Multiclass Text Classification with

Feed-forward Neural Networks and Word Embeddings

First, we will do some initialization.

Este código realiza una configuración inicial para una clasificación de texto multiclase utilizandouna red neuronal feed-forward implementada con PyTorch y pérdida de entropía cruzada (Cross-Entropy Loss).

Se empieza por configurar el entorno para trabajar con PyTorch y realizar una clasificación de texto multiclase de manera eficiente en la GPU (si está disponible). La barra de progreso, la configuración del dispositivo, y la semilla aleatoria son elementos para facilitar el entrenamiento, mejorar la eficiencia y reproducibilidad del modelo.

In [5]:
```python
# 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 [6]:
```
# Carga del conjunto de datos de entrenamiento
# La opción header=None indica que el archivo CSV no tiene encabezado, por lo que se añadirá manualmente después
train_df = pd.read_csv('/kaggle/input/datainfo/train.csv', header=None)

# Asignación de nombres a las columnas del DataFrame
# 'class index' indica la clase o etiqueta de categoría
# 'title' contiene el título del artículo de noticias
# 'description' contiene la descripción del artículo de noticias
train_df.columns = ['class index', 'title', 'description']

# Mostrar dataframe
train_df
```

Out[6]:

	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 worries\ab...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
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 acknowled...
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.

In [7]:
```
# 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/datainfo/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 actualizado con la columna de nombres de clases
train_df
```

Out[7]:

	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 worries\ab...
3	3	Business	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
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 acknowled...
119997	2	Sports	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of...
119998	2	Sports	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: ...
119999	2	Sports	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad...

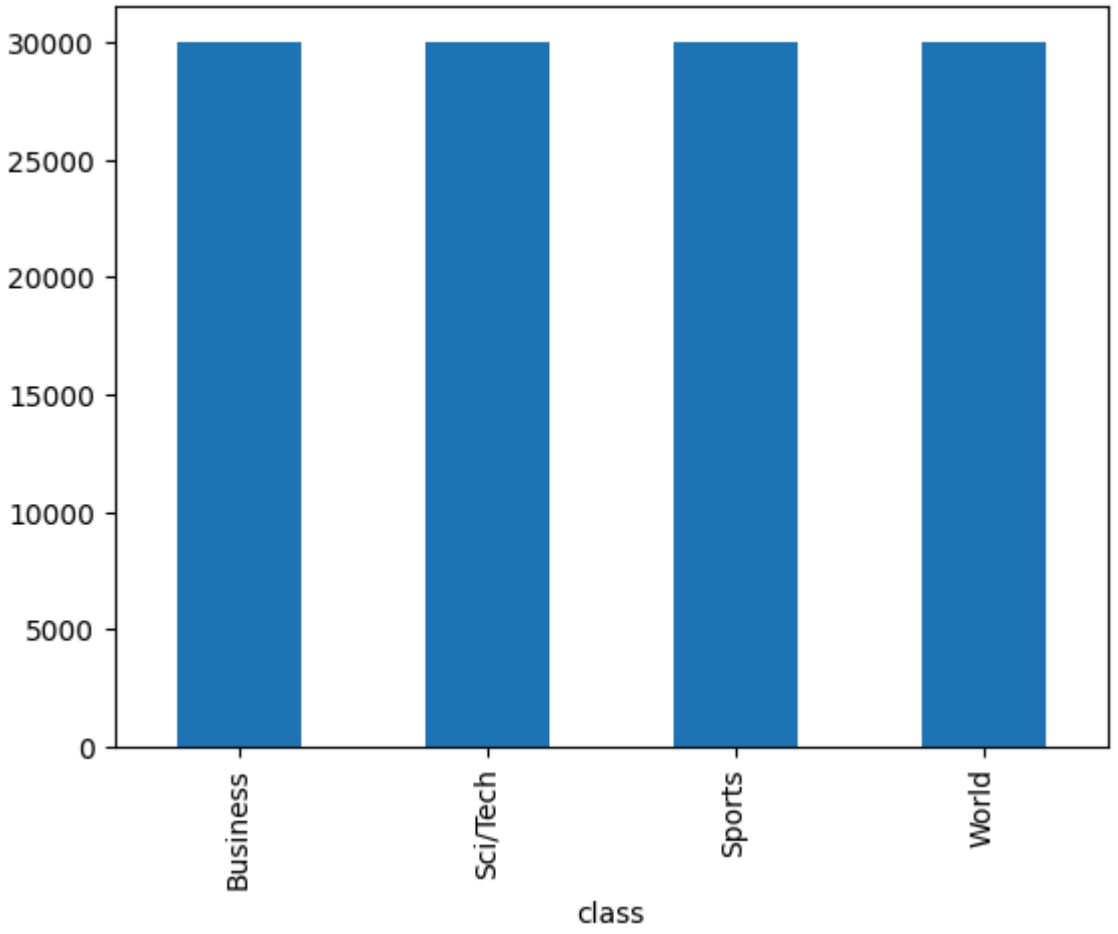
120000 rows × 4 columns

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

In [8]:
```
# 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[8]: <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 [9]:
```
# Imprimir la descripción de la fila para visualizar el texto
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.

El código combina las columnas `'title'` y `'description'` en minúsculas dentro de una nueva columna `'text'` para unificar el contenido textual en un solo lugar. Luego, reemplaza las barras invertidas (`\`) en `'text'` con espacios usando `replace()` con `regex=False` para tratarlas como caracteres literales.

In [10]:
```
# Concatenar las columnas 'title' y 'description' en una sola columna 'text'
# Ambas columnas se convierten a minúsculas para uniformidad y facilitar el procesamiento de texto
train_df['text'] = train_df['title'].str.lower() + " " + train_df['description'].str.lower()

# Reemplazar todas las barras invertidas '\' en la columna 'text' con espacios
# El parámetro 'regex=False' asegura que el reemplazo interprete '\' como un carácter literal, no como un patrón reg
train_df['text'] = train_df['text'].str.replace('\\', ' ', regex=False)

