DistilBert_Benchmark

August 10, 2021

Question answering comes in many forms. In this example, we'll look at the particular type of extractive QA that involves answering a question about a passage by highlighting the segment of the passage that answers the question. This involves fine-tuning a model which predicts a start position and an end position in the passage. We will use the Stanford Question Answering Dataset (SQuAD) 2.0.

0.1 Prerequisites:

1. Download and install the required libraries below.
2. Import the required libraries

[ ]:
```
!pip install pytorch-lightning
!pip install transformers
!pip install sentencepiece
!pip install wandb
```

```
Requirement already satisfied: pytorch-lightning in
/usr/local/lib/python3.7/dist-packages (1.4.0)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from pytorch-lightning) (3.7.4.3)
Requirement already satisfied: torch>=1.6 in /usr/local/lib/python3.7/dist-
packages (from pytorch-lightning) (1.9.0+cu102)
Requirement already satisfied: pyDeprecate==0.3.1 in
/usr/local/lib/python3.7/dist-packages (from pytorch-lightning) (0.3.1)
Requirement already satisfied: future>=0.17.1 in /usr/local/lib/python3.7/dist-
packages (from pytorch-lightning) (0.18.2)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.7/dist-
packages (from pytorch-lightning) (4.41.1)
Requirement already satisfied: numpy>=1.17.2 in /usr/local/lib/python3.7/dist-
packages (from pytorch-lightning) (1.19.5)
Requirement already satisfied: torchmetrics>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from pytorch-lightning) (0.4.1)
Requirement already satisfied: tensorboard!=2.5.0,>=2.2.0 in
/usr/local/lib/python3.7/dist-packages (from pytorch-lightning) (2.4.1)
Requirement already satisfied: packaging>=17.0 in /usr/local/lib/python3.7/dist-
packages (from pytorch-lightning) (21.0)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.7/dist-
packages (from pytorch-lightning) (5.4.1)
Requirement already satisfied: fsspec[http]!=2021.06.0,>=2021.05.0 in
```

```
/usr/local/lib/python3.7/dist-packages (from pytorch-lightning) (2021.7.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.7/dist-packages
(from fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (3.7.4.post0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (2.26.0)
Requirement already satisfied: pyparsing>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging>=17.0->pytorch-lightning)
(2.4.7)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-
packages (from tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (1.34.1)
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.7/dist-
packages (from tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (3.17.3)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (57.2.0)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-
packages (from tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (0.36.2)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (1.32.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-
packages (from tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (1.15.0)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (1.0.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (1.8.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-
packages (from tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (3.3.4)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (0.4.4)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-
packages (from tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (0.12.0)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<2,>=1.6.3->tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (4.2.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-
packages (from google-auth<2,>=1.6.3->tensorboard!=2.5.0,>=2.2.0->pytorch-
lightning) (4.7.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<2,>=1.6.3->tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (1.3.0)
Requirement already satisfied: importlib-metadata in
```

```
/usr/local/lib/python3.7/dist-packages (from
markdown>=2.6.8->tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (4.6.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<2,>=1.6.3->tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (0.4.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from
requests->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (2.0.2)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from
requests->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (2021.5.30)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from
requests->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (1.26.6)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning)
(2.10)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard!=2.5.0,>=2.2.0->pytorch-lightning) (3.1.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.7/dist-packages (from
aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (5.1.0)
Requirement already satisfied: chardet<5.0,>=2.0 in
