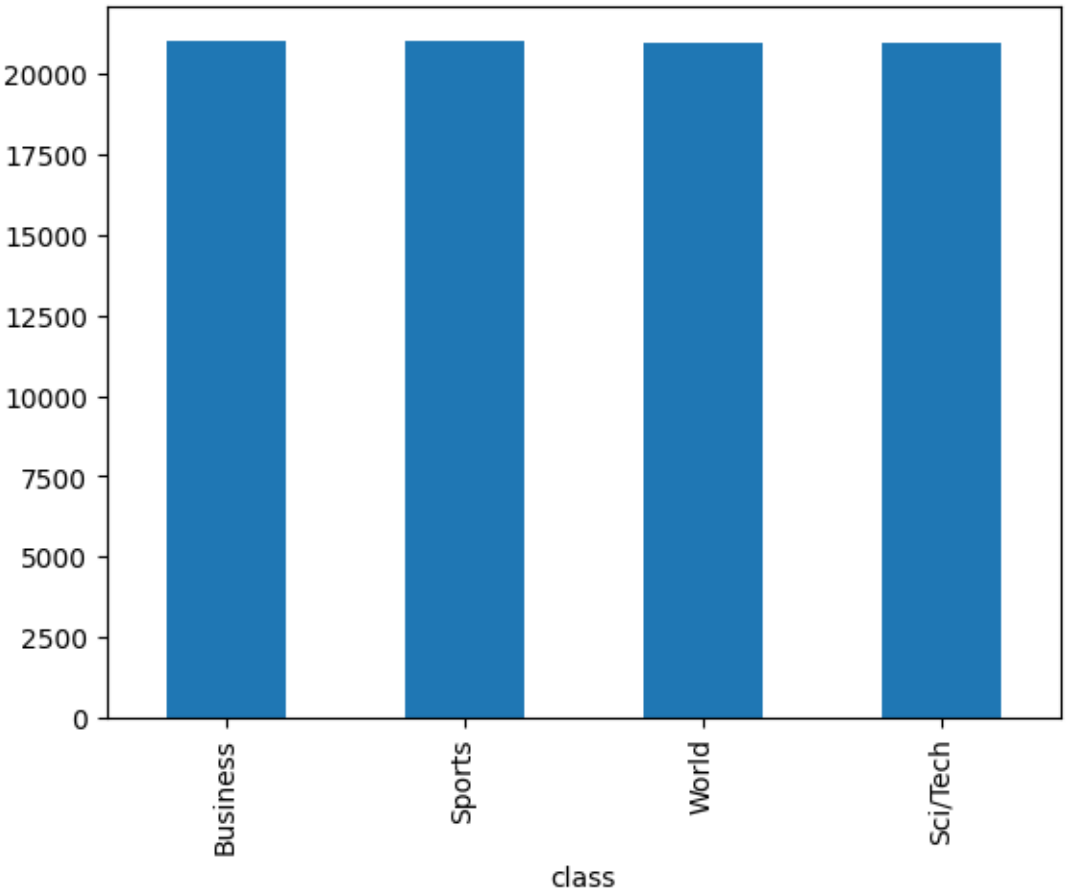
Let's inspect how balanced our examples are by using a bar plot.

En este chunk de código generamos una gráfica de barras para verificar si las clases se distribuyen de manera uniforme, porque cuando trabajamos con problemas de clasificación lo que no queremos es que haya un desequilibrio ya que no queremos que el modelo favorezca más una clase que otra, como vemos aqui no tenemos ese problema porque la grafica se ve equilibrada.

[4]:
```
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()
```

[4]: <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".

En este chunk de código inspeccionamos una de las descripciones en el conjunto de datos para identificar posibles problemas en el texto. Vemos por ejemplo el error del #39;, y estos caracteres pueden interferir con el procesamiento de texto posterior,por lo que se busca identificarlos para limpiarlos y que el texto no reciba errores ni ruido.

