Lecture_16_ML_ecosystem_code

October 7, 2024

ML Ecosystem

In this notebook, we will use the example of HuggingFace to demonstrate how to easily load a pretrained model, load the dataset, and finetune the model for a downstream task.

To prepare, make sure PyTorch is installed in your environment. Also install the following packages

`pip install transformers datasets accelerate peft`

0.0.1 Loading the model

Our goal is to finetune a model for sentiment classification, that is given a text, classify whether it has a positive or negative sentiment.

To start, let's load a pretrained model from HuggingFace model hub. The name of the model we are loading is "bert-base-uncased" and its webpage can be found at https://huggingface.co/google-bert/bert-base-uncased. Here, 'bert' is a language model Google developed in 2018, 'base' means it's the base version of the model (that has a relatively small size), and 'uncased' means it does not distinguash upper and lower cases.

To load the model, we use `transformers.AutoModelForSequenceClassification.from_pretrained` and provide the model name - it will automatically load the model from HuggingFace. Note that the 'ForSequenceClassification' indicates we will load this model for sequence classification purposes, which will add an additional linear layer to the bert model such that the output size equals the number of classes in our problem.

We also need to use a tokenizer, which maps words in string format into feature vectors. We again will load this from pretrained.

[14]:
```
from transformers import AutoModelForSequenceClassification,␣
↪AutoTokenizer,pipeline
import torch

model_name = "bert-base-uncased"

id2label = {0: "Negative", 1: "Positive"}
label2id = {"Negative":0, "Positive":1}

# loading the model from pretrained, setting number of classes to be 2, and␣
↪setting the class label names.
```

```
model = AutoModelForSequenceClassification.from_pretrained(model_name,␣
  ↪num_labels=2,id2label=id2label, label2id = label2id)

# loading the tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-base-uncased and are newly initialized:
['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
/Users/coolq/mambaforge/envs/pytorch_ecosystem/lib/python3.12/site-
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be depracted in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

[15]:
```
print("isinstance(model,torch.nn.Module) = ",isinstance(model,torch.nn.Module))␣
  ↪# the loaded model is a torch.nn.Module

# let's print out the model structure
model
```

```
isinstance(model,torch.nn.Module) =  True
```

[15]:
```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
```

```
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (activation): Tanh()
  )
 )
 (dropout): Dropout(p=0.1, inplace=False)
 (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

You will see the model has many layers, in particular the 12 transformer blocks (the BertLayer). The original bert-base-uncased model ends at the "BertPooler" layer, which outputs a 768 dimensional feature vector. The final Linear layer is added when we load the model to convert the output dimension to 2.

Let's see how we can use the model to make prediction.

[18]:
```
from transformers import pipeline

# pipeline connects a tokenizer and a model. It is designed such that one can␣
 ↪make inference easily.
classifier = pipeline("sentiment-analysis",model=model,tokenizer=tokenizer)

# to classify a sentiment of a sentence
classifier("I feel very sad today. ")
```

```
Hardware accelerator e.g. GPU is available in the environment, but no `device`
argument is passed to the `Pipeline` object. Model will be on CPU.
```

[18]:
```
[{'label': 'Positive', 'score': 0.7406206727027893}]
```

The model is pretrained to "understand" texts, and the 768 dimensional output of BERT contains

useful features of the texts. However, the pretrained model is not specifically designed to predict sentiment. Further, the final linear layer is added when we load the model and it is NOT trained yet. So we wouldn't expect the model to work well now. To make it work well, we need to fine-tune it on some dataset.

0.0.2 Loading the dataset

We will use the `datasets` package to load datasets from HuggingFace. Today we will load the IMDB dataset consisting of movie reviews.

