f-forward-nn-word-embeddings

November 1, 2024

1 Multiclass Text Classification with

2 Feed-forward Neural Networks and Word Embeddings

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 ésta primera celda de código, primero de importan las librerías `random`, `torch`, `numpy`, `pandas`, además del módulo `notebook` perteneciente a la librería `tqdm`, mismas que tienen funciones para realizar tareas como la generación de valores de forma aleatoria, realizacón de operaciones que involucran deep learning, manipular arreglos y hacer operaciones matemáticas con ellos, manejar y analizar datos en forma de tabla, además de poder desplegar barras de progreso en los notebooks de Jupyter, mismas que en este caso particular se habilitarán exclusivamente para aquellas operaciones realizadas con `pandas`. Además de lo anterior, después de la importación de librerías, se procede a habilitar las funciones de `tqdm` en `pandas`, con lo cual será posible que todos los procesos iterativos realizados mediante funciones de la librería `pandas`, despliegue una barra de progreso en el notebook. Ahora, se procede a establecer el dispositivo computacional, ya sea el GPU o CPU, por lo que se establece que se utilizará GPU si se encuentra disponible, de lo contrario, se usará la CPU y posteriormente, se definen 3 semillas aleatorias: la primera para la generación de valores numéricos al azar en Python, la segunda para generar valores aleatorios en base a la librería de `numpy` y la tercera para generar nuevamente números de forma aleatoria pero basado específicamente en la librería de `PyTorch`.

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('train.csv', header=None).iloc[1:, :]
train_df.columns = ['class index', 'title', 'description']
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]:
```
# Convertir la columna 'class index' a numérica de tipo entero

train_df['class index'] = train_df['class index'].astype("int")

# Leer datos del archivo, separar información en renglones e incorporar las␣
 ↪clases al train_df

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

[3]:
```
        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
```

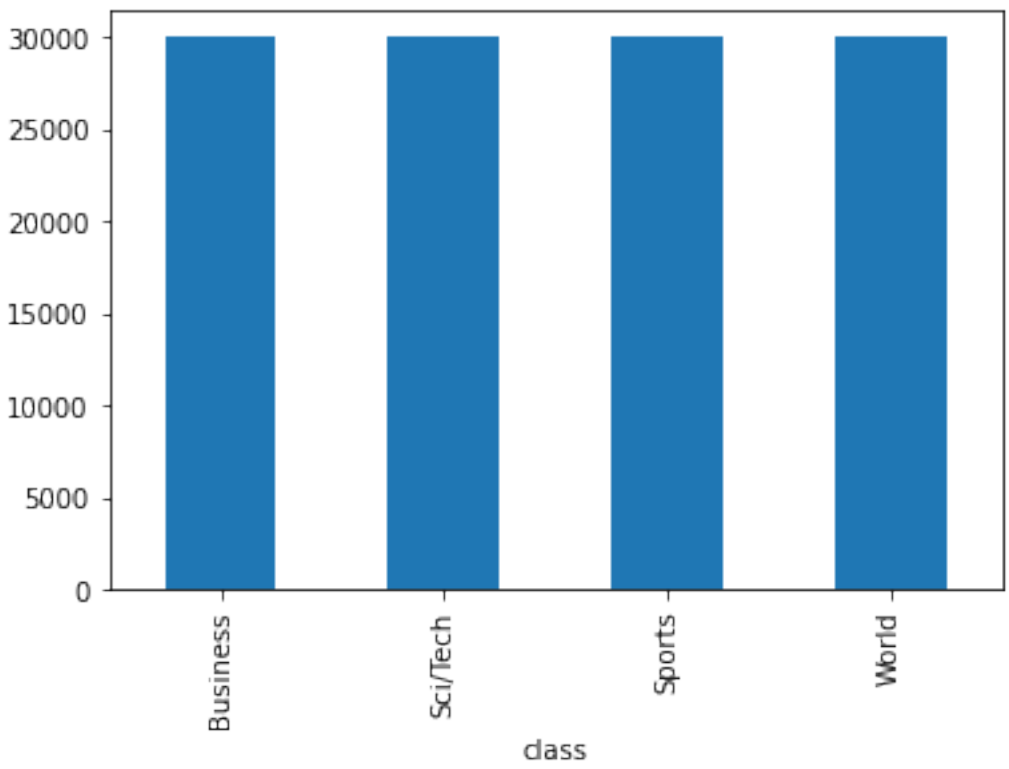
```
120000                        Nets get Carter from Raptors

                                                  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…

[120000 rows x 4 columns]
```

En el código previo, primero se convierte la columna `class index` a numérica de tipo entero para poder posteriormente acceder a cada una de las claves del archivo de `classes.txt` y después se procede a abrir el archivo de clases, se incorporan las posibles clases extraídas del archivo `.txt` en una variable llamada `classes` y a continuación se inserta la columna de clases en la posición 1 de las columnas del dataframe `train_df` (en realidad es la segunda columna del mismo).

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]:
```
train_df['text'] = train_df['title'].str.lower() + " " +␣
 ↪train_df['description'].str.lower()
train_df['text'] = train_df['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 el bloque anterior de código, básicamente se genera una nueva columna llamada `text` para el dataframe `train_df`, la cual se calcula, primero convirtiendo a texto en minúsculas los contenidos de la columna `title` de `train_df` y posteriormente, se agrega un espacio en blanco a dicho texto y posteriormente se concatena con los contenidos de la columna `description` también en letras minúsculas, para finamente, reeplazar los caractéres \ por un espacio en blanco .

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	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…	
119997	2	Sports	Renteria signing a top-shelf deal	
119998	2	Sports	Saban not going to Dolphins yet	
119999	2	Sports	Today's NFL games	
120000	2	Sports	Nets get Carter from Raptors	

```
…                                                       …
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]
```

En la celda previa, se importa la función `word_tokenize()` que pertenece al módulo `tokenize` de la librería `nltk`, esto para obtener los tokens de las columnas `title` y `description`, para lo cual, dicha función es aplicada sobre la columna `text` que representa el resultado de la concatenación de `title` y `description`, por medio de la función `progress_map()` para que al final, se muestre una barra de progreso indicando la proporción del proceso de tokenización que ya ha sido correctamente realizado y finalmente, los tokens resultantes se almacenan en una nueva columna del dataframe con el nombre `tokens`. que representa la forma tokenizada de los textos.