[5]:
```
print(train_df.loc[0, 'description'])
```

```
London - The British Broadcasting Corporation, the world #39;s biggest public
broadcaster, is to cut almost a quarter of its 28 000-strong workforce, in the
biggest shake-up in its 82-year history, The Times newspaper in London said on
Monday.
```

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

En este chunk de código se hace una limpieza, se convierten el titulo 'title' y la descripcion 'description' en minusculas y luego se unifican al juntarse en una misma y nueva columna llamada 'text'. Se reemplaza el caracter " por un espacio en blanco para la limpieza.

[6]:
```
title = train_df['title'].str.lower()
descr = train_df['description'].str.lower()
text = title + " " + descr
train_df['text'] = text.str.replace('\\', ' ', regex=False)
train_df
```

[6]:

	class index	class	\
0	3	Business	
1	4	Sci/Tech	
2	3	Business	
3	4	Sci/Tech	
4	1	World	
…	…	…	
83995	3	Business	
83996	2	Sports	
83997	1	World	
83998	4	Sci/Tech	
83999	2	Sports	

	title	\
0	BBC set for major shake-up, claims newspaper	
1	Taking Microsoft for a spin?	
2	September sales at Target stores beat retail a…	
3	Macromedia launches Flex Builder	
4	Rocket lands near Afghan school as President K…	
…	…	
83995	Boston techies envision TV's on-demand future	
83996	Levy is the glue that holds Wake together	
83997	Three tested for bird flu discharged from Mala…	
83998	Lexmark Recalls Laser Printers	
83999	Hopkins leaves chink in the Golden Boy #39;s a…	

	description	\
0	London - The British Broadcasting Corporation,…	
1	The software juggernaut that conquered the des…	

```
2      MINNEAPOLIS - While other retailers struggled …
3      Macromedia this week will ship Flex Builder, w…
4      AFP - A rocket landed near a school in southea…
…                                                    …
83995  At the end of a boisterous dinner, after sever…
83996  We can talk all you want about Chris Paul, Jus…
83997  KUALA LUMPUR : Three Malaysians who fell ill i…
83998  Lexmark is recalling 39,400 laser printers bec…
83999  On paper, it looked like a matchup of youth ag…

                                                   text
0      bbc set for major shake-up, claims newspaper l…
1      taking microsoft for a spin? the software jugg…
2      september sales at target stores beat retail a…
3      macromedia launches flex builder macromedia th…
4      rocket lands near afghan school as president k…
…                                                    …
83995  boston techies envision tv's on-demand future …
83996  levy is the glue that holds wake together we c…
83997  three tested for bird flu discharged from mala…
83998  lexmark recalls laser printers lexmark is reca…
83999  hopkins leaves chink in the golden boy #39;s a…

[84000 rows x 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.

En este chunk de código es donde se lleva a cabo el proceso de tokenización de texto, es importante que se haga adecuadamente ya que esta parte es fundamental en el preprocesamiento para modelos de NLP. Usamos la función word_tokenize de la libreria nltk se divide el texto de cada registro en palabras individuales (tokens) . word_tokenize va a crear una lista de tokens para cada fila en la columna text, la lsita se va a almacenar en una columna nueva llamda 'tokens'. Con la función progress_map vamos a poder visualizar el progreso que va llevando del proceso de tokenización para cada resgitro.

[7]:
```
from nltk.tokenize import word_tokenize

train_df['tokens'] = train_df['text'].progress_map(word_tokenize)
train_df
```

```
  0%|          | 0/84000 [00:00<?, ?it/s]
```

[7]:
```
       class index     class  \
0                3  Business
1                4  Sci/Tech
2                3  Business
3                4  Sci/Tech
```

```
4                1     World
…              …       …
83995            3  Business
83996            2    Sports
83997            1     World
83998            4  Sci/Tech
83999            2    Sports

                                                    title  \
0           BBC set for major shake-up, claims newspaper
1                             Taking Microsoft for a spin?
2      September sales at Target stores beat retail a…
3                        Macromedia launches Flex Builder
4      Rocket lands near Afghan school as President K…
…                                                      …
83995      Boston techies envision TV's on-demand future
83996           Levy is the glue that holds Wake together
83997  Three tested for bird flu discharged from Mala…
83998                       Lexmark Recalls Laser Printers
83999  Hopkins leaves chink in the Golden Boy #39;s a…