[19]:
```
from datasets import load_dataset

# let's load the dataset from HuggingFace
dataset = load_dataset("imdb")
train_dataset = dataset["train"]
test_dataset = dataset["test"]

# let's see how the dataset looks like
for i in range(10):
    print(train_dataset[i])
```

```
{'text': 'I rented I AM CURIOUS-YELLOW from my video store because of all the
controversy that surrounded it when it was first released in 1967. I also heard
that at first it was seized by U.S. customs if it ever tried to enter this
country, therefore being a fan of films considered "controversial" I really had
to see this for myself.<br /><br />The plot is centered around a young Swedish
drama student named Lena who wants to learn everything she can about life. In
particular she wants to focus her attentions to making some sort of documentary
on what the average Swede thought about certain political issues such as the
Vietnam War and race issues in the United States. In between asking politicians
and ordinary denizens of Stockholm about their opinions on politics, she has sex
with her drama teacher, classmates, and married men.<br /><br />What kills me
about I AM CURIOUS-YELLOW is that 40 years ago, this was considered
pornographic. Really, the sex and nudity scenes are few and far between, even
then it\'s not shot like some cheaply made porno. While my countrymen mind find
it shocking, in reality sex and nudity are a major staple in Swedish cinema.
Even Ingmar Bergman, arguably their answer to good old boy John Ford, had sex
scenes in his films.<br /><br />I do commend the filmmakers for the fact that
any sex shown in the film is shown for artistic purposes rather than just to
shock people and make money to be shown in pornographic theaters in America. I
AM CURIOUS-YELLOW is a good film for anyone wanting to study the meat and
potatoes (no pun intended) of Swedish cinema. But really, this film doesn\'t
have much of a plot.', 'label': 0}
{'text': '"I Am Curious: Yellow" is a risible and pretentious steaming pile. It
doesn\'t matter what one\'s political views are because this film can hardly be
taken seriously on any level. As for the claim that frontal male nudity is an
automatic NC-17, that isn\'t true. I\'ve seen R-rated films with male nudity.
Granted, they only offer some fleeting views, but where are the R-rated films
```

with gaping vulvas and flapping labia? Nowhere, because they don\'t exist. The same goes for those crappy cable shows: schlongs swinging in the breeze but not a clitoris in sight. And those pretentious indie movies like The Brown Bunny, in which we\'re treated to the site of Vincent Gallo\'s throbbing johnson, but not a trace of pink visible on Chloe Sevigny. Before crying (or implying) "double-standard" in matters of nudity, the mentally obtuse should take into account one unavoidably obvious anatomical difference between men and women: there are no genitals on display when actresses appears nude, and the same cannot be said for a man. In fact, you generally won\'t see female genitals in an American film in anything short of porn or explicit erotica. This alleged double-standard is less a double standard than an admittedly depressing ability to come to terms culturally with the insides of women\'s bodies.', 'label': 0}
{'text': "If only to avoid making this type of film in the future. This film is interesting as an experiment but tells no cogent story.

One might feel virtuous for sitting thru it because it touches on so many IMPORTANT issues but it does so without any discernable motive. The viewer comes away with no new perspectives (unless one comes up with one while one's mind wanders, as it will invariably do during this pointless film).

One might better spend one's time staring out a window at a tree growing.

", 'label': 0}
{'text': "This film was probably inspired by Godard's Masculin, féminin and I urge you to see that film instead.

The film has two strong elements and those are, (1) the realistic acting (2) the impressive, undeservedly good, photo. Apart from that, what strikes me most is the endless stream of silliness. Lena Nyman has to be most annoying actress in the world. She acts so stupid and with all the nudity in this film,…it's unattractive. Comparing to Godard's film, intellectuality has been replaced with stupidity. Without going too far on this subject, I would say that follows from the difference in ideals between the French and the Swedish society.

A movie of its time, and place. 2/10.", 'label': 0}
{'text': 'Oh, brother…after hearing about this ridiculous film for umpteen years all I can think of is that old Peggy Lee song..

"Is that all there is??" …I was just an early teen when this smoked fish hit the U.S. I was too young to get in the theater (although I did manage to sneak into "Goodbye Columbus"). Then a screening at a local film museum beckoned - Finally I could see this film, except now I was as old as my parents were when they schlepped to see it!!

