```
!pip uninstall torch torchvision torchaudio torchtext torchdata -y
!pip cache purge
!pip install torch==2.0.1 torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
!pip install torchtext torchdata==0.6.1


import torch
import torchtext

print(f"PyTorch Version: {torch.__version__}")
print(f"TorchText Version: {torchtext.__version__}")
print(f"CUDA Available: {torch.cuda.is_available()}")
```

```
      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.4/4.4 MB 87.7 MB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch==2.0.1) (3
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->torch
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->torchvision) (3.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->torchvision
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->torchvision
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->torch==2.0.1)
Building wheels for collected packages: lit
  Building wheel for lit (setup.py) ... done
  Created wheel for lit: filename=lit-15.0.7-py3-none-any.whl size=89991 sha256=f406022657e5212531194899f8f46da2fd2d66de
  Stored in directory: /root/.cache/pip/wheels/fc/5d/45/34fe9945d5e45e261134e72284395be36c2d4828af38e2b0fe
Successfully built lit
Installing collected packages: lit, triton, torch, torchvision, torchaudio
  Attempting uninstall: triton
    Found existing installation: triton 3.2.0
    Uninstalling triton-3.2.0:
      Successfully uninstalled triton-3.2.0
Successfully installed lit-15.0.7 torch-2.0.1+cu118 torchaudio-2.0.2+cu118 torchvision-0.15.2+cu118 triton-2.0.0
Collecting torchtext
  Downloading torchtext-0.18.0-cp311-cp311-manylinux1_x86_64.whl.metadata (7.9 kB)
Collecting torchdata==0.6.1
  Downloading torchdata-0.6.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Requirement already satisfied: urllib3>=1.25 in /usr/local/lib/python3.11/dist-packages (from torchdata==0.6.1) (2.3.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from torchdata==0.6.1) (2.32.3)
Requirement already satisfied: torch==2.0.1 in /usr/local/lib/python3.11/dist-packages (from torchdata==0.6.1) (2.0.1+cu
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch==2.0.1->torchdata==0.6.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from torch==2.0.1->torchdat
Requirement already satisfied: sympy in /usr/local/lib/python3.11/dist-packages (from torch==2.0.1->torchdata==0.6.1) (1
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch==2.0.1->torchdata==0.6.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.1->torchdata==0.6.1) (
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.1->torchdata==0
Requirement already satisfied: cmake in /usr/local/lib/python3.11/dist-packages (from triton==2.0.0->torch==2.0.1->torch
Requirement already satisfied: lit in /usr/local/lib/python3.11/dist-packages (from triton==2.0.0->torch==2.0.1->torchda
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from torchtext) (4.67.1)
INFO: pip is looking at multiple versions of torchtext to determine which version is compatible with other requirements.
Collecting torchtext
  Downloading torchtext-0.17.2-cp311-cp311-manylinux1_x86_64.whl.metadata (7.9 kB)
  Downloading torchtext-0.17.1-cp311-cp311-manylinux1_x86_64.whl.metadata (7.6 kB)
  Downloading torchtext-0.17.0-cp311-cp311-manylinux1_x86_64.whl.metadata (7.6 kB)
  Downloading torchtext-0.16.2-cp311-cp311-manylinux1_x86_64.whl.metadata (7.5 kB)
  Downloading torchtext-0.16.1-cp311-cp311-manylinux1_x86_64.whl.metadata (7.5 kB)
  Downloading torchtext-0.16.0-cp311-cp311-manylinux1_x86_64.whl.metadata (7.5 kB)
  Downloading torchtext-0.15.2-cp311-cp311-manylinux1_x86_64.whl.metadata (7.4 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from torchtext) (2.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->torch
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->torchdata==0.6.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->torchdata==
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch==2.0.1->to
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->torch==2.0.1->
Downloading torchdata-0.6.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.6/4.6 MB 94.9 MB/s eta 0:00:00
Downloading torchtext-0.15.2-cp311-cp311-manylinux1_x86_64.whl (2.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 67.2 MB/s eta 0:00:00
Installing collected packages: torchdata, torchtext
Successfully installed torchdata-0.6.1 torchtext-0.15.2
PyTorch Version: 2.0.1+cu118
TorchText Version: 0.15.2+cpu
CUDA Available: True
```