/usr/local/lib/python3.7/dist-packages (from
aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (3.0.4)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.7/dist-
packages (from aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning)
(21.2.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.7/dist-
packages (from aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning)
(1.6.3)
Requirement already satisfied: async-timeout<4.0,>=3.0 in
/usr/local/lib/python3.7/dist-packages (from
aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (3.0.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-
metadata->markdown>=2.6.8->tensorboard!=2.5.0,>=2.2.0->pytorch-lightning)
(3.5.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-
packages (4.9.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from transformers) (1.19.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from transformers) (3.0.12)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: importlib-metadata in
```

```
/usr/local/lib/python3.7/dist-packages (from transformers) (4.6.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from transformers) (21.0)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/dist-
packages (from transformers) (0.0.45)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-
packages (from transformers) (4.41.1)
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in
/usr/local/lib/python3.7/dist-packages (from transformers) (0.10.3)
Requirement already satisfied: huggingface-hub==0.0.12 in
/usr/local/lib/python3.7/dist-packages (from transformers) (0.0.12)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from transformers) (2.26.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-
packages (from transformers) (5.4.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from huggingface-
hub==0.0.12->transformers) (3.7.4.3)
Requirement already satisfied: pyparsing>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging->transformers) (2.4.7)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->transformers) (3.5.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (2.0.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->transformers) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.26.6)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (2021.5.30)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (1.15.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (1.0.1)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (7.1.2)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.7/dist-
packages (0.1.96)
Requirement already satisfied: wandb in /usr/local/lib/python3.7/dist-packages
(0.11.2)
Requirement already satisfied: requests<3,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.26.0)
Requirement already satisfied: Click!=8.0.0,>=7.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (7.1.2)
Requirement already satisfied: shortuuid>=0.5.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (1.0.1)
Requirement already satisfied: subprocess32>=3.5.3 in
/usr/local/lib/python3.7/dist-packages (from wandb) (3.5.4)
```

```
Requirement already satisfied: configparser>=3.8.1 in
/usr/local/lib/python3.7/dist-packages (from wandb) (5.0.2)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (5.4.8)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages
(from wandb) (5.4.1)
Requirement already satisfied: sentry-sdk>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (1.3.1)
Requirement already satisfied: pathtools in /usr/local/lib/python3.7/dist-
packages (from wandb) (0.1.2)
Requirement already satisfied: python-dateutil>=2.6.1 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.8.1)
Requirement already satisfied: six>=1.13.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (1.15.0)
Requirement already satisfied: promise<3,>=2.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (2.3)
Requirement already satisfied: GitPython>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (3.1.18)
Requirement already satisfied: protobuf>=3.12.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (3.17.3)
Requirement already satisfied: urllib3>=1.26.5 in /usr/local/lib/python3.7/dist-
packages (from wandb) (1.26.6)
Requirement already satisfied: docker-pycreds>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (0.4.0)
Requirement already satisfied: typing-extensions>=3.7.4.0 in
/usr/local/lib/python3.7/dist-packages (from GitPython>=1.0.0->wandb) (3.7.4.3)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.7/dist-
packages (from GitPython>=1.0.0->wandb) (4.0.7)
Requirement already satisfied: smmap<5,>=3.0.1 in /usr/local/lib/python3.7/dist-
packages (from gitdb<5,>=4.0.1->GitPython>=1.0.0->wandb) (4.0.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.0.0->wandb) (2.0.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests<3,>=2.0.0->wandb) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.0.0->wandb)
(2021.5.30)
```

