Homework Assignment - 4

Name: Sagar Patel

NETID: sp5894

Question 1

RNNs for images. In this exercise, we will develop an RNN-type architecture for processing multi-dimensional data (such as RGB images). Here, hidden units are themselves arranged in the form of a grid (as opposed to a sequence). Each hidden unit is connected from the corresponding node from the input layer, as well as recurrently connected to its "top" and "left" neighbors. Here is a picture:

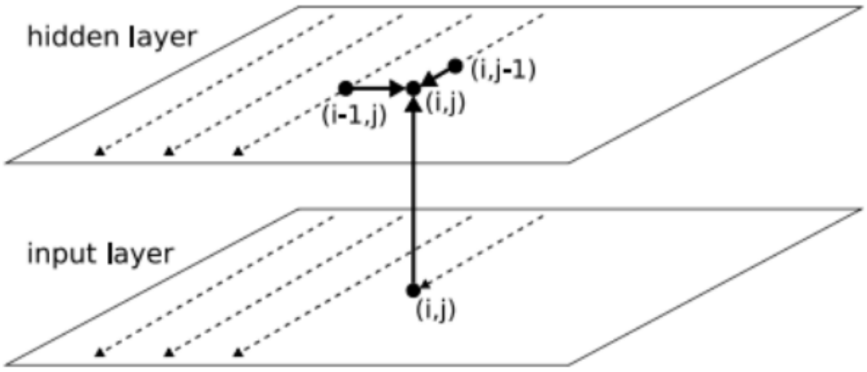


Figure 1: 2D RNNs

The equation for the hidden unit is given as follows (assume no bias parameters for simplicity):

$h^{i,j} = \sigma(W_{in}x^{i,j} + W_{left}h^{i-1,j} + W_{top}h^{i,j-1})$

1.a

Assume that the input grid is $n \times n$, each input $x_{i,j}$ is a c-channel vector, and the hidden state $h_{i,j}$ has dimension h. How many trainable parameters does this architecture have? You can assume zero padding as needed to avoid boundary effects.

Solution:

Let us consider that the size of the input image is = Input Size = $c \times n \times n$

The number of parameters would then we equal to $\Rightarrow \dim(W_{in}) + \dim(W_{left}) + \dim(W_{top})$

$\Rightarrow$ (Input Size $\times\ h$) + $(h \times h) + (h \times h)$

$\Rightarrow (c \times n \times n) + (h \times h) + (h \times h)$

$\Rightarrow cn^2h + h^2 + h^2$

1.b

How many arithmetic operations (scalar adds and multiplies) are required to compute all the hidden activations? You can write your answer in big-Oh notation.

Solution:

$O(cn^2h + h^2 + h^2)$ or $O(cn^2h + 2h^2)$

1.c

Compare the above architecture with a regular convnet, and explain advantages/disadvantages.

Solution: The main difference between CNN and RNN is the ability to process temporal information or data that comes in sequences, such as a sentence for example. Moreover, convolutional neural networks and recurrent neural networks are used for completely different purposes, and there are differences in the structures of the neural networks themselves to fit those different use cases.

CNNs employ filters within convolutional layers to transform data. Whereas, RNNs reuse activation functions from other data points in the sequence to generate the next output in a series.

CNN is a type of feed-forward artificial neural network with variations of multilayer perceptrons designed to use minimal amounts of preprocessing. However, RNN unlike feed forward neural networks - can use their internal memory to process arbitrary sequences of inputs.

RNNs are ideal for text and speech analysis and CNNs are ideal fro images and video processing.

Question 2

Attention! My code takes too long. In class, we showed that a regular self-attention layer takes $O(T^2)$ time to evaluate for an input with T tokens. Propose a way to reduce this running time to, say, $O(T)$, and comment on it's possible pros vs cons.

Solution:

In order to reduce the runtime, we can use this 2-step procedure:

Step 1:

Consider "autoregressive self attestation" where each of the given token attending only to it's current position and it's previous positions.

Step 2:

The alternative method is by using a windowed self attestation where all the 'n' tokens are partitioned into "windows". Here, every token attends to all the positions within it's designated window and prior to it.

Question 3

Making skip-gram training practical. Suppose we have a dictionary containing d words, and we are trying to learn skip-gram embeddings of each word using a very large training corpus. Suppose that our embedding dimension is chosen to be h.