Now we will load the GloVe word embeddings.

[8]:
```
from gensim.models import KeyedVectors
glove = KeyedVectors.load_word2vec_format("glove.6B.300d.txt", no_header=True)
glove.vectors.shape
```

[8]: (400000, 300)

En primera instancia, en el código anterior se procede a la importación de la función `KeyedVectors` del módulo `models`, a su vez perteneciente a la librería `gensim`, la cual tiene la función principal de operar con palabras a manera de vectores, después se procede a cargar los datos del archivo `glove.6B.300d.txt` con el parámetro `no_header = True`, con el objetivo de que al momento de cargar los datos, indicar que el primer registro del archivo no se considere como un encabezado, para finalmente, mostrar la forma (tupla con cantidad de filas y columnas) de la matriz de vectores correspondiente al modelo cargado, por lo que la primer cantidad de la tupla representa en este caso, el número de palabras del vocabulario (400,000 palabras), mientras que la segunda cantidad de la tupla hace referencia al número de dimensiones que poseen los vectores mencionados con anterioridad, que en este caso dichos vectores poseen 300 dimensiones cada uno.

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.

[9]:
```
from collections import Counter

def count_unknown_words(data, vocabulary):
    counter = Counter()
    for row in tqdm(data):
        counter.update(tok for tok in row if tok not in vocabulary)
    return counter

# find out how many times each unknown token occurrs in the corpus
c = count_unknown_words(train_df['tokens'], glove.key_to_index)

# find the total number of tokens in the corpus
total_tokens = train_df['tokens'].map(len).sum()

# find some statistics about occurrences of unknown tokens
unk_tokens = sum(c.values())
percent_unk = unk_tokens / total_tokens
distinct_tokens = len(list(c))

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%}')
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: 5,273,364
number of unknown tokens: 65,817
number of distinct unknown tokens: 24,621
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
```

En el código anterior, en primer lugar se procede a importar la función `Counter` que permite calcular la frecuencia absoluta de los elementos de un conjunto, después se procede a programar la función `count_unknown_words(data, vocabulary)` que recibe como parámetros de entrada `data` que será un listado de tokens y `vocabulary` que se refiere a un vocabulario ya definido previamente, por ejemplo un diccionario o algún otro grupo de palabras que el modelo aún desconozca, por lo que una vez recibidos los argumentos, ésta función en primer lugar, genera un contador vacío para registrar la frecuencia de aparición de palabras no conocidas por el modelo, para después iterar sobre los registros del dataset utilizando la función `tqdm` para mostrar cuánto progreso se ha realizado hasta el momento, luego se procede a actualizar el valor del contador solamente en el caso de las palabras que no existan en el vocabulario conocido por el modelo.

Por otra parte, después de lo anterior, se procede a invocar a la función `count_unknown_words()` para realizar un conteo de las palabras no conocidas para el modelo, además para mandar a llamar a la función antes mencionada, se emplea como argumento de entrada la columna `tokens` del `train_df` junto con `glove.key_to_index` que hace alusión al vocabulario que contiene las palabras conocidas por el modelo. Posterior a ello, se procede a calcular la cantidad total de tokens dentro del corpus, contabilizando los tokens presentes en cada registro de la columna `tokens` de `train_df` y luego sumando éstos mismos, para después calcular el total de palabras no conocidas en el corpus y a su vez calcular la cifra porcentual de palabras no conocidas en el corpus, realizando la división de la cantidad de palabras no conocidas entre la cantidad total de tokens, además de calcular también el número de palabras no conocidas diferentes en el mismo corpus.

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.

[10]:
```
# string values corresponding to the new embeddings
unk_tok = '[UNK]'
pad_tok = '[PAD]'

# initialize the new embedding values
```

```
unk_emb = glove.vectors.mean(axis=0)
pad_emb = np.zeros(300)

# add new embeddings to glove
glove.add_vectors([unk_tok, pad_tok], [unk_emb, pad_emb])

# get token ids corresponding to the new embeddings
unk_id = glove.key_to_index[unk_tok]
pad_id = glove.key_to_index[pad_tok]

unk_id, pad_id
```

[10]: (400000, 400001)

En primer lugar, en el código anterior, se definen 2 tokens especiales: `[UNK]` y `[PAD]`, el primero de ellos, tiene la función de representar aquellas palabras que el modelo no conoce y por tanto que se encuentran fuera del vocabulario del mismo, mientras que el segundo token `[PAD]` se emplea para llevar a cabo un proceso de *padding* en secuencias de menor longitud cuando se requiere que la longitud de todas las secuencias sea idéntica, posteriormente se procede a inicializar los vectores para el embedding de los tokens, el primer vector será `unk_emb` y se asociará con el token de desconocido `[UNK]` y también será inicializado como el promedio de los vectores pertenecientes al conjunto de datos de glove, mientras que de manera similar, el segundo vector se llamará `pad_emb` y estará asociado al token `[PAD]` para realizar el proceso de *padding* y a su vez éste segundo vector será inicializado como un vector que solamente contenga un total de 300 ceros, ésto último debido a que glove usa únicamente vectores de 300 dimensiones. Por otro lado, el siguiente paso a realizar en el proceso es agregar los vectores que contienen los embeddings al modelo glove y vincularlos a los tokens `unk_tok` y `pad_tok`, además de luego proceder a realizar una búsqueda de los identificadores numéricos que correspondan a los tokens recientemente agregados `[UNK]` y `[PAD]`, mismos que son requeridos para trabajar con representaciones en niveles distintos del modelo y por último, se retornan los valores de los identificadores cuantitativos relacionados con los tokens `unk_tok` y `pad_tok`.