[1]:
```python
import torch
import transformers as tfs
import pytorch_lightning as pl
from pytorch_lightning.loggers import WandbLogger
from pytorch_lightning.metrics import Accuracy
from pytorch_lightning import loggers as pl_loggers
import json
from pathlib import Path
from torch.utils.data import DataLoader
```

```
from tqdm.notebook import tqdm
from torch.utils.tensorboard import SummaryWriter
from transformers import AdamW, DistilBertForQuestionAnswering,␣
↪DistilBertTokenizerFast
import string, re
```

```
2021-08-03 20:00:32.041474: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open
shared object file: No such file or directory; LD_LIBRARY_PATH: /usr/local/cuda/
extras/CUPTI/lib64:/usr/lib/cuda/include:/usr/lib/cuda/lib64:/usr/local/cuda-10.
1/lib64
2021-08-03 20:00:32.041499: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
```

1 1. Data Understanding

In this section we will import the data & convert it correctly into paralell lists of contexts, questions and answers provided in the SQuAD 2.0 Dataset.

1.1 Download SQuAD 2.0 Data

Note : This dataset can be explored in the Hugging Face model hub (SQuAD V2), and can be alternatively downloaded with the NLP library with load_dataset("squad_v2").

[ ]:
```
## Create a squad directory and download the train and evaluation datasets␣
↪directly into the library
!mkdir squad
!wget https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json -O␣
↪squad/train-v2.0.json
!wget https://rajpurkar.github.io/SQuAD-explorer/dataset/dev-v2.0.json -O squad/
↪dev-v2.0.json
```

```
mkdir: cannot create directory ‘squad’: File exists
--2021-08-03 10:44:06--  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, 185.199.110.153, …
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: ‘squad/train-v2.0.json’

squad/train-v2.0.js 100%[===================>]  40.17M   116MB/s    in 0.3s

2021-08-03 10:44:06 (116 MB/s) - ‘squad/train-v2.0.json’ saved
[42123633/42123633]
```

```
--2021-08-03 10:44:06--  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, 185.199.110.153, …
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: ‘squad/dev-v2.0.json’

squad/dev-v2.0.json 100%[===================>]   4.17M  --.-KB/s    in 0.05s

2021-08-03 10:44:07 (75.9 MB/s) - ‘squad/dev-v2.0.json’ saved [4370528/4370528]
```

Below we will import the data and convert it into parallel lists of contexts, questions, and answers.

[ ]:
```
def read_squad(path):
    path = Path(path)
    with open(path, 'rb') as f:
        squad_dict = json.load(f)

    contexts = []
    questions = []
    answers = []
    combined_qac=[] #combined contexts, questions & answers
    counter=0
    for group in squad_dict['data']:
        for passage in group['paragraphs']:
            context = passage['context']
            for qa in passage['qas']:
                question = qa['question']
                q_answers = qa['answers'].copy()
                q_answers = list(map(lambda x:x['text'], q_answers))
                for answer in qa['answers']:
                    contexts.append(context)
                    questions.append(question)
                    answers.append(answer)
                    combined_qac.append({'context':context,'question':
 ↪question,'answers':q_answers})
    return contexts, questions, answers, combined_qac

train_contexts, train_questions, train_answers,train_qac = read_squad('squad/
 ↪train-v2.0.json')
val_contexts, val_questions, val_answers, val_qac = read_squad('squad/dev-v2.0.
 ↪json')
```

Now that we have converted the data into parallel lists, let us assess what the dataset holds.

[ ]:
```
len(train_contexts)
```

[ ]:
```
86821
```

[ ]:
```
train_contexts[0]
```

[ ]:
```
'Beyoncé Giselle Knowles-Carter (/bi j nse / bee-YON-say) (born September 4,
1981) is an American singer, songwriter, record producer and actress. Born and
raised in Houston, Texas, she performed in various singing and dancing
competitions as a child, and rose to fame in the late 1990s as lead singer of
R&B girl-group Destiny\'s Child. Managed by her father, Mathew Knowles, the
group became one of the world\'s best-selling girl groups of all time. Their
hiatus saw the release of Beyoncé\'s debut album, Dangerously in Love (2003),
which established her as a solo artist worldwide, earned five Grammy Awards and
featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby
Boy".'
```

[ ]:
```
train_questions[0]
```

[ ]:
```
'When did Beyonce start becoming popular?'
```

[ ]:
```
train_answers[0]
```

[ ]:
```
{'answer_start': 269, 'text': 'in the late 1990s'}
```

[ ]:
```
len(train_qac)
```

[ ]:
```
86821
```

[ ]:
```
train_qac[0]
```

[ ]:
```
{'answers': ['in the late 1990s'],
 'context': 'Beyoncé Giselle Knowles-Carter (/bi j nse / bee-YON-say) (born
September 4, 1981) is an American singer, songwriter, record producer and
actress. Born and raised in Houston, Texas, she performed in various singing and
dancing competitions as a child, and rose to fame in the late 1990s as lead
singer of R&B girl-group Destiny\'s Child. Managed by her father, Mathew
Knowles, the group became one of the world\'s best-selling girl groups of all
time. Their hiatus saw the release of Beyoncé\'s debut album, Dangerously in
Love (2003), which established her as a solo artist worldwide, earned five
Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in
Love" and "Baby Boy".',
 'question': 'When did Beyonce start becoming popular?'}
```