3.a

Calculate the gradient of the cross-entropy loss for a single target-context training pair of words. What is the running time of calculating this gradient in terms of d and h?

Solution:

Let d be the number of words in the dictionary and h be the embedding dimension.

For our reference, we assume that $d >> h$

We can start by computing the gradient of the logarithmic conditional probability for the central word vector v_c and the conteext word vector u_o.

$log(P(w_o|w_c)) = u_o^T v_c - log(\sum_{i \in d} e^{u_i^T v_c})$

Calculating the gradient for the aforementioned formula (differentiating with respect to v_c), we get -

$$\frac{\partial log(P(w_o|w_c))}{\partial v_c} = u_o - \frac{\sum_{j \in d} e^{u_j^T v_c} u_j}{\sum_{i \in d} e^{u_i^T v_c}}$$

$$\Rightarrow u_o - \sum_{j \in d} (\frac{e^{u_j^T v_c}}{\sum_{i \in d} e^{u_i^T v_c}}) u_j$$

Since we are only interested in computing the loss for a single (context word and central target word) pair, we do not need to iterate index j through the d words in the dictionary. Therefore, the time complexity is bounded and dominated by the "softmax term". Thus, the time complexity is given by $O(h \times d)$. Since we assumed that $d >> h$, we can approximate the time complexity to be $O(d)$.

3.b

Here is a second approach. Argue that the gradient from part a can be written as a weighted average of some number of terms, and intuitively describe a sampling-based method to improve the running time of the gradient calculation.

Solution:

Question 4

Open the (incomplete) Jupyter notebook provided as an attachment to this homework in Google Colab (or other cloud service of your choice) and complete the missing items. Save your finished notebook in PDF format and upload along with your answers to the above theory questions in a single PDF.

Solution:

Link to the Notebook

Homework Assignment - 4

Question 4

In this problem we will use the BERT model for sentiment analysis. We will start with a pre-trained BERT model and fine-tune it on a dataset of Google Play store reviews.

Setup

Install the Transformers library by Hugging Face:

In [1]:
```
!pip install -q -U watermark
```

In [2]:
```
!pip install -qq transformers
```

```
     |████████████████████████████████| 2.0MB 8.6MB/s
     |████████████████████████████████| 890kB 49.5MB/s
     |████████████████████████████████| 3.2MB 51.2MB/s
  Building wheel for sacremoses (setup.py) ... done
```

In [44]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from transformers import BertTokenizer, BertForPreTraining
from torch.utils.data import Dataset
from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader
```

In [ ]:
```
import torch
import torchvision
from transformers import BertModel
import torch.nn.functional as F
from torch import nn
from transformers import AdamW,get_linear_schedule_with_warmup
```

In [81]:
```
from collections import defaultdict
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

In [4]:
```
%reload_ext watermark
%watermark -v -p numpy,pandas,torch,transformers
```

```
Python implementation: CPython
Python version       : 3.7.10
IPython version      : 5.5.0

numpy       : 1.19.5
pandas      : 1.1.5
```

```
torch       : 1.8.1+cu101
transformers: 4.4.2
```

Data Exploration

Download the Google Play app reviews dataset using the following commands:

In [5]:
```
!gdown --id 1S6qMioqPJjyBLpLVz4gmRTnJHnjitnuV
!gdown --id 1zdmewp7ayS4js4VtrJEHzAheSW-5NBZv
```

```
Downloading...
From: https://drive.google.com/uc?id=1S6qMioqPJjyBLpLVz4gmRTnJHnjitnuV
To: /content/apps.csv
100% 134k/134k [00:00<00:00, 48.4MB/s]
Downloading...
From: https://drive.google.com/uc?id=1zdmewp7ayS4js4VtrJEHzAheSW-5NBZv
To: /content/reviews.csv
7.17MB [00:00, 43.7MB/s]
```

Here is how it looks like:

