

✓ Multiclass Text Classification with

Logistic Regression Implemented with PyTorch and CE Loss

First, we will do some initialization.

Como primer paso se importan las librerías necesarias, como torch para manejar tensores y modelos de machine learning, numpy y pandas para manipulación de datos, y tqdm para mostrar una barra de progreso al ejecutar las celdas. Se activa la opción para mostrar el progreso en las operaciones de pandas. Se configura la opción de usar GPU si está disponible, seleccionando automáticamente el dispositivo para el procesamiento (cuda o cpu). Además, se define una semilla aleatoria para asegurar la reproducibilidad en la generación de números aleatorios en Python, numpy, y torch.

```
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.

A continuación se carga el archivo `train.csv` mediante pandas y se asignan nombres a las columnas: 'class index', 'title', y 'description'. Este archivo contiene los datos de entrenamiento para el modelo.

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

	class	index	title	description
0	3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...	
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...	
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worrieslab...	
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil exportf...	
4	3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...	
...
119995	1	Pakistan's Musharraf Says Won't Quit as Army C...	KARACHI (Reuters) - Pakistani President Perve...	
119996	2	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowl...	
119997	2	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of...	
119998	2	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: ...	
119999	2	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad...	

120000 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.

Después se cargan las etiquetas de las clases desde un archivo classes.txt asignándolas a los valores en la columna 'class index' del DataFrame train_df para crear una nueva columna, 'class'. Esta columna adicional permite trabajar con las etiquetas de las clases de manera más clara y legible.

```
labels = open('/kaggle/input/ag-news/ag_news_csv/classes.txt').read().splitlines()
classes = train_df['class index'].map(lambda i: labels[i-1])
train_df.insert(1, 'class', classes)
train_df
```



	class index	class	title	description
0	3	Business	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...
1	3	Business	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...
2	3	Business	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worrieslab...
3	3	Business	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil exportf...
4	3	Business	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...
...
119995	1	World	Pakistan's Musharraf Says Won't Quit as Army C...	KARACHI (Reuters) - Pakistani President Perve...
119996	2	Sports	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowl...

◀

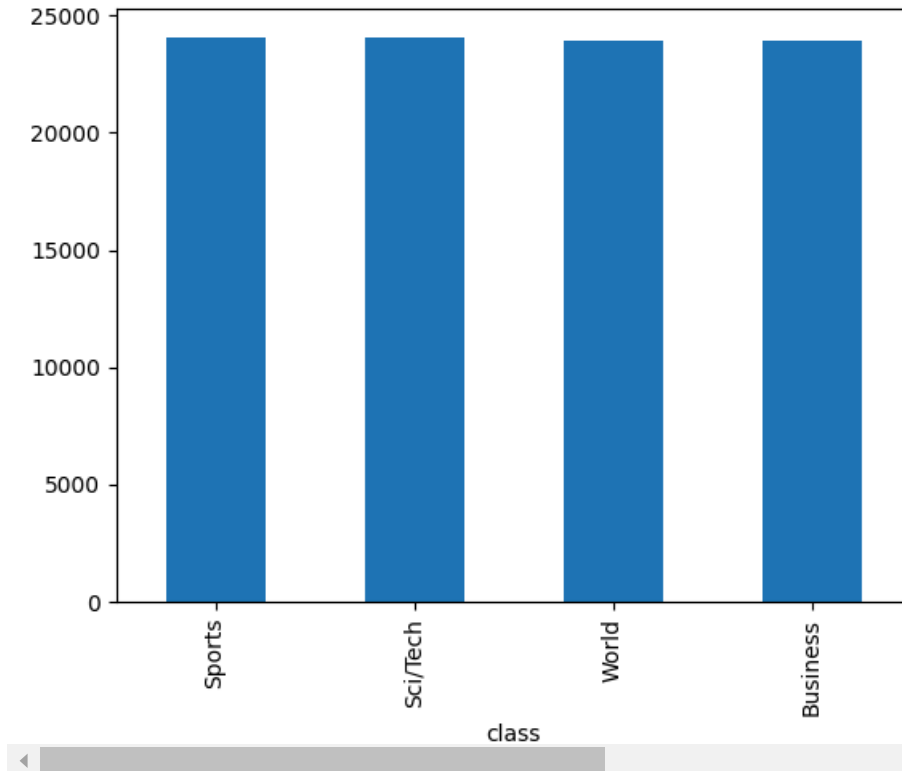
▶

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

En este punto se toma una muestra aleatoria del 80% de los datos originales, estableciendo una semilla para asegurar reproducibilidad, esto ya que al usarlo por completo nos quedamos sin memoria. Y se genera un gráfico de barras que muestra la distribución de las clases en el conjunto de entrenamiento.