[11]:
```
from sklearn.model_selection import train_test_split

train_df, dev_df = train_test_split(train_df, train_size=0.8)
train_df.reset_index(inplace=True)
dev_df.reset_index(inplace=True)
```

En el código, previo primero se importa la función `train_test_split()` que pertenece al módulo `model_selection` de la librería Scikit-Learn, cuyo propósito principal es dividir el conjunto total de datos en un subconjunto para el entrenamiento del modelo y otro para su puesta a prueba, por lo que dicha función es aplicada sobre el dataframe de entrenamiento `train_df` para producir 2 nuevos dataframes: `train_df` y `dev_df`, el primer para entrenar el modelo y el segundo para ponerlo a prueba, además, luego se procede a resetear los índices de las filas de ambos conjuntos de datos y dicha modificación se efectúa en el dataframe de origen para no tener que crear otros dataframes por separado con dicha modificación incorporada.

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

[12]:
```
threshold = 10
tokens = train_df['tokens'].explode().value_counts()
vocabulary = set(tokens[tokens > threshold].index.tolist())
print(f'vocabulary size: {len(vocabulary):,}')
```

```
vocabulary size: 17,445
```

En primera instancia, en el código anterior, primero se establece un umbral mínimo de 10 unidades de longitud para los tokens, el cual señala que en el vocabulario solamente serán incluidas aquellos tokens que tengan una frecuencia de aparición superior a 10 ocasiones, luego se aplica el método `explode()` a los datos de la columna `tokens` del dataframe `train_df`, el cual se encarga de expandir los listados de tokens contenidos en la columna con el objetivo de que a cada token le corresponda una única fila del dataframe, siendo ésto especialmente útil en caso de que la columna `tokens` contenga listas de tokens por cada registro de la misma, posteriormente con `value_counts()` se calcula la frecuencia de aparación correspondiente a cada token, retornando como índice una serie que contiene a los tokens y como valores, las frecuencias de los mismos. Posteriormente, se procede a elegir aquellos tokens que poseen una frecuencia de aparición superior a 10 ocasiones y dichos tokens son transformados en una lista, además de qye también luego ésta lista es transformada a un conjunto sin valores repetidos denominado `vocabulary` y finalmente, se despliega el tamaño del vocabulario resultante de todo el proceso anteriormente descrito.

[13]:
```
# find the length of the longest list of tokens
max_tokens = train_df['tokens'].map(len).max()

# return unk_id for infrequent tokens too
def get_id(tok):
    if tok in vocabulary:
        return glove.key_to_index.get(tok, unk_id)
    else:
        return unk_id

# function that gets a list of tokens and returns a list of token ids,
# with padding added accordingly
def token_ids(tokens):
    tok_ids = [get_id(tok) for tok in tokens]

    pad_len = max_tokens - len(tok_ids)

    return tok_ids + [pad_id] * pad_len

# add new column to the dataframe
train_df['token ids'] = train_df['tokens'].progress_map(token_ids)
train_df
```

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

[13]:

	index	class index	class	\
0	9117	1	World	
1	99832	3	Business	

```
2       10664            3  Business
3       73176            4  Sci/Tech
4      104495            4  Sci/Tech
…       …          …       …
95995   89461            1     World
95996   60621            1     World
95997   34087            1     World
95998   58068            1     World
95999   92976            4  Sci/Tech

                                                    title  \
0          Najaf's Residents Feel Trapped in Battle (AP)
1                U.S. FDA Adds Restrictions to Acne Drug
2              Smithfield Foods Profit More Than Doubles
3      PluggedIn: The OQO Is Not Just Another Handhel…
4                       IBM invigorates LTO tape storage
…                                                     …
95995    Bush, Blair See Hope for Palestinian State (AP)
95996    Ex-Soldiers Vow to Bring Order to Haiti Capital
95997  Musharraf says U.S. must address root of terro…
95998         Nuclear materials  #39;vanish #39; in Iraq
95999  In Brief: Bowstreet unveils pre-packaged porta…

                                                description  \
0      AP - For nearly three weeks, Amer al-Jamali ha…
1       WASHINGTON (Reuters) - Roche's acne drug Accu…
2      Smithfield Foods Inc. (SFD.N: Quote, Profile, …
3       SAN FRANCISCO (Reuters) - A full-fledged Wind…
4      LTO (linear tape open)-based drives are invigo…
…                                                     …
95995  AP - As Yasser Arafat was buried, President Bu…
95996  Ex-soldiers who helped topple former President…
95997  Reuters - The United States could lose its war…
95998  Equipment and materials that could be used to …
95999  Bowstreet this week launched its Enterprise Po…

                                                     text  \
0      najaf's residents feel trapped in battle (ap) …
1      u.s. fda adds restrictions to acne drug  washi…
2      smithfield foods profit more than doubles smit…
3      pluggedin: the oqo is not just another handhel…
4      ibm invigorates lto tape storage lto (linear t…
…                                                     …
95995  bush, blair see hope for palestinian state (ap…
95996  ex-soldiers vow to bring order to haiti capita…
95997  musharraf says u.s. must address root of terro…
95998  nuclear materials  #39;vanish #39; in iraq equ…
```

```
95999  in brief: bowstreet unveils pre-packaged porta…

                                                  tokens  \
0      [najaf, 's, residents, feel, trapped, in, batt…
1      [u.s., fda, adds, restrictions, to, acne, drug…
2      [smithfield, foods, profit, more, than, double…
3      [pluggedin, :, the, oqo, is, not, just, anothe…
4      [ibm, invigorates, lto, tape, storage, lto, (,…
…                                                     …
95995  [bush, ,, blair, see, hope, for, palestinian, …
95996  [ex-soldiers, vow, to, bring, order, to, haiti…
95997  [musharraf, says, u.s., must, address, root, o…
95998  [nuclear, materials, #, 39, ;, vanish, #, 39, …
95999  [in, brief, :, bowstreet, unveils, pre-package…

                                                token ids
0      [10709, 9, 1048, 998, 4799, 6, 903, 23, 1582, …
1      [99, 5584, 2144, 3252, 4, 400000, 780, 289, 23…
2      [34026, 5008, 1269, 56, 73, 4229, 34026, 5008,…
3      [400000, 45, 0, 293697, 14, 36, 120, 170, 2099…
4      [5199, 400000, 400000, 4143, 4418, 400000, 23,…
…                                                     …
95995  [272, 1, 2356, 253, 824, 10, 463, 92, 23, 1582…
95996  [223970, 12887, 4, 938, 460, 4, 3836, 351, 223…
95997  [3820, 210, 99, 390, 1476, 5440, 3, 1291, 23, …
95998  [490, 2176, 2749, 3403, 89, 25736, 2749, 3403,…
95999  [6, 2461, 45, 400000, 20465, 400000, 12174, 83…

[96000 rows x 8 columns]
```

