regresion-logistica-multiclase-pytorch

October 29, 2024

1 Multiclass Text Classification with

2 Logistic Regression Implemented with PyTorch and CE Loss

Realizado por: Rodolfo Jesús Cruz Rebollar

Matrícula: A01368326

Grupo: 101

First, we will do some initialization.

[1]:
```
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: cpu
random seed: 1234
```

En la celda de código anterior, se habilita el uso del módulo tqdm en pandas, además de que también se establece el dispositivo a emplear para almacenar la información que se procece durante todo el proceso de análisis y modelación de los datos, por lo que si existe un dispositivo gpu disponible, éste será el que se utilizará, en caso contrario, se usará un dispositivo cpu. Además se define una semilla aleatoria para todos los procesos posteriores.

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.

[2]:
```
train_df = pd.read_csv('/kaggle/input/ag-news-classification-dataset/train.
 ↪csv', header=None)
train_df.columns = ['class_index', 'title', 'description']
train_df = train_df.iloc[1:, :]
train_df
```

[2]:
```
        class_index                                              title  \
1                 3  Wall St. Bears Claw Back Into the Black (Reuters)
2                 3  Carlyle Looks Toward Commercial Aerospace (Reu…
3                 3    Oil and Economy Cloud Stocks' Outlook (Reuters)
4                 3  Iraq Halts Oil Exports from Main Southern Pipe…
5                 3  Oil prices soar to all-time record, posing new…
…               …                                                …
119996            1  Pakistan's Musharraf Says Won't Quit as Army C…
119997            2                  Renteria signing a top-shelf deal
119998            2                    Saban not going to Dolphins yet
119999            2                                Today's NFL games
120000            2                     Nets get Carter from Raptors

                                              description
1       Reuters - Short-sellers, Wall Street's dwindli…
2       Reuters - Private investment firm Carlyle Grou…
3       Reuters - Soaring crude prices plus worries\ab…
4       Reuters - Authorities have halted oil export\f…
5       AFP - Tearaway world oil prices, toppling reco…
…                                                     …
119996   KARACHI (Reuters) - Pakistani President Perve…
119997  Red Sox general manager Theo Epstein acknowled…
119998  The Miami Dolphins will put their courtship of…
119999  PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: …
120000  INDIANAPOLIS -- All-Star Vince Carter was trad…

[120000 rows x 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.

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

[3]:

	class_index	class	title	description
1	3	Business	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli…
2	3	Business	Carlyle Looks Toward Commercial Aerospace (Reu…	Reuters - Private investment firm Carlyle Grou…
3	3	Business	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab…
4	3	Business	Iraq Halts Oil Exports from Main Southern Pipe…	Reuters - Authorities have halted oil export\f…
5	3	Business	Oil prices soar to all-time record, posing new…	AFP - Tearaway world oil prices, toppling reco…
…	…	…	…	…
119996	1	World	Pakistan's Musharraf Says Won't Quit as Army C…	KARACHI (Reuters) - Pakistani President Perve…
119997	2	Sports	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowled…
119998	2	Sports	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of…
119999	2	Sports	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: …
120000	2	Sports	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad…

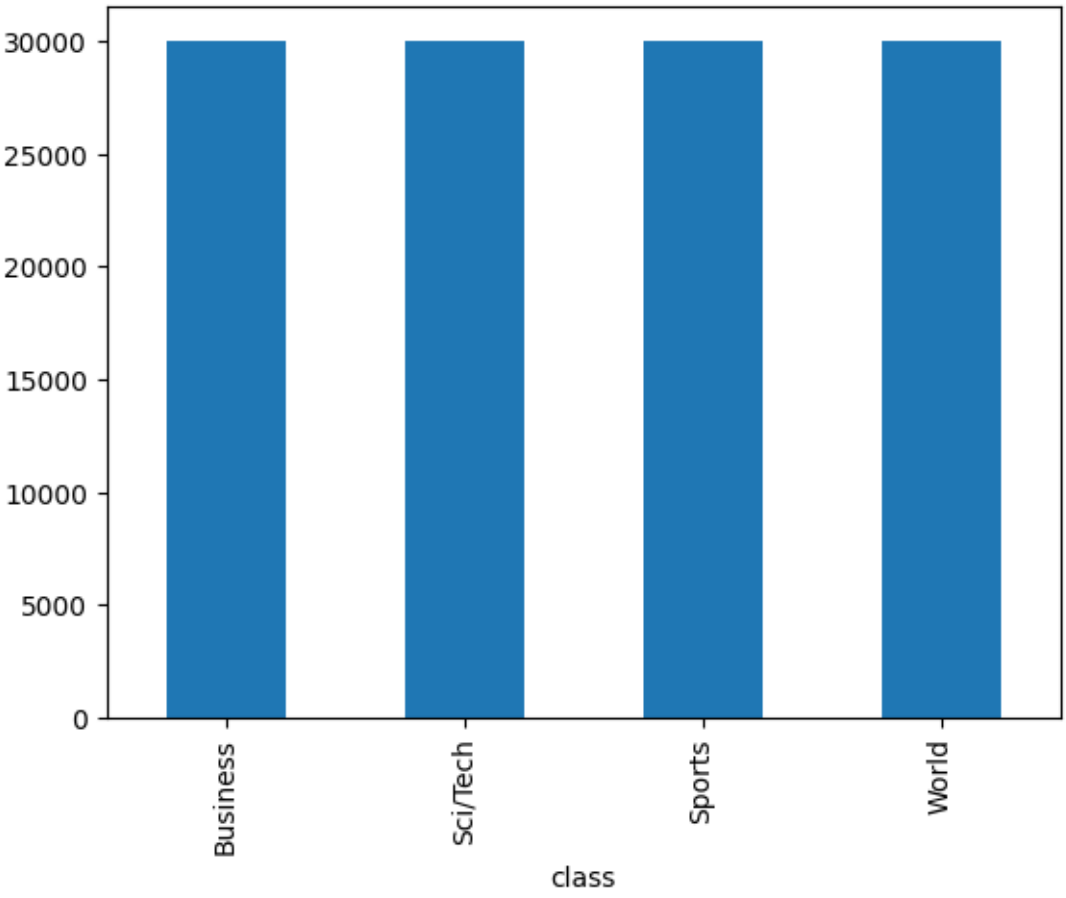
```
[120000 rows x 4 columns]
```

El código anterior tiene la función principal de leer las clases del archivo y separar la información leída en renglones individuales. Además, se agrega la columna `class` al dataframe, que contiene los nombres de las categorías que se desean predecir posteriormente en función del índice numérico de las mismas, además la columna `class` se inserta en el índice 1 de las columnas del dataframe.

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

[4]:
```
train_df['class'].value_counts().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".