In [6]:
```
df = pd.read_csv("reviews.csv")
df.head()
```

Out[6]:

	userName	userImage	content	score	thumbsUpCount
0	Andrew Thomas	https://lh3.googleusercontent.com/a-/AOh14GiHd...	Update: After getting a response from the deve...	1	21
1	Craig Haines	https://lh3.googleusercontent.com/-hoe0kwSJgPQ...	Used it for a fair amount of time without any ...	1	11
2	steven adkins	https://lh3.googleusercontent.com/a-/AOh14GiXw...	Your app sucks now!!!!! Used to be good but no...	1	17
3	Lars Panzerbjørn	https://lh3.googleusercontent.com/a-/AOh14Gg-h...	It seems OK, but very basic. Recurring tasks n...	1	192
4	Scott Prewitt	https://lh3.googleusercontent.com/-K-X1-YsVd6U...	Absolutely worthless. This app runs a prohibit...	1	42

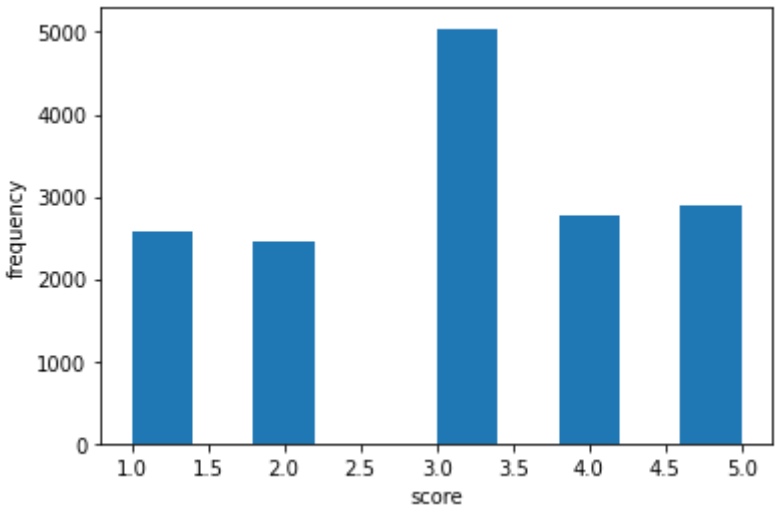
Let's first check the size of the dataset.

In [7]:
```
# TODO: Q1. How many samples are there in this dataset?

print(len(df.index))
```

```
15746
```

In [11]:
```
# TODO: Q2. Plot a histogram of review scores. These can be accessed in the df.s

plt.hist(df.score)
plt.xlabel('score')
plt.ylabel('frequency')
print("Most common score is", 3)
```

```
Most common score is 3
```



If correctly plotted, you should be able to see that this is a somewhat imbalanced dataset. Let's first convert the dataset into three classes: negative, neutral, and positive sentiment:

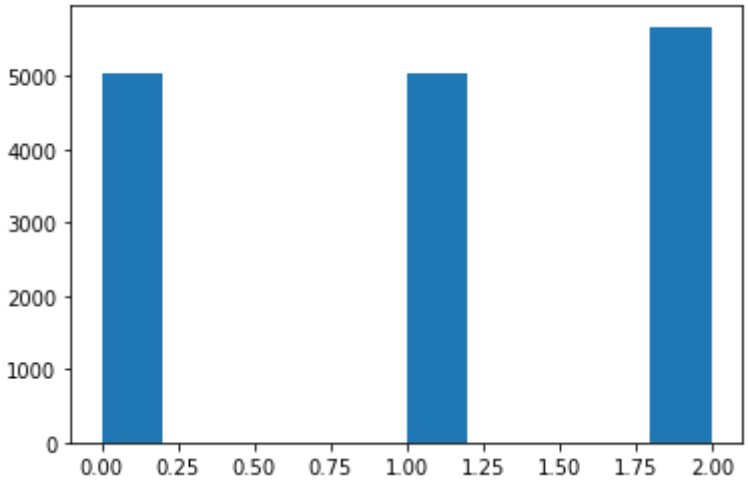
In [12]:
```
def to_sentiment(rating):
  rating = int(rating)
  if rating <= 2:
    return 0
  elif rating == 3:
    return 1
  else:
    return 2

df['sentiment'] = df.score.apply(to_sentiment)
```

In [13]:
```
class_names = ['negative', 'neutral', 'positive']
```

In [14]:
```
# TODO: Q3. Plot the histogram of review sentiments, and show that it is now app

plt.hist(df.sentiment, label= class_names)
```

Out[14]:
```
(array([5028.,    0.,    0.,    0.,    0., 5042.,    0.,    0.,    0.,
```

```
       5676.]),
array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
<a list of 10 Patch objects>)
```



Data Preprocessing

Let's now load a pre-trained BERT model and the corresponding tokenizer, which converts text data into tokens.