# Mostrar el DataFrame actualizado con la nueva columna 'text'
train_df
```

Out[10]:

	class index	class	title	description	text
0	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...
1	3	Business	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...	carlyle looks toward commercial aerospace (reu...
2	3	Business	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...	oil and economy cloud stocks' outlook (reuters...
3	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...

4	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...
...	...	...	...	...	...
119995	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...
119996	2	Sports	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowled...	renteria signing a top-shelf deal red sox gene...
119997	2	Sports	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of...	saban not going to dolphins yet the miami dolp...
119998	2	Sports	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: ...	today's nfl games pittsburgh at ny giants time...
119999	2	Sports	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad...	nets get carter from raptors indianapolis -- a...

120000 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.

In [11]:
```
# 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%|          | 0/120000 [00:00<?, ?it/s]
```

Out[11]:

	class index	class	title	description	text	tokens
0	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...	[wall, st., bears, claw, back, into, the, blac...
1	3	Business	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...	carlyle looks toward commercial aerospace (reu...	[carlyle, looks, toward, commercial, aerospace...
2	3	Business	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...	oil and economy cloud stocks' outlook (reuters...	[oil, and, economy, cloud, stocks, ', outlook,...
3	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...	[iraq, halts, oil, exports, from, main, southe...
4	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...	[oil, prices, soar, to, all-time, record, ,, p...
...	...	...	...	...	...	...
119995	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...	[pakistan, 's, musharraf, says, wo, n't, quit,...
119996	2	Sports	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowled...	renteria signing a top-shelf deal red sox gene...	[renteria, signing, a, top-shelf, deal, red, s...
119997	2	Sports	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of...	saban not going to dolphins yet the miami dolp...	[saban, not, going, to, dolphins, yet, the, mi...
119998	2	Sports	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: ...	today's nfl games pittsburgh at ny giants time...	[today, 's, nfl, games, pittsburgh, at, ny, gi...

119999	2	Sports	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad...	nets get carter from raptors indianapolis -- a...	[nets, get, carter, from, raptors, indianapoli...

120000 rows × 6 columns

Now we will load the GloVe word embeddings.

Cargamos los embeddings de palabras de GloVe usando la función load_word2vec_format() de Gensim, que convierte el archivo de GloVe al formato Word2Vec compatible.

In [13]:
```
from gensim.models import KeyedVectors

# Cargar los embeddings de GloVe en el formato Word2Vec utilizando Gensim
# 'no_header=True' indica que el archivo no tiene encabezado, lo cual es necesario para el formato de GloVe
glove = KeyedVectors.load_word2vec_format("/kaggle/input/datainfo/glove.6B.300d.txt", no_header=True)

# Mostrar la forma (dimensiones) de los vectores de embeddings de GloVe
glove.vectors.shape
```

Out[13]: (400000, 300)

The word embeddings have been pretrained in a different corpus, so it would be a good idea to estimate how good our tokenization matches the GloVe vocabulary.

Se evalúa qué tan bien la tokenización del corpus coincide con el vocabulario de GloVe. La función `count_unknown_words` recorre cada token en los datos y cuenta los tokens desconocidos que no están en el vocabulario de GloVe. Luego, calcula el total de tokens en el corpus y el porcentaje de tokens desconocidos (`unk_tokens`) frente a los tokens totales (`total_tokens`). También muestra el número de tokens desconocidos distintos y los 10 tokens desconocidos más comunes, lo que brinda información útil para evaluar la compatibilidad entre el vocabulario del corpus y el de los embeddings de GloVe.

In [14]:
```
from collections import Counter

# Definición de la función para contar palabras desconocidas
# La función toma los datos tokenizados y el vocabulario de GloVe como entradas
def count_unknown_words(data, vocabulary):
```

```
    counter = Counter()  # Inicializa un contador para los tokens desconocidos
    for row in tqdm(data):  # Recorre cada fila del conjunto de datos con una barra de progreso
        # Cuenta tokens en cada fila que no están en el vocabulario
        counter.update(tok for tok in row if tok not in vocabulary)
    return counter  # Devuelve el conteo de tokens desconocidos

# Contar las veces que ocurre cada token desconocido en el corpus
# train_df['tokens'] representa la columna con los tokens en el corpus, y glove.key_to_index es el vocabulario de G
c = count_unknown_words(train_df['tokens'], glove.key_to_index)

# Calcular el número total de tokens en el corpus
total_tokens = train_df['tokens'].map(len).sum()  # Suma el tamaño de cada lista de tokens para obtener el total

# Estadísticas sobre la ocurrencia de tokens desconocidos
unk_tokens = sum(c.values())  # Suma el conteo de todos los tokens desconocidos
percent_unk = unk_tokens / total_tokens  # Calcula el porcentaje de tokens desconocidos
distinct_tokens = len(list(c))  # Cuenta la cantidad de tokens desconocidos distintos

# Imprimir las estadísticas de los tokens
print(f'total number of tokens: {total_tokens:,}')
print(f'number of unknown tokens: {unk_tokens:,}')
print(f'number of distinct unknown tokens: {distinct_tokens:,}')
print(f'percentage of unkown tokens: {percent_unk:.2%}')

# Imprimir los 10 tokens desconocidos más comunes
print('top 50 unknown words:')
for token, n in c.most_common(10):
    print(f'\t{n}\t{token}')
```

```
  0%|          | 0/120000 [00:00<?, ?it/s]
total number of tokens: 5,273,465
number of unknown tokens: 66,035
number of distinct unknown tokens: 24,799
percentage of unkown tokens: 1.25%
top 50 unknown words:
        2984    /b
        2119    href=
        2117    /a
        1813    //www.investor.reuters.com/fullquote.aspx
        1813    target=/stocks/quickinfo/fullquote
        537     /p
```

```
510     newsfactor
471     cbs.mw
431     color=
417     /font
```

Podemos observar que algunos tokens desconocidos provienen de etiquetas HTML y URLs, lo cual puede indicar que sería útil limpiar mejor los datos eliminando o filtrando este tipo de secuencias antes de procesar el texto.

Glove embeddings seem to have a good coverage on this dataset -- only 1.25% of the tokens in the dataset are unknown, i.e., don't appear in the GloVe vocabulary.

Still, we will need a way to handle these unknown tokens. Our approach will be to add a new embedding to GloVe that will be used to represent them. This new embedding will be initialized as the average of all the GloVe embeddings.

We will also add another embedding, this one initialized to zeros, that will be used to pad the sequences of tokens so that they all have the same length. This will be useful when we train with mini-batches.

En esta sección se definen dos tokens especiales, `[UNK]` para palabras desconocidas y `[PAD]` para el padding en secuencias, y luego inicializa sus embeddings: el de `[UNK]` como el promedio de los vectores de GloVe y el de `[PAD]` como un vector de ceros. Después, añade estos nuevos embeddings al vocabulario de GloVe y obtiene los IDs correspondientes de ambos tokens para poder referenciarlos durante el entrenamiento o predicción, lo que permite manejar eficazmente palabras desconocidas y padding en el modelo.

In [15]:
```
# Definir tokens especiales
unk_tok = '[UNK]'  # Token para representar palabras desconocidas
pad_tok = '[PAD]'  # Token para representar el padding en secuencias