Inspecting Validation Data

[ ]:
```
len(val_contexts)
```

[ ]: 20302

[ ]:
```
val_contexts[0]
```

[ ]:
```
'The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the
people who in the 10th and 11th centuries gave their name to Normandy, a region
in France. They were descended from Norse ("Norman" comes from "Norseman")
raiders and pirates from Denmark, Iceland and Norway who, under their leader
Rollo, agreed to swear fealty to King Charles III of West Francia. Through
generations of assimilation and mixing with the native Frankish and Roman-
Gaulish populations, their descendants would gradually merge with the
Carolingian-based cultures of West Francia. The distinct cultural and ethnic
identity of the Normans emerged initially in the first half of the 10th century,
and it continued to evolve over the succeeding centuries.'
```

[ ]:
```
val_questions[0]
```

[ ]:
```
'In what country is Normandy located?'
```

[ ]:
```
val_answers[0]
```

[ ]:
```
{'answer_start': 159, 'text': 'France'}
```

[ ]:
```
val_qac[0]
```

[ ]:
```
{'answers': ['France', 'France', 'France', 'France'],
 'context': 'The Normans (Norman: Nourmands; French: Normands; Latin: Normanni)
were the people who in the 10th and 11th centuries gave their name to Normandy,
a region in France. They were descended from Norse ("Norman" comes from
"Norseman") raiders and pirates from Denmark, Iceland and Norway who, under
their leader Rollo, agreed to swear fealty to King Charles III of West Francia.
Through generations of assimilation and mixing with the native Frankish and
Roman-Gaulish populations, their descendants would gradually merge with the
Carolingian-based cultures of West Francia. The distinct cultural and ethnic
identity of the Normans emerged initially in the first half of the 10th century,
and it continued to evolve over the succeeding centuries.',
 'question': 'In what country is Normandy located?'}
```

1.2 Observations:

- We have successfully created 3 subsets of both the training and validation sets
- We gathered the following stats:
  - **Training Data**
    - Length: 86821
    - The combined_qac shows the way things will work, i.e.: We submit a context & a question to the model & receive the answer already highlighted
    - train_answers shows the answer for a particular question and the start index value
  - **Validation Data**

* Length: 20302
* Similar to the train_qac we have created a val_qac to understand the validation dataset better as well

2 2. Data processing

In this section we will prepare the data appropriately for modelling and training.

We will extract token positions where answers begins & ends for train & validation data.

The contexts and questions are just strings. The answers are dicts containing the subsequence of the passage with the correct answer as well as an integer indicating the character at which the answer begins. In order to train a model on this data we need (1) the tokenized context/question pairs, and (2) integers indicating at which token positions the answer begins and ends.

First, let's get the character position at which the answer ends in the passage (we are given the starting position). Sometimes SQuAD answers are off by one or two characters, so we will also adjust for that.

[ ]:
```
## Index the answers and contexts in the training and validation sets. This␣
↪will help us generate the tokens
## and helpp get better answers for our questions
def add_end_idx(answers, contexts):
    for answer, context in zip(answers, contexts):
        gold_text = answer['text']
        start_idx = answer['answer_start']
        end_idx = start_idx + len(gold_text)

        # sometimes squad answers are off by a character or two - fix this
        if context[start_idx:end_idx] == gold_text:
            answer['answer_end'] = end_idx
        elif context[start_idx-1:end_idx-1] == gold_text:
            answer['answer_start'] = start_idx - 1
            answer['answer_end'] = end_idx - 1     # When the gold label is off␣
↪by one character
        elif context[start_idx-2:end_idx-2] == gold_text:
            answer['answer_start'] = start_idx - 2
            answer['answer_end'] = end_idx - 2     # When the gold label is off␣
↪by two characters

add_end_idx(train_answers, train_contexts)
add_end_idx(val_answers, val_contexts)
```

observation

[ ]:
```
## Initialize a tokenizer using DistilBERT which will help us tokenize our␣
↪training questions and answers

tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')
```

```
## obtain encoded training and validation sets from the tokenizer
train_encodings = tokenizer(train_contexts, train_questions, truncation=True,␣
↪padding=True)
val_encodings = tokenizer(val_contexts, val_questions, truncation=True,␣
↪padding=True)
```

observation

[ ]:
```
## Create a function to add token positions
def add_token_positions(encodings, answers):
    start_positions = []
    end_positions = []
    for i in range(len(answers)):
        start_positions.append(encodings.char_to_token(i,␣
↪answers[i]['answer_start']))
        end_positions.append(encodings.char_to_token(i,␣
↪answers[i]['answer_end'] - 1))
        # if None, the answer passage has been truncated
        if start_positions[-1] is None:
            start_positions[-1] = tokenizer.model_max_length
        if end_positions[-1] is None:
            end_positions[-1] = tokenizer.model_max_length
    encodings.update({'start_positions': start_positions, 'end_positions':␣
↪end_positions})

add_token_positions(train_encodings, train_answers)
add_token_positions(val_encodings, val_answers)
```