```
train_df=train_df.sample(frac=0.8, random_state=42)
pd.value_counts(train_df['class']).plot.bar()
```

```
⚡ /tmp/ipykernel_30/1641164020.py:2: FutureWarning: pandas.value_counts is deprecated and will be removed in a fut
pd.value_counts(train_df['class']).plot.bar()
<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".

Se imprime el contenido de la columna 'description' para el primer registro en train_df, con el objetivo de inspeccionar una descripción de muestra del conjunto de datos.

```
print(train_df.loc[0, '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.

Se convierten los textos en las columnas "title" y "description" a minúsculas y se concatenan en una nueva columna llamada "text". Aquí también se reemplazan los caracteres \ por espacios, lo cual es útil para limpiar el texto para futuras tareas de procesamiento.

```

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

```



	class index	class	title	description	text
71787	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	3	Business	Marsh averts cash crunch	Embattled insurance broker #39;s banks agree t...	marsh averts cash crunch embattled insurance b...
54066	2	Sports	Jeter, Yankees Look to Take Control (AP)	AP - Derek Jeter turned a season that started ...	jeter, yankees look to take control (ap) ap - ...
7168	4	Sci/Tech	Flying the Sun to Safety	When the Genesis capsule comes back to Earth w...	flying the sun to safety when the genesis caps...
29618	3	Business	Stocks Seen Flat as Nortel and Oil Weigh	NEW YORK (Reuters) - U.S. stocks were set to ...	stocks seen flat as nortel and oil weigh new ...
...
59228	4	Sci/Tech	Investors Flock to Web Networking Sites	Internet whiz kids Marc Andreessen, Josh Kopel...	investors flock to web networking sites intern...
64447	3	Business	Samsung Electric Quarterly	Samsung Electronics Co. Ltd.	samsung electric quarterly profit

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.

Aquí se utiliza la función word_tokenize de NLTK para dividir cada texto en palabras (tokens). Los tokens resultantes se almacenan en una nueva columna "tokens" en el DataFrame train_df.

```

from nltk.tokenize import word_tokenize

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

```

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

	class index	class	title	description	text	tokens
71787	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	3	Business	Marsh averts cash crunch	Embattled insurance broker #39;s banks agree t...	marsh averts cash crunch embattled insurance b...	[marsh, averts, cash, crunch, embattled, insur...
54066	2	Sports	Jeter, Yankees Look to Take Control (AP)	AP - Derek Jeter turned a season that started ...	jeter, yankees look to take control (ap) ap - ...	[jeter, ,, yankees, look, to, take, control, (...]
7168	4	Sci/Tech	Flying the Sun to Safety	When the Genesis capsule comes back to Earth w...	flying the sun to safety when the genesis caps...	[flying, the, sun, to, safety, when, the, gene...
29618	3	Business	Stocks Seen Flat as Nortel and Oil Weigh	NEW YORK (Reuters) - U.S. stocks were set to ...	stocks seen flat as nortel and oil weigh new ...	[stocks, seen, flat, as, nortel, and, oil, wei...
...

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

Se establece un umbral mínimo de 10 apariciones para los tokens y se cuenta la frecuencia de cada token en el DataFrame, excluyendo aquellos que aparecen menos de 10 veces. Se construyen dos diccionarios: uno para mapear cada token a un identificador único (token_to_id) y otro para el mapeo inverso. También se calcula el tamaño del vocabulario y se imprime.