En la celda de código anterior, el primer paso consiste en calcular la cantidad de elementos en cada lista de tokens de la columna `tokens` del dataframe `train_df` y posterior a ello, calcula la longitud máxima de entre todas las listas de tokens, luego la función `get_id(tok)` recibe como parámetro de entrada un token y retorna su correspondiente identificador o id, por lo que si dicho id sí existe dentro del vocabulario del modelo, se procede a buscar ese id en el modelo de embeddings de glove, además, en caso de que el token buscado no exista en el vocabulario, entonces se retorna un valor almacenado en la variable `unk_id` que a su vez simboliza un token que el modelo desconoce. Después de todo lo anterior, la función `token_ids(tokens)` se encarga de transformar una lista de tokens en una lista de identificadores por medio de la función `get_id` antes descrita, posteriormente calcula la cantidad necesaria de *padding* para que la longitud de la lista sea idéntica a la definida en la variable `max_tokens` y finalmente, retorna la lista de identificadores con el *padding* agregado al final de los mismos, además el valor correspondiente al *padding* viene representado por `pad_id`. Además como último paso del proceso, la función `token_ids(tokens)` es aplicada a cada uno de los registros de la columna `tokens` del dataframe `train_df` y a su vez se genera una nueva columna denominada `token ids` con las listas de los identificadores respectivos ya con valores en su interior.

[14]:
```
max_tokens = dev_df['tokens'].map(len).max()
dev_df['token ids'] = dev_df['tokens'].progress_map(token_ids)
dev_df
```

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

[14]:
```
        index  class index     class  \
0       60975            1     World
1       50392            4  Sci/Tech
2        9308            3  Business
3       35222            3  Business
4       40082            1     World
…       …             …        …
23995   49573            1     World
23996   40410            4  Sci/Tech
23997   70471            2    Sports
23998    7942            4  Sci/Tech
23999   42304            1     World

                                                   title  \
0       Sharon Accepts Plan to Reduce Gaza Army Operat…
1       Internet Key Battleground in Wildlife Crime Fight
2               July Durable Good Orders Rise 1.7 Percent
3               Growing Signs of a Slowing on Wall Street
4                           The New Faces of Reality TV
…                                                    …
23995         Iraqi Kidnappers Release 2 Indonesian Women
23996                  Big Wi-Fi Project for Philadelphia
23997                                    Owen scores again
23998   US Online Retail Sales Expected To Double In S…
23999   Egyptian holding company says it has heard fou…

                                             description  \
0       Israeli Prime Minister Ariel Sharon accepted a…
1       Why trawl through a sweaty illegal\wildlife ma…
2       America's factories saw orders for costly manu…
3       all Street #39;s earnings growth, fueled by tw…
4       The introduction of children to the genre was …
…                                                    …
23995   Two Indonesian women held hostage for several …
23996   What would Benjamin Franklin say? Philadelphia…
23997   Michael Owen scored the winner for Real Madrid…
23998   Online retail sales in the US are expected to …
23999   Egypt said Tuesday that Iraqi kidnappers had f…

                                                    text  \
0       sharon accepts plan to reduce gaza army operat…
1       internet key battleground in wildlife crime fi…
```

```
2      july durable good orders rise 1.7 percent amer…
3      growing signs of a slowing on wall street all …
4      the new faces of reality tv the introduction o…
…                                                    …
23995  iraqi kidnappers release 2 indonesian women tw…
23996  big wi-fi project for philadelphia what would …
23997  owen scores again michael owen scored the winn…
23998  us online retail sales expected to double in s…
23999  egyptian holding company says it has heard fou…

                                                  tokens  \
0      [sharon, accepts, plan, to, reduce, gaza, army…
1      [internet, key, battleground, in, wildlife, cr…
2      [july, durable, good, orders, rise, 1.7, perce…
3      [growing, signs, of, a, slowing, on, wall, str…
4      [the, new, faces, of, reality, tv, the, introd…
…                                                    …
23995  [iraqi, kidnappers, release, 2, indonesian, wo…
23996  [big, wi-fi, project, for, philadelphia, what,…
23997  [owen, scores, again, michael, owen, scored, t…
23998  [us, online, retail, sales, expected, to, doub…
23999  [egyptian, holding, company, says, it, has, he…

                                               token ids
0      [2548, 9889, 394, 4, 1680, 1166, 330, 957, 1, …
1      [925, 638, 14944, 6, 4446, 1340, 838, 738, 400…
2      [375, 10699, 219, 1949, 1027, 6262, 72, 453, 9…
3      [988, 1867, 3, 7, 6515, 13, 1015, 491, 64, 491…
4      [0, 50, 1919, 3, 2532, 816, 0, 4344, 3, 271, 4…
…                                                    …
23995  [710, 9349, 713, 232, 2656, 266, 55, 2656, 266…
23996  [365, 39300, 716, 10, 2201, 102, 54, 4067, 503…
23997  [7116, 2776, 378, 785, 7116, 878, 0, 1364, 10,…
23998  [95, 1292, 2645, 526, 287, 4, 1278, 6, 228, 82…
23999  [2434, 1383, 128, 210, 20, 31, 1435, 133, 2434…

[24000 rows x 8 columns]
```

En las líneas de código anteriores, primero se lleva a cabo el cálculo de la longitud de cada una de las listas de tokens antes obtenidas y en base a todos los valores de longitud calculados, se selecciona el mayor de todos ellos y posteriormente, la función `token_ids` se aplica sobre la columna `tokens` de `dev_df` y los resultados a su vez se almacenan en una nueva columna llamada `token ids` dentro de ese mismo dataframe, por lo que al final, los resultados almacenados en esa columna son básicamente el conteo de la aparación de cada token de la lista contenida en cada fila de la columna `tokens` en el respectivo resultado de esa misma fila en la columna `text`.

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.

