ELECTRA for Question Answering on SQUAD

Trong notebook này ta sẽ làm quen với mô hình Electra ứng dụng cho bài toán Question Answering. Electra là một phương pháp học biểu diễn ngôn ngữ (language representation learning) được ứng dụng cho nhiều bài toán khác nhau, ví dụ như Classification, QA, Text chunking. Đây là một phương pháp học biểu diễn mới, cho phép chúng ta đạt được hiệu năng cao với các Benchmark task trong NLP như SQUAD và GLUE (chi tiết xem tại paper ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators).

Stanford Question Answering Dataset (SQuAD) là một dataset cho bài toán đọc hiểu và trả lời câu hỏi được phát triển bởi đại học Stanford. Trong đó, với mỗi bản ghi, một hệ thống AI sẽ được cung cấp một đoạn văn bản để đọc hiểu và một câu hỏi, nhiệm vụ của hệ thống AI đó là trả lời câu hỏi đó bằng một đoạn trích từ đoạn văn bản được cung cấp nếu có thể, hoặc báo lại là không thể trả lời nếu đoạn văn cung cấp không thể dùng để trả lời câu hỏi.

ELECTRA được công bố với ba phiên bản theo kích thước tăng dần như sau: Small, Base, Large. Vì giới hạn về thời gian cũng như khả năng tính toán, trong notebook này ta sẽ tiến hành thử nghiệm với mô hình ELECTRA Small. Học viên nên chạy bài thực hành này trên notebook nếu không có server để hỗ trợ

Bước 1: Setup môi trường trên Google Colab

Học viên sử dụng nền tảng tính toán khác ngoài Google Colan có thể bỏ qua bước này. Trước khi chạy những câu lệnh dưới, ta chọn cấu hình GPU bằng cách ấn: **Runtime** -> **Change runtime type** -> **GPU**

1.1. Mount máy ảo vào drive của chúng ta

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

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

1.2. Cài đặt thư viện và tải mã nguồn ELECTRA

Trong bài thực hành này, ta sẽ sử dụng mã nguồn ELECTRA do bên Google Research phát triển. Để sử dụng mã nguồn này ta sẽ phải cài thư viện tensorflow==1.15 và

Đầu tiên ta cài đặt tensorflow phiên bản 1.15

```
!pip install tensorflow==1.15
```

```
Collecting tensorflow==1.15
  Downloading https://files.pythonhosted.org/packages/3f/98/5a99af92fb911d7a88a0005ad55005f35b4c1
     |████████████████████████████████| 412.3MB 40kB/s
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from tensor
```

```
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.6/dist-packages (from t
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-packages (from tens
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/python3.6/dist-packages (from te
Collecting keras-applications>=1.0.8
  Downloading https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf6342d71b28fbfd
     |████████████████████████████████| 51kB 7.9MB/s
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-packages (from ten
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-packages (from tens
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.6/dist-packages (from tensor
Collecting tensorflow-estimator==1.15.1
  Downloading https://files.pythonhosted.org/packages/de/62/2ee9cd74c9fa2fa450877847ba560b260f5d0
     |████████████████████████████████| 512kB 53.7MB/s
Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/lib/python3.6/dist-packages (fro
Collecting tensorboard<1.16.0,>=1.15.0
  Downloading https://files.pythonhosted.org/packages/1e/e9/d3d747a97f7188f48aa5eda486907f3b345cd
     |████████████████████████████████| 3.8MB 50.4MB/s
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from te
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.6/dist-packages (from tenso
Collecting gast==0.2.2
  Downloading https://files.pythonhosted.org/packages/4e/35/11749bf99b2d4e3cceb4d55ca22590b0d7c2c
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from protobu
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras-applica
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packages (from te
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-packages (from t
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/local/lib/pytho
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8" in /usr/local/lib/
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from importli
Building wheels for collected packages: gast
  Building wheel for gast (setup.py) ... done
  Created wheel for gast: filename=gast-0.2.2-cp36-none-any.whl size=7540 sha256=86e41042670edf23
  Stored in directory: /root/.cache/pip/wheels/5c/2e/7e/a1d4d4fcebe6c381f378ce7743a3ced3699feb89b
Successfully built gast
ERROR: tensorflow-probability 0.12.1 has requirement gast>=0.3.2, but you'll have gast 0.2.2 whic
Installing collected packages: keras-applications, tensorflow-estimator, tensorboard, gast, tenso
  Found existing installation: tensorflow-estimator 2.4.0
    Uninstalling tensorflow-estimator-2.4.0:
      Successfully uninstalled tensorflow-estimator-2.4.0
  Found existing installation: tensorboard 2.4.1
    Uninstalling tensorboard-2.4.1:
      Successfully uninstalled tensorboard-2.4.1
  Found existing installation: gast 0.3.3
    Uninstalling gast-0.3.3:
      Successfully uninstalled gast-0.3.3
  Found existing installation: tensorflow 2.4.1
    Uninstalling tensorflow-2.4.1:
      Successfully uninstalled tensorflow-2.4.1
Successfully installed gast-0.2.2 keras-applications-1.0.8 tensorboard-1.15.0 tensorflow-1.15.0 te
```