In [15]:
```
PRE_TRAINED_MODEL_NAME = 'bert-base-cased'
```

In [16]:
```
tokenizer = BertTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME)
```

Let's see how tokenization works. Here is the test sentence. Convert into tokens using the `tokenizer.tokenize` and `tokenizer.convert_tokens_to_ids` methods.

In [17]:
```
sample_txt = 'Every day feels like the same during the lock down.'
```

In [18]:
```
# TODO: Q4. Print the tokens and token ids of the sample text above.

tokens= tokenizer.tokenize(sample_txt)
print(tokens)
token_ids= tokenizer.convert_tokens_to_ids(tokens)
print(token_ids)
```

```
['Every', 'day', 'feels', 'like', 'the', 'same', 'during', 'the', 'lock', 'dow
n', '.']
[4081, 1285, 5115, 1176, 1103, 1269, 1219, 1103, 5842, 1205, 119]
```

BERT has special tokens for sentence separators [SEP] and unknown words [UNK]. This can be done using the `encode_plus()` method, which takes the test sentence and encodes it into `input_ids`.

In [19]:
```
encoding = tokenizer.encode_plus(
  sample_txt,
  max_length=32,
  add_special_tokens=True, # Add '[CLS]' and '[SEP]'
  return_token_type_ids=False,
  pad_to_max_length=True,
  return_attention_mask=True,
  return_tensors='pt',  # Return PyTorch tensors
)

encoding.keys()
```

```
Truncation was not explicitly activated but `max_length` is provided a specific
value, please use `truncation=True` to explicitly truncate examples to max lengt
h. Defaulting to 'longest_first' truncation strategy. If you encode pairs of seq
uences (GLUE-style) with the tokenizer you can select this strategy more precise
ly by providing a specific strategy to `truncation`.
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
```

Out[19]:
```
dict_keys(['input_ids', 'attention_mask'])
```

The token ids are now stored in a Tensor and padded to a length of 32:

In [20]:
```
print(len(encoding['input_ids'][0]))
encoding['input_ids'][0]
```

```
32
```

Out[20]:
```
tensor([ 101, 4081, 1285, 5115, 1176, 1103, 1269, 1219, 1103, 5842, 1205,  119,
         102,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0])
```

The attention mask has the same length:

In [21]:
```
print(len(encoding['attention_mask'][0]))
encoding['attention_mask']
```

```
32
```

Out[21]:
```
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Use the `tokenizer.convert_ids_to_tokens` method to invert the encoded token ids (the above tensor of length 32) and visualize the sentence.

In [22]:
```
# TODO: Q5. Invert the encoded token ids.
invert_tokens =  tokenizer.convert_ids_to_tokens(encoding['input_ids'][0])
print(invert_tokens)
```

```
['[CLS]', 'Every', 'day', 'feels', 'like', 'the', 'same', 'during', 'the', 'loc
k', 'down', '.', '[SEP]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]',
'[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]',
'[PAD]', '[PAD]', '[PAD]', '[PAD]']
```

Most reviews in the dataset contain less than around 120 tokens, but let us choose a maximum

length of 160.

In [23]:
```
MAX_LEN = 160
```

Building the dataset

Let's now create a dataset using the tokenizer. Here is some code that does this:

In [24]:
```
class GPReviewDataset(Dataset):

  def __init__(self, reviews, targets, tokenizer, max_len):
    self.reviews = reviews
    self.targets = targets
    self.tokenizer = tokenizer
    self.max_len = max_len

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

  def __getitem__(self, item):
    review = str(self.reviews[item])
    target = self.targets[item]

    encoding = self.tokenizer.encode_plus(
      review,
      add_special_tokens=True,
      max_length=self.max_len,
      return_token_type_ids=False,
      pad_to_max_length=True,
      return_attention_mask=True,
      return_tensors='pt',
    )

    return {
      'review_text': review,
      'input_ids': encoding['input_ids'].flatten(),
      'attention_mask': encoding['attention_mask'].flatten(),
      'targets': torch.tensor(target, dtype=torch.long)
    }
```

The tokenizer is doing most of the heavy lifting for us. We also return the review texts, so it'll be easier to evaluate the predictions from our model. Let's split the data into 90-5-5 train-validation-test.

In [103…
```
# TODO: Q6. Create three data frames: df_train, df_val, df_train as above and pr