The ONLY reason this film was not condemned to the anonymous sands of time was because of the obscenity case sparked by its U.S. release. MILLIONS of people flocked to this stinker, thinking they were going to see a sex film…Instead, they got lots of closeups of gnarly, repulsive Swedes, on-street interviews in bland shopping malls, asinie political pretension…and feeble who-cares simulated sex scenes with saggy, pale actors.

Cultural icon, holy grail, historic artifact..whatever this thing was, shred it, burn it, then stuff the ashes in a lead box!

Elite esthetes still scrape to find value in its boring pseudo revolutionary political spewings..But if it weren\'t for the censorship scandal, it would have been ignored, then forgotten.

Instead, the "I Am Blank, Blank" rhythymed title was repeated endlessly for years as a titilation for porno films (I am Curious, Lavender - for gay films, I Am Curious, Black - for blaxploitation films, etc..)

and every ten years or so the thing rises from the dead, to be viewed by a new generation of suckers who want to see that "naughty sex film" that "revolutionized the film industry"…

Yeesh, avoid like the plague..Or if you MUST see it - rent the video and fast forward to the "dirty" parts, just to get it over with.

', 'label': 0}
{'text': "I would put this at the top of my list of films in the category of unwatchable trash! There are films that are bad, but the worst kind are the ones that are unwatchable but you are suppose to like them because they are supposed to be good for you! The sex sequences, so shocking in its day, couldn't even arouse a rabbit. The so called controversial politics is strictly high school sophomore amateur night Marxism. The film is self-consciously arty in the worst sense of the term. The photography is in a harsh grainy black and white. Some scenes are out of focus or taken from the wrong angle. Even the sound is bad! And some people call this art?

", 'label': 0}
{'text': "Whoever wrote the screenplay for this movie obviously never consulted any books about Lucille Ball, especially her autobiography. I've never seen so many mistakes in a biopic, ranging from her early years in Celoron and Jamestown to her later years with Desi. I could write a whole list of factual errors, but it would go on for pages. In all, I believe that Lucille Ball is one of those inimitable people who simply cannot be portrayed by anyone other than themselves. If I were Lucie Arnaz and Desi, Jr., I would be irate at how many mistakes were made in this film. The filmmakers tried hard, but the movie seems awfully sloppy to me.", 'label': 0}
{'text': 'When I first saw a glimpse of this movie, I quickly noticed the actress who was playing the role of Lucille Ball. Rachel York\'s portrayal of Lucy is absolutely awful. Lucille Ball was an astounding comedian with incredible talent. To think about a legend like Lucille Ball being portrayed the way she was in the movie is horrendous. I cannot believe out of all the actresses in the world who could play a much better Lucy, the producers decided to get Rachel York. She might be a good actress in other roles but to play the role of Lucille Ball is tough. It is pretty hard to find someone who could resemble Lucille Ball, but they could at least find someone a bit similar in looks and talent. If you noticed York\'s portrayal of Lucy in episodes of I Love Lucy like the chocolate factory or vitavetavegamin, nothing is similar in any way-her expression, voice, or movement.

To top it all off, Danny Pino playing Desi Arnaz is horrible. Pino does not qualify to play as Ricky. He\'s small and skinny, his accent is unreal, and once again, his acting is unbelievable. Although Fred and Ethel were not similar either, they were not as bad as the characters of Lucy and Ricky.

Overall, extremely horrible casting and the story is badly told. If people want to understand the real life situation of Lucille Ball, I suggest watching A&E Biography of Lucy and Desi, read the book from Lucille Ball herself, or PBS\' American Masters: Finding Lucy. If you want to see a docudrama, "Before the Laughter" would be a better choice. The casting of Lucille Ball and Desi Arnaz in "Before the Laughter" is much better compared to this. At least, a similar aspect is shown rather than nothing.', 'label': 0}
{'text': 'Who are these "They"- the actors? the filmmakers? Certainly couldn\'t be the audience- this is among the most air-puffed productions in existence.

It\'s the kind of movie that looks like it was a lot of fun to shoot\x97 TOO much fun, nobody is getting any actual work done, and that almost always makes for a movie that\'s no fun to watch.

Ritter dons glasses so as to hammer home his character\'s status as a sort of doppleganger of the bespectacled Bogdanovich; the scenes with the breezy Ms. Stratten are sweet, but have an embarrassing, look-guys-I\'m-dating-the-prom-queen feel to them. Ben Gazzara sports his usual cat\'s-got-canary grin in a futile attempt to elevate the meager plot, which requires him to pursue Audrey Hepburn with all the interest of a narcoleptic at an insomnia clinic. In the meantime, the budding couple\'s respective children (nepotism alert: Bogdanovich\'s daughters) spew cute and pick up some fairly disturbing pointers on \'love\' while observing their parents. (Ms. Hepburn, drawing on her dignity, manages to rise above the proceedings- but she has the monumental challenge of playing herself, ostensibly.) Everybody looks great, but so what? It\'s a movie and we can expect that much, if that\'s what you\'re looking for you\'d be better off picking up a copy of Vogue.

Oh- and it has to be mentioned that Colleen Camp thoroughly annoys, even apart from her singing, which, while competent, is wholly unconvincing… the country and western numbers are woefully mismatched with the standards on the soundtrack. Surely this is NOT what Gershwin (who wrote the song from which the movie\'s title is derived) had in mind; his stage musicals of the 20\'s may have been slight, but at least they were long on charm. "They All Laughed" tries to coast on its good intentions, but nobody- least of all Peter Bogdanovich - has the good sense to put on the brakes.

Due in no small part to the tragic death of Dorothy Stratten, this movie has a special place in the heart of Mr. Bogdanovich- he even bought it back from its producers, then distributed it on his own and went bankrupt when it didn\'t prove popular. His rise and fall is among the more sympathetic and tragic of Hollywood stories, so there\'s no joy in criticizing the film… there _is_ real emotional investment in Ms. Stratten\'s scenes. But "Laughed" is a faint echo of "The Last Picture Show", "Paper Moon" or "What\'s Up, Doc"- following "Daisy Miller" and "At Long Last Love", it was a thundering confirmation of the phase from which P.B. has never emerged.

All in all, though, the movie is harmless, only a waste of rental. I want to watch people having a good time, I\'ll go to the park on a sunny day. For filmic expressions of joy and love, I\'ll stick to Ernest Lubitsch and Jaques Demy…', 'label': 0}
{'text': "This is said to be a personal film for Peter Bogdonavitch. He based it on his life but changed things around to fit the characters, who are detectives. These detectives date beautiful models and have no problem getting them. Sounds more like a millionaire playboy filmmaker than a detective, doesn't it? This entire movie was written by Peter, and it shows how out of touch with real people he was. You're supposed to write what you know, and he did that, indeed. And leaves the audience bored and confused, and jealous, for that matter. This is a curio for people who want to see Dorothy Stratten, who was murdered right after filming. But Patti Hanson, who would, in real life, marry Keith Richards, was also a model, like Stratten, but is a lot better and has a more ample part. In fact, Stratten's part seemed forced; added. She doesn't have a lot to do with the story, which is pretty convoluted to begin with. All in all, every character in this film is somebody that very few people can relate with, unless you're

millionaire from Manhattan with beautiful supermodels at your beckon call. For the rest of us, it's an irritating snore fest. That's what happens when you're out of touch. You entertain your few friends with inside jokes, and bore all the rest.", 'label': 0}

[20]:
```
# Tokenize the dataset - convert the 'text' from string to a vector
def tokenize_function(examples):
    # the tokenizer will tranform the text into 'input_ids', which is a token␣
 ↪ID that represents the word; and "attention_masks"
    return tokenizer(examples["text"], padding="max_length", truncation=True)


train_dataset = train_dataset.map(tokenize_function, batched=True)
test_dataset = test_dataset.map(tokenize_function, batched=True)

train_dataset.set_format("torch", columns=["input_ids", "attention_mask",␣
 ↪"label"])
test_dataset.set_format("torch", columns=["input_ids", "attention_mask",␣
 ↪"label"])

print(train_dataset[0])
```

```
Map: 100%|      | 25000/25000 [00:08<00:00, 3117.37 examples/s]
Map: 100%|      | 25000/25000 [00:06<00:00, 3783.35 examples/s]
```

```
{'label': tensor(0), 'input_ids': tensor([  101,  1045, 12524,  1045,  2572,
8025,  1011,  3756,  2013,  2026,
         2678,  3573,  2138,  1997,  2035,  1996,  6704,  2008,  5129,  2009,
         2043,  2009,  2001,  2034,  2207,  1999,  3476,  1012,  1045,  2036,
         2657,  2008,  2012,  2034,  2009,  2001,  8243,  2011,  1057,  1012,
         1055,  1012,  8205,  2065,  2009,  2412,  2699,  2000,  4607,  2023,
         2406,  1010,  3568,  2108,  1037,  5470,  1997,  3152,  2641,  1000,
         6801,  1000,  1045,  2428,  2018,  2000,  2156,  2023,  2005,  2870,
         1012,  1026,  7987,  1013,  1028,  1026,  7987,  1013,  1028,  1996,
         5436,  2003,  8857,  2105,  1037,  2402,  4467,  3689,  3076,  2315,
        14229,  2040,  4122,  2000,  4553,  2673,  2016,  2064,  2055,  2166,
         1012,  1999,  3327,  2016,  4122,  2000,  3579,  2014,  3086,  2015,
         2000,  2437,  2070,  4066,  1997,  4516,  2006,  2054,  1996,  2779,
        25430, 14728,  2245,  2055,  3056,  2576,  3314,  2107,  2004,  1996,
         5148,  2162,  1998,  2679,  3314,  1999,  1996,  2142,  2163,  1012,
         1999,  2090,  4851,  8801,  1998,  6623,  7939,  4697,  3619,  1997,
         8947,  2055,  2037, 10740,  2006,  4331,  1010,  2016,  2038,  3348,
         2007,  2014,  3689,  3836,  1010, 19846,  1010,  1998,  2496,  2273,
         1012,  1026,  7987,  1013,  1028,  1026,  7987,  1013,  1028,  2054,
         8563,  2033,  2055,  1045,  2572,  8025,  1011,  3756,  2003,  2008,
         2871,  2086,  3283,  1010,  2023,  2001,  2641, 26932,  1012,  2428,
         1010,  1996,  3348,  1998, 16371, 25469,  5019,  2024,  2261,  1998,
         2521,  2090,  1010,  2130,  2059,  2009,  1005,  1055,  2025,  2915,
```

```
         2066,  2070, 10036,  2135,  2081, 22555,  2080,  1012,  2096,  2026,
         2406,  3549,  2568,  2424,  2009, 16880,  1010,  1999,  4507,  3348,
         1998, 16371, 25469,  2024,  1037,  2350, 18785,  1999,  4467,  5988,
         1012,  2130, 13749,  7849, 24544,  1010, 15835,  2037,  3437,  2000,
         2204,  2214,  2879,  2198,  4811,  1010,  2018,  3348,  5019,  1999,
         2010,  3152,  1012,  1026,  7987,  1013,  1028,  1026,  7987,  1013,
         1028,  1045,  2079,  4012,  3549,  2094,  1996, 16587,  2005,  1996,
         2755,  2008,  2151,  3348,  3491,  1999,  1996,  2143,  2003,  3491,
         2005,  6018,  5682,  2738,  2084,  2074,  2000,  5213,  2111,  1998,
         2191,  2769,  2000,  2022,  3491,  1999, 26932, 12370,  1999,  2637,
         1012,  1045,  2572,  8025,  1011,  3756,  2003,  1037,  2204,  2143,
         2005,  3087,  5782,  2000,  2817,  1996,  6240,  1998, 14629,  1006,
         2053, 26136,  3832,  1007,  1997,  4467,  5988,  1012,  2021,  2428,
         1010,  2023,  2143,  2987,  1005,  1056,  2031,  2172,  1997,  1037,
         5436,  1012,   102,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0]), 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0])}
```

0.0.3 FineTuning

Now let's conduct the actual fine-tuning. First, let's see the size of the model.