Sau đó, ta clone git repo của ELECTRA về không gian làm việc và cd và thư mục `electra`

```
!git clone https://github.com/google-research/electra.git
```

```
Cloning into 'electra'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
```

```
remote: Total 113 (delta 0), reused 2 (delta 0), pack-reused 104
Receiving objects: 100% (113/113), 124.09 KiB | 12.41 MiB/s, done.
Resolving deltas: 100% (46/46), done.
```

```
cd electra/
```

```
/content/electra
```

Tiếp theo, ta download và unzip file mô hình của phiên bản ELECTRA Small

```
!wget https://storage.googleapis.com/electra-data/electra_small.zip
!unzip electra_small.zip
```

```
--2021-02-22 02:35:05--  https://storage.googleapis.com/electra-data/electra_small.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 172.253.122.128, 142.250.31.128, 172
Connecting to storage.googleapis.com (storage.googleapis.com)|172.253.122.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 171877176 (164M) [application/zip]
Saving to: ‘electra_small.zip’

electra_small.zip   100%[===================>] 163.91M   203MB/s    in 0.8s

2021-02-22 02:35:06 (203 MB/s) - ‘electra_small.zip’ saved [171877176/171877176]

Archive:  electra_small.zip
   creating: electra_small/
  inflating: electra_small/checkpoint
  inflating: electra_small/electra_small.meta
  inflating: electra_small/electra_small.data-00000-of-00001
  inflating: electra_small/electra_small.index
  inflating: electra_small/vocab.txt
```

Bước 2: Download và quan sát dữ liệu

2.1. Download training và validation data của bộ Squad 2.0

```
!wget https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json
!wget https://rajpurkar.github.io/SQuAD-explorer/dataset/dev-v2.0.json
```

```
--2021-02-22 02:35:26--  https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json
Resolving rajpurkar.github.io (rajpurkar.github.io)... 185.199.108.153, 185.199.109.153, 185.199.1
Connecting to rajpurkar.github.io (rajpurkar.github.io)|185.199.108.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 42123633 (40M) [application/json]
Saving to: ‘train-v2.0.json’

train-v2.0.json     100%[===================>]  40.17M   202MB/s    in 0.2s

2021-02-22 02:35:27 (202 MB/s) - ‘train-v2.0.json’ saved [42123633/42123633]

--2021-02-22 02:35:27--  https://rajpurkar.github.io/SQuAD-explorer/dataset/dev-v2.0.json
```

```
Resolving rajpurkar.github.io (rajpurkar.github.io)... 185.199.108.153, 185.199.109.153, 185.199.1
Connecting to rajpurkar.github.io (rajpurkar.github.io)|185.199.108.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4370528 (4.2M) [application/json]
Saving to: ‘dev-v2.0.json’

dev-v2.0.json       100%[===================>]   4.17M  --.-KB/s    in 0.08s

2021-02-22 02:35:27 (53.8 MB/s) - ‘dev-v2.0.json’ saved [4370528/4370528]
```

2.2. Tạo thư mục để chứa dữ liệu huấn luyện và chuyển data vào thư mục đó

Đầu tiên, ta tạo một thư mục tên là `data` để chứa dữ liệu và file models

Trong đó, theo quy ước của mã nguồn:

- `finetuning_data/<tên tác vụ>` là thư mục chứa data cho tác vụ tương ứng, vì chúng ta đang làm bài toán Question Answering, nên tên thư mục con sẽ để là `squad`.
- `models` là thư mục chứa model của electra mà ta muốn sử dụng

Sau khi tạo hai thư mục này rồi, ta chuyển hai file json chứa dữ liệu của SQuAD 2.0 vào thư mục `squad`

```
!mkdir -p data/finetuning_data/squad
!mkdir -p data/models/
!mv dev-v2.0.json data/finetuning_data/squad/dev.json
!mv train-v2.0.json data/finetuning_data/squad/train.json
```

Tiếp theo, ta copy file vocab.txt từ thư mục `electra_small` sang thư mục `data`

```
!cp electra_small/vocab.txt data/vocab.txt
```

Cuối cùng, ta copy thư mục `electra_small` vào trong thư mục `data/models`

```
import shutil
shutil.copytree('electra_small', 'data/models/electra_small', copy_function = shu
```

```
'data/models/electra_small'
```