                                                description  \
0      London - The British Broadcasting Corporation,…
1      The software juggernaut that conquered the des…
2      MINNEAPOLIS - While other retailers struggled …
3      Macromedia this week will ship Flex Builder, w…
4      AFP - A rocket landed near a school in southea…
…                                                      …
83995  At the end of a boisterous dinner, after sever…
83996  We can talk all you want about Chris Paul, Jus…
83997  KUALA LUMPUR : Three Malaysians who fell ill i…
83998  Lexmark is recalling 39,400 laser printers bec…
83999  On paper, it looked like a matchup of youth ag…

                                                     text  \
0      bbc set for major shake-up, claims newspaper l…
1      taking microsoft for a spin? the software jugg…
2      september sales at target stores beat retail a…
3      macromedia launches flex builder macromedia th…
4      rocket lands near afghan school as president k…
…                                                      …
83995  boston techies envision tv's on-demand future …
83996  levy is the glue that holds wake together we c…
83997  three tested for bird flu discharged from mala…
83998  lexmark recalls laser printers lexmark is reca…
83999  hopkins leaves chink in the golden boy #39;s a…
```

```
                                                   tokens
0       [bbc, set, for, major, shake-up, ,, claims, ne…
1       [taking, microsoft, for, a, spin, ?, the, soft…
2       [september, sales, at, target, stores, beat, r…
3       [macromedia, launches, flex, builder, macromed…
4       [rocket, lands, near, afghan, school, as, pres…
…                                                     …
83995   [boston, techies, envision, tv, 's, on-demand,…
83996   [levy, is, the, glue, that, holds, wake, toget…
83997   [three, tested, for, bird, flu, discharged, fr…
83998   [lexmark, recalls, laser, printers, lexmark, i…
83999   [hopkins, leaves, chink, in, the, golden, boy,…

[84000 rows x 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.

Aquí vamos a crear un vocabulario con los datos de entrenamiento, solamente se van a conservar los términos que aparecen con una frecuencia mayor al establecido, o sea 10 . Con value_counts en la columna tokens se cuenta la frecuencia de cada palabra. Despues se filtran esos tokens que tienen una frecuencia mayor a la del umbral pre establecido. Creamos una lista de tokens única ' id_to_token' que va a incluir un marcador especial [UNK] para las palabras desconocidas , o sea las que no cumplen con el umbral, despues se crea un diccionario token_to_id que asigna un índice único a cada palabra en el vocabulario y por ultimo se imprime el numero del tamaño del vocabulario, que son el numero de palabras frecuente retenidas.

[8]:
```
threshold = 10
tokens = train_df['tokens'].explode().value_counts()
tokens = tokens[tokens > threshold]
id_to_token = ['[UNK]'] + tokens.index.tolist()
token_to_id = {w:i for i,w in enumerate(id_to_token)}
vocabulary_size = len(id_to_token)
print(f'vocabulary size: {vocabulary_size:,}')
```

```
vocabulary size: 16,197
```

Aqui construimos un vector de características para cada registro del conjunto de datos, es clave para el entrenamiento para modelos de este tipo, NLP. Definimos una fucnión llamada make_feature_vector, que toma una lista de tokens y devuelve un vector donde cada posición representa un token del vocabulario. Con defaultdict (diccionario) cada token en el texto se convierte a su índice correspondiente en el vocabulario (token_to_id) y aumenta el conteo en esa posición, para crear un vector de frecuencia de palabras. En dado caso que un token no esté en el vocabulario se le asigna el índice de [UNK]. La función que se creó se aplica a la columna tokens usando progress_map, se agrega una nueva columna de features con el vector de características para cada entrada en el DataFrame. Con este vector de características se va a representar cada texto en formato numérico que el mpdelo es capaz de interpretar y procesar duarnte esta fase de entrenamiento.