[5]:
```
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.

[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  \
1                 3  Business
2                 3  Business
3                 3  Business
4                 3  Business
5                 3  Business
…               …        …
119996            1     World
119997            2    Sports
119998            2    Sports
119999            2    Sports
120000            2    Sports

                                                    title  \
1       Wall St. Bears Claw Back Into the Black (Reuters)
2       Carlyle Looks Toward Commercial Aerospace (Reu…
3         Oil and Economy Cloud Stocks' Outlook (Reuters)
4       Iraq Halts Oil Exports from Main Southern Pipe…
5       Oil prices soar to all-time record, posing new…
…                                                     …
119996  Pakistan's Musharraf Says Won't Quit as Army C…
119997                  Renteria signing a top-shelf deal
119998                    Saban not going to Dolphins yet
119999                                Today's NFL games
120000                     Nets get Carter from Raptors

                                                description  \
1       Reuters - Short-sellers, Wall Street's dwindli…
2       Reuters - Private investment firm Carlyle Grou…
3       Reuters - Soaring crude prices plus worries\ab…
4       Reuters - Authorities have halted oil export\f…
5       AFP - Tearaway world oil prices, toppling reco…
…                                                     …
119996   KARACHI (Reuters) - Pakistani President Perve…
119997  Red Sox general manager Theo Epstein acknowled…
119998  The Miami Dolphins will put their courtship of…
119999  PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: …
```

```
120000  INDIANAPOLIS -- All-Star Vince Carter was trad…

                                                    text
1       wall st. bears claw back into the black (reute…
2       carlyle looks toward commercial aerospace (reu…
3       oil and economy cloud stocks' outlook (reuters…
4       iraq halts oil exports from main southern pipe…
5       oil prices soar to all-time record, posing new…
…                                                     …
119996  pakistan's musharraf says won't quit as army c…
119997  renteria signing a top-shelf deal red sox gene…
119998  saban not going to dolphins yet the miami dolp…
119999  today's nfl games pittsburgh at ny giants time…
120000  nets get carter from raptors indianapolis -- a…

[120000 rows x 5 columns]
```