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("kazanova/sentiment140")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/kazanova/sentiment140?dataset_version_number=2...
100%|██████████| 80.9M/80.9M [00:04<00:00, 17.2MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/kazanova/sentiment140/versions/2
```

```
import numpy as np
import pandas as pd
import re
import random

from sklearn.model_selection import train_test_split

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset


RANDOM_SEED = 3539
torch.manual_seed(RANDOM_SEED)

VOCABULARY_SIZE = 20000
LEARNING_RATE = 1e-4
BATCH_SIZE = 128
NUM_EPOCHS = 10
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

EMBEDDING_DIM = 128
HIDDEN_DIM = 256
OUTPUT_DIM = 2
```

```
A module that was compiled using NumPy 1.x cannot be run in
NumPy 2.0.2 as it may crash. To support both 1.x and 2.x
versions of NumPy, modules must be compiled with NumPy 2.0.
Some module may need to rebuild instead e.g. with 'pybind11>=2.12'.

If you are a user of the module, the easiest solution will be to
downgrade to 'numpy<2' or try to upgrade the affected module.
We expect that some modules will need time to support NumPy 2.

Traceback (most recent call last):  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "/usr/local/lib/python3.11/dist-packages/colab_kernel_launcher.py", line 37, in <module>
    ColabKernelApp.launch_instance()
  File "/usr/local/lib/python3.11/dist-packages/traitlets/config/application.py", line 992, in launch_instance
    app.start()
  File "/usr/local/lib/python3.11/dist-packages/ipykernel/kernelapp.py", line 712, in start
    self.io_loop.start()
  File "/usr/local/lib/python3.11/dist-packages/tornado/platform/asyncio.py", line 205, in start
    self.asyncio_loop.run_forever()
  File "/usr/lib/python3.11/asyncio/base_events.py", line 608, in run_forever
    self._run_once()
  File "/usr/lib/python3.11/asyncio/base_events.py", line 1936, in _run_once
    handle._run()
  File "/usr/lib/python3.11/asyncio/events.py", line 84, in _run
    self._context.run(self._callback, *self._args)
  File "/usr/local/lib/python3.11/dist-packages/ipykernel/kernelbase.py", line 510, in dispatch_queue
    await self.process_one()
  File "/usr/local/lib/python3.11/dist-packages/ipykernel/kernelbase.py", line 499, in process_one
    await dispatch(*args)
  File "/usr/local/lib/python3.11/dist-packages/ipykernel/kernelbase.py", line 406, in dispatch_shell
    await result
  File "/usr/local/lib/python3.11/dist-packages/ipykernel/kernelbase.py", line 730, in execute_request
    reply_content = await reply_content
  File "/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py", line 383, in do_execute
    res = shell.run_cell(
  File "/usr/local/lib/python3.11/dist-packages/ipykernel/zmqshell.py", line 528, in run_cell
    return super().run_cell(*args, **kwargs)
  File "/usr/local/lib/python3.11/dist-packages/IPython/core/interactiveshell.py", line 2975, in run_cell
    result = self._run_cell(
  File "/usr/local/lib/python3.11/dist-packages/IPython/core/interactiveshell.py", line 3030, in _run_cell
    return runner(coro)
  File "/usr/local/lib/python3.11/dist-packages/IPython/core/async_helpers.py", line 78, in _pseudo_sync_runner
    coro.send(None)
  File "/usr/local/lib/python3.11/dist-packages/IPython/core/interactiveshell.py", line 3257, in run_cell_async
    has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
  File "/usr/local/lib/python3.11/dist-packages/IPython/core/interactiveshell.py", line 3473, in run_ast_nodes
    if (await self.run_code(code, result,  async_=asy)):
  File "/usr/local/lib/python3.11/dist-packages/IPython/core/interactiveshell.py", line 3553, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "<ipython-input-3-ee5ea4139b6a>", line 16, in <cell line: 0>
    torch.manual_seed(RANDOM_SEED)
  File "/usr/local/lib/python3.11/dist-packages/torch/random.py", line 46, in manual_seed
    return default_generator.manual_seed(seed)
/usr/local/lib/python3.11/dist-packages/torch/random.py:46: UserWarning: Failed to initialize NumPy: _ARRAY_API not foun
  return default_generator.manual_seed(seed)
```

```
dataset_path = "/root/.cache/kagglehub/datasets/kazanova/sentiment140/versions/2/training.1600000.processed.noemoticon.csv"
```