[9]:
```
from collections import defaultdict

def make_feature_vector(tokens, unk_id=0):
    vector = defaultdict(int)
    for t in tokens:
        i = token_to_id.get(t, unk_id)
        vector[i] += 1
    return vector

train_df['features'] = train_df['tokens'].progress_map(make_feature_vector)
train_df
```

```
  0%|          | 0/84000 [00:00<?, ?it/s]
```

[9]:

	class index	class	title	description
0	3	Business	BBC set for major shake-up, claims newspaper	London - The British Broadcasting Corporation,…
1	4	Sci/Tech	Taking Microsoft for a spin?	The software juggernaut that conquered the des…
2	3	Business	September sales at Target stores beat retail a…	MINNEAPOLIS - While other retailers struggled …
3	4	Sci/Tech	Macromedia launches Flex Builder	Macromedia this week will ship Flex Builder, w…
4	1	World	Rocket lands near Afghan school as President K…	AFP - A rocket landed near a school in southea…
…	…	…	…	…
83995	3	Business	Boston techies envision TV's on-demand future	
83996	2	Sports	Levy is the glue that holds Wake together	
83997	1	World	Three tested for bird flu discharged from Mala…	
83998	4	Sci/Tech	Lexmark Recalls Laser Printers	
83999	2	Sports	Hopkins leaves chink in the Golden Boy #39;s a…	

```
83995  At the end of a boisterous dinner, after sever…
83996  We can talk all you want about Chris Paul, Jus…
83997  KUALA LUMPUR : Three Malaysians who fell ill i…
83998  Lexmark is recalling 39,400 laser printers bec…
83999  On paper, it looked like a matchup of youth ag…

                                                   text  \
0      bbc set for major shake-up, claims newspaper l…
1      taking microsoft for a spin? the software jugg…
2      september sales at target stores beat retail a…
3      macromedia launches flex builder macromedia th…
4      rocket lands near afghan school as president k…
…                                                    …
83995  boston techies envision tv's on-demand future …
83996  levy is the glue that holds wake together we c…
83997  three tested for bird flu discharged from mala…
83998  lexmark recalls laser printers lexmark is reca…
83999  hopkins leaves chink in the golden boy #39;s a…

                                                 tokens  \
0      [bbc, set, for, major, shake-up, ,, claims, ne…
1      [taking, microsoft, for, a, spin, ?, the, soft…
2      [september, sales, at, target, stores, beat, r…
3      [macromedia, launches, flex, builder, macromed…
4      [rocket, lands, near, afghan, school, as, pres…
…                                                    …
83995  [boston, techies, envision, tv, 's, on-demand,…
83996  [levy, is, the, glue, that, holds, wake, toget…
83997  [three, tested, for, bird, flu, discharged, fr…
83998  [lexmark, recalls, laser, printers, lexmark, i…
83999  [hopkins, leaves, chink, in, the, golden, boy,…

                                                features
0      {2683: 1, 166: 1, 11: 1, 209: 1, 7243: 2, 2: 5…
1      {627: 1, 85: 1, 11: 1, 5: 1, 4395: 1, 88: 1, 1…
2      {438: 1, 129: 2, 22: 1, 791: 2, 596: 1, 376: 1…
3      {5264: 2, 969: 1, 7996: 3, 7247: 2, 59: 1, 93:…
4      {1142: 2, 3985: 1, 363: 2, 704: 1, 553: 2, 21:…
…                                                    …
83995  {332: 1, 14651: 1, 0: 4, 605: 1, 24: 1, 7686: …
83996  {10896: 1, 23: 1, 1: 2, 0: 1, 18: 1, 1713: 1, …
83997  {100: 2, 3427: 1, 11: 1, 1582: 2, 1264: 2, 0: …
83998  {5938: 3, 3883: 1, 5358: 2, 5175: 3, 23: 1, 62…
83999  {5312: 1, 1718: 1, 0: 2, 7: 1, 1: 1, 2488: 1, …

[84000 rows x 7 columns]
```

En este chunk de código el vector de características disperso lo convertimos en un vector denso y se prepara el conjunto de datos en formato tensor para su uso en PyTorch. Se crea la función make_dense que esta crea un vector de tomaño vocabulary_size (inicializado en ceros) y asigna los valores de frecuencia en las posiciones que le corresponde de acuerdo con los índices de los tokens. Despues la función (make_dense) se aplica a cada vector de características en la columna features, generando una matriz densa X_train con los vectores completos.