[15]:
```
from torch.utils.data import Dataset

class MyDataset(Dataset):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __len__(self):
        return len(self.y)

    def __getitem__(self, index):
        x = torch.tensor(self.x[index])
        y = torch.tensor(self.y[index])
        return x, y
```

En la celda de código previa, el primer paso es importar la función `Dataset` que pertenece al módulo `utils.data` de la librería torch, la cual tiene el principal propósito de generar una interfaz para la creación de datasets personalizados, posteriormente se procede a definir una clase llamada `MyDataset(Dataset)`, la cual prácticamente es una clase derivada o hija de la clase `Dataset`, por lo que podrá ser usada como parte de la infraestructura de la librería PyTorch para garantizar un adecuado manejo de los datos durate todo el proceso de modelación, además, el primer método de la clase `MyDataset` es su constructor que se llama igual que la clase y se encarga de asignar los valores iniciales a los atributos de dicha clase `x` y `y` al momento de crear un objeto de la clase `MyDataset`. Adicionalmente, el método `_len_(self)` retorna el tamaño del dataset, en otras palabras, el número de ejemplos que éste mismo contiene con base en la cantidad de elementos del atributo `y`, lo cual requiere la clase `Dataset` para poder llevar a cabo un proceso iterativo sobre el dataset.

Finalmente, por otro lado, el último método definido en la clase `_getitem_(self, index)` tiene el objetivo principal de retornar el ejemplo que ocupa la posición especificada por la variable `index` dentro del dataset, además se encarga de la transformación de los datos en tensores de PyTorch para luego poder efectuar operaciones matemáticas con ellos en base a las funciones de PyTorch, por lo que en última instancia, se devuelve como resultado una tupla que contiene un ejemplo en particular, además de su respectiva label o etiqueta.

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

[17]:
```
from torch import nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self, vectors, pad_id, hidden_dim, output_dim, dropout):
        super().__init__()
        # embeddings must be a tensor
        if not torch.is_tensor(vectors):
            vectors = torch.tensor(vectors)
        # keep padding id
```

```
        self.padding_idx = pad_id
        # embedding layer
        self.embs = nn.Embedding.from_pretrained(vectors, padding_idx=pad_id)
        # feedforward layers
        self.layers = nn.Sequential(
            nn.Dropout(dropout),
            nn.Linear(vectors.shape[1], hidden_dim),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, output_dim),
        )

    def forward(self, x):
        # get boolean array with padding elements set to false
        not_padding = torch.isin(x, self.padding_idx, invert=True)
        # get lengths of examples (excluding padding)
        lengths = torch.count_nonzero(not_padding, axis=1)
        # get embeddings
        x = self.embs(x)
        # calculate means
        x = x.sum(dim=1) / lengths.unsqueeze(dim=1)
        # pass to rest of the model
        output = self.layers(x)
        # calculate softmax if we're not in training mode
        #if not self.training:
        #    output = F.softmax(output, dim=1)
        return output
```

En el bloque anterior de código, lo primero que se hace es importar la función `nn` de la librería de PyTorch, además del módulo `nn.functional` igualmente de la librería PyTorch, además de asignar el nombre personalizado `F` a `nn.functional`, para posteriormente, definir la primera clase llamada `Model(nn.Module)`, en la cual, primero se define el método `_init_`, mismo que se encarga básicamente de inicializar los atributos de la clase al crear un objeto o instancia de la misma, además dicho método recibe como argumentos de entrada: `vectors`, `pad_id`, `hidden_dim`, `output_dim` y `dropout`, por un lado, `vectors` hace referencia a una matriz de embeddings preentrenados en la que cada registro corresponde a la representación a manera de vector de una cierta palabra en el vocabulario, además, `pad_id` se refiere al índice del token para *padding*, usado para el manejo eficiente de secuencias cuyo tamaño es cambiante (palaras completadas con el *padding* en caso de que las longitudes de palabras sean menores), además, `self.embs` se refiere a una capa de embedding que recibe índices de palabras como argumentos de entrada y retorna sus correspondientes representaciones a manera de vectores mediante la utilización de los embeddings preentrenados ubicados en `vectors`, por otro lado, `self.layers` es una secuencia de capas densas que poseen ciertas funciones de activación, tales como Relu, lo cual sirve para el procesamiento de los vectores de input y a su vez calcular la predicción definitiva del modelo. Finalmente, el método `forward` es aquel que define todo el procedimiento hacia adelante de la red, en el cual se reciben los argumentos de entrada y éstos mismos son enviados a través de la red y como resultado se obtiene una determinada predicción de parte del modelo, además el método `forward` posee los siguientes componentes:

- `not_padding` -> máscara de tipo booleano para detectar posiciones que no correspondan a *padding*.
- `lengths` -> Calcula los tokens verdaderos de una secuencia sin tomar en cuenta tokens de *padding*.
- **Embeddings:** Transforman secuencias de índices de palabras en sus vectores respectivos por medio de una capa de embedding.
- **Promedio:** obtiene el promedio de embeddings a lo largo de una determinada secuencia sin tomar en consideración componentes de *padding*.
- **Capas densas:** la representación de los embeddings medios (calculados con promedio) pasa por aquellas capas densas de `self.layers`, mismas que se encargan de generar los resultados finales derivados del modelo.

Nota: Es posible añadir un argumento opcional llamado `softmax` para indicar que los resultados de salida del modelo tienen que ser probabilidades en caso de que el modelo no se encuentre en entrenamiento.