Our data is ready. Let's just put it in a PyTorch dataset so that we can easily use it for training. In PyTorch, we define a custom Dataset class.

3 3. Train & Validation Dataset Creation

[ ]:
```
## Creating the taining and validation datasets using the encoded training and␣
↪validation sets we created in
## the section above
class SquadDataset(torch.utils.data.Dataset):
    def __init__(self, encodings):
        self.encodings = encodings

    def __getitem__(self, idx):
        return {key: torch.tensor(val[idx]) for key, val in self.encodings.
↪items()}

    def __len__(self):
        return len(self.encodings.input_ids)
```

```
train_dataset = SquadDataset(train_encodings)
val_dataset = SquadDataset(val_encodings)
```

3.0.1 Observations

4 4. Model Building & Training

[ ]:
```
model = DistilBertForQuestionAnswering.
 ↪from_pretrained("distilbert-base-uncased")
```

```
Some weights of the model checkpoint at distilbert-base-uncased were not used
when initializing DistilBertForQuestionAnswering: ['vocab_projector.bias',
'vocab_layer_norm.bias', 'vocab_transform.bias', 'vocab_layer_norm.weight',
'vocab_transform.weight', 'vocab_projector.weight']
- This IS expected if you are initializing DistilBertForQuestionAnswering from
the checkpoint of a model trained on another task or with another architecture
(e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing DistilBertForQuestionAnswering
from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
Some weights of DistilBertForQuestionAnswering were not initialized from the
model checkpoint at distilbert-base-uncased and are newly initialized:
['qa_outputs.weight', 'qa_outputs.bias']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

4.0.1 Observations

[ ]:
```
# Training the created model using the available cuda gpu or cpu
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model.to(device) # send the model to the available device for training.
model.train()

train_dataloader = DataLoader(train_dataset, batch_size=8, shuffle=False)
val_dataloader = torch.utils.data.
 ↪DataLoader(val_dataset,batch_size=8,shuffle=False)

optim = AdamW(model.parameters(), lr=5e-5)
```

4.0.2 Observations

[ ]:
```
# Removing articles and punctuation, and standardizing whitespace are all␣
↪typical text processing steps


def normalize_text(s):

    def remove_articles(text):
        regex = re.compile(r"\b(a|an|the)\b", re.UNICODE)
        return re.sub(regex, " ", text)

    def white_space_fix(text):
        return " ".join(text.split())

    def remove_punc(text):
        exclude = set(string.punctuation)
        return "".join(ch for ch in text if ch not in exclude)

    def lower(text):
        return text.lower()

    return white_space_fix(remove_articles(remove_punc(lower(s))))
```

[ ]:
```
# Function to compute the exact match for an answer.
# This will help us determine how accurately do our answers match with the␣
↪suggested answers
def compute_exact_match(prediction, truth):
    return int(normalize_text(prediction) == normalize_text(truth))
```

[ ]:
```
# Function to compute the F1 Statistic

def compute_f1(prediction, truth):
    pred_tokens = normalize_text(prediction).split()
    truth_tokens = normalize_text(truth).split()

    # if either the prediction or the truth is no-answer then f1 = 1 if they␣
↪agree, 0 otherwise
    if len(pred_tokens) == 0 or len(truth_tokens) == 0:
        return int(pred_tokens == truth_tokens)

    common_tokens = set(pred_tokens) & set(truth_tokens)