df_train1, df_test= train_test_split(df, test_size=0.05)
df_train, df_val= train_test_split(df_train1, test_size=0.05263)

print(df_train.shape, df_val.shape, df_test.shape)
```

```
(14170, 12) (788, 12) (788, 12)
```

We also need to create a couple of data loaders:

In [52]:
```
def create_data_loader(df, tokenizer, max_len, batch_size):
  ds = GPReviewDataset(
    reviews=df.content.to_numpy(),
    targets=df.sentiment.to_numpy(),
    tokenizer=tokenizer,
    max_len=max_len
  )

  return DataLoader(
    ds,
    batch_size=batch_size,
    num_workers=4
  )
```

In [53]:
```
BATCH_SIZE = 16

train_data_loader = create_data_loader(df_train, tokenizer, MAX_LEN, BATCH_SIZE)
val_data_loader = create_data_loader(df_val, tokenizer, MAX_LEN, BATCH_SIZE)
test_data_loader = create_data_loader(df_test, tokenizer, MAX_LEN, BATCH_SIZE)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477: UserW
arning: This DataLoader will create 4 worker processes in total. Our suggested m
ax number of worker in current system is 2, which is smaller than what this Data
Loader is going to create. Please be aware that excessive worker creation might
get DataLoader running slow or even freeze, lower the worker number to avoid pot
ential slowness/freeze if necessary.
  cpuset_checked))
```

Let's have a look at an example batch from our training data loader:

In [54]:
```
data = next(iter(train_data_loader))
data.keys()
print(data['input_ids'].shape)
print(data['attention_mask'].shape)
print(data['targets'].shape)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477: UserW
arning: This DataLoader will create 4 worker processes in total. Our suggested m
ax number of worker in current system is 2, which is smaller than what this Data
Loader is going to create. Please be aware that excessive worker creation might
get DataLoader running slow or even freeze, lower the worker number to avoid pot
ential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
```

```
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
torch.Size([16, 160])
torch.Size([16, 160])
torch.Size([16])
```

Let's now load the basic BertModel and build our sentiment classifier on top of it. Load the model using:

In [55]:
```
bert_model = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME)
```

And encode our sample text:

In [56]:
```
last_hidden_state, pooled_output = bert_model(
  input_ids=encoding['input_ids'],
  attention_mask=encoding['attention_mask']
)
```

The `last_hidden_state` is the sequence of hidden states of the last layer of the model. The `pooled_output` can be thought of as a summary of the content in the test sentence. Try printing out the sizes of `last_hidden_state` and `pooled_output`:

In [57]:
```
# TODO: Q7. Print the sizes of the hidden states and the pooled output.

print(len(last_hidden_state))
print(len(pooled_output))
```

```
17
13
```

We can use all of this knowledge to create a classifier that uses the BERT model:

In [58]:
```
class SentimentClassifier(nn.Module):

  def __init__(self, n_classes):
    super(SentimentClassifier, self).__init__()
    self.bert = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME,return_dict = F
    self.drop = nn.Dropout(p=0.3)
    self.out = nn.Linear(self.bert.config.hidden_size, n_classes)

  def forward(self, input_ids, attention_mask):
    _, pooled_output = self.bert(
      input_ids=input_ids,
      attention_mask=attention_mask
    )
```