# Inicializar los valores de los nuevos embeddings
# Para [UNK], se usa el promedio de todos los vectores en GloVe como valor inicial
unk_emb = glove.vectors.mean(axis=0)  # El promedio de todos los embeddings en GloVe
# Para [PAD], se usa un vector de ceros de 300 dimensiones
pad_emb = np.zeros(300)  # Un vector de ceros de longitud 300

# Agregar los nuevos vectores (embeddings) al modelo de GloVe
# add_vectors toma dos listas: la lista de tokens y la lista de vectores correspondientes
glove.add_vectors([unk_tok, pad_tok], [unk_emb, pad_emb])

# Obtener los IDs de los tokens correspondientes a los nuevos embeddings agregados
```

```
unk_id = glove.key_to_index[unk_tok]  # ID del token [UNK]
pad_id = glove.key_to_index[pad_tok]  # ID del token [PAD]

# Mostrar los IDs de los tokens [UNK] y [PAD]
unk_id, pad_id
```

Out[15]: (400000, 400001)

In [16]:
```
from sklearn.model_selection import train_test_split

# Dividir el conjunto de datos en entrenamiento y validación
# 'train_size=0.8' significa que el 80% de los datos se asignarán al conjunto de entrenamiento
# y el 20% restante se asignará al conjunto de validación
train_df, dev_df = train_test_split(train_df, train_size=0.8)

# Reiniciar los índices del DataFrame para ambos conjuntos (entrenamiento y validación)
train_df.reset_index(inplace=True)
dev_df.reset_index(inplace=True)
```

We will now add a new column to our dataframe that will contain the padded sequences of token ids.

Se crea un vocabulario basado en el conjunto de entrenamiento filtrando los tokens que aparecen más de 10 veces. Primero, se usa explode() para convertir las listas de tokens en filas individuales, luego cuenta la frecuencia de cada token con value_counts(). Solo los tokens con una frecuencia mayor al umbral especificado (en este caso, 10) se incluyen en el vocabulario final, cuyo tamaño se imprime. Este proceso ayuda a reducir el vocabulario eliminando palabras poco frecuentes.

In [17]:
```
# Establecer un umbral para la frecuencia de tokens
threshold = 10

# Obtener todos los tokens en el conjunto de entrenamiento
# 'explode()' transforma listas de tokens en filas individuales para luego contar la frecuencia de cada token
tokens = train_df['tokens'].explode().value_counts()

# Crear un vocabulario con solo los tokens que aparecen más veces que el umbral especificado
# El uso de 'tokens > threshold' selecciona solo los tokens con una frecuencia mayor al umbral
vocabulary = set(tokens[tokens > threshold].index.tolist())
```

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

```
vocabulary size: 17,442
```

Aquí se procesan las listas de tokens en el conjunto de datos y las convierte en listas de IDs correspondientes a los embeddings de GloVe. Primero, se encuentra la longitud máxima de las listas de tokens (max_tokens) para añadir el padding necesario. La función get_id() obtiene el ID del token o devuelve el unk_id si el token es infrecuente. Luego, la función token_ids() genera listas de IDs para cada secuencia, agregando el padding necesario al final para igualar la longitud máxima. Finalmente, se añade una nueva columna 'token ids' al DataFrame que contiene estas listas de IDs ya procesadas.

In [18]:
```
# Encontrar la longitud máxima de las listas de tokens en el conjunto de entrenamiento
# Esto es importante para determinar cuántos elementos de padding agregar a las secuencias más cortas
max_tokens = train_df['tokens'].map(len).max()

# Definir una función que devuelve el ID del token, devolviendo unk_id para tokens infrecuentes
# Si el token está en el vocabulario, devuelve su ID en GloVe, si no, devuelve unk_id (ID para tokens desconocidos)
def get_id(tok):
    if tok in vocabulary:
        return glove.key_to_index.get(tok, unk_id)
    else:
        return unk_id

# Definir una función que convierte una lista de tokens en una lista de IDs
# También agrega padding al final de la lista si es más corta que la secuencia más larga (max_tokens)
def token_ids(tokens):
    # Convertir los tokens a sus IDs correspondientes usando la función get_id
    tok_ids = [get_id(tok) for tok in tokens]

    # Calcular cuántos tokens de padding se necesitan para alcanzar la longitud máxima
    pad_len = max_tokens - len(tok_ids)

    # Devolver la lista de IDs con el padding añadido
    return tok_ids + [pad_id] * pad_len  # Agregar el token de padding (pad_id) hasta que la lista tenga la longitu

# Agregar una nueva columna al DataFrame que contenga los IDs de los tokens y el padding añadido
train_df['token ids'] = train_df['tokens'].progress_map(token_ids)
```

```
# Mostrar el DataFrame
train_df
```

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

Out[18]:

	index	class index	class	title	description	text	tokens	token ids
0	9116	1	World	Najaf's Residents Feel Trapped in Battle (AP)	AP - For nearly three weeks, Amer al-Jamali ha...	najaf's residents feel trapped in battle (ap) ...	[najaf, 's, residents, feel, trapped, in, batt...	[10709, 9, 1048, 998, 4799, 6, 903, 23, 1582, ...
1	99831	3	Business	U.S. FDA Adds Restrictions to Acne Drug	WASHINGTON (Reuters) - Roche's acne drug Accu...	u.s. fda adds restrictions to acne drug washi...	[u.s., fda, adds, restrictions, to, acne, drug...	[99, 5584, 2144, 3252, 4, 400000, 780, 289, 23...
2	10663	3	Business	Smithfield Foods Profit More Than Doubles	Smithfield Foods Inc. (SFD.N: Quote, Profile, ...	smithfield foods profit more than doubles smit...	[smithfield, foods, profit, more, than, double...	[34026, 5008, 1269, 56, 73, 4229, 34026, 5008,...
3	73175	4	Sci/Tech	PluggedIn: The OQO Is Not Just Another Handhel...	SAN FRANCISCO (Reuters) - A full-fledged Wind...	pluggedin: the oqo is not just another handhel...	[pluggedin, :, the, oqo, is, not, just, anothe...	[400000, 45, 0, 293697, 14, 36, 120, 170, 2099...
4	104494	4	Sci/Tech	IBM invigorates LTO tape storage	LTO (linear tape open)-based drives are invigo...	ibm invigorates lto tape storage lto (linear t...	[ibm, invigorates, lto, tape, storage, lto, (,...	[5199, 400000, 400000, 4143, 4418, 400000, 23,...
...	...	...	...	...	...	...	...	...
95995	89460	1	World	Bush, Blair See Hope for Palestinian State (AP)	AP - As Yasser Arafat was buried, President Bu...	bush, blair see hope for palestinian state (ap...	[bush, ,, blair, see, hope, for, palestinian, ...	[272, 1, 2356, 253, 824, 10, 463, 92, 23, 1582...

95996	60620	1	World	Ex-Soldiers Vow to Bring Order to Haiti Capital	Ex-soldiers who helped topple former President...	ex-soldiers vow to bring order to haiti capita...	[ex-soldiers, vow, to, bring, order, to, haiti...	[223970, 12887, 4, 938, 460, 4, 3836, 351, 223...
95997	34086	1	World	Musharraf says U.S. must address root of terro...	Reuters - The United States could lose its war...	musharraf says u.s. must address root of terro...	[musharraf, says, u.s., must, address, root, o...	[3820, 210, 99, 390, 1476, 5440, 3, 1291, 23, ...
95998	58067	1	World	Nuclear materials #39;vanish #39; in Iraq	Equipment and materials that could be used to ...	nuclear materials #39;vanish #39; in iraq equ...	[nuclear, materials, #, 39, ;, vanish, #, 39, ...	[490, 2176, 2749, 3403, 89, 25736, 2749, 3403,...
95999	92975	4	Sci/Tech	In Brief: Bowstreet unveils pre-packaged porta...	Bowstreet this week launched its Enterprise Po...	in brief: bowstreet unveils pre-packaged porta...	[in, brief, :, bowstreet, unveils, pre-package...	[6, 2461, 45, 400000, 20465, 400000, 12174, 83...

96000 rows × 8 columns

El código primero encuentra la longitud máxima de las listas de tokens en el conjunto de validación (max_tokens) para asegurarse de agregar el padding correcto a las secuencias. Luego, usa la función token_ids() para convertir las listas de tokens en listas de IDs, agregando padding cuando sea necesario.

In [19]:
```
# Encontrar la longitud máxima de las listas de tokens en el conjunto de validación
max_tokens = dev_df['tokens'].map(len).max()

# Aplicar la función token_ids a cada lista de tokens en el conjunto de validación
dev_df['token ids'] = dev_df['tokens'].progress_map(token_ids)

# Mostrar el DataFrame
dev_df
```

```
  0%|          | 0/24000 [00:00<?, ?it/s]
```

Out[19]:

	index	class index	class	title	description	text	tokens	token ids

0	60974	1	World	Sharon Accepts Plan to Reduce Gaza Army Operat...	Israeli Prime Minister Ariel Sharon accepted a...	sharon accepts plan to reduce gaza army operat...	[sharon, accepts, plan, to, reduce, gaza, army...	[2548, 9889, 394, 4, 1680, 1166, 330, 957, 1, ...
1	50391	4	Sci/Tech	Internet Key Battleground in Wildlife Crime Fight	Why trawl through a sweaty illegal\wildlife ma...	internet key battleground in wildlife crime fi...	[internet, key, battleground, in, wildlife, cr...	[925, 638, 14944, 6, 4446, 1340, 838, 738, 400...
2	9307	3	Business	July Durable Good Orders Rise 1.7 Percent	America's factories saw orders for costly manu...	july durable good orders rise 1.7 percent amer...	[july, durable, good, orders, rise, 1.7, perce...	[375, 10699, 219, 1949, 1027, 6262, 72, 453, 9...
3	35221	3	Business	Growing Signs of a Slowing on Wall Street	all Street #39;s earnings growth, fueled by tw...	growing signs of a slowing on wall street all ...	[growing, signs, of, a, slowing, on, wall, str...	[988, 1867, 3, 7, 6515, 13, 1015, 491, 64, 491...
4	40081	1	World	The New Faces of Reality TV	The introduction of children to the genre was ...	the new faces of reality tv the introduction o...	[the, new, faces, of, reality, tv, the, introd...	[0, 50, 1919, 3, 2532, 816, 0, 4344, 3, 271, 4...
...	...	...	...	...	...	...	...	...
23995	49572	1	World	Iraqi Kidnappers Release 2 Indonesian Women	Two Indonesian women held hostage for several ...	iraqi kidnappers release 2 indonesian women tw...	[iraqi, kidnappers, release, 2, indonesian, wo...	[710, 9349, 713, 232, 2656, 266, 55, 2656, 266...
23996	40409	4	Sci/Tech	Big Wi-Fi Project for Philadelphia	What would Benjamin Franklin say? Philadelphia...	big wi-fi project for philadelphia what would ...	[big, wi-fi, project, for, philadelphia, what,...	[365, 39300, 716, 10, 2201, 102, 54, 4067, 503...

23997	70470	2	Sports	Owen scores again	Michael Owen scored the winner for Real Madrid...	owen scores again michael owen scored the winn...	[owen, scores, again, michael, owen, scored, t...	[7116, 2776, 378, 785, 7116, 878, 0, 1364, 10,...
23998	7941	4	Sci/Tech	US Online Retail Sales Expected To Double In S...	Online retail sales in the US are expected to ...	us online retail sales expected to double in s...	[us, online, retail, sales, expected, to, doub...	[95, 1292, 2645, 526, 287, 4, 1278, 6, 228, 82...
23999	42303	1	World	Egyptian holding company says it has heard fou...	Egypt said Tuesday that Iraqi kidnappers had f...	egyptian holding company says it has heard fou...	[egyptian, holding, company, says, it, has, he...	[2434, 1383, 128, 210, 20, 31, 1435, 133, 2434...

24000 rows × 8 columns

Now we will get a numpy 2-dimensional array corresponding to the token ids, and a 1-dimensional array with the gold classes. Note that the classes are one-based (i.e., they start at one), but we need them to be zero-based, so we need to subtract one from this array.

En esta parte del código, se define una clase personalizada MyDataset que hereda de Dataset en PyTorch. La clase toma dos argumentos, **x** (los datos de entrada) y **y** (las etiquetas), y almacena estos en la clase. El método **len** devuelve la cantidad de ejemplos en el conjunto de datos, mientras que **getitem** devuelve el par (x, y) correspondiente a un índice dado, convirtiéndolos a tensores de PyTorch. Esto hace que el conjunto de datos sea compatible con los cargadores de datos (DataLoader) de PyTorch para su uso en modelos de aprendizaje profundo.

In [20]:
```
from torch.utils.data import Dataset

# Definir una clase personalizada de Dataset para PyTorch
# Esta clase permitirá crear un conjunto de datos compatible con PyTorch para entrenamiento y validación
class MyDataset(Dataset):

    # El constructor (__init__) toma los datos de entrada (x) y las etiquetas (y)
    def __init__(self, x, y):
        self.x = x  # Asignar las características (inputs) a un atributo de la clase
```

```
        self.y = y  # Asignar las etiquetas (targets) a un atributo de la clase

    # Este método devuelve el tamaño del conjunto de datos (cantidad de ejemplos)
    def __len__(self):
        return len(self.y)  # Devuelve la longitud del conjunto de etiquetas, que debe coincidir con la de las carac

    # Este método permite acceder a los datos y etiquetas de forma indexada
    # Devuelve los datos de entrada (x) y la etiqueta (y) en la posición indicada por 'index'
    def __getitem__(self, index):
        # Convertir los datos de entrada y etiquetas a tensores de PyTorch
        x = torch.tensor(self.x[index])
        y = torch.tensor(self.y[index])
        return x, y  # Devolver el par de (características, etiqueta) como tensores
```

Next, we construct our PyTorch model, which is a feed-forward neural network with two layers:

Este modelo en PyTorch utiliza una capa de embeddings preentrenados (nn.Embedding.from_pretrained) que asigna una representación vectorial a cada token de entrada. Los embeddings se promedian excluyendo los tokens de padding, utilizando un tensor booleano para identificar las posiciones de padding. El resultado se pasa a través de una red neuronal feedforward, que incluye capas lineales con activación ReLU y Dropout para regularización. El modelo devuelve una salida sin aplicar softmax.

In [21]:
```
from torch import nn
import torch.nn.functional as F

# Definir la clase Model que hereda de nn.Module
class Model(nn.Module):

    # Constructor para inicializar el modelo
    def __init__(self, vectors, pad_id, hidden_dim, output_dim, dropout):
        super().__init__()  # Llamar al constructor de la clase base (nn.Module)

        # Si los embeddings no son un tensor, los convertimos a tensor
        if not torch.is_tensor(vectors):
            vectors = torch.tensor(vectors)

        # Guardar el índice del token de padding (pad_id)
        self.padding_idx = pad_id

        # Crear una capa de embeddings usando los vectores preentrenados
```

```
        # 'padding_idx' se asegura de que el token de padding no afecte los cálculos de gradiente
        self.embs = nn.Embedding.from_pretrained(vectors, padding_idx=pad_id)

        # Definir una secuencia de capas completamente conectadas (feedforward)
        # Estas capas incluyen Dropout, capas lineales, y una función de activación ReLU
        self.layers = nn.Sequential(
            nn.Dropout(dropout),  # Añadir Dropout para regularización
            nn.Linear(vectors.shape[1], hidden_dim),  # Capa lineal que toma la salida de los embeddings
            nn.ReLU(),  # Función de activación ReLU
            nn.Dropout(dropout),  # Otro Dropout para evitar el sobreajuste
            nn.Linear(hidden_dim, output_dim),  # Capa lineal que produce las predicciones finales
        )

    # Método forward que define el flujo de datos a través del modelo
    def forward(self, x):
        # Crear un array booleano que marca los elementos que no son padding (True para no-padding)
        not_padding = torch.isin(x, self.padding_idx, invert=True)

        # Calcular la longitud de cada ejemplo (excluyendo el padding) para cada entrada
        lengths = torch.count_nonzero(not_padding, axis=1)

        # Obtener las representaciones de los embeddings para las secuencias de entrada
        x = self.embs(x)

        # Calcular la media de los embeddings (sumar y dividir por la longitud de cada ejemplo)
        x = x.sum(dim=1) / lengths.unsqueeze(dim=1)

        # Pasar las representaciones a través de las capas restantes del modelo
        output = self.layers(x)

        # Retornar el output final (no se aplica softmax ya que será manejado externamente)
        return output
```

Next, we implement the training procedure. We compute the loss and accuracy on the development partition after each epoch.

Se entrena un modelo de clasificación de texto en PyTorch usando embeddings preentrenados de GloVe. Se definen hiperparámetros y se inicializan el modelo, la función de pérdida, el optimizador, y los conjuntos de datos de entrenamiento y validación. En cada epoch, el modelo se entrena sobre batches de datos, calculando la pérdida y optimizando los parámetros. Al final de cada epoch, se evalúa el

modelo en el conjunto de validación, calculando y almacenando las métricas de pérdida y precisión tanto para el entrenamiento como para la validación.

In [22]:
```
from torch import optim
from torch.utils.data import DataLoader
from sklearn.metrics import accuracy_score

# Hiperparámetros para el entrenamiento
lr = 1e-3                # Tasa de aprendizaje para el optimizador
weight_decay = 0         # Decaimiento de peso (regularización L2)
batch_size = 500         # Tamaño de batch
shuffle = True           # Barajar los datos antes de cada epoch
n_epochs = 5             # Número de épocas para el entrenamiento
hidden_dim = 50          # Dimensión de la capa oculta
output_dim = len(labels)  # Dimensión de salida (número de clases)
dropout = 0.1            # Proporción de dropout para regularización
vectors = glove.vectors  # Vectores de GloVe preentrenados

# Inicializar el modelo, la función de pérdida, el optimizador y los DataLoaders
model = Model(vectors, pad_id, hidden_dim, output_dim, dropout).to(device)  # Inicializar el modelo en el dispositiv
loss_func = nn.CrossEntropyLoss()  # Función de pérdida de entropía cruzada para clasificación multiclase
optimizer = optim.Adam(model.parameters(), lr=lr, weight_decay=weight_decay)  # Optimizador Adam

# Crear el conjunto de datos y DataLoader para entrenamiento
train_ds = MyDataset(train_df['token ids'], train_df['class index'] - 1)  # Conjunto de datos de entrenamiento (resu
train_dl = DataLoader(train_ds, batch_size=batch_size, shuffle=shuffle)  # DataLoader para cargar los datos de entre

# Crear el conjunto de datos y DataLoader para validación
dev_ds = MyDataset(dev_df['token ids'], dev_df['class index'] - 1)  # Conjunto de datos de validación
dev_dl = DataLoader(dev_ds, batch_size=batch_size, shuffle=shuffle)  # DataLoader para cargar los datos de validaci

# Variables para almacenar las pérdidas y precisiones a lo largo del entrenamiento
train_loss = []
train_acc = []
dev_loss = []
dev_acc = []

# Bucle principal de entrenamiento
for epoch in range(n_epochs):
    losses = []  # Lista para almacenar las pérdidas de cada batch durante la época
```

```
    gold = []    # Lista para almacenar las etiquetas reales
    pred = []    # Lista para almacenar las predicciones del modelo
    model.train()  # Poner el modelo en modo de entrenamiento

    # Iterar sobre los batches de datos de entrenamiento
    for X, y_true in tqdm(train_dl, desc=f'epoch {epoch+1} (train)'):
        model.zero_grad()  # Limpiar los gradientes acumulados del modelo
        X = X.to(device)  # Enviar los datos de entrada al dispositivo (CPU o GPU)
        y_true = y_true.to(device)  # Enviar las etiquetas al dispositivo

        y_pred = model(X)  # Realizar las predicciones para el batch
        loss = loss_func(y_pred, y_true)  # Calcular la pérdida para el batch

        # Almacenar la pérdida y las etiquetas/predicciones para cálculo de métricas
        losses.append(loss.detach().cpu().item())  # Guardar la pérdida del batch
        gold.append(y_true.detach().cpu().numpy())  # Guardar las etiquetas reales
        pred.append(np.argmax(y_pred.detach().cpu().numpy(), axis=1))  # Guardar las predicciones (índice de clase m

        loss.backward()  # Calcular gradientes mediante retropropagación
        optimizer.step()  # Actualizar los parámetros del modelo

    # Al final de la época, calcular y almacenar la pérdida y precisión de entrenamiento
    train_loss.append(np.mean(losses))  # Calcular la pérdida promedio de la época
    train_acc.append(accuracy_score(np.concatenate(gold), np.concatenate(pred)))  # Calcular la precisión de entrena

    # Validación (sin calcular gradientes)
    model.eval()  # Poner el modelo en modo de evaluación
    with torch.no_grad():  # No calcular gradientes en la validación
        losses = []  # Lista para almacenar las pérdidas de cada batch en la validación
        gold = []    # Lista para etiquetas reales de validación
        pred = []    # Lista para predicciones en validación

        # Iterar sobre los batches de datos de validación
        for X, y_true in tqdm(dev_dl, desc=f'epoch {epoch+1} (dev)'):
            X = X.to(device)  # Enviar los datos de entrada al dispositivo
            y_true = y_true.to(device)  # Enviar las etiquetas al dispositivo

            y_pred = model(X)  # Realizar las predicciones para el batch de validación
            loss = loss_func(y_pred, y_true)  # Calcular la pérdida para el batch de validación

            # Almacenar la pérdida y las etiquetas/predicciones para cálculo de métricas
```

```
            losses.append(loss.cpu().item())  # Guardar la pérdida del batch de validación
            gold.append(y_true.cpu().numpy())  # Guardar las etiquetas reales
            pred.append(np.argmax(y_pred.cpu().numpy(), axis=1))  # Guardar las predicciones

        # Al final de la época, calcular y almacenar la pérdida y precisión de validación
        dev_loss.append(np.mean(losses))  # Calcular la pérdida promedio de validación
        dev_acc.append(accuracy_score(np.concatenate(gold), np.concatenate(pred)))  # Calcular la precisión en vali
```

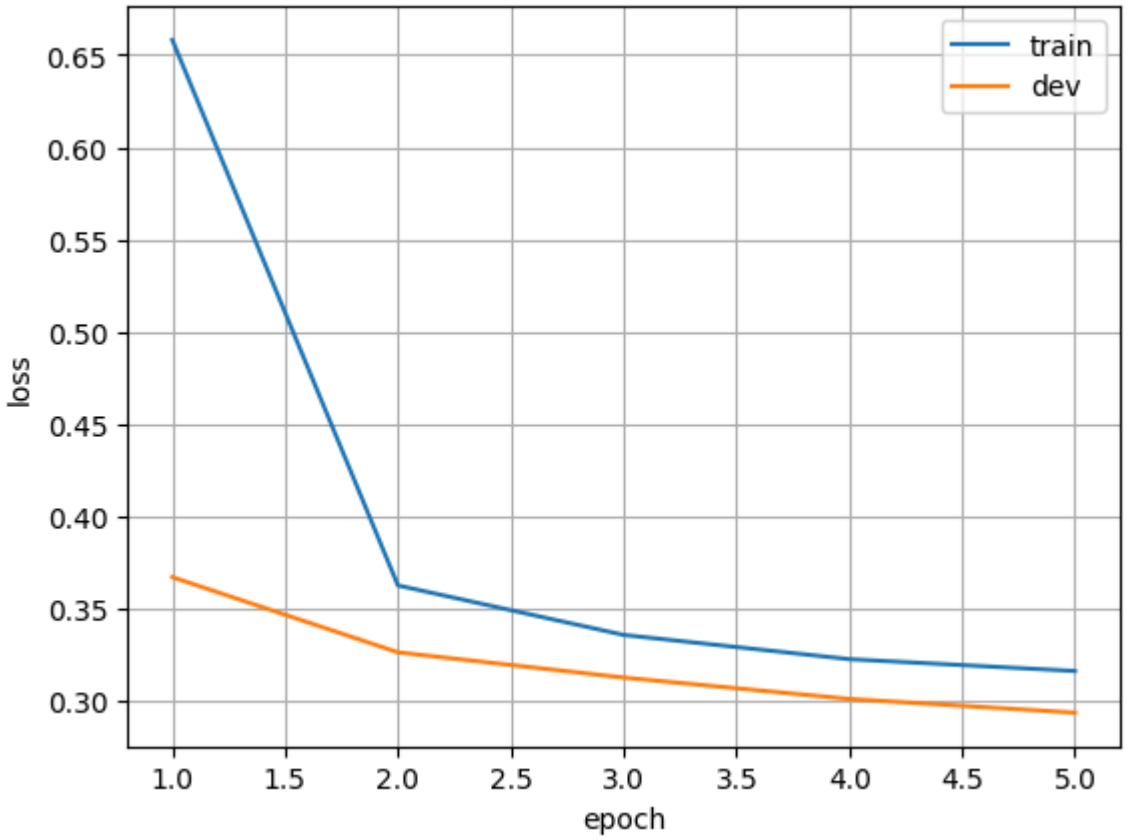
```
epoch 1 (train):   0%|          | 0/192 [00:00<?, ?it/s]
epoch 1 (dev):   0%|          | 0/48 [00:00<?, ?it/s]
epoch 2 (train):   0%|          | 0/192 [00:00<?, ?it/s]
epoch 2 (dev):   0%|          | 0/48 [00:00<?, ?it/s]
epoch 3 (train):   0%|          | 0/192 [00:00<?, ?it/s]
epoch 3 (dev):   0%|          | 0/48 [00:00<?, ?it/s]
epoch 4 (train):   0%|          | 0/192 [00:00<?, ?it/s]
epoch 4 (dev):   0%|          | 0/48 [00:00<?, ?it/s]
epoch 5 (train):   0%|          | 0/192 [00:00<?, ?it/s]
epoch 5 (dev):   0%|          | 0/48 [00:00<?, ?it/s]
```

Let's plot the loss and accuracy on dev:

Se genera un gráfico de líneas para visualizar la pérdida de entrenamiento (train_loss) y validación (dev_loss) a lo largo de las épocas.

In [23]:
```
import matplotlib.pyplot as plt
%matplotlib inline