2.3. Quan sát dữ liệu

Bây giờ, ta sẽ thực hiện một vài thao tác thống kê để hiểu thêm về dữ liệu của Squad

```
import os
os.listdir("data/finetuning_data/squad")
```

```
['dev.json', 'train.json']
```

```
import json
from pprint import pprint
import numpy as np


def view_squad_info(subset = 'train', get_impossible_exp = False):
    with open("data/finetuning_data/squad/{}.json".format(subset), "r") as f:
      data = json.load(f)

    # Thống kê số văn bản
    numOfParagraph = 0
    # YOUR CODE HERE
    for i in range(len(data["data"])):
      numOfParagraph += len(data["data"][0]['paragraphs'])
    # YOUR CODE HERE


    # Thống kê số cặp câu hỏi câu trả lời
    numOfQaPair = 0
     # YOUR CODE HERE
    for i in range(len(data["data"])):
      for j in range(len(data["data"][i]['paragraphs'])):
        numOfQaPair += len(data["data"][i]['paragraphs'][j]["qas"])
    # YOUR CODE HERE


    # Thống kê độ dài của context
    ContextLen = []
    # YOUR CODE HERE
    for i in range(len(data["data"][0])):
      for j in range(len(data["data"][i]['paragraphs'])):
        ContextLen.append(len(data["data"][i]['paragraphs'][j]["context"]))
    maxContextLen = np.max(ContextLen)
    # YOUR CODE HERE


    # Thống kê độ dài của query và answer
    queryLen = [] # Độ dài của các query
    ansLen = [] # Độ dài của các câu trả lời

    # YOUR CODE HERE
    for i in range(len(data["data"])):
      for j in range(len(data["data"][i]['paragraphs'])):
        for k in range(len(data["data"][i]['paragraphs'][j]["qas"])):
          queryLen.append(len(data["data"][i]['paragraphs'][j]["qas"][k]["questic
          if len(data["data"][i]['paragraphs'][j]["qas"][k]["answers"]) > 0:
            ansLen.append(len(data["data"][i]['paragraphs'][j]["qas"][k]["answers
    # YOUR CODE HERE


    avgQueLen = np.mean(queryLen)
    avgAnsLen = np.mean(ansLen)


    print("Phiên bản SQuAd là {}".format(data["version"]))
    print("Số văn bản trong dataset là {}".format(len(data["data"])))
```

```
    print("Mỗi văn bản có những key sau: {}".format(data["data"][0].keys()))
    print("Số đoạn văn trong dataset là: {}".format(numOfParagraph))
    print("Số cặp câu hỏi và trả lời trong dataset là: {}".format(numOfQaPair))
    print("Độ dài tối đa của một đoạn văn là: {}".format(maxContextLen))
    print("Độ dài trung bình của một câu hỏi là: {}".format(avgQueLen))
    print("Độ dài trung bình của một trả lời là: {}".format(avgAnsLen))

    print("------MỘT SỐ CẶP CÂU VÍ DỤ-----")
    pprint(data["data"][0]['paragraphs'][0]["qas"][0:2])
    print("------------------------------")
    pprint(data["data"][-1]['paragraphs'][0]["qas"][0:2])

    if get_impossible_exp:
      for i in range(len(data["data"])):
        for j in range(len(data["data"][i]['paragraphs'])):
          for k in range(len(data["data"][i]['paragraphs'][j]["qas"])):
            if data["data"][i]['paragraphs'][j]["qas"][k]['is_impossible']:
              pprint(data["data"][i]['paragraphs'][j]["qas"][k])
```