```
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: 17,430

Luego se define una función para crear un vector de características que cuenta las apariciones de cada token en un texto, utilizando un diccionario de tokens e identificadores. Este vector se asigna a la nueva columna 'features' del DataFrame train_df, generando vectores de características para cada registro.

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

	class index	class	title	description	text	tokens	features
71787	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...	{2451: 1, 167: 1, 11: 1, 201: 1, 6778: 2, 2: 5...
67218	3	Business	Marsh averts cash crunch	Embattled insurance broker #39;s banks agree t...	marsh averts cash crunch embattled insurance b...	[marsh, averts, cash, crunch, embattled, insur...	{1945: 2, 0: 2, 723: 1, 5100: 1, 2891: 1, 752:...
54066	2	Sports	Jeter, Yankees Look to Take Control (AP)	AP - Derek Jeter turned a season that started ...	jeter, yankees look to take control (ap) ap - ...	[jeter, ,, yankees, look, to, take, control, (...	{6670: 2, 2: 1, 508: 1, 599: 1, 4: 1, 193: 1, ...
7168	4	Sci/Tech	Flying the Sun to Safety	When the Genesis capsule comes back to Earth w...	flying the sun to safety when the genesis caps...	[flying, the, sun, to, safety, when, the, gene...	{2601: 1, 1: 4, 416: 2, 4: 3, 1061: 1, 96: 1, ...
29618	3	Business	Stocks Seen Flat as Nortel and Oil Weigh	NEW YORK (Reuters) - U.S. stocks were set to ...	stocks seen flat as nortel and oil weigh new ...	[stocks, seen, flat, as, nortel, and, oil, wei...	{158: 2, 646: 1, 1523: 1, 21: 1, 2035: 2, 9: 1...
...
59228	4	Sci/Tech	Investors Flock to Web Networking Sites	Internet whiz kids Marc Andreessen, Josh Kopel...	investors flock to web networking sites intern...	[investors, flock, to, web, networking, sites	{366: 1, 8544: 1, 4: 1, 227: 1, 2620: 1, 992: ...

Ahora se define la función `make_dense` para convertir cada vector de características disperso en un vector denso del tamaño del vocabulario. Se apilan estos vectores en `X_train` y se establecen las etiquetas de clase en `y_train`. Luego, ambos son convertidos a tensores de `torch`, lo cual permite su uso en modelos de machine learning.

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

A continuación se define y entrena el modelo de clasificación utilizando `torch`. Se configuran los hiperparámetros, como la tasa de aprendizaje, el número de épocas, y las dimensiones de entrada y salida. El modelo se entrena utilizando el algoritmo de descenso de gradiente estocástico y la función de pérdida `CrossEntropyLoss`. Durante cada época, se barajan los índices de los ejemplos y se actualizan los parámetros del modelo en función de la pérdida calculada para cada instancia.

```
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%|          | 0/96000 [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]

```

Next, we evaluate on the test dataset

Se repiten los pasos de preprocesamiento realizados en el conjunto de entrenamiento pero ahora aplicados a un conjunto de prueba (test_df). Este procesamiento incluye convertir los textos a minúsculas, tokenización y creación de vectores de características para cada texto. Finalmente, los vectores de características y etiquetas del conjunto de prueba se convierten a tensores de torch.

```

# repeat all preprocessing done above, this time on the test set
test_df = pd.read_csv('/kaggle/input/ag-news/ag_news_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('\n', ' ', 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))
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]

```


Y finalmente se evalúa el rendimiento del modelo en el conjunto de prueba utilizando la métrica classification_report de sklearn. Se establece el modo de evaluación en el modelo y se obtienen las predicciones sin almacenar gradientes. Las

predicciones finales se comparan con las etiquetas reales y se imprime un reporte de clasificación que muestra métricas como precisión, recall, y F1 para cada clase.

```
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	0.86	0.91	0.89	1900
...