En la celda de código previa, se convierten a minúsculas todos los títulos contenidos en la columna `title` para luego convertir también a minúsculas todas las descripciones del `train_df` contenidas en la columna `description` y posteriormente concatenar los títulos y las descripciones para formar el texto contenido en la variable text y una vez teniendo dicha variable, se procede a reemplazar las diagonales presentes en el texto por un espacio en blanco y con los resultados de ese reemplazo, se forma una nueva columna llamada `text` en el dataframe `train_df`.

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.

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

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

```
  0%|          | 0/120000 [00:00<?, ?it/s]
```

[7]:
```
        class_index     class  \
1                 3  Business
2                 3  Business
3                 3  Business
4                 3  Business
5                 3  Business
…               …        …
119996            1     World
119997            2    Sports
119998            2    Sports
119999            2    Sports
120000            2    Sports

                                                    title  \
1       Wall St. Bears Claw Back Into the Black (Reuters)
```

```
2       Carlyle Looks Toward Commercial Aerospace (Reu…
3         Oil and Economy Cloud Stocks' Outlook (Reuters)
4       Iraq Halts Oil Exports from Main Southern Pipe…
5       Oil prices soar to all-time record, posing new…
…                                                     …
119996  Pakistan's Musharraf Says Won't Quit as Army C…
119997                  Renteria signing a top-shelf deal
119998                    Saban not going to Dolphins yet
119999                                Today's NFL games
120000                     Nets get Carter from Raptors

                                                 description  \
1       Reuters - Short-sellers, Wall Street's dwindli…
2       Reuters - Private investment firm Carlyle Grou…
3       Reuters - Soaring crude prices plus worries\ab…
4       Reuters - Authorities have halted oil export\f…
5       AFP - Tearaway world oil prices, toppling reco…
…                                                     …
119996   KARACHI (Reuters) - Pakistani President Perve…
119997  Red Sox general manager Theo Epstein acknowled…
119998  The Miami Dolphins will put their courtship of…
119999  PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: …
120000  INDIANAPOLIS -- All-Star Vince Carter was trad…

                                                    text  \
1       wall st. bears claw back into the black (reute…
2       carlyle looks toward commercial aerospace (reu…
3       oil and economy cloud stocks' outlook (reuters…
4       iraq halts oil exports from main southern pipe…
5       oil prices soar to all-time record, posing new…
…                                                     …
119996  pakistan's musharraf says won't quit as army c…
119997  renteria signing a top-shelf deal red sox gene…
119998  saban not going to dolphins yet the miami dolp…
119999  today's nfl games pittsburgh at ny giants time…
120000  nets get carter from raptors indianapolis -- a…

                                                  tokens
1       [wall, st., bears, claw, back, into, the, blac…
2       [carlyle, looks, toward, commercial, aerospace…
3       [oil, and, economy, cloud, stocks, ', outlook,…
4       [iraq, halts, oil, exports, from, main, southe…
5       [oil, prices, soar, to, all-time, record, ,, p…
…                                                     …
119996  [pakistan, 's, musharraf, says, wo, n't, quit,…
119997  [renteria, signing, a, top-shelf, deal, red, s…
119998  [saban, not, going, to, dolphins, yet, the, mi…
```

```
119999  [today, 's, nfl, games, pittsburgh, at, ny, gi…
120000  [nets, get, carter, from, raptors, indianapoli…

[120000 rows x 6 columns]
```