Ta sử dụng hàm `view_squad_info` để xem thông tin của dataset:

```
view_squad_info(subset = 'train')
```

```
     Phiên bản SQuAd là v2.0
     Số văn bản trong dataset là 442
     Mỗi văn bản có những key sau: dict_keys(['title', 'paragraphs'])
     Số đoạn văn trong dataset là: 29172
     Số cặp câu hỏi và trả lời trong dataset là: 130319
     Độ dài tối đa của một đoạn văn là: 1895
     Độ dài trung bình của một câu hỏi là: 58.50773870272178
     Độ dài trung bình của một trả lời là: 20.149168979855105
     ------MỘT SỐ CẶP CÂU VÍ DỤ-----
     [{'answers': [{'answer_start': 269, 'text': 'in the late 1990s'}],
       'id': '56be85543aeaaa14008c9063',
       'is_impossible': False,
       'question': 'When did Beyonce start becoming popular?'},
      {'answers': [{'answer_start': 207, 'text': 'singing and dancing'}],
       'id': '56be85543aeaaa14008c9065',
       'is_impossible': False,
       'question': 'What areas did Beyonce compete in when she was growing up?'}]
     ------------------------------
     [{'answers': [],
       'id': '5a7db48670df9f001a87505f',
       'is_impossible': True,
       'plausible_answers': [{'answer_start': 50,
                              'text': 'ordinary matter composed of atoms'}],
       'question': 'What did the term matter include after the 20th century?'},
      {'answers': [],
       'id': '5a7db48670df9f001a875060',
       'is_impossible': True,
       'plausible_answers': [{'answer_start': 59, 'text': 'matter'}],
       'question': 'What are atoms composed of?'}]
```

```
view_squad_info(subset = 'dev', get_impossible_exp= False)
```

```
     Phiên bản SQuAd là v2.0
```

```
Số văn bản trong dataset là 35
Mỗi văn bản có những key sau: dict_keys(['title', 'paragraphs'])
Số đoạn văn trong dataset là: 1365
Số cặp câu hỏi và trả lời trong dataset là: 11873
Độ dài tối đa của một đoạn văn là: 1765
Độ dài trung bình của một câu hỏi là: 59.50619051629748
Độ dài trung bình của một trả lời là: 20.916160593792174
------MỘT SỐ CẶP CÂU VÍ DỤ-----
[{'answers': [{'answer_start': 159, 'text': 'France'},
              {'answer_start': 159, 'text': 'France'},
              {'answer_start': 159, 'text': 'France'},
              {'answer_start': 159, 'text': 'France'}],
  'id': '56ddde6b9a695914005b9628',
  'is_impossible': False,
  'question': 'In what country is Normandy located?'},
 {'answers': [{'answer_start': 94, 'text': '10th and 11th centuries'},
              {'answer_start': 87, 'text': 'in the 10th and 11th centuries'},
              {'answer_start': 94, 'text': '10th and 11th centuries'},
              {'answer_start': 94, 'text': '10th and 11th centuries'}],
  'id': '56ddde6b9a695914005b9629',
  'is_impossible': False,
  'question': 'When were the Normans in Normandy?'}]
------------------------------
[{'answers': [{'answer_start': 46, 'text': 'force'},
              {'answer_start': 46, 'text': 'force'},
              {'answer_start': 31, 'text': 'the concept of force'},
              {'answer_start': 31, 'text': 'the concept of force'},
              {'answer_start': 46, 'text': 'force'},
              {'answer_start': 46, 'text': 'force'}],
  'id': '573735e8c3c5551400e51e71',
  'is_impossible': False,
  'question': 'What concept did philosophers in antiquity use to study simple '
              'machines?'},
 {'answers': [{'answer_start': 387, 'text': 'fundamental error'},
              {'answer_start': 385, 'text': 'A fundamental error'},
              {'answer_start': 385, 'text': 'A fundamental error'},
              {'answer_start': 385, 'text': 'A fundamental error'},
              {'answer_start': 385, 'text': 'A fundamental error'},
              {'answer_start': 385, 'text': 'A fundamental error'}],
  'id': '573735e8c3c5551400e51e72',
  'is_impossible': False,
  'question': 'What was the belief that maintaining motion required force?'}]
```

Bên trên là một vài thông số cơ bản của SQuAD dataset. Nếu như ta muốn sử dụng lại mô hình ELECTRA cho bài toán Question-Answering, ta có thể làm hai việc sau:

- Xây dựng ngữ liệu cho ngôn ngữ mà bạn muốn xây dựng mô hình từ đó và tạo file vocab.txt tương ứng, sau đó chạy file run_finetuning.py trong mã nguồn để có được mô hình electra custom của bạn
- Thiết kế một dataset có cấu trúc giống như trên và lắp ghép với mã nguồn trong bài thực hành này.

Và đương nhiên, là phải có một server thật khỏe để chạy!

Bước 3: Training

Ta chạy dòng lệnh dưới đây để thực hiện training, chúng ta có thể thay đổi các tham số trong hparams và theo dõi sự khác biệt trong quá trình huấn luyện

```
# YOUR CODE HERE
!python3 run_finetuning.py --data-dir data --model-name electra_small --hparams '
# YOUR CODE HERE


# YOUR CODE HERE
with open("data/models/electra_small/results/squad_results.txt", "r") as f:
  result_file = f.read()

print(result_file.replace(" - ", "\n"))
# YOUR CODE HERE
```

Mô hình đang đạt được độ chính xác tầm 65% và f1 là 67% trên tập test của Squad, ta có thể cải thiện mô hình bằng cách chỉnh các tham số cho phù hợp và chạy mô hình với số epoch lớn hơn. Nếu như bạn có hệ thống máy mạnh hơn, bạn có thể thử nghiệm với các phiên bản lớn hơn của electra