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

[18]:
```
from torch import optim
from torch.utils.data import DataLoader
from sklearn.metrics import accuracy_score

# hyperparameters
lr = 1e-3
weight_decay = 0
batch_size = 500
shuffle = True
n_epochs = 5
hidden_dim = 50
output_dim = len(labels)
dropout = 0.1
vectors = glove.vectors

# initialize the model, loss function, optimizer, and data-loader
model = Model(vectors, pad_id, hidden_dim, output_dim, dropout).to(device)
loss_func = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=lr, weight_decay=weight_decay)
train_ds = MyDataset(train_df['token ids'], train_df['class index'] - 1)
train_dl = DataLoader(train_ds, batch_size=batch_size, shuffle=shuffle)
dev_ds = MyDataset(dev_df['token ids'], dev_df['class index'] - 1)
dev_dl = DataLoader(dev_ds, batch_size=batch_size, shuffle=shuffle)

train_loss = []
train_acc = []
```

```
dev_loss = []
dev_acc = []

# train the model
for epoch in range(n_epochs):
    losses = []
    gold = []
    pred = []
    model.train()
    for X, y_true in tqdm(train_dl, desc=f'epoch {epoch+1} (train)'):
        # clear gradients
        model.zero_grad()
        # send batch to right device
        X = X.to(device)
        y_true = y_true.to(device).long()
        # predict label scores
        y_pred = model(X)
        # compute loss
        loss = loss_func(y_pred, y_true)
        # accumulate for plotting
        losses.append(loss.detach().cpu().item())
        gold.append(y_true.detach().cpu().numpy())
        pred.append(np.argmax(y_pred.detach().cpu().numpy(), axis=1))
        # backpropagate
        loss.backward()
        # optimize model parameters
        optimizer.step()
    train_loss.append(np.mean(losses))
    train_acc.append(accuracy_score(np.concatenate(gold), np.concatenate(pred)))

    model.eval()
    with torch.no_grad():
        losses = []
        gold = []
        pred = []
        for X, y_true in tqdm(dev_dl, desc=f'epoch {epoch+1} (dev)'):
            X = X.to(device)
            y_true = y_true.to(device).long()
            y_pred = model(X)
            loss = loss_func(y_pred, y_true)
            losses.append(loss.cpu().item())
            gold.append(y_true.cpu().numpy())
            pred.append(np.argmax(y_pred.cpu().numpy(), axis=1))
        dev_loss.append(np.mean(losses))
        dev_acc.append(accuracy_score(np.concatenate(gold), np.
 ↪concatenate(pred)))
```

```
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]
```

En el bloque previo de código, primero se procede a la importación de librerías y módulos tales como `optim` de PyTorch para establecer el tipo de optimizador con el que se ajustarán los parámetros del modelo, `utils.data.loader` también de PyTorch, para simplificar ciertas tareas involucradas en el manejo de datos, además de `metrics.accuracy_score` de la librería Scikit-Learn. Posterior a ello, ahora se procede a definir los valores iniciales de los hiperparámetros del modelo, en este caso, el parámetro `lr` (tasa de aprendizaje) se inicia en 1e-03, mientras que `weight_decay` se establece en 0 para indicar que se aplicará al modelo la regularización tipo L2 para evitar que el modelo se sobreajuste, además de inicializar el parámetro `batch_size` en 500, indicando que en cada iteración de la fase de entrenamiento del modelo, se utilizarán 500 muestras por cada lote de datos procesado, además con `shuffle = True` se indica que los datos deberán ser dispuestos al azar antes de ser procesados, además el entrenamiento se llevará a cabo en un total de 5 épocas, además de que también el modelo en cuestión tendrá una capa oculta de 50 dimensiones, junto con otra capa de salida igual a la cantidad de etiquetas que haya, además se usará una proporción de 0.1 para la regularización aplicada al modelo durante su entrenamiento, junto con el conjunto de embeddings preentrenados.

Por otra parte, después de definir los valores iniciales de los hiperparámetros, el siguiente paso consiste en crear una instancia del modelo, mismo que consistirá en una red neuronal conformada por embeddings preentrenados, una función para llevar a cabo el *padding*, además de capas de salida y ocultas con droput incluido, después se procede a establecer el tipo de función de pérdida para entrenar el modelo que en este caso se usará la función de entropía cruzada como función de pérdida `CrossEntropyLoss`, misma que es bastante usada en problemas clasificatorios. Adicionalmente, el modelo también se inicializará con el optimizador **Adam** cuya función consiste en ajustar los parámetros del modelo a partir del gradiente obtenido mediante un proceso de backpropagation y como último paso de la etapa de inicialización del modelo, se generan 2 DataLoaders, el primero para establecer el dataset de entrenamiento del modelo y el segundo para definir el dataset de prueba para el mismo, los cuales se encargan de cargar los datos por lotes y para el caso particular del entrenamiento, los datos son "mezclados" de forma aleatoria.

Además de lo anterior, también cabe señalar que respecto al entrenamiento del modelo, el primer paso consiste en que dado que el modelo es entrenado con 5 iteraciones en total, en cada época primero se define el modo de entrenamiento y para cada lote de datos se realizan las acciones a continuación en el orden especificado:

1. Se borran los datos que tengan los gradientes acumulados de los parámetros del modelo actualmente.

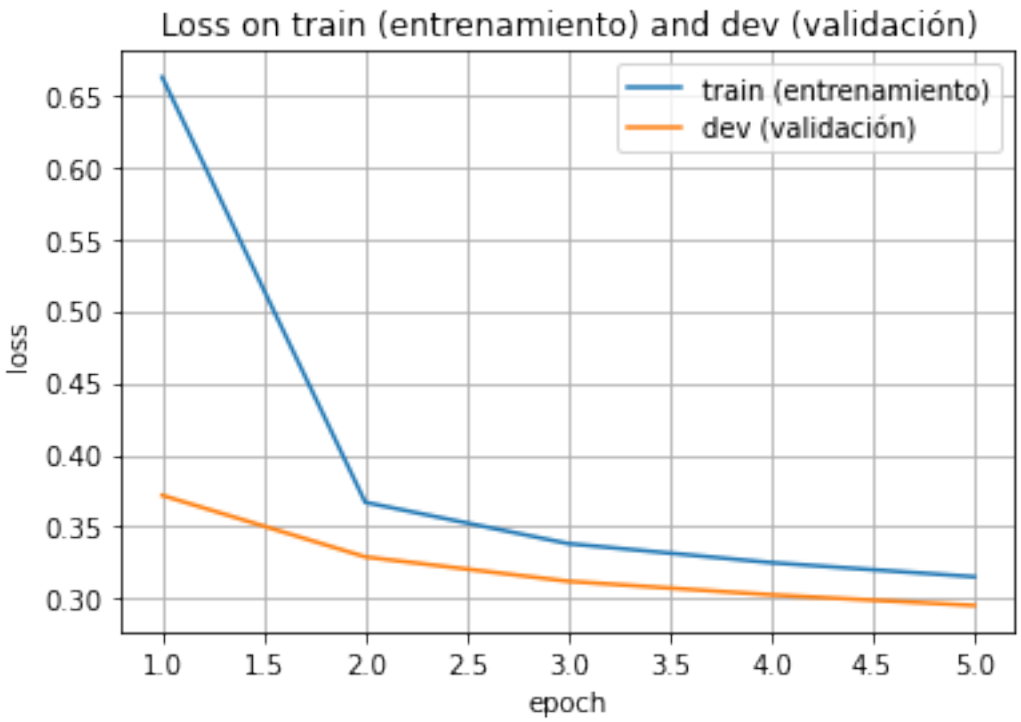
2. Los datos de entrada y las labels reales son enviadas al dispositivo computacional correspondiente.
3. El modelo calcula las predicciones correspondientes a los datos de entrada.
4. Efectuar el cálculo de la función de pérdida del modelo, mediante la diferencia entre la label real y la predicha por el modelo.
5. Calcular gradientes de la función de pérdida en relación a los valores actuales de los parámetros del modelo.
6. Finalmente, el optimizador Adam se encarga de ajustar los parámetros del modelo a partir de los gradientes anteriormente calculados.