```
df = pd.read_csv(dataset_path, encoding="ISO-8859-1", header=None)
df.columns = ["sentiment", "id", "date", "query", "user", "text"]
print(df.head())
print(df.shape)
```

```
   sentiment          id                          date     query  \
0          0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1          0  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY
2          0  1467810917  Mon Apr 06 22:19:53 PDT 2009  NO_QUERY
3          0  1467811184  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
4          0  1467811193  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY

              user                                               text
0  _TheSpecialOne_  @switchfoot http://twitpic.com/2y1zl - Awww, t...
1    scotthamilton  is upset that he can't update his Facebook by ...
2         mattycus  @Kenichan I dived many times for the ball. Man...
3          ElleCTF    my whole body feels itchy and like its on fire
4           Karoli  @nationwideclass no, it's not behaving at all....
(1600000, 6)
```

*Data Prepocessing *

```
# we only want label and text

df_ = df[["sentiment", "text"]]

#there is only positive and negative tone
print('type of tone',df_['sentiment'].unique())

#we turn 4 into 1 to make it binary
df_.loc[:, "sentiment"] = df_["sentiment"].replace(4, 1)


print(len(df_))
df_sample=df_
# df_sample = df_.sample(n=20000, random_state=RANDOM_SEED)
print(df_sample.head)
```

```
type of tone [0 4]
1600000
<bound method NDFrame.head of          sentiment                                               text
0                0  @switchfoot http://twitpic.com/2y1zl - Awww, t...
1                0  is upset that he can't update his Facebook by ...
2                0  @Kenichan I dived many times for the ball. Man...
3                0    my whole body feels itchy and like its on fire
4                0  @nationwideclass no, it's not behaving at all....
...            ...                                                ...
1599995          1  Just woke up. Having no school is the best fee...
1599996          1  TheWDB.com - Very cool to hear old Walt interv...
1599997          1  Are you ready for your MoJo Makeover? Ask me f...
1599998          1  Happy 38th Birthday to my boo of alll time!!! ...
1599999          1  happy #charitytuesday @theNSPCC @SparksCharity...

[1600000 rows x 2 columns]>
```

```
#def text cleaning function
def clean_text(text):
    text = text.lower()  # 转换为小写
    text = re.sub(r"http\S+|www\S+|https\S+", "", text, flags=re.MULTILINE)  # remove URL
    text = re.sub(r'\@\w+|\#', '', text)  # remove @ and #
    text = re.sub(r'[^a-zA-Z\s]', '', text)  #only use words
    return text.strip()

df_sample['text'] = df_sample['text'].apply(clean_text)
df_sample.head()
```

```
<ipython-input-6-fad821d6abd7>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df_sample['text'] = df_sample['text'].apply(clean_text)
```

	sentiment	text
0	0	a thats a bummer you shoulda got david carr o...
1	0	is upset that he cant update his facebook by t...
2	0	i dived many times for the ball managed to sav...
3	0	my whole body feels itchy and like its on fire
4	0	no its not behaving at all im mad why am i her...

```
import torchtext
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator
from collections import Counter

# use tokenizer from torchtext
tokenizer = get_tokenizer("basic_english")


df_sample["tokens"] = df_sample["text"].apply(tokenizer)
df_sample.head()
```

```
<ipython-input-7-fe558fd85c28>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df_sample["tokens"] = df_sample["text"].apply(tokenizer)
```

	sentiment	text	tokens
0	0	a thats a bummer you shoulda got david carr o...	[a, thats, a, bummer, you, shoulda, got, david...
1	0	is upset that he cant update his facebook by t...	[is, upset, that, he, cant, update, his, faceb...
2	0	i dived many times for the ball managed to sav...	[i, dived, many, times, for, the, ball, manage...
3	0	my whole body feels itchy and like its on fire	[my, whole, body, feels, itchy, and, like, its...
4	0	no its not behaving at all im mad why am i her...	[no, its, not, behaving, at, all, im, mad, why...