Anteriormente se crea una columna adicional en el dataframe de entrenamiento llamada “tokens” que almacena los tokens creados a partir del texto en la columna “text”, además los tokens se crean con el propósito principal de generar el vocabulario con el que será entrenado el modelo de lenguaje, mismo que a su vez contendrá solamente aquellas palabras o frases que tengan una frecuencia de ocurrencia superior a un determinado umbral establecido posteriormente, esto con la finalidad principal de establecer el espacio de palabras que el modelo conocerá al ser entrenado, por lo que aquellas otras palabras no presentes en dicho vocalubario de entrenamiento, no tendrán ninguna probabilidad de ocurrencia para el modelo.

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

[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: 19,667
```

Como se aprecia en el resultado del bloque de código anterior, el tamaño del vocabulario de entrenamiento es 19,667, lo cual significa que el modelo de lenguaje actualmente conoce dicha cantidad de palabras, no obstante, también es importante señalar que en ese vocabulario de entrenamiento, también se ubica el token especial [UNK] que se utilizará para representar en el vocabulario aquel conjunto de palabras que el modelo aún no haya visto y al que es posible aplicar una amplia gama de métodos de suavizamiento como el Laplace Smoothing, abnsolute discounting, entre otros para garantizar que no haya palabras del vocabulario del modelo que queden con probabilidad nula.

[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%|          | 0/120000 [00:00<?, ?it/s]
```

[9]:

	class_index	class	title	description	text
1	3	Business	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli…	wall st. bears claw back into the black (reute…
2	3	Business	Carlyle Looks Toward Commercial Aerospace (Reu…	Reuters - Private investment firm Carlyle Grou…	carlyle looks toward commercial aerospace (reu…
3	3	Business	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab…	oil and economy cloud stocks' outlook (reuters…
4	3	Business	Iraq Halts Oil Exports from Main Southern Pipe…	Reuters - Authorities have halted oil export\f…	iraq halts oil exports from main southern pipe…
5	3	Business	Oil prices soar to all-time record, posing new…	AFP - Tearaway world oil prices, toppling reco…	oil prices soar to all-time record, posing new…
…	…	…	…	…	…
119996	1	World	Pakistan's Musharraf Says Won't Quit as Army C…	KARACHI (Reuters) - Pakistani President Perve…	pakistan's musharraf says won't quit as army c…
119997	2	Sports	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowled…	
119998	2	Sports	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of…	
119999	2	Sports	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: …	
120000	2	Sports	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad…	

```
119997  renteria signing a top-shelf deal red sox gene…
119998  saban not going to dolphins yet the miami dolp…
119999  today's nfl games pittsburgh at ny giants time…
120000  nets get carter from raptors indianapolis -- a…

                                                  tokens  \
1       [wall, st., bears, claw, back, into, the, blac…
2       [carlyle, looks, toward, commercial, aerospace…
3       [oil, and, economy, cloud, stocks, ', outlook,…
4       [iraq, halts, oil, exports, from, main, southe…
5       [oil, prices, soar, to, all-time, record, ,, p…
…                                                     …
119996  [pakistan, 's, musharraf, says, wo, n't, quit,…
119997  [renteria, signing, a, top-shelf, deal, red, s…
119998  [saban, not, going, to, dolphins, yet, the, mi…
119999  [today, 's, nfl, games, pittsburgh, at, ny, gi…
120000  [nets, get, carter, from, raptors, indianapoli…

                                                features
1       {427: 2, 566: 1, 1608: 1, 14927: 1, 120: 1, 73…
2       {16143: 2, 1077: 1, 854: 1, 1287: 1, 4237: 1, …
3       {66: 1, 9: 2, 351: 2, 4572: 1, 158: 1, 119: 1,…
4       {77: 2, 7434: 1, 66: 3, 1782: 1, 32: 2, 901: 2…
5       {66: 2, 99: 2, 4377: 1, 4: 2, 3598: 1, 149: 1,…
…                                                     …
119996  {383: 1, 23: 1, 1626: 2, 91: 1, 1809: 1, 285: …
119997  {8419: 2, 2632: 1, 5: 4, 0: 3, 127: 1, 203: 3,…
119998  {7711: 2, 68: 1, 661: 1, 4: 2, 1439: 2, 703: 1…
119999  {106: 1, 23: 1, 729: 1, 225: 1, 1586: 1, 22: 1…
120000  {2163: 2, 226: 1, 2409: 2, 32: 1, 2995: 2, 219…

[120000 rows x 7 columns]
```