La variable y_train la vamos a obtener de la columna 'class index' y se ajustó el indice para que comience desde cero, compatible con PyTorch.

Por ultimo x_train y y_train se convierten a tensores de Pytorch, se tiene que especificar que x_train debe de ser tipo float (float32) ya que es el tipo de dato que comúnmente se usa para el entrenamiento en PyTorch. Los tensores x_train y y_train representan las entradas y etiquetas del conjunto de entrenamiento y ya se pueden utilizar en el modelo.

[10]:
```
def make_dense(feats):
    x = np.zeros(vocabulary_size)
    for k,v in feats.items():
        x[k] = v
    return x

X_train = np.stack(train_df['features'].progress_map(make_dense))
y_train = train_df['class index'].to_numpy() - 1

X_train = torch.tensor(X_train, dtype=torch.float32)
y_train = torch.tensor(y_train)
```

```
  0%|          | 0/84000 [00:00<?, ?it/s]
```

En este chunk de código se define, inicializa y se entrena un modelo de red neuronal simple para clasificación de texto usando PyTorch. Primeramente configuramos los hiperparámetros principales: la tasa de aprendizaje (lr), el número de épocas (n_epochs), el número de ejemplos, características y clases, que se obtienen de los datos de entrenamiento y el conjunto de etiquetas.

Despues se inicializa un modelo de capa lineal (nn.Linear), el cual toma como entrada el número de características (n_feats) y produce una salida de tamaño igual al número de clases (n_classes). Tambien tenemos que definir la función de pérdida (CrossEntropyLoss)

Y en el bucle de entrenamiento hacemos las iteraciones para cada epoch, para que el modelo sea más robusto se bajan los ínidces de los datos. El modelo despues va a predecir los puntajes de clase, o sea y_pred, se calcula la pérdida comparando y_pred con la etiqueta real (y_true), y se ejecuta la retropropagación de esta pérdida para ajustar los parámetros del modelo. Proceso que se repite para cada época, y ya que se completan todas las épocas y apartir de los datos de entrenamiento el modelo ajusta los parámetros.

[11]:
```
from torch import nn
from torch import optim

# hyperparameters
lr = 1.0
```

```
n_epochs = 5
n_examples = X_train.shape[0]
n_feats = X_train.shape[1]
n_classes = len(labels)

# initialize the model, loss function, optimizer, and data-loader
model = nn.Linear(n_feats, n_classes).to(device)
loss_func = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=lr)

# train the model
indices = np.arange(n_examples)
for epoch in range(n_epochs):
    np.random.shuffle(indices)
    for i in tqdm(indices, desc=f'epoch {epoch+1}'):
        # clear gradients
        model.zero_grad()
        # send datum to right device
        x = X_train[i].unsqueeze(0).to(device)
        y_true = y_train[i].unsqueeze(0).to(device)
        # predict label scores
        y_pred = model(x)
        # compute loss
        loss = loss_func(y_pred, y_true)
        # backpropagate
        loss.backward()
        # optimize model parameters
        optimizer.step()
```

```
epoch 1:   0%|          | 0/84000 [00:00<?, ?it/s]
epoch 2:   0%|          | 0/84000 [00:00<?, ?it/s]
epoch 3:   0%|          | 0/84000 [00:00<?, ?it/s]
epoch 4:   0%|          | 0/84000 [00:00<?, ?it/s]
epoch 5:   0%|          | 0/84000 [00:00<?, ?it/s]
```

Next, we evaluate on the test dataset

En este chunk de código se hace el mismo preprocesamiento que se hizo en el conjunto de etrenamiento pero ahora para el conjunto de prueba para que los dos tengan el mismo formato y estructura y no haya ruidos externis, se incluye la lectura del archivo test.csv y se le asignan los nombres a las columnas: 'class index', 'title', 'description' y ya despues creamos la columna text que es el conjunto del titulo y la descripcion (en minusculas). Despues se tokeniza el texto en palabras individuales y se genera el vector de características utilizando el vocabulario creado previamente. E igual que lo habiamos hecho anteriormente el índice de clase en test_df se reindexa y convierte a tipo numérico. Y ya por ultimo los vectores de 'features' con la función make_dense se convierten en una representación densa y de igual manera que en el conjunto de entrenamiento

tanto las características (X_test) como las etiquetas (y_test) se convierten a tensores de PyTorch para su uso en el modelo.