Por otro lado, posterior a cada época de entrenamiento del modelo, se procede a evaluar el modelo con el conjunto de validación o prueba, lo cual implica que primeramente se define la manera de evaluación (por ejemplo normalización batch o batch_normalization) y acto seguido, se verifica que no se calculen gradientes en el transcurso de la etapa de evaluación del modelo y posteriormente se repite el procedimiento de predecir y calcular los valores de pérdida, pero ahora aplicado al dataset de prueba o validación.

Por último, los valores promedio de pérdida para el entrenamiento y validación del modelo se guardan en un par de variables llamadas `train_loss` y `dev_loss`, respectivamente, mientras que de forma similar, los niveles de precisión del modelo en cada época de entrenamiento y validación también son almacenados, en este caso, éstos se almacenan en otras 2 variables denominadas `train_acc` y `dev_acc`, respectivamente.

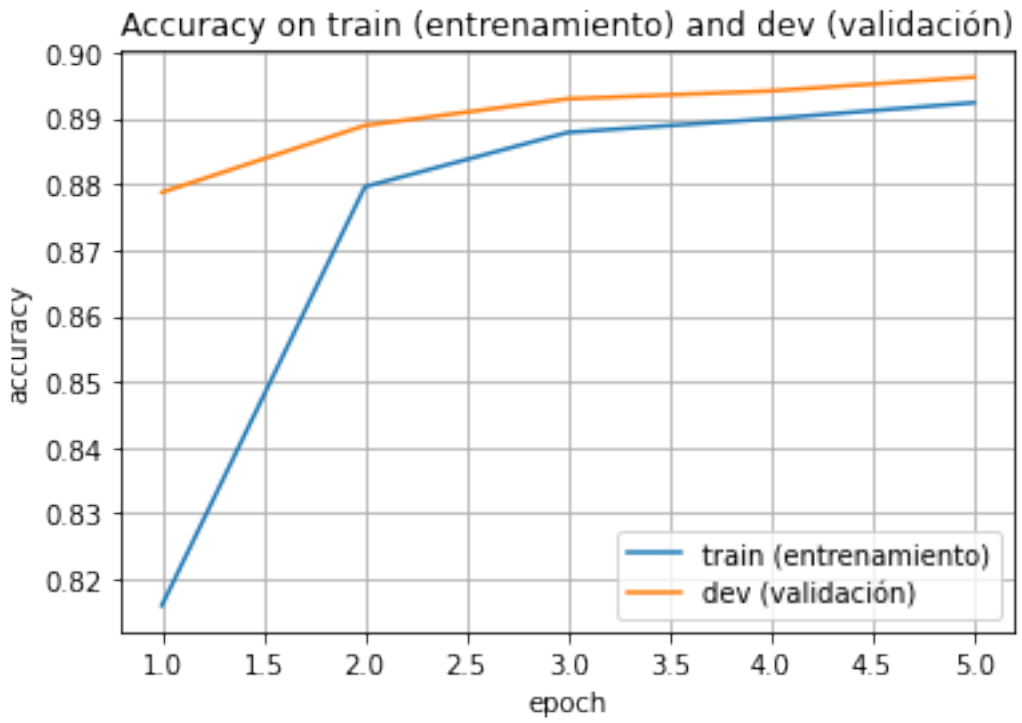
Let's plot the loss and accuracy on dev:

[19]:
```
import matplotlib.pyplot as plt
%matplotlib inline

x = np.arange(n_epochs) + 1

plt.plot(x, train_loss)
plt.plot(x, dev_loss)
plt.legend(['train (entrenamiento)', 'dev (validación)'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.title('Loss on train (entrenamiento) and dev (validación)')
plt.grid(True)
```



En el gráfico anterior, se grafican los valores de pérdida del modelo contra el número de épocas de entrenamiento del mismo, en el cual es posible apreciar que los valores de la función de pérdida tanto para entrenamiento como para validación del modelo presentan en ambos casos un comportamiento o tendencia decreciente, dado que conforme incrementa el número de épocas de entrenamiento transcurridas, los valores de la función de pérdida tienden a disminuir, lo cual es un indicativo de que a medida que transcurren más épocas de entrenamiento, el modelo aumenta su capacidad para aprender patrones o tendencias significativas en sus datos de entrenamiento, provocando que las predicciones derivadas del mismo tengan un margen de error cada vez menor con respecto a los datos reales (menor sesgo de las predicciones), por lo que de forma similar, en la etapa de validación también se observa un decremento de la pérdida conforme avanzan las épocas de validación del modelo, señalando que conforme avanza la etapa de validación, los parámetros del modelo se ajustan cada vez de mejor manera al dataset destinado específicamente para la validación, por lo que el modelo mejora gradualmente su capacidad para generalizar su conocimiento de los patrones aprendidos en su entrenamiento a nuevos datos que aún no conozca y realizar una predicción mayormente precisa de dichos datos desconocidos.