```
#create a counter for vocab
counter = Counter()
for tokens in df_sample["tokens"]:
    counter.update(tokens)


#vocab = Vocab(counter, specials=["<PAD>", "<UNK>"], max_size=VOCABULARY_SIZE)
#this not work becausethe version of torchtext, using build_vocab_from_iterator instead

def yield_tokens(data):
    for tokens in data:
        yield tokens

vocab = build_vocab_from_iterator(yield_tokens(df_sample["tokens"]), max_tokens=20000, specials=["<pad>", "<unk>"])
vocab.set_default_index(vocab["<unk>"])


def text_to_indices(text):
    return [vocab[token] for token in text]

df_sample["indices"] = df_sample["tokens"].apply(text_to_indices)


#train,test,valid split
train_texts, test_texts, train_labels, test_labels = train_test_split(
    df_sample["indices"], df_sample["sentiment"], test_size=0.2, random_state=RANDOM_SEED
)
train_texts, val_texts, train_labels, val_labels = train_test_split(
    train_texts, train_labels, test_size=0.2, random_state=RANDOM_SEED
)
```

```
<ipython-input-8-ffc8ead6565d>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df_sample["indices"] = df_sample["tokens"].apply(text_to_indices)
```

```
#build dataset class
class SentimentDataset(Dataset):
    def __init__(self, texts, labels):
        """
        Custom dataset for tweets dataset.
        Args:
            texts: List of tokenized text sequences (e.g., token indices)
            labels: List of sentiment labels (0 or 1).
        """
        self.texts = list(texts)  # Convert to list to prevent indexing issues
        self.labels = list(labels)  # Convert to list to prevent indexing issues

    def __len__(self):
        """Return dataset size"""
        return len(self.texts)

    def __getitem__(self, idx):
        """
        Get item by index.
        Returns:
            text_tensor: Tensor of token indices.
            label_tensor: Tensor of the corresponding label.
        """
        text_tensor = torch.tensor(self.texts[idx], dtype=torch.long)
        label_tensor = torch.tensor(self.labels[idx], dtype=torch.long)
        return text_tensor, label_tensor



train_dataset = SentimentDataset(train_texts, train_labels)
val_dataset = SentimentDataset(val_texts, val_labels)
test_dataset = SentimentDataset(test_texts, test_labels)

#why we do not directly use dataloader?
# NLP tasks, our text data is usually variable-length sequences, but PyTorch's DataLoader does not perform padding by defaul
#which can cause shape mismatch errors during model training.

# train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
# val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)
# test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

from torch.nn.utils.rnn import pad_sequence
def collate_fn(batch):
    """
    Custom collate function for handling batches of different lengths.
    Args:
        batch: List of (text, label) tuples.
    Returns:
        texts: Padded tensor of token indices.
        labels: Tensor of sentiment labels.
    """
    texts, labels = zip(*batch)
    texts = pad_sequence(texts, batch_first=True, padding_value=vocab["<pad>"])
    labels = torch.tensor(labels, dtype=torch.long)
    return texts, labels


PAD_IDX = vocab["<pad>"]


train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True, collate_fn=collate_fn)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=False, collate_fn=collate_fn)
test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False, collate_fn=collate_fn)
```

```
print('Train')
for batch in train_loader:
    text_batch, label_batch = batch  # Unpack the tuple
    print(f'Text matrix size: {text_batch.size()}')  # (batch_size, seq_length)
    print(f'Target vector size: {label_batch.size()}')  # (batch_size,)
    break
```

```
print('\nValid:')
for batch in val_loader:
    text_batch, label_batch = batch  # Unpack the tuple
    print(f'Text matrix size: {text_batch.size()}')  # (batch_size, seq_length)
    print(f'Target vector size: {label_batch.size()}')  # (batch_size,)
    break

print('\nTest:')
for batch in test_loader:
    text_batch, label_batch = batch  # Unpack the tuple
    print(f'Text matrix size: {text_batch.size()}')  # (batch_size, seq_length)
    print(f'Target vector size: {label_batch.size()}')  # (batch_size,)
    break
```

```
Train
Text matrix size: torch.Size([128, 29])
Target vector size: torch.Size([128])

Valid:
Text matrix size: torch.Size([128, 30])
Target vector size: torch.Size([128])

Test:
Text matrix size: torch.Size([128, 30])
Target vector size: torch.Size([128])
```

Model

