Albert_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

```
[ ]: !pip3 install torch

     !pip3 install transformers
     !pip3 install sentencepiece
     !pip3 install wandb
```

Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages
(1.9.0+cu102)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from torch) (3.7.4.3)
Collecting transformers
  Downloading transformers-4.9.1-py3-none-any.whl (2.6 MB)
     | | 2.6 MB 15.8 MB/s
Collecting huggingface-hub==0.0.12
  Downloading huggingface_hub-0.0.12-py3-none-any.whl (37 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from transformers) (3.0.12)
Collecting sacremoses
  Downloading sacremoses-0.0.45-py3-none-any.whl (895 kB)
     | | 895 kB 61.6 MB/s
Requirement already satisfied: tqdm>=4.27 in
/usr/local/lib/python3.7/dist-packages (from transformers) (4.41.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from transformers) (1.19.5)
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)

1

```
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers) (2019.12.20)
Collecting pyyaml>=5.1
  Downloading PyYAML-5.4.1-cp37-cp37m-manylinux1_x86_64.whl (636 kB)
     |                              | 636 kB 70.5 MB/s
Collecting tokenizers<0.11,>=0.10.1
  Downloading tokenizers-0.10.3-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_6
4.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (3.3 MB)
     |                              | 3.3 MB 65.2 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from transformers) (2.23.0)
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: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->transformers) (2.10)
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: click in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (7.1.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (1.0.1)
Installing collected packages: tokenizers, sacremoses, pyyaml, huggingface-hub,
transformers
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.0.12 pyyaml-5.4.1 sacremoses-0.0.45
tokenizers-0.10.3 transformers-4.9.1
Collecting sentencepiece
  Downloading
sentencepiece-0.1.96-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(1.2 MB)
     |                              | 1.2 MB 13.3 MB/s
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.96
Collecting wandb
```

```
Downloading wandb-0.11.2-py2.py3-none-any.whl (1.8 MB)
     |                    | 1.8 MB 14.6 MB/s
Collecting subprocess32>=3.5.3
  Downloading subprocess32-3.5.4.tar.gz (97 kB)
     |                    | 97 kB 9.3 MB/s
Requirement already satisfied: protobuf>=3.12.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (3.17.3)
Requirement already satisfied: Click!=8.0.0,>=7.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (7.1.2)
Collecting sentry-sdk>=1.0.0
  Downloading sentry_sdk-1.3.1-py2.py3-none-any.whl (133 kB)
     |                    | 133 kB 79.8 MB/s
Requirement already satisfied: psutil>=5.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (5.4.8)
Requirement already satisfied: python-dateutil>=2.6.1 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.8.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages
(from wandb) (5.4.1)
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)
Collecting urllib3>=1.26.5
  Downloading urllib3-1.26.6-py2.py3-none-any.whl (138 kB)
     |                    | 138 kB 92.3 MB/s
Requirement already satisfied: requests<3,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.23.0)
Collecting configparser>=3.8.1
  Downloading configparser-5.0.2-py3-none-any.whl (19 kB)
Collecting GitPython>=1.0.0
  Downloading GitPython-3.1.18-py3-none-any.whl (170 kB)
     |                    | 170 kB 93.9 MB/s
Collecting shortuuid>=0.5.0
  Downloading shortuuid-1.0.1-py3-none-any.whl (7.5 kB)
Collecting pathtools
  Downloading pathtools-0.1.2.tar.gz (11 kB)
Collecting docker-pycreds>=0.4.0
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
Collecting gitdb<5,>=4.0.1
  Downloading gitdb-4.0.7-py3-none-any.whl (63 kB)
     |                    | 63 kB 2.4 MB/s
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)
Collecting smmap<5,>=3.0.1
  Downloading smmap-4.0.0-py2.py3-none-any.whl (24 kB)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.0.0->wandb) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
```

```
packages (from requests<3,>=2.0.0->wandb) (2.10)
Collecting requests<3,>=2.0.0
  Downloading requests-2.26.0-py2.py3-none-any.whl (62 kB)
     |                          | 62 kB 1.1 MB/s
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: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.0.0->wandb)
(2021.5.30)
Building wheels for collected packages: subprocess32, pathtools
  Building wheel for subprocess32 (setup.py) … done
  Created wheel for subprocess32: filename=subprocess32-3.5.4-py3-none-any.whl
size=6502
sha256=24286522b9a6fb0b1179a017970325995708ab7583fc562be6303ad068bad41b
  Stored in directory: /root/.cache/pip/wheels/50/ca/fa/8fca8d246e64f19488d07567
547ddec8eb084e8c0d7a59226a
  Building wheel for pathtools (setup.py) … done
  Created wheel for pathtools: filename=pathtools-0.1.2-py3-none-any.whl
size=8806
sha256=2367f1f481439325ecee6f60313387077c2e3270b65981910b74c4b78650cdeb
  Stored in directory: /root/.cache/pip/wheels/3e/31/09/fa59cef12cdcfecc627b3d24
273699f390e71828921b2cbba2
Successfully built subprocess32 pathtools
Installing collected packages: smmap, urllib3, gitdb, subprocess32, shortuuid,
sentry-sdk, requests, pathtools, GitPython, docker-pycreds, configparser, wandb
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.24.3
    Uninstalling urllib3-1.24.3:
      Successfully uninstalled urllib3-1.24.3
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
ERROR: pip's dependency resolver does not currently take into account all

the packages that are installed. This behaviour is the source of the following

dependency conflicts.

google-colab 1.0.0 requires requests~=2.23.0, but you have requests 2.26.0 which

is incompatible.

datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is

incompatible.
Successfully installed GitPython-3.1.18 configparser-5.0.2 docker-pycreds-0.4.0
gitdb-4.0.7 pathtools-0.1.2 requests-2.26.0 sentry-sdk-1.3.1 shortuuid-1.0.1
smmap-4.0.0 subprocess32-3.5.4 urllib3-1.26.6 wandb-0.11.2
```

```
[1]: import torch
     import transformers as tfs

     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, AlbertForQuestionAnswering, AlbertTokenizerFast
     import string, re
```

```
2021-08-04 14:21:44.017878: 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
2021-08-04 14:21:44.017900: 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
```

```
--2021-08-03 11:20:08--  https://rajpurkar.github.io/SQuAD-
explorer/dataset/train-v2.0.json
Resolving rajpurkar.github.io (rajpurkar.github.io)… 185.199.108.153,
185.199.110.153, 185.199.111.153, …
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    121MB/s       in 0.3s

2021-08-03 11:20:10 (121 MB/s) - 'squad/train-v2.0.json' saved
[42123633/42123633]

--2021-08-03 11:20:10--  https://rajpurkar.github.io/SQuAD-
explorer/dataset/dev-v2.0.json
Resolving rajpurkar.github.io (rajpurkar.github.io)… 185.199.108.153,
185.199.110.153, 185.199.111.153, …
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 0.07s

2021-08-03 11:20:10 (60.6 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.