# Crear un arreglo que represente el número de épocas para el eje x
x = np.arange(n_epochs) + 1  # np.arange(n_epochs) crea un arreglo [0, 1, ..., n_epochs-1], y al sumar 1 se ajusta a

# Graficar la pérdida de entrenamiento y validación
plt.plot(x, train_loss)
plt.plot(x, dev_loss)

# Añadir una leyenda para identificar cada línea
plt.legend(['train', 'dev'])

# Etiquetas de los ejes
plt.xlabel('epoch')
plt.ylabel('loss')
```

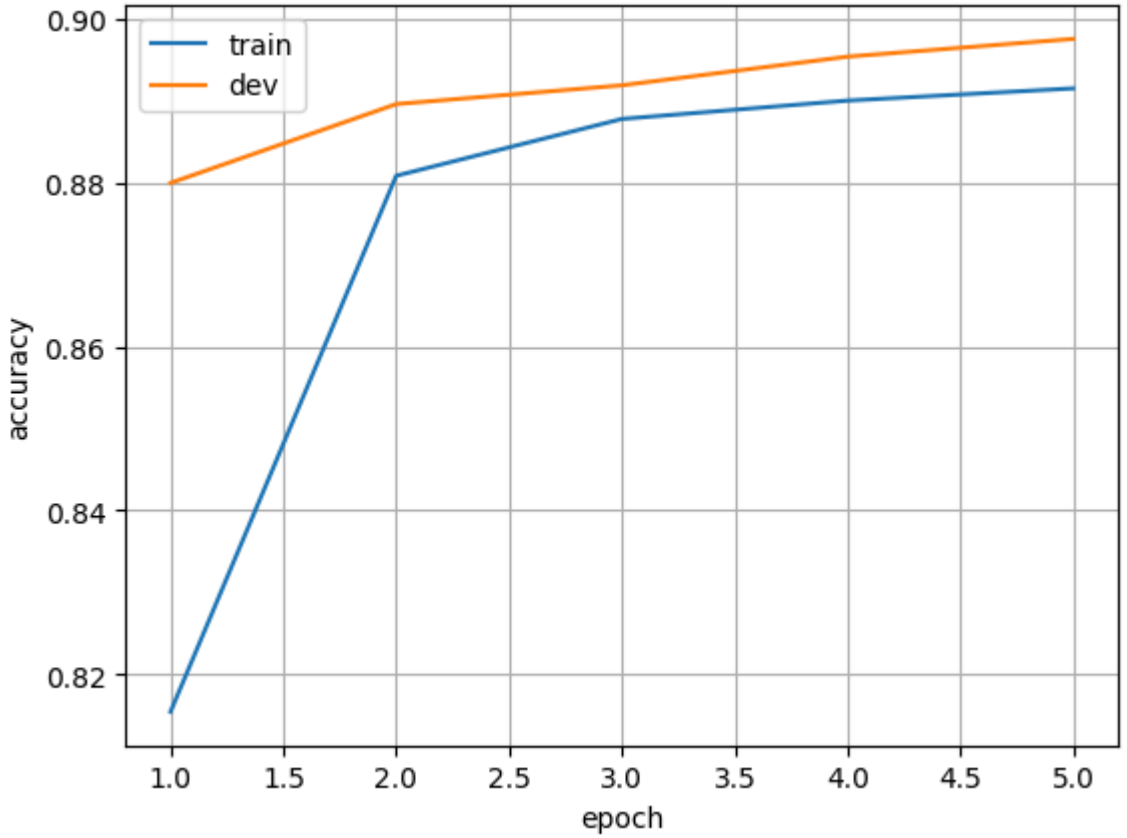
```
# Añadir una cuadrícula
plt.grid(True)
```



La gráfica muestra que la pérdida tanto en el conjunto de entrenamiento como en el de validación disminuye consistentemente a lo largo de las 5 épocas, lo que indica que el modelo está aprendiendo de manera efectiva. La pérdida de entrenamiento baja rápidamente en las primeras épocas, mientras que la de validación también disminuye de manera constante, lo que sugiere una buena generalización. Al final de las 5 épocas, las pérdidas en ambos conjuntos son bastante cercanas, lo que indica que el modelo está bien equilibrado y no está sobreajustando.

Se crea una gráfica que muestra cómo varía la precisión (accuracy) tanto en el conjunto de entrenamiento como en el de validación a lo largo de las épocas.

In [24]:
```
# Graficar la precisión del entrenamiento y validación
plt.plot(x, train_acc)
plt.plot(x, dev_acc)