```
import torch.nn.utils.rnn as rnn_utils  # ✅ Import this at the top of your script

class RNN(nn.Module):
    def __init__(self, input_dim, embedding_dim, hidden_dim, output_dim, padding_idx):
        super().__init__()
        self.embedding = nn.Embedding(input_dim, embedding_dim, padding_idx=padding_idx)
        self.layer_norm = nn.LayerNorm(embedding_dim)

        self.rnn = nn.LSTM(embedding_dim, hidden_dim, num_layers=2,
                           bidirectional=True, batch_first=True, dropout=0.6)

        self.batch_norm = nn.BatchNorm1d(hidden_dim * 2)
        self.fc = nn.Linear(hidden_dim * 2, output_dim)
        self.dropout = nn.Dropout(0.7) #set high dropout to avoid overfitting

    def forward(self, text, text_length):
        embedded = self.embedding(text)
        embedded = self.layer_norm(embedded)

        text_length = text_length.cpu()
        packed_embedded = rnn_utils.pack_padded_sequence(embedded, text_length, batch_first=True, enforce_sorted=False)
        packed_output, (hidden, cell) = self.rnn(packed_embedded)

        hidden = torch.cat((hidden[-2,:,:], hidden[-1,:,:]), dim=1)
        hidden = self.batch_norm(hidden)
        hidden = self.dropout(hidden)

        out = self.fc(hidden)
        return out
```

```
from torch.optim import AdamW
INPUT_DIM = len(vocab)

torch.manual_seed(RANDOM_SEED)
model = RNN(INPUT_DIM, EMBEDDING_DIM, HIDDEN_DIM, OUTPUT_DIM,PAD_IDX)
model = model.to(DEVICE)
optimizer = AdamW(model.parameters(), lr=LEARNING_RATE, weight_decay=1e-2)
```

Training

```
def compute_binary_accuracy(model, data_loader, device):
    model.eval()
    correct_pred, num_examples = 0, 0
    with torch.no_grad():
        for text, labels in data_loader:
            text_lengths = text.ne(PAD_IDX).sum(dim=1)  # Efficient length calculation
            valid_indices = text_lengths > 0 #ensure length is not zero
            if valid_indices.sum() == 0:
                continue

            text, labels = text[valid_indices], labels[valid_indices]
            text_lengths = text_lengths[valid_indices]

            text, labels = text.to(device), labels.to(device)
            text_lengths = text_lengths.cpu()

            logits = model(text, text_lengths)
            predicted_labels = torch.argmax(logits, dim=1)

            num_examples += labels.size(0)
            correct_pred += (predicted_labels == labels).sum().item()

    return correct_pred / num_examples * 100 # if num_examples > 0 else 0
```

```
import time
start_time = time.time()
for epoch in range(NUM_EPOCHS):
    model.train()

    for batch_idx, (text, labels) in enumerate(train_loader):
        text_lengths = text.ne(PAD_IDX).sum(dim=1)

        #Finding valid index
        valid_indices = text_lengths > 0 #similar to compute_binary_accuracy function, ensure valid sequence lengths
        if valid_indices.sum() == 0:   #if equal to zero, escape this batch
            continue

        text, labels = text[valid_indices], labels[valid_indices] #we only keep the valid input
        text_lengths = text_lengths[valid_indices]

        #recommended by chatgpt for robust, sometimes we do not need these two lines of codes
        text, labels = text.to(DEVICE), labels.to(DEVICE)
        text_lengths = text_lengths.cpu()

        #Forward and Backprop
        logits = model(text, text_lengths)  # Output shape: [batch_size, 2]
        cost = F.cross_entropy(logits, labels)  # Use cross-entropy loss for multi-class classification

        optimizer.zero_grad()
        cost.backward()

        #UPDATE MODEL PARAMETERS
        optimizer.step()

        #Logging
        if batch_idx % 1000 == 0:
            print(f'Epoch: {epoch+1:03d}/{NUM_EPOCHS:03d} | '
                  f'Batch {batch_idx:03d}/{len(train_loader):03d} | '
                  f'Cost: {cost:.4f}')


    with torch.no_grad():
        train_acc = compute_binary_accuracy(model, train_loader, DEVICE)
        val_acc = compute_binary_accuracy(model, val_loader, DEVICE)

        print(f'Training Accuracy: {train_acc:.2f}% | Validation Accuracy: {val_acc:.2f}%')

    print(f'Time elapsed: {(time.time() - start_time)/60:.2f} min')

#Final Test Accuracy
print(f'Total Training Time: {(time.time() - start_time)/60:.2f} min')
print(f'Test Accuracy: {compute_binary_accuracy(model, test_loader, DEVICE):.2f}%')
```