[12]:
```
# repeat all preprocessing done above, this time on the test set
test_df = pd.read_csv('/kaggle/input/test-csv/test.csv', header=None)
test_df.columns = ['class index', 'title', 'description']
test_df['text'] = test_df['title'].str.lower() + " " + test_df['description'].
 ↪str.lower()
test_df['text'] = test_df['text'].str.replace('\\', ' ', regex=False)
test_df['tokens'] = test_df['text'].progress_map(word_tokenize)
test_df['features'] = test_df['tokens'].progress_map(make_feature_vector)

test_df = test_df.drop(index=0).reset_index(drop=True)
test_df['class index'] = pd.to_numeric(test_df['class index'], errors='coerce')

X_test = np.stack(test_df['features'].progress_map(make_dense))
y_test = test_df['class index'].to_numpy() - 1
X_test = torch.tensor(X_test, dtype=torch.float32)
y_test = torch.tensor(y_test)
```

```
  0%|          | 0/7601 [00:00<?, ?it/s]
  0%|          | 0/7601 [00:00<?, ?it/s]
  0%|          | 0/7600 [00:00<?, ?it/s]
```

Los resultados muestran el desempeño del modelo en términos de precisión (exactitud de las predicciones correctas frente a todas las predicciones), recall (capacidad de detectar todas las instancias de una clase) y F1-score (media armónica entre precisión y recall). La métrica de accuracy representa el rendimiento global del modelo.

En este ultimo chunk de codigo es donde vamos a evaluar el renidmiento del modelo entrenado en el conjunto de prueba con métricas de clasificación. Primeramente estabecemos el modelo en modo de evaluación 'model.eval()', para desactivar ciertas funciones de entrenamiento, como el dropout y tener aseguradas predicciones consistentes. Despues en torch.no_grad() (que desactiva el cálculo de gradientes para optimizar el rendimiento) se pasa se pasa el tensor X_test al GPU y obtenemos la predicción del modelo con argmax (selecciona la clase con la mayor probabilidad para cada muestra). Las predicciones se convierten a un arreglo de NumPy y se comparan con las etiquetas reales (y_test) utilizando classification_report de scikit-learn (calcula las métricas de precisión, recall y F1-score para cada clase), con estas metricas se evalua la eficiencia del modelo en cada categoría, y ya los resultados van a mostrar el desempeño del modelo en términos de precisión (exactitud de las predicciones correctas frente a todas las predicciones), recall (capacidad de detectar todas las instancias de una clase) y F1-score (media armónica entre precisión y recall). La métrica de accuracy representa el rendimiento global del modelo.

Los resultados obtenidos del desempeño del modelo en cada clase:

Precision: El porcentaje de las predicciones de cada clase fue correcto. Para "World" el 93%. Recall: El porcentaje de los casos reales de cada clase fue identificado correctamente. Para "Business" 90%. F1-score: El balance entre precisión y recall, es útil para ver el rendimiento general en cada clase.

Support: La cantidad de ejemplos reales por clase en el conjunto de prueba. El modelo tiene una precisión y recall altos '89%'.

[13]:
```
from sklearn.metrics import classification_report

# set model to evaluation mode
model.eval()

# don't store gradients
with torch.no_grad():
    X_test = X_test.to(device)
    y_pred = torch.argmax(model(X_test), dim=1)
    y_pred = y_pred.cpu().numpy()
    print(classification_report(y_test, y_pred, target_names=labels))
```

```
              precision    recall  f1-score   support

       World       0.93      0.86      0.89      1900
      Sports       0.95      0.96      0.96      1900
    Business       0.79      0.90      0.84      1900
    Sci/Tech       0.88      0.82      0.85      1900

    accuracy                           0.89      7600
   macro avg       0.89      0.89      0.89      7600
weighted avg       0.89      0.89      0.89      7600
```