# Añadir leyenda para identificar las líneas
plt.legend(['train', 'dev'])

# Etiquetas de los ejes
plt.xlabel('epoch')
plt.ylabel('accuracy')

# Añadir cuadrícula
plt.grid(True)

# Mostrar la gráfica
plt.show()
```



Next, we evaluate on the testing partition:

La gráfica muestra un aumento constante en la precisión tanto en el conjunto de entrenamiento como en el de validación a lo largo de 5 épocas. La precisión del entrenamiento y la de validación alcanzan aproximadamente un 90%, indicando que el modelo está aprendiendo correctamente y generaliza bien en datos no vistos, sin signos de sobreajuste significativo.

En esta sección se aplica el mismo preprocesamiento al conjunto de prueba que se realizó en los conjuntos de entrenamiento y validación. Primero, carga el conjunto de prueba, luego combina el título y la descripción en una sola columna de texto en minúsculas. A continuación, reemplaza las barras invertidas por espacios y tokeniza el texto, convirtiendo cada entrada en una lista de palabras. Finalmente, convierte las listas de tokens en listas de IDs correspondientes a los embeddings de GloVe y añade padding para que todas las secuencias tengan la misma longitud, tomando como referencia la longitud máxima calculada en el conjunto de validación.

In [25]:
```
# Cargar el conjunto de datos de prueba desde un archivo CSV
test_df = pd.read_csv('/kaggle/input/datainfo/test.csv', header=None)

# Asignar nombres a las columnas
test_df.columns = ['class index', 'title', 'description']

# Crear una nueva columna 'text' que contiene el título y la descripción en minúsculas
test_df['text'] = test_df['title'].str.lower() + " " + test_df['description'].str.lower()

# Reemplazar las barras invertidas '\' por espacios en la columna 'text'
test_df['text'] = test_df['text'].str.replace('\\', ' ', regex=False)

# Tokenizar el texto (crear una lista de palabras para cada texto) utilizando la función word_tokenize
test_df['tokens'] = test_df['text'].progress_map(word_tokenize)

# Obtener la longitud máxima de tokens del conjunto de validación para agregar padding de acuerdo a esa longitud
max_tokens = dev_df['tokens'].map(len).max()

# Convertir las listas de tokens en listas de IDs correspondientes a los embeddings de GloVe, añadiendo padding si e
test_df['token ids'] = test_df['tokens'].progress_map(token_ids)
```

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

Se evalúa el modelo en el conjunto de prueba. Primero, se pone el modelo en modo de evaluación (model.eval()) para desactivar el cálculo de gradientes y el dropout, optimizando la eficiencia en esta fase. Se crea un conjunto de datos de prueba a partir de los tokens y las etiquetas reales, y se utiliza un DataLoader para cargar los datos en batches. Durante la evaluación, no se almacenan gradientes (torch.no_grad()), y se realiza la predicción para cada batch, almacenando las clases predichas. Finalmente, se utiliza classification_report de Scikit-learn para comparar las etiquetas reales con las predicciones del modelo y generar un informe de clasificación, incluyendo métricas como precisión, recall y F1-score por clase.

In [26]:
```
from sklearn.metrics import classification_report

# Poner el modelo en modo de evaluación
model.eval()  # Desactiva el dropout y el cálculo de gradientes para la fase de evaluación

# Crear el conjunto de datos de prueba utilizando los 'token ids' y las 'class index'
# Se resta 1 a los índices de clase para ajustarlos a índices basados en cero
```

```python
dataset = MyDataset(test_df['token ids'], test_df['class index'] - 1)

# Crear un DataLoader para cargar los datos de prueba en batches
data_loader = DataLoader(dataset, batch_size=batch_size)

# Inicializar una lista para almacenar las predicciones del modelo
y_pred = []

# Evaluar el modelo sin calcular gradientes (reduce el uso de memoria y mejora la eficiencia)
with torch.no_grad():  # Bloque que indica que no queremos que PyTorch calcule gradientes
    for X, _ in tqdm(data_loader):  # Iterar sobre el DataLoader para obtener batches de datos de prueba
        X = X.to(device)  # Enviar los datos al dispositivo (CPU o GPU)
        y = torch.argmax(model(X), dim=1)  # Realizar predicciones y obtener la clase más probable para cada ejemplo
        y_pred.append(y.cpu().numpy())  # Convertir las predicciones a numpy y almacenarlas en y_pred

# Combinar las predicciones de todos los batches en un solo array y mostrar el informe de clasificación
# classification_report compara las etiquetas reales (dataset.y) con las predicciones (y_pred)
print(classification_report(dataset.y, np.concatenate(y_pred), target_names=labels))
```

```
  0%|          | 0/16 [00:00<?, ?it/s]
              precision    recall  f1-score   support

       World       0.92      0.88      0.90      1900
      Sports       0.95      0.97      0.96      1900
    Business       0.85      0.86      0.85      1900
    Sci/Tech       0.86      0.88      0.87      1900

    accuracy                           0.90      7600
   macro avg       0.90      0.90      0.90      7600
weighted avg       0.90      0.90      0.90      7600
```

El informe de clasificación muestra que el modelo tiene un buen rendimiento general con una precisión del 90%. Las métricas de precisión, recall y F1-score están bien equilibradas, con "Sports" obteniendo los mejores resultados (F1-score de 0.96) y "Business" ligeramente más bajo (F1-score de 0.85). La exactitud global del modelo es del 90%, y tanto el promedio macro como el ponderado de las métricas reflejan un equilibrio entre las clases, lo que indica que el modelo clasifica correctamente la mayoría de las instancias en todas las categorías del conjunto de prueba.

Conclusión: Este pipeline para la clasificación de texto multiclase realiza un flujo completo de preprocesamiento, entrenamiento y evaluación utilizando embeddings preentrenados de GloVe y una red neuronal alimentada por capas (feed-forward). Inicia con la carga y preparación de los datos, tokenización y mapeo de tokens a IDs basados en embeddings, y manejo de padding y palabras desconocidas. Luego, el modelo se entrena mediante mini-lotes y se optimiza utilizando pérdida de entropía cruzada. Durante el entrenamiento, el pipeline controla la precisión y la pérdida en los conjuntos de entrenamiento y validación, asegurando que el modelo generalice correctamente sin sobreajuste. Finalmente, el modelo se evalúa en el conjunto de prueba, mostrando métricas equilibradas con una precisión general del 90%, lo que indica un desempeño sólido y consistente en las distintas categorías. Este pipeline demuestra un enfoque estructurado y efectivo para la clasificación de textos utilizando redes neuronales y embeddings preentrenados.