    # if there are no common tokens then f1 = 0
    if len(common_tokens) == 0:
        return 0
```

```
    prec = len(common_tokens) / len(pred_tokens)
    rec = len(common_tokens) / len(truth_tokens)

    return 2 * (prec * rec) / (prec + rec)
```

[ ]:
```
# Function to calculate exact match and exact F1 score for a particular␣
 ↪training epoch
def calculate_stats(input_ids,start,end,idx):
    batch_start = 8*idx
    batch_end = batch_start+8
    data = val_qac[batch_start:batch_end]
    em = 0
    ef1 = 0
    for i,d in enumerate(data):
        answer_start = start[i]
        answer_end = end[i]
        answer = tokenizer.convert_tokens_to_string(tokenizer.
 ↪convert_ids_to_tokens(input_ids[i][answer_start:answer_end]))
        gold_ans = d['answers']
        if len(gold_ans)==0:
            gold_ans.append("")
        em_s= max((compute_exact_match(answer, g_answer)) for g_answer in␣
 ↪gold_ans)
        ef1_s = max((compute_f1(answer, g_answer)) for g_answer in gold_ans)
        em+=em_s
        ef1+=ef1_s
    return em,ef1
```

4.0.3 Observations

[26]:
```
# Train for the model, perform validation on it per epoch and generate files␣
 ↪for a tensorboard
num_epochs = 10

writer = SummaryWriter()

for epoch in range(num_epochs):
    print('Epoch {}/{}'.format(epoch, num_epochs - 1))
    print('-' * 10)
    model.train()
    running_loss = 0.0
    tk0 = tqdm(train_dataloader, total=int(len(train_dataloader)))
    counter = 0
    for idx,batch in enumerate(tk0):
        optim.zero_grad()
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
```

```
        start_positions = batch['start_positions'].to(device)
        end_positions = batch['end_positions'].to(device)
        outputs = model(input_ids, attention_mask=attention_mask,␣
↪start_positions=start_positions, end_positions=end_positions)
        loss = outputs[0]
        loss.backward()
        optim.step()
        running_loss += loss.item() *  batch['input_ids'].size(0)
        counter += 1
        tk0.set_postfix(loss=(running_loss / (counter * train_dataloader.
↪batch_size)))
    epoch_loss = running_loss / len(train_dataloader)
    writer.add_scalar('Train/Loss', epoch_loss,epoch)
    print('Training Loss: {:.4f}'.format(epoch_loss))

    model.eval()
    running_val_loss=0
    running_val_em=0
    running_val_f1=0
    tk1 = tqdm(val_dataloader, total=int(len(val_dataloader)))
    for idx,batch in enumerate(tk1):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        start_positions = batch['start_positions'].to(device)
        end_positions = batch['end_positions'].to(device)
        outputs = model(input_ids, attention_mask=attention_mask,␣
↪start_positions=start_positions, end_positions=end_positions)
        running_val_loss += loss.item() *  batch['input_ids'].size(0)
        counter += 1
        tk1.set_postfix(loss=(running_loss / (counter * val_dataloader.
↪batch_size)))
        answer_start = torch.argmax(outputs['start_logits'], dim=1)
        answer_end = torch.argmax(outputs['end_logits'], dim=1) + 1
        em_score, f1_score =␣
↪calculate_stats(input_ids,answer_start,answer_end,idx)
        running_val_em += em_score
        running_val_f1 += f1_score
    l = len(val_qac)
    epoch_v_loss = running_val_loss /l
    epoch_v_em = running_val_em/l
    epoch_val_f1 = running_val_f1/l
    writer.add_scalar('Val/Loss', epoch_v_loss,epoch)
    writer.add_scalar('Val/EM', epoch_v_em,epoch)
    writer.add_scalar('Val/F1', epoch_val_f1,epoch)
    print('Val Loss: {:.4f}, EM: {:.4f}, F1: {:.4f} '.
↪format(epoch_v_loss,epoch_v_em,epoch_val_f1))
```

```
Epoch 0/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 11.9780
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 1.9800, EM: 0.6107, F1: 0.7186
Epoch 1/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 7.8185
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 1.7960, EM: 0.6163, F1: 0.7188
Epoch 2/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 5.7666
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 0.0275, EM: 0.6131, F1: 0.7211
Epoch 3/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 4.4696
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 0.8989, EM: 0.5981, F1: 0.7045
Epoch 4/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 3.6274
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))
```