```
Epoch: 001/010 | Batch 000/8000 | Cost: 0.7929
Epoch: 001/010 | Batch 1000/8000 | Cost: 0.5663
Epoch: 001/010 | Batch 2000/8000 | Cost: 0.6028
Epoch: 001/010 | Batch 3000/8000 | Cost: 0.3930
Epoch: 001/010 | Batch 4000/8000 | Cost: 0.4876
Epoch: 001/010 | Batch 5000/8000 | Cost: 0.5991
Epoch: 001/010 | Batch 6000/8000 | Cost: 0.4820
Epoch: 001/010 | Batch 7000/8000 | Cost: 0.4515
Training Accuracy: 79.23% | Validation Accuracy: 78.88%
Time elapsed: 3.05 min
Epoch: 002/010 | Batch 000/8000 | Cost: 0.4784
Epoch: 002/010 | Batch 1000/8000 | Cost: 0.4810
Epoch: 002/010 | Batch 2000/8000 | Cost: 0.5592
Epoch: 002/010 | Batch 3000/8000 | Cost: 0.4169
```

```
Epoch: 002/010 | Batch 4000/8000 | Cost: 0.4640
Epoch: 002/010 | Batch 5000/8000 | Cost: 0.3703
Epoch: 002/010 | Batch 6000/8000 | Cost: 0.4481
Epoch: 002/010 | Batch 7000/8000 | Cost: 0.4305
Training Accuracy: 81.07% | Validation Accuracy: 80.36%
Time elapsed: 6.07 min
Epoch: 003/010 | Batch 000/8000 | Cost: 0.4387
Epoch: 003/010 | Batch 1000/8000 | Cost: 0.3700
Epoch: 003/010 | Batch 2000/8000 | Cost: 0.4224
Epoch: 003/010 | Batch 3000/8000 | Cost: 0.5151
Epoch: 003/010 | Batch 4000/8000 | Cost: 0.3931
Epoch: 003/010 | Batch 5000/8000 | Cost: 0.2975
Epoch: 003/010 | Batch 6000/8000 | Cost: 0.4486
Epoch: 003/010 | Batch 7000/8000 | Cost: 0.4053
Training Accuracy: 82.11% | Validation Accuracy: 81.09%
Time elapsed: 9.07 min
Epoch: 004/010 | Batch 000/8000 | Cost: 0.3876
Epoch: 004/010 | Batch 1000/8000 | Cost: 0.3984
Epoch: 004/010 | Batch 2000/8000 | Cost: 0.3607
Epoch: 004/010 | Batch 3000/8000 | Cost: 0.3546
Epoch: 004/010 | Batch 4000/8000 | Cost: 0.3708
Epoch: 004/010 | Batch 5000/8000 | Cost: 0.3641
Epoch: 004/010 | Batch 6000/8000 | Cost: 0.3905
Epoch: 004/010 | Batch 7000/8000 | Cost: 0.4271
Training Accuracy: 82.73% | Validation Accuracy: 81.45%
Time elapsed: 12.07 min
Epoch: 005/010 | Batch 000/8000 | Cost: 0.4300
Epoch: 005/010 | Batch 1000/8000 | Cost: 0.3768
Epoch: 005/010 | Batch 2000/8000 | Cost: 0.4356
Epoch: 005/010 | Batch 3000/8000 | Cost: 0.4657
Epoch: 005/010 | Batch 4000/8000 | Cost: 0.4408
Epoch: 005/010 | Batch 5000/8000 | Cost: 0.4006
Epoch: 005/010 | Batch 6000/8000 | Cost: 0.3762
Epoch: 005/010 | Batch 7000/8000 | Cost: 0.3210
Training Accuracy: 83.39% | Validation Accuracy: 81.71%
Time elapsed: 15.06 min
Epoch: 006/010 | Batch 000/8000 | Cost: 0.3226
Epoch: 006/010 | Batch 1000/8000 | Cost: 0.3792
Epoch: 006/010 | Batch 2000/8000 | Cost: 0.3928
Epoch: 006/010 | Batch 3000/8000 | Cost: 0.4206
Epoch: 006/010 | Batch 4000/8000 | Cost: 0.3445
Epoch: 006/010 | Batch 5000/8000 | Cost: 0.4131
Epoch: 006/010 | Batch 6000/8000 | Cost: 0.4445
Epoch: 006/010 | Batch 7000/8000 | Cost: 0.3747
```