```
    output = self.drop(pooled_output)
    return self.out(output)
```

Note that our sentiment classifier takes the BERT backbone and adds a dropout layer (for regularization) and a linear dense layer, which we train using cross-entropy. Let's create an instance and move it to the GPU:

In [60]:
```
device = torch.device('cuda:0')
model = SentimentClassifier(len(class_names))
model = model.to(device)
```

We'll move the example batch of our training data to the GPU:

In [61]:
```
input_ids = data['input_ids'].to(device)
attention_mask = data['attention_mask'].to(device)
```

To get the predicted probabilities from our trained model, we'll apply the softmax function to the outputs:

In [62]:
```
F.softmax(model(input_ids, attention_mask), dim=1)
```

Out[62]:
```
tensor([[0.3667, 0.3875, 0.2458],
        [0.4513, 0.2976, 0.2511],
        [0.4222, 0.2626, 0.3152],
        [0.3998, 0.2818, 0.3183],
        [0.2722, 0.4380, 0.2898],
        [0.5963, 0.1957, 0.2079],
        [0.4070, 0.2777, 0.3153],
        [0.4183, 0.3288, 0.2529],
        [0.2683, 0.3602, 0.3715],
        [0.4586, 0.2818, 0.2596],
        [0.4676, 0.2418, 0.2907],
        [0.4276, 0.3481, 0.2243],
        [0.3975, 0.2828, 0.3197],
        [0.4794, 0.2431, 0.2775],
        [0.3272, 0.3103, 0.3624],
        [0.3041, 0.2749, 0.4209]], device='cuda:0', grad_fn=<SoftmaxBackward>)
```

Training

To train the model, we will use the AdamW optimizer and a linear learning-rate scheduler with no warmup steps, along with the cross-entropy loss. Five epochs (full passes through the training data should be enough) should be enough, but you can experiment with more epochs.

In [63]:
```
EPOCHS = 5

optimizer = AdamW(model.parameters(), lr=2e-5, correct_bias=False)
total_steps = len(train_data_loader) * EPOCHS

scheduler = get_linear_schedule_with_warmup(
  optimizer,
  num_warmup_steps=0,
  num_training_steps=total_steps
)
```

```
loss_fn = nn.CrossEntropyLoss().to(device)
```

Let's continue with writing a helper function for training our model for one epoch:

In [65]:

```
def train_epoch(
  model,
  data_loader,
  loss_fn,
  optimizer,
  device,
  scheduler,
  n_examples
):
  model = model.train()

  losses = []
  correct_predictions = 0

  for d in data_loader:
    # TODO Q8. Complete the incomplete code snippets below to finish training.

    input_ids = d["input_ids"].to(device)
    attention_mask = d["attention_mask"].to(device)
    targets = d["targets"].to(device)

    outputs = model(
      input_ids=input_ids,
      attention_mask=attention_mask
    )

    _, preds = torch.max(outputs, dim=1)
    loss = loss_fn(outputs, targets)

    correct_predictions += torch.sum(preds == targets)
    losses.append(loss.item())

    loss.backward()
    nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
    optimizer.step()
    scheduler.step()
    optimizer.zero_grad()

  return correct_predictions.double() / n_examples, np.mean(losses)
```

Let's write another function that helps us evaluate the model on a given data loader.

In [66]:

```
def eval_model(model, data_loader, loss_fn, device, n_examples):
  model = model.eval()

  # TODO: Q9. Reproduce the above code but only evaluate the model (without any

  losses = []
  correct_predictions = 0

  for d in data_loader:
    input_ids = d["input_ids"].to(device)
    attention_mask = d["attention_mask"].to(device)
```

```
    targets = d["targets"].to(device)

    outputs = model(
      input_ids=input_ids,
      attention_mask=attention_mask
    )

    _, preds = torch.max(outputs, dim=1)
    loss = loss_fn(outputs, targets)

    correct_predictions += torch.sum(preds == targets)
    losses.append(loss.item())

  return correct_predictions.double() / n_examples, np.mean(losses)
```

Using those two, we can write our training loop.

In [70]:

```
%%time

history = defaultdict(list)
best_accuracy = 0

for epoch in range(EPOCHS):

  print(f'Epoch {epoch + 1}/{EPOCHS}')
  print('-' * 10)

  # TODO: Q10. Complete the code below to track train and test accuracy.losses

  train_acc, train_loss = train_epoch(model,train_data_loader,loss_fn,optimizer,

  print(f'Train loss {train_loss} accuracy {train_acc}')

  val_acc, val_loss = eval_model(model,val_data_loader,loss_fn,device,len(df_val

  print(f'Val loss {val_loss} accuracy {val_acc}')
  print()

  history['train_acc'].append(train_acc)
  history['train_loss'].append(train_loss)
  history['val_acc'].append(val_acc)
  history['val_loss'].append(val_loss)

  if val_acc > best_accuracy:
    torch.save(model.state_dict(), 'best_model_state.bin')
    best_accuracy = val_acc
```