```
Val Loss: 0.0398, EM: 0.6093, F1: 0.7167
Epoch 5/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 2.9996
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 0.3840, EM: 0.6107, F1: 0.7175
Epoch 6/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 2.5831
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 0.0306, EM: 0.5998, F1: 0.7124
Epoch 7/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 2.2640
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 0.0585, EM: 0.5909, F1: 0.7017
Epoch 8/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 2.0188
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))

Val Loss: 0.0178, EM: 0.5839, F1: 0.7003
Epoch 9/9
----------
HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 1.8150
```

```
HBox(children=(FloatProgress(value=0.0, max=2538.0), HTML(value='')))


Val Loss: 0.0019, EM: 0.5798, F1: 0.6940
```

4.0.4 Observations

[27]:
```
# We save our model so that it can be reused later

torch.save(model,'./distilBertModel.pt')
```

[2]:
```
# Generate a Tensorboard

%load_ext tensorboard
%tensorboard --logdir runs
```

```
<IPython.core.display.HTML object>
```

4.0.5 Observations

We have created a Tensorboard to map the loss and accuracy across the various epochs that the model has trained at.

We will now run some examples to see how our model is performing & is it responding correctly to our questions.

5 5. Running The Model

We will now test the model on some contexts and questions to see if we are getting the correct answers

[29]:
```
test_context = """The Normans (Norman: Nourmands; French: Normands; Latin:␣
↪Normanni) were the people who in the 10th and 11th centuries gave their name␣
↪to Normandy, a region in France. They were descended from Norse ("Norman"␣
↪comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway␣
↪who, under their leader Rollo, agreed to swear fealty to King Charles III of␣
↪West Francia. Through generations of assimilation and mixing with the native␣
↪Frankish and Roman-Gaulish populations, their descendants would gradually␣
↪merge with the Carolingian-based cultures of West Francia. The distinct␣
↪cultural and ethnic identity of the Normans emerged initially in the first␣
↪half of the 10th century, and it continued to evolve over the succeeding␣
↪centuries."""


test_question = """Who was the Norse leader?"""

test_answer =  "Rollo"
```

[30]:
```
def question_answer(question, context, model):
    inputs = tokenizer(question,context, return_tensors='pt')

    input_ids = inputs['input_ids'].to(device)

    attention_mask = inputs['attention_mask'].to(device)
    inputs.to(device)
    start_scores, end_scores = model(input_ids, attention_mask=attention_mask,␣
 ↪output_attentions=False)[:2]

    all_tokens = tokenizer.convert_ids_to_tokens(input_ids[0])
    answer = ' '.join(all_tokens[torch.argmax(start_scores) : torch.
 ↪argmax(end_scores)+1])
    answer = tokenizer.convert_tokens_to_ids(answer.split())
    answer = tokenizer.decode(answer)
    return answer
```

[31]:
```
question_answer(test_question, test_context, model)
```

[31]:
```
'rollo'
```

[32]:
```
question_answer(val_questions[0], val_contexts[0], model)
```

[32]:
```
'france'
```

[37]:
```
model_loaded = torch.load('./distilBertModel.pt')
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

[38]:
```
## we will now take some text at random from Wikipedia and test our model. This␣
 ↪excerpt can be found at:
## https://en.wikipedia.org/wiki/Long_short-term_memory under the Idea heading.
context = """In theory, classic (or "vanilla") RNNs can keep track of arbitrary␣
 ↪long-term dependencies in the input sequences. The problem with vanilla RNNs␣
 ↪is computational (or practical) in nature: when training a vanilla RNN using␣
 ↪back-propagation, the gradients which are back-propagated can "vanish" (that␣
 ↪is, they can tend to zero) or "explode" (that is, they can tend to␣
 ↪infinity), because of the computations involved in the process, which use␣
 ↪finite-precision numbers. RNNs using LSTM units partially solve the␣
 ↪vanishing gradient problem, because LSTM units allow gradients to also flow␣
 ↪unchanged. However, LSTM networks can still suffer from the exploding␣
 ↪gradient problem."""
question = """What problem can LSTM suffer from?"""
answer = """exploding gradient problem"""
```

[39]:
```
question_answer(question, context, model_loaded)
```

[39]: 'vanishing gradient problem'

[32]:

[32]:

[32]:

[32]:

[32]: