ELECTRA cho bài toán Text Classification

Trong notebook này, ta sẽ thực hành bài toán phân loại văn bản với mô hình ELECTRA

Hiện tại, mô hình ELECTRA đã được đóng gói thành một module mà ta có thể gọi qua thư viện transformers. Điều này giúp cho việc thử nghiệm với mô hình trở nên dễ dàng hơn rất nhiều.

General Language Understanding Evaluation (GLUE) là một bộ Benchmark được dùng để đánh giá khả năng đọc hiểu ngôn ngữ của các mô hình AI dành cho ngôn ngữ tiếng Anh. Bộ benchmark GLUE được tổng hợp từ 10 bài toán nhỏ khác nhau, đại diện cho những bài toán tiêu biểu trong xử lý ngôn ngữ. Trong đó, bài toán text classification được đại diện bởi dataset Stanford Sentiment Tree-2 (SST-2). Ta có thể đọc thêm về GLUE tại đây.

Trong khuôn khổ của notebook này, ta sẽ thực hiện bài toán phân loại văn bản với bộ data SST-2 (đã được cung cấp)

Sau khi hoàn thiện notebook này, học viên sẽ có:

- Hiểu biết sơ bộ về thư viện Pytorch dùng trong Deep Learning và Transformers cho các tác vụ xử lý ngôn ngữ
- Biết cách xây dựng một mô hình phân loại văn bản dùng ELECTRA

Bước 1: Thiết lập môi trường lập trình và cài đặt thư viện

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

Ta nên chuyển thư mục làm việc vào nơi chứa dataset của bộ SST-2 vào thư mục mà ta muốn làm việc.

```
import os
os.chdir('/content/drive/My Drive/Colab Notebooks/VinBDI-Daotao/Deep Learning/Question Answering')
```

Sử dụng lệnh `!ls` để kiểm tra các file đã đầy đủ chưa.

```
!ls glue_data/SST-2
```

```
dev.tsv  original  test.tsv  train.tsv
```

Ngoài bộ SST-2, ta cũng có thể download các bộ dữ liệu của GLUE thông qua mã nguồn tại đây

Sau khi đã có môi trường lập trình, ta cài đặt thư viện `transformers`

```
!pip install transformers
```

```
Collecting transformers
  Downloading https://files.pythonhosted.org/packages/d8/f4/9f93f06dd2c57c7cd7aa515ffbf9fcfd8a084
     |████████████████████████████████| 1.0MB 8.8MB/s
Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (from transfor
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from transforme
Collecting sentencepiece!=0.1.92
  Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a78cca907a6ff9a5e9fe4f0
     |████████████████████████████████| 1.1MB 31.1MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from transformers
Collecting tokenizers==0.8.1.rc2
  Downloading https://files.pythonhosted.org/packages/80/83/8b9fccb9e48eeb575ee19179e2bdde0ee9a19
     |████████████████████████████████| 3.0MB 53.0MB/s
Requirement already satisfied: dataclasses; python_version < "3.7" in /usr/local/lib/python3.6/di
Requirement already satisfied: filelock in /usr/local/lib/python3.6/dist-packages (from transforme
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.6/dist-packages (from transfo
Collecting sacremoses
  Downloading https://files.pythonhosted.org/packages/7d/34/09d19aff26edcc8eb2a01bed8e98f13a15370
     |████████████████████████████████| 890kB 49.6MB/s
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.6/dist-packages (from pa
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from packaging->tran
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from reques
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from sacremoses->t
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from sacremoses-
Building wheels for collected packages: sacremoses
  Building wheel for sacremoses (setup.py) ... done
  Created wheel for sacremoses: filename=sacremoses-0.0.43-cp36-none-any.whl size=893257 sha256=79
  Stored in directory: /root/.cache/pip/wheels/29/3c/fd/7ce5c3f0666dab31a50123635e6fb5e19ceb42ce38
Successfully built sacremoses
Installing collected packages: sentencepiece, tokenizers, sacremoses, transformers
Successfully installed sacremoses-0.0.43 sentencepiece-0.1.91 tokenizers-0.8.1rc2 transformers-3.2
```

Bước 2: Làm quen với thư viện và dữ liệu

2.1: Làm quen với thư viện `transformers`

Trong thư viện này, ta sẽ được cung cấp API để sử dụng nhiều mô hình transformers khác nhau chỉ với một vài dòng lệnh.

Để có thể sử dụng mô hình Electra cho bài toán phân loại văn bản, ta sẽ dùng ba class:

- transformers.ElectraConfig: để cấu hình cho mô hình ELECTRA
- transformers.ElectraTokenizer: để mã hóa văn bản thành các token để đưa vào mô hình Electra
- transformers.ElectraForSequenceClassification: module cung cấp cho chúng ta kiến trúc mô hình ELECTRA đi kèm với một tầng neuron để phân loại sau cùng.

Ví dụ về cách sử dụng như sau:

```
from transformers import ElectraTokenizer, ElectraForSequenceClassification, ElectraConfig
import torch
```

```
#1. Khai báo hai đối tượng tokenizer với phiên bản small
tokenizer = ElectraTokenizer.from_pretrained('google/electra-small-discriminator')
config = ElectraConfig(num_labels =2)
model = ElectraForSequenceClassification(config).from_pretrained('google/electra-small-discriminator',


#2. Chuyển đổi văn bản thành chuỗi các token và định nghĩa label cho mô hình
inputs = tokenizer("Hello, my dog is cute", return_tensors="pt")
labels = torch.tensor([1]).unsqueeze(0)  # Batch size 1

#3 Nạp đầu vào và đầu ra cho mô hình
outputs = model(**inputs, labels=labels)

# Lấy ra giá trị loss và giá trị dự đoán của mô hình
loss = outputs.loss
logits = outputs.logits

print(loss)
print(logits)
```

```
Some weights of the model checkpoint at google/electra-small-discriminator were not used when init
- This IS expected if you are initializing ElectraForSequenceClassification from the checkpoint of
- This IS NOT expected if you are initializing ElectraForSequenceClassification from the checkpoin
Some weights of ElectraForSequenceClassification were not initialized from the model checkpoint at
You should probably TRAIN this model on a down-stream task to be able to use it for predictions an
tensor(0.7003, grad_fn=<NllLossBackward>)
tensor([[-0.0148, -0.0291]], grad_fn=<AddmmBackward>)
```

Sau khi khởi tạo, object `model` sẽ có tính chất như một model thông thường của Pytorch. Cách huấn luyện model như thế nào ta sẽ cùng tìm hiểu trong notebook dưới đây.

Bước 2.2: Import và quan sát dữ liệu

Các file dữ liệu của SST-2 được lưu dưới định dạng `.tsv`, chia thành ba subset là train, dev, test. Đầu tiên, hãy import 3 file dữ liệu vào 3 dataframe trong notebook này. Ta có thể dùng `pd.read_csv()` để thực hiện điều đó.

```
import pandas as pd
import torch.optim as optim
from torch import nn
from transformers import ElectraTokenizer, ElectraForSequenceClassification, ElectraConfig
import torch
import torch.optim as optim
from torch import nn


# Dữ liệu training
# YOUR CODE HERE
train_df =  pd.read_csv("glue_data/SST-2/train.tsv", sep='\t', error_bad_lines=False)
print(train_df.shape)
train_df.head()
# YOUR CODE HERE
```

```
(67349, 2)
```

	sentence	label
0	hide new secretions from the parental units	0
1	contains no wit , only labored gags	0
2	that loves its characters and communicates som...	1
3	remains utterly satisfied to remain the same t...	0
4	on the worst revenge-of-the-nerds clichés the ...	0

```
# Dữ liệu valuation
# YOUR CODE HERE
dev_df =  pd.read_csv("glue_data/SST-2/dev.tsv", sep='\t', error_bad_lines=False)
print(dev_df.shape)
dev_df.head()
# YOUR CODE HERE
```

```
(872, 2)
```

	sentence	label
0	it 's a charming and often affecting journey .	1
1	unflinchingly bleak and desperate	0
2	allows us to hope that nolan is poised to emba...	1
3	the acting , costumes , music , cinematography...	1
4	it 's slow -- very , very slow .	0

```
# Dữ liệu test
# YOUR CODE HERE
test_df =  pd.read_csv("glue_data/SST-2/test.tsv", sep='\t', error_bad_lines=False)
print(test_df.shape)
test_df.head()
# YOUR CODE HERE
```

```
(1821, 2)
```

	index	sentence
0	0	uneasy mishmash of styles and genres .
1	1	this film 's relationship to actual tension is...
2	2	by the end of no such thing the audience , lik...
3	3	director rob marshall went out gunning to make...
4	4	lathan and diggs have considerable personal ch...

Trong các file data mà ta được cung cấp, có 2 file có nhãn là file `train` và file `dev`. Ta sẽ dùng dữ liệu từ 2 file này trong mã nguồn của chúng ta: file train để train model và file dev để kiểm thử

Bước 3: Tiền xử lý và xây dựng mô hình

3.1. Lập trình hàm `tokenize` để mã hóa văn bản thành các token

Đầu vào là một hoặc một list các văn bản, đầu ra là object đã được biến đổi, đồng thời, hàm này sẽ cho ta lựa chọn giữa các phiên bản small, base, large của ELECTRA

```
def tokenize(text, version = "small"):
  # YOUR CODE HERE
  tokenizer = ElectraTokenizer.from_pretrained('google/electra-{}-discriminator'.format(version))
  transformed_data = tokenizer(text, return_tensors="pt", padding = True)
  return transformed_data
  # YOUR CODE HERE
```

```
#tokenize dữ liệu huấn luyện với hàm tokenize()
train_data = tokenize(train_df.sentence.values.tolist())
type(train_data)
```

```
transformers.tokenization_utils_base.BatchEncoding
```

```
# tokenize dữ liệu validation với hàm tokenize()
dev_data = tokenize(dev_df.sentence.values.tolist())
type(dev_data)
```

```
transformers.tokenization_utils_base.BatchEncoding
```

3.2. Lấy nhãn của train và dev data và chuyển đổi thành kiểu dữ liệu `torch.Tensor`

```
# chuyển đổi nhãn của tập train thành Tensor sử dụng torch.Tensor()
train_labels = torch.Tensor(train_df.label)
type(train_labels)
```

```
torch.Tensor
```

```
# chuyển đổi nhãn của tập validation thành Tensor sử dụng torch.Tensor()
dev_labels = torch.Tensor(dev_df.label)
type(dev_labels)
```

```
torch.Tensor
```

3.3. Lập trình hàm để huấn luyện mô hình

Lập trình hàm `train` để khởi tạo và huấn luyện mô hình cho bài toán của notebook. Hàm train sẽ nhận vào những tham số như sau:

- `train_data`: dữ liệu để huấn luyện, thuộc kiểu `transformers.tokenization_utils_base.BatchEncoding`

- `batch_size` : Kích thước của dữ liệu huấn luyện,
- `epochs` : Số vòng huấn luyện
- `use_cuda` : Để là `True` nếu ta muốn dùng GPU để huấn luyện
- `display_every` : Hiển thị thông số training sau bao nhiêu bước
- `version` : Phiên bản ELECTRA mà ta muốn sử dụng

Hàm train trong notebook này đã được viết một phần, nhiệm vụ của bạn là hoàn thành phần đọc dữ liệu theo từng batch để đưa vào mô hình huấn luyện.

Lưu ý: Mô hình ElectraForSequenceClassification() sẽ nhận ba đầu vào là input_ids, token_type_ids và attention_mask (tương ứng với ba attribute cùng tên trong object `train_data` . Trong đó, input_ids là bắt buộc.

```
# 3 input của train_data mà ta sẽ phải đưa vào mô hình
train_data.keys()
```

```
     dict_keys(['input_ids', 'token_type_ids', 'attention_mask'])
```

```
def train(train_data, batch_size, epochs, use_cuda = True, display_every = 200, version = "small"):
  # Lấy kích thước dữ liệu và khởi tạo mô hình
  DATA_SIZE = len(train_data.input_ids)
  model = ElectraForSequenceClassification.from_pretrained('google/electra-{}-discriminator'.format(ver

  # Chuyển đổi mô hình sang chạy trên GPU nếu option use_cuda=True
  if use_cuda:
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    model.to(device)

  # Định nghĩa hàm loss và thuật toán tối ưu cho mô hình
  criterion = nn.BCELoss()
  optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)


  for epoch in range(2):  # loop over the dataset multiple times

    train_batch = batch_size
    cursor = 0
    running_loss = 0.0

    # Đọc dữ liệu theo từng batch và đưa vào mô hình để huấn luyện

    # YOUR CODE HERE
    while cursor < DATA_SIZE:
      step = cursor/train_batch + 1
      if cursor + train_batch>=DATA_SIZE:
        train_batch = DATA_SIZE - cursor

      # get the inputs; data is a list of [inputs, labels]
      input_id = train_data.input_ids[cursor:cursor+train_batch].type(torch.LongTensor)
      token_id = train_data.token_type_ids[cursor:cursor+train_batch].type(torch.LongTensor)
      attn_mask = train_data.attention_mask[cursor:cursor+train_batch].type(torch.LongTensor)
      label = train_labels[cursor:cursor+train_batch].type(torch.LongTensor)
```

```
    # YOUR CODE HERE


      # Chuyển dữ liệu sang chế độ GPU nếu có
      # YOUR CODE HERE
      if use_cuda:
          input_id = input_id.cuda()
          token_id = token_id.cuda()
          attn_mask = attn_mask.cuda()
          label = label.cuda()

      # YOUR CODE HERE

      # zero the parameter gradients
      optimizer.zero_grad()


      # Tiếp theo, ta viết phần để mô hình thực hiện ba phần: forward_propagation, back_propagation, op

      # YOUR CODE HERE
      # Giai đoạn forward propagation (truyền xuôi) của mô hình
      outputs = model(input_ids = input_id, token_type_ids = token_id, attention_mask = attn_mask, labe

      # Minimize hàm loss qua câu lệnh loss.backward()

      loss = outputs.loss
      loss.backward()

      # Tối ưu tham số mô hình  qua hàm optimizer.step()
      optimizer.step()

      # YOUR CODE HERE
      # Dịch chuyển sang batch tiếp theo để huấn luyện
      cursor += train_batch

      # In thông số huấn luyện
      running_loss += loss.item()
      if step%display_every == 0:
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, step, running_loss /(display_every*train_batch)))
            running_loss = 0.0

  print('Finished Training')

  return model
```