[21]:
```
print("number of total parameters in model = ", model.num_parameters(),"number␣
↪of TRAINABLE parameters in model = ", model.num_parameters(only_trainable =␣
↪True))
```

```
number of total parameters in model =  109483778 number of TRAINABLE parameters
in model =  109483778
```

To reduce the memory needed for training, a common method is to use peft to reduce the number of trainable parameters of the model. If we don't use peft, all the model parameters will be trained by default, which will take a lot of memory. peft, standing for "Parameter Efficient Fine Tuning", uses "LoRA" to significantly reduce the number of trainable parameters (but still keeping good performance).

[24]:
```
from transformers import Trainer, TrainingArguments
from peft import LoraConfig,get_peft_model
import numpy as np

# use peft to reduce the number of trainable parameters
peft_config = LoraConfig(
    r=8,
)
model = get_peft_model(model, peft_config)

print("number of total parameters with LoRA ", model.num_parameters(),"number␣
↪of TRAINABLE parameters with LoRA = ", model.num_parameters(only_trainable =␣
↪True))
```

```
number of total parameters with LoRA  109778690 number of TRAINABLE parameters
with LoRA =  294912
```

You can see the number of trainable parameters goes down from 109,483,778 to 294,912, which is 3 orders of magnitudes less!

Let's now conduct the finetuneing. We will use the Trainer class for this, which conducts all the training loops. As an alternative, you can also write your training loop as this is just a PyTorch module. Given the large memory need, you would need a GPU to run the below code.

[ ]:
```
training_args = TrainingArguments(
    output_dir='./results',
```

```
    evaluation_strategy="epoch",
    num_train_epochs=3,
    per_device_train_batch_size=64,
    per_device_eval_batch_size=64
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset
)

trainer.train()
```

0.0.4 Checking out the fine-tuned model.

[25]:
```
# replace the path to where you stored the fine tuned model
model_finetuned = AutoModelForSequenceClassification.from_pretrained('/Users/
 ↪coolq/Library/CloudStorage/Box-Box/Teaching/Tool Chain/Toolchain 2024 Fall/
 ↪notebooks/fine_tune_models/fine_tuned_model_fullfinetuning', num_labels = 2,␣
 ↪id2label=id2label, label2id=label2id)
```

[26]:
```
classifier_finetuned = pipeline("sentiment-analysis", model=model_finetuned,␣
 ↪tokenizer=tokenizer)
```

```
Hardware accelerator e.g. GPU is available in the environment, but no `device`
argument is passed to the `Pipeline` object. Model will be on CPU.
```

[27]:
```
classifier_finetuned("I am very happy today.")
```

[27]:
```
[{'label': 'Positive', 'score': 0.9922531247138977}]
```

[28]:
```
classifier_finetuned("This is my first semester at CMU. The courses are␣
 ↪challenging, but I have learned a lot. ")
```

[28]:
```
[{'label': 'Positive', 'score': 0.9922582507133484}]
```

[29]:
```
classifier_finetuned("A pleasant surprise, the cinematography is impeccable,␣
 ↪the characters quite well done, the plot looks like a link between the␣
 ↪stories of the First Age, the Silmarillion and the stories of the Lord of␣
 ↪the Rings of the Third Age, the rhythm of narration is pleasant albeit a bit␣
 ↪slow. If the outcome of the series will be to narrate how Sauron forged the␣
 ↪Rings of Power, it will definitely be something to watch. Until this moment,␣
 ↪I think that in general terms, at least the first chapter delivers. I think␣
 ↪enough to be cautiously optimistic about what the next 7 episodes might turn␣
 ↪out to be. I must add, again that I am pleasantly surprised.")
```

[29]: [{'label': 'Positive', 'score': 0.9936692118644714}]