```
Epoch 1/5
----------
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477: UserW
arning: This DataLoader will create 4 worker processes in total. Our suggested m
ax number of worker in current system is 2, which is smaller than what this Data
Loader is going to create. Please be aware that excessive worker creation might
get DataLoader running slow or even freeze, lower the worker number to avoid pot
ential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
```

```
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Train loss 0.7356017599424713 accuracy 0.6656316160903317
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Val   loss 0.5804714962840081 accuracy 0.7538071065989848

Epoch 2/5
----------
```

```
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Train loss 0.42440733684407667 accuracy 0.8400846859562456
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
```

```
e model (e.g. 512 for Bert).
  FutureWarning,
Val   loss 0.4956962388381362 accuracy 0.8185279187817258

Epoch 3/5
----------
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Train loss 0.2307982091049813 accuracy 0.9270995059985886
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
```

```
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Val   loss 0.4909455148782581 accuracy 0.8553299492385786

Epoch 4/5
----------
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Train loss 0.14757165490582894 accuracy 0.9568807339449542
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
```

```
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Val   loss 0.581008961526677 accuracy 0.8591370558375634

Epoch 5/5
----------
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Train loss 0.11645558438008666 accuracy 0.9669724770642203
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
```

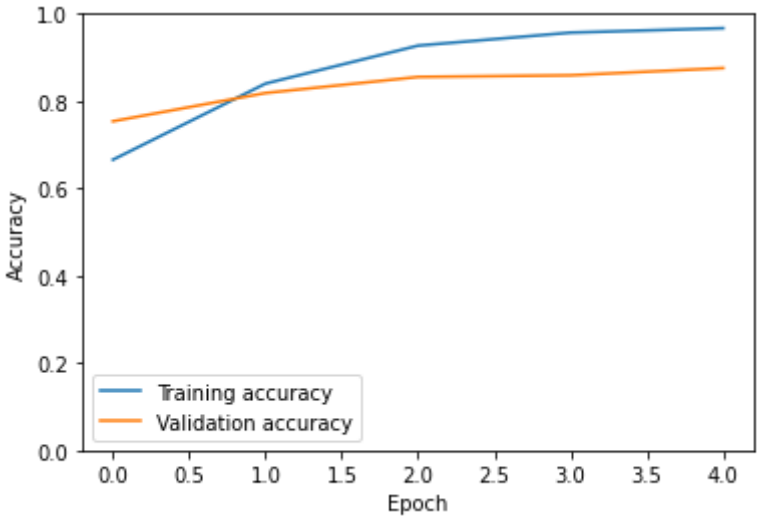
```
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
Val   loss 0.5889394129044376 accuracy 0.8756345177664974

CPU times: user 18min 28s, sys: 15min 46s, total: 34min 15s
Wall time: 34min 27s
```

Note that we're storing the best model, indicated by the highest validation accuracy.

Plot train and validation accuracy as a function of epoch count.

In [72]:

```
# TODO: Q11. Plot train/validation accuracies.

plt.plot(history['train_acc'], label='Training accuracy')
plt.plot(history['val_acc'], label='Validation accuracy')

plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.ylim([0, 1]);
```



You might try to fine-tune the parameters (learning rate, batch size) a bit more if accuracy is not good enough.

Evaluation

So how good is our model on predicting sentiment?

We'll define a helper function to get the predictions from our model:

In [73]:
```
def get_predictions(model, data_loader):
  model = model.eval()

  review_texts = []
  predictions = []
  prediction_probs = []
  real_values = []

  with torch.no_grad():
    for d in data_loader:

      texts = d["review_text"]
      input_ids = d["input_ids"].to(device)
      attention_mask = d["attention_mask"].to(device)
      targets = d["targets"].to(device)

      outputs = model(
        input_ids=input_ids,
        attention_mask=attention_mask
      )
      _, preds = torch.max(outputs, dim=1)

      probs = F.softmax(outputs, dim=1)

      review_texts.extend(texts)
      predictions.extend(preds)
      prediction_probs.extend(probs)
      real_values.extend(targets)