3.4. Huấn luyện mô hình

```
model = train(train_data, 32, 2, use_cuda = True, display_every = 200, version = "small")
```

```
    Some weights of the model checkpoint at google/electra-small-discriminator were not used when ini
    - This IS expected if you are initializing ElectraForSequenceClassification from the checkpoint o
    - This IS NOT expected if you are initializing ElectraForSequenceClassification from the checkpoi
```

```
Some weights of ElectraForSequenceClassification were not initialized from the model checkpoint at
You should probably TRAIN this model on a down-stream task to be able to use it for predictions an
[1,   200] loss: 0.021
[1,   400] loss: 0.014
[1,   600] loss: 0.011
[1,   800] loss: 0.009
[1,  1000] loss: 0.009
[1,  1200] loss: 0.008
[1,  1400] loss: 0.008
[1,  1600] loss: 0.007
[1,  1800] loss: 0.007
[1,  2000] loss: 0.007
[2,   200] loss: 0.007
[2,   400] loss: 0.006
[2,   600] loss: 0.006
[2,   800] loss: 0.005
[2,  1000] loss: 0.005
[2,  1200] loss: 0.005
[2,  1400] loss: 0.005
[2,  1600] loss: 0.004
[2,  1800] loss: 0.005
[2,  2000] loss: 0.005
Finished Training
```

```
#Sau khi training, ta có thể lưu mô hình vào một file `.pth` để dùng lại khi cần
PATH = './ELECTRA_net.pth'
torch.save(model.state_dict(), PATH)
```

3.5 Đánh giá mô hình trên tập kiểm thử

Dưới đây ta có một hàm để đánh giá hiệu năng của mô hình, hãy hoàn thành phần so sánh giữa nhãn dự đoán và nhãn thực của code để hoàn thiện hàm.

```
def evaluate(dev_data, model,  batch_size = 32, use_cuda = True):
  test_size = len(dev_data.input_ids)
  correct = 0
  total = 0
  with torch.no_grad():
    cursor = 0
    running_loss = 0.0
    while cursor < test_size:
      step = cursor/batch_size + 1
      if cursor + batch_size>=test_size:
        batch_size = test_size - cursor

      # get the inputs; data is a list of [inputs, labels]
      input_id = dev_data.input_ids[cursor:cursor+batch_size].type(torch.LongTensor)
      token_id = dev_data.token_type_ids[cursor:cursor+batch_size].type(torch.LongTensor)
      attn_mask = dev_data.attention_mask[cursor:cursor+batch_size].type(torch.LongTensor)
      label = dev_labels[cursor:cursor+batch_size].type(torch.LongTensor)

      if use_cuda:
          input_id = input_id.cuda()
          token_id = token_id.cuda()
```

```
            attn_mask = attn_mask.cuda()
            label = label.cuda()

        outputs = model(input_ids = input_id, token_type_ids = token_id, attention_mask = attn_mask, labe
        _, predicted = torch.max(outputs.logits, 1)

        total = len(dev_labels)

        # Hoàn thiện phần thống kê số nhãn dự đoán giống nhãn thực tế
        # YOUR CODE HERE
        correct += (predicted == label.cuda()).sum().item()
        # YOUR CODE HERE

        cursor += batch_size
    print('Accuracy of the network on data: %d %%' % (100 * correct / total))

    return 100 * correct / total



evaluate(dev_data, model)
```

```
    Accuracy of the network on data: 89 %
    89.79357798165138
```

Với ELECTRA Small, ta có thể đạt độ chính xác 89% trên tập dev. Bạn có thể thử tăng cao độ chính xác bằng cách thay đổi tham số mô hình hoặc sử dụng phiên bản khác của ELECTRA.

Nếu bạn đã làm đến đây, xin chúc mừng! Bây giờ bạn đã có trong tay kiến thức về:

- Một thư viện học sâu phổ biến hàng đầu thế giới
- Một thư viện nổi tiếng trong NLP
- Và trên hết, cách sử dụng một mô hình State-of-the-art trong Deep Learning.

That's quite an achievement! Hy vọng những kiến thức bạn đã tích lũy được sẽ giúp ích cho bạn trong tương lai!