En el bloque de código previo, en primer lugar, se realiza la importación de la clase defaultdict perteneciente al módulo collections, además ésta clase consiste en un tipo especial de diccionario que no retornará un error al momento de que se intente acceder a una clave que no exista dentro del mismo, sino que en su lugar, creará en automático un nuevo input para esa clave con un cierto valor predefinido, por lo que en este caso en particular, dicho valor por defecto es el entero 0.

Adicionalmente, también cabe mencionar que después de importar la clase defaultdict, se procede a programar la función `make_feature_vector()` que recibe como parámetros de entrada, por un lado, un listado de tokens, además de un parámetro llamado unk_id que se emplea para asignar un idenificador único a aquellos tokens desconocidos por el modelo, además dentro de ésta función, se genera un diccionario denominado `vector` mediante la utilización de la función `defaultdict(int)`, por lo cual, cualquier clave inexistente en el diccionario `vector` será inicializada con un valor de 0.

Después de lo anterior, la función procede a incrementar en 1 unidad el valor localizado en `vector[i]` donde i representa al id o identificador de cada token, por lo cual, se lleva a cabo un conteo de las ocurrencias de cada uno de los tokens en la lista de tokens. Además , tras definir

las acciones de la función `make_feature_vector()` se procede a aplicar dicha función sobre los valores de la columna `tokens` del dataframe de entrenamiento `train_df`, esto a través de la función `progress_map` para mostrar un indicador de progreso, además dado que cada registro del dataframe contiene una lista de tokens, para cada una de esas listas se crea un vector de características a manera de un diccionario y dichos diccionarios se almacenan en una nueva columna llamada `features`, por lo que en última instancia, cada fila `features` tiene un diccionario que simboliza un vector de características calculado en base a la frecuencia de los tokens para cada fila en particular.

[29]:
```
from tqdm import tqdm

def make_dense(feats):
    indices, values = [], []
    for k,v in feats.items():
        indices.append([k])
        values.append(v)

    indices = torch.tensor(indices, dtype = torch.long).T
    values = torch.tensor(values, dtype = torch.float32)

    tensor_sparse = torch.sparse_coo_tensor(indices, values, size =␣
 ↪(vocabulary_size,), dtype=torch.float32)

    return tensor_sparse.to_dense()

X_train = torch.stack([make_dense(feats).to_dense() for feats in␣
 ↪tqdm(train_df['features'])])
y_train = torch.tensor(train_df['class_index'].astype(int).to_numpy() - 1,␣
 ↪dtype = torch.long)
```

```
100%|      | 120000/120000 [00:23<00:00, 5108.09it/s]
```

En la celda de código anterior, en primer lugar se realiza la importación de la librería tqdm que sirve para desplegar barras de progreso al momento de ejecutar ciclos o bucles, luego se procede a configurar la función `make_dense()` que recibe un diccionario denominado `feats` como input donde sus claves simbolizan índices de cualidades y los valores representan a su vez aquellos valores numéricos relacionados con dichas cualidades, para posteriormente crear 2 listas: la primera es `indices` para almacenar las claves asociadas con las cualiades y la segunda es `values` par almacenar los valores de cada una de las características, posteriormente esas listas se transforman en tensores de PyTorch, por lo que por un lado, la lista `indices` se transforma en un tensor de enteros largos y después es transpuesto para convertirlo al formato aceptado por los tensores dispersos y a su vez la segunda lista `values` se transforma en concreto en un tensor de valores de punto flotante y posteriormente se genera un nuevo tensor con los índices y valores de los tensores previamente mencionados y de un tamaño igual al definido en la variable `vocabulary_size`.