```python
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.

```python
## Index the answers and contexts in the training and validation sets. This
 will help us generate the tokens
## and help 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)
```

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

tokenizer = AlbertTokenizerFast.from_pretrained('albert-base-v2')

## 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)
```

HBox(children=(FloatProgress(value=0.0, description='Downloading', max=760289.0,
 ↪style=ProgressStyle(descripti…

HBox(children=(FloatProgress(value=0.0, description='Downloading', max=1312669.
 ↪0, style=ProgressStyle(descript…

HBox(children=(FloatProgress(value=0.0, description='Downloading', max=684.0,
 ↪style=ProgressStyle(description_…

```python
## 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)
```

3   3. Train & Validation Dataset Creation

```python
## 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

Our training and validation sets have been successully created. We will now use these to train, validate and score our model below.

4   4. Model Building & Training

```python
model = AlbertForQuestionAnswering.from_pretrained('albert-base-v2')
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=47376696.
→0, style=ProgressStyle(descrip…

Some weights of the model checkpoint at albert-base-v2 were not used when
initializing AlbertForQuestionAnswering: ['predictions.decoder.bias',
'predictions.dense.bias', 'predictions.dense.weight',
'predictions.LayerNorm.weight', 'predictions.decoder.weight',
'predictions.bias', 'predictions.LayerNorm.bias']
- This IS expected if you are initializing AlbertForQuestionAnswering 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 AlbertForQuestionAnswering from
the checkpoint of a model that you expect to be exactly identical (initializing
a BertForSequenceClassification model from a BertForSequenceClassification
model).
Some weights of AlbertForQuestionAnswering were not initialized from the model
checkpoint at albert-base-v2 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.

```python
# 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)
```

```python
# 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))))
```

```python
# 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))
```

```python
# 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
```

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

     writer = SummaryWriter()
```

```python
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: 9.4228

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

Val Loss: 0.8846, EM: 0.6720, F1: 0.7747
Epoch 1/9

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 10.4198

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

Val Loss: 1.4451, EM: 0.5506, F1: 0.6797
Epoch 2/9

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 6.5823

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

Val Loss: 0.0370, EM: 0.6623, F1: 0.7672
Epoch 3/9

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))

Training Loss: 5.4899
```

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


Val Loss: 0.0286, EM: 0.6682, F1: 0.7775
Epoch 4/9
----------

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))


Training Loss: 4.6742

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


Val Loss: 0.3087, EM: 0.6920, F1: 0.7962
Epoch 5/9
----------

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))


Training Loss: 4.1192

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


Val Loss: 0.0018, EM: 0.6834, F1: 0.7840
Epoch 6/9
----------

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))


Training Loss: 3.6522

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


Val Loss: 0.0042, EM: 0.6826, F1: 0.7868
Epoch 7/9
----------

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))


Training Loss: 3.1741

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


Val Loss: 0.0088, EM: 0.6823, F1: 0.7868
Epoch 8/9
----------

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))
```

```
Training Loss: 2.8555

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


Val Loss: 0.1793, EM: 0.6827, F1: 0.7876
Epoch 9/9
----------

HBox(children=(FloatProgress(value=0.0, max=10853.0), HTML(value='')))


Training Loss: 2.4778

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


Val Loss: 0.0006, EM: 0.6648, F1: 0.7730
```

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

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

[3]:
```python
# Generate a Tensorboard

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

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 295325), started 0:01:10 ago. (Use '!kill␣
↪295325' to kill it.)
```

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

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"
```

```python
[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
```

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

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

```python
[32]: ## Checking the response is the same as we had in the validation set.
      question_answer(val_questions[0], val_contexts[0], model)
```

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

```python
[33]: model_loaded = torch.load('./albertModel.pt')
      tokenizer = AlertTokenizerFast.from_pretrained('albert-base-v2')
      device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
  File "<ipython-input-33-155bc8814cc7>", line 2
    tokenizer = AlertTokenizerFast.from_pretrained('albert-base-v2'')
                                                                    ^
SyntaxError: EOL while scanning string literal
```

```python
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"
```

```python
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
```

```python
question_answer(test_question, test_context, model_loaded)
```

```python
## 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"""
```

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

6 END OF FILE