  predictions = torch.stack(predictions).cpu()
  prediction_probs = torch.stack(prediction_probs).cpu()
  real_values = torch.stack(real_values).cpu()
  return review_texts, predictions, prediction_probs, real_values
```

This is similar to the evaluation function, except that we're storing the text of the reviews and the predicted probabilities (by applying the softmax on the model outputs):

In [74]:
```
y_review_texts, y_pred, y_pred_probs, y_test = get_predictions(
  model,
  test_data_loader
)
```

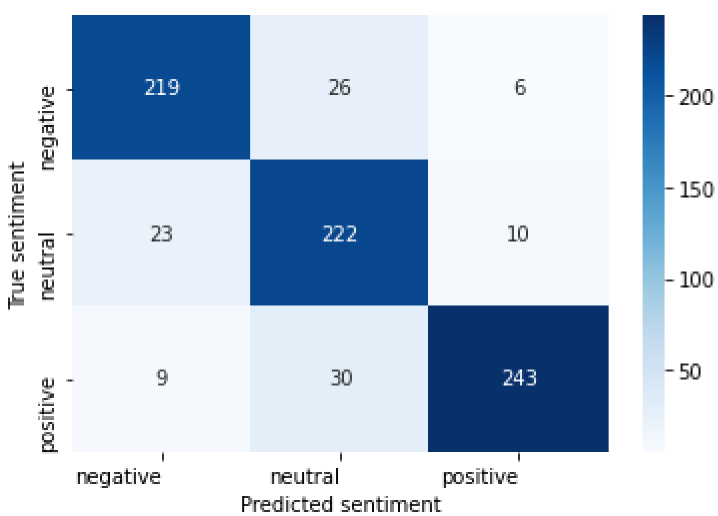
```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477: UserW
arning: This DataLoader will create 4 worker processes in total. Our suggested m
ax number of worker in current system is 2, which is smaller than what this Data
Loader is going to create. Please be aware that excessive worker creation might
get DataLoader running slow or even freeze, lower the worker number to avoid pot
ential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
```

```
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
```

Let us compare true sentiment vs predicted sentiment by plotting a confusion matrix of `y_test` vs `y_pred`.

In [85]:

```
# TODO. Q12. Plot the 3x3 confusion matrix and show that the model finds it a bi

cmat = confusion_matrix(y_test, y_pred)
df_cmat = pd.DataFrame(cmat, index=class_names, columns=class_names)
hmap = sns.heatmap(df_cmat, annot=True, fmt="d", cmap="Blues")

hmap.xaxis.set_ticklabels(hmap.xaxis.get_ticklabels(), ha='right')
hmap.yaxis.set_ticklabels(hmap.yaxis.get_ticklabels(), ha='right')

plt.xlabel('Predicted sentiment');
plt.ylabel('True sentiment')
```



Predicting on Raw Text

Let's use our model to predict the sentiment of some raw text:

In [86]:
```
review_text = "I love Deep Learning! Best course evah!!!1!!"
```

Use your trained model to predict the sentiment expressed in `review_text`.

In [87]:
```
# TODO: Q13. Print the predicted sentiment in `review_text`.

encoding = tokenizer.encode_plus(
  review_text,
  max_length=32,
  add_special_tokens=True, # Add '[CLS]' and '[SEP]'
  return_token_type_ids=False,
  pad_to_max_length=True,
  return_attention_mask=True,
  return_tensors='pt',  # Return PyTorch tensors
)

encoding.keys()
```

```
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2
074: FutureWarning: The `pad_to_max_length` argument is deprecated and will be r
emoved in a future version, use `padding=True` or `padding='longest'` to pad to
the longest sequence in the batch, or use `padding='max_length'` to pad to a max
length. In this case, you can give a specific length with `max_length` (e.g. `ma
x_length=45`) or leave max_length to None to pad to the maximal input size of th
e model (e.g. 512 for Bert).
  FutureWarning,
```

Out[87]:
```
dict_keys(['input_ids', 'attention_mask'])
```

In [94]:
```
input_ids = encoding['input_ids'].to(device)
attention_mask = encoding['attention_mask'].to(device)
outputs = model(input_ids, attention_mask)
_, preds = torch.max(outputs, dim=1)

print(f'Review text: {review_text}')
print(f'Sentiment  : {class_names[prediction]}')
```

```
Review text: I love Deep Learning! Best course evah!!!1!!
Sentiment  : positive
```

References

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- L11 Language Models - Alec Radford (OpenAI)
- The Illustrated BERT, ELMo, and co.
- BERT Fine-Tuning Tutorial with PyTorch
- How to Fine-Tune BERT for Text Classification?
- Huggingface Transformers
- BERT Explained: State of the art language model for NLP