Posteriormente, para crear el conjunto de entrenamiento de X `X_train`, se ejecuta un ciclo con barras de progresosobre cada grupo de cualidades en la columna `features` de `train_df`, para luego transformar los datos dispersos en un tensor denso para finalmente integrar todos los tensores formados en otro tensor de mayor tamaño para así formar el conjunto de entrenamiento para X (`X_train`). Por otra parte, para crear el conjunto de prueba y (`y_train`), primero las clases a

predecir se transforman en un arreglo de numpy de números enteros y el resultado a su vez se transforma en un tensor de PyTorch, además también se resta 1 a los índices de las clases con el propósito de que inicien en 0 en vez de 1 y finalmente, el tensor resultante del proceso anterior, será el conjunto de entrenamiento para y `y_train` cuyos valores serán enteros largos.

[20]:
```
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: 100%|      | 120000/120000 [01:00<00:00, 1998.85it/s]
epoch 2: 100%|      | 120000/120000 [00:57<00:00, 2085.76it/s]
epoch 3: 100%|      | 120000/120000 [00:57<00:00, 2085.86it/s]
epoch 4: 100%|      | 120000/120000 [00:57<00:00, 2101.85it/s]
epoch 5: 100%|      | 120000/120000 [00:57<00:00, 2072.48it/s]
```

En el código anterior, en primera instancia se importan las librerías `torch.nn` y `torch.optim` que contienen módulos con funciones destinadas a la implementación de redes neuronales y optimizadores para entrenar modelos, respectivamente, luego se procede a definir los valores de los hiperparámetros: la tasa de aprendizaje para el optimizador se establece en 1, se establece que el

entrenamiento del modelo tendrá una duración de 5 épocas, además se tendrán tantos ejemplos de entrenamiento como registros se tengan en el conjunto de entrenamiento de X `X_train`, además se tendrán tantas características de entrenamiento como columnas se tengan en `X_train` y finalmente se establece que la cantidad de clases de salida. Además de lo anterior, después de definir los valores para los hiperparámetros, se procede a definir un modelo de red neuronal que recibe como entrada el número de características de entrenamiento y el número de características de salida, después se define la función de pérdida que será específicamente, una función de pérdida de entropía cruzada además de inicializar el optimizador de SGD (Stochastic Gradient Descent), usando los parámetros del modelo antes definidos.

Adicionalmente, también se genera un arreglo de índices para cada uno de los ejemplos de training y esos índices son reorganizados de forma aleatoria para mejorar el nivel de aprendizaje del modelo durante su etapa de entrenamiento y posteriormente se realiza el entrenamiento como tal del modelo durante 5 épocas, y en cada una, se combinan los ejemplos de entrenamiento y se procesan de 1 en 1, además, luego se toma el ejemplo con índice i y se manda al dispositivo correspondiente, además de agregar una dimensión extra con la función `unsqueeze` y de manera similar, también se toma la etiqueta con índice i de `y_true` y también es enviada al mismo dispositivo, luego el modelo predice la salida correspondiente a cada entrada, posteriormente, una vez hechas las predicciones, se calculan los valores de la función de pérdida entre las clases predichas por del modelo y las clases verdaderas, para luego llevar a cabo el proceso de backpropagation con el objetivo de calcular los gradientes de los parámetros del modelo en relación al nivel de pérdida calculado, finalmente se actualizan los parámetros del modelo empleando el optimizador, junto con gradientes calculados previamente.