[20]:
```
plt.plot(x, train_acc)
plt.plot(x, dev_acc)
plt.legend(['train (entrenamiento)', 'dev (validación)'])
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.title('Accuracy on train (entrenamiento) and dev (validación)')
plt.grid(True)
```



En el gráfico anterior de precisión (accuracy) tanto en la etapa de entrenamiento como de validación del modelo, es posible observar que conforme la cantidad de épocas de entrenamiento y validación aumenta, los valores de precisión o accuracy también tienden a incrementar, lo cual indica que a medida que se aumenta el número o cantidad de épocas de validación y entrenamiento del modelo, las diferencias entre los datos reales que se desean predecir y aquellos datos predichos por el modelo, son cada vez de menor magnitud, lo cual significa que las predicciones derivadas del modelo son cada vez más cercanas a los datos verdaderos, motivo por el cual, se afirma que la precisión de las predicciones resulta ser mayor conforme transcurre también una mayor cantidad de épocas de entrenamiento y validación, motivo por el cual, en última instancia, se logra tener un modelo con un porcentaje de precisión ubicado entre el 89% y el 90%, indicando que en términos generales, se tiene un modelo mayormente adecuado para realizar predicciones.

Next, we evaluate on the testing partition:

[21]:
```
# repeat all preprocessing done above, this time on the test set
test_df = pd.read_csv('test.csv', header=None).iloc[1:, :]
test_df.columns = ['class index', 'title', 'description']
test_df['text'] = test_df['title'].str.lower() + " " + test_df['description'].
 ↪str.lower()
test_df['text'] = test_df['text'].str.replace('\\', ' ', regex=False)
test_df['tokens'] = test_df['text'].progress_map(word_tokenize)
max_tokens = dev_df['tokens'].map(len).max()
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]
```

En el bloque de código previo prácticamente se repite el mismo preprocesamiento de texto realizado inicialmente para el conjunto de datos de entrenamiento, pero en esta ocasión aplicado al dataset de validación o prueba, por lo que en primera instancia, se procede a leer los datos del archivo `test.csv` para validar y/o probar el modelo para posteriormente asignar nombres a las columnas de dicho dataset, además de crear la columna `text` también para éste dataset de prueba, igualmente concatenando los contenidos de las columnas `title` y `description` correspondientes a ese dataset y agregando ciertos caractéres adicionales tales como espacios en blanco entre palabras, además de convertir todas las letras del texto a minúsculas y reemplazar todas las ocurrencias de las diagonales \\ o \ por espacios en blanco. Además, también se procede a crear la columna `tokens` para el dataset de prueba, que contiene los contenidos de la columna `text`, pero ya con la tokenización aplicada sobre ellos, además también se calcula la longitud máxima de entre todas las longitudes correspondientes a todas las listas de tokens del dataset de prueba y por último, se calcula la columna `token ids` para el dataset de prueba, reemplazando cada token de las listas de tokens por su correspondiente id numérico, por lo que en última instancia, lo que se almacenará en la columna `token ids` serán listas numéricas enteras, donde cada valor corresponderá al id numérico de un token particular.

[22]:
```
from sklearn.metrics import classification_report

# set model to evaluation mode
model.eval()

# Convertir los valores de la columna 'class index' a numéricos enteros

test_df['class index'] = test_df['class index'].astype("int") - 1

"""Detectar si es necesario truncar la lista de token ids para
que sea de la longitud máxima especificada en max_tokens"""

if len(test_df['token ids']) > max_tokens:

    tokenIds_reduced = test_df[['token ids', 'class index']][:max_tokens]

else:

    tokenIds_reduced = test_df[['token ids', 'class index']]

dataset = MyDataset(tokenIds_reduced['token ids'], tokenIds_reduced['class␣
 ↪index'])
dataset.x.reset_index(drop=True, inplace=True)
dataset.y.reset_index(drop=True, inplace=True)
data_loader = DataLoader(dataset, batch_size=batch_size)
y_pred = []

# don't store gradients
with torch.no_grad():
```

```
    for X, _ in tqdm(data_loader):
        X = X.to(device)
        # predict one class per example
        y = torch.argmax(model(X), dim=1)
        # convert tensor to numpy array (sending it back to the cpu if needed)
        y_pred.append(y.cpu().numpy())

    print(classification_report(dataset.y, np.concatenate(y_pred),␣
↪target_names=labels))
```

```
  0%|          | 0/1 [00:00<?, ?it/s]
              precision    recall  f1-score   support

       World       0.95      0.85      0.90        67
      Sports       0.96      0.96      0.96        71
    Business       0.71      0.83      0.76        41
    Sci/Tech       0.90      0.90      0.90        70

    accuracy                           0.89       249
   macro avg       0.88      0.88      0.88       249
weighted avg       0.90      0.89      0.89       249
```

En el reporte de métricas de desempeño anterior, se aprecia que el modelo fue capaz de predecir con un nivel de precisión del 96% aquellas instancias que pertenecen a la categoría “World”, siendo el nivel de precisión máximo, por lo cual, eso significa que la categoría “World” es aquella que tuvo el mayor resultado de precisión, es decir, que del total de instancias que fueron predichas por el modelo, el 96% de ellas fue clasificado correctamente por el mismo, mientras que de forma similar, el mayor porcentaje de recall lo obtuvo la clase “Sports”, siendo dicho porcentaje del 96%, por lo cual eso indica que del total de instancias que realmente pertenecen a la clase “Sports”, el 96% de ellas fueron clasificadas como tales por el modelo. Adicionalmente, también es importante mencionar que en cuanto al F1-score, el mayor porcentaje de ésta métrica, específicamente del 96% fue igualmente para la clase “Sports”, lo cual hace referencia a que combinando las métricas de recall y precisión, el modelo posee una capacidad muy elevada para clasificar correctamente instancias que pertenecen específicamente a la clase “Sports”, no obstante el modelo posee una capacidad también muy alta del 90% para clasificar datos pertenecientes a las categorías “World”, “Sci/Tech”, además, dado que todos los valores de F1-score para el modelo son superiores al 80%, es posible afirmar que en general, el modelo generado y entrenado anteriormente es mayormente adecuado para predecir la clase verdadera a la que pertenecen las instancias o datos, por lo que resulta ser de buena calidad.