Next, we evaluate on the test dataset

[45]:
```
test_df = pd.read_csv('/kaggle/input/ag-news-classification-dataset/test.csv',␣
 ↪header=None).iloc[1:, :]
test_df.columns = ['class_index', 'title', 'description']

# repeat all preprocessing done above, this time on the test set

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)

X_test = np.stack(test_df['features'].progress_map(make_dense))
test_df['class_index'] = test_df['class_index'].astype("int")
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%|          | 0/7600 [00:00<?, ?it/s]
  0%|          | 0/7600 [00:00<?, ?it/s]
  0%|          | 0/7600 [00:00<?, ?it/s]
```

En el código anterior, se realiza el mismo tipo de preprocesamiento que con el dataframe de entrenamiento, pero ahora aplicado al dataframe de test, por lo que en primera instancia, después de leer el archivo de los datos de prueba `test.csv` y definir los nombres de sus columnas, se procede a nuevamente a definir la columna `text` como la concatenación de las columnas `title` y `description` en minúsculas y de ese resultado, se reemplazan la diagonales por un espacio en blanco y una vez hecho el reemplazo, los resultados finales se almacenan en la columna `text`, además nuevamente se dividen los textos de la columna `text` en tokens para formar la columna `tokens` pero esta vez para el dataset de prueba y también se calcula nuevamente la columna `features` en base a la frecuencia de los tokens del dataset de prueba en la lista de tokens de dicho dataset. Además, después se procede a crear tensores densos en base a los diccionarios contenidos en la columna `features` y dichos tensores posteriormente se apilan para formar el conjunto de prueba para X (`X_test`).

Por otro lado, también se procede a convertir la columna `class_index` a tipo entero para que sea posible restar 1 unidad a cada uno de los valores de `class_index` y de esa manera, calcular el conjunto de prueba para y (`y_test`), convirtiendo finalmente el resultado de la resta anteriormente mencionada a un tensor que contenga todos los datos para poner a prueba el modelo.

[46]:
```
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       0.94      0.82      0.88      1900
      Sports       0.91      0.98      0.94      1900
    Business       0.78      0.90      0.84      1900
    Sci/Tech       0.88      0.79      0.83      1900

    accuracy                           0.87      7600
   macro avg       0.88      0.87      0.87      7600
weighted avg       0.88      0.87      0.87      7600
```

En la última celda de código, en primer lugar se realiza la importación de la función `classification_report()` del módulo `metrics` de la librería sklearn, misma que tiene el propósito de desplegar a manera de una tabla, los valores de distintas métricas de desempeño del modelo tales como precisión, F1 Score, entre otras, para cada posible clase predicha, posteriormente el método `model.eval()` activa el modo de evaluación del modelo, lo cual implica que se deshabilitan ciertas funciones usadas en el entrenamiento como aplicación de dropout, asegurando que las clases predichas por el modelo tengan coherencia al momento de evaluar el desempeño del modelo implementado. Adicionalmente, por medio del comando `with torch.no_grad()` se ingresa en un

contexto en el cual, se omite el cálculo y el almacenamiento de gradientes, produciendo una reducción en la cantidad de memoria utilizada y a su vez agiliza las operaciones, además, luego de esto, se procede a mandar los datos de prueba para X `X_test` al dispositivo donde se encuentra almacenado el modelo de lenguaje (CPU O GPU), después de esto se pasa el dataset de prueba al modelo para calcular las predicciones y posteriormente se determina la clase predicha, siendo ésta misma, aquella que posee la mayor probabilidad de entre todas las posibles clases y una vez obtenidas las clases predichas se envían de un dispositivo a otro (de la CPU a la GPU o viceversa) y éstas a su vez pasan de ser un tensor de PyTorch, a ser un arreglo de Numpy, lo cual resulta necesario realizar, puesto que para que la función `classification_report()` funcione adecuadamente, los datos que reciba deben estar en un arreglo de Numpy.

Finalmente, se procede a desplegar la tabla con los valores de las métricas de desempeño del modelo, esto se realiza mediante la función `classification_report()` comentada previamente, misma que realiza la comparación entre las labels de prueba verdaderas y las labels predichas por el modelo, además con el parámetro `target_names` las clases se representan con sus respectivos nombres en lugar ser representadas con números.