Develop New Transformer Architecture For Question and Answering

Nirbhay P. Tandon
Master's of Science in Data Science
Liverpool John Moore's University

August, 2021

Contents

List of Figures

List of Tables

List of Abbreviations

ALBERT	A LITE BERT
Ans-Ptr	Answer-Pointer layer
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional LSTM
BLEU	Bilingual Evaluation Understudy
BPPTs	Back-Propagation Through Time
CEC	Constant Error Carousel
CNNs	Convolutional Neural Networks
DistilBERT	Distilled Version of BERT
EM	Exact Match
FFN	Feed Forward Network(s)
GloVe	Global Vectors for Word Representation
GPUs	Graphical Processing Units
LSTMs	Long Short-Term Memory Architectures
MCTest	Machine Comprehension of Text
MLM	Masked Language Model
NLP	Natural Language Processing
NLU	Natural Language Understanding
QASENT	A challenge dataset for open-domain question answering
QnA	Question and Answering
RACE	ReAding Comprehension Dataset From Examinations
RNNs	Recurrent Neural Networks
RoBERTa	Robustly Optimized BERT

SQuAD ..... Stanford Question Answering Dataset

SuBiLSTM .. Suffix Bidirectional LSTM

WMT ..... Workshop on Statistical Machine Translation

Abstract

Attention-based Transformer architectures have become the norm of current Natural Language Processing applications. Google began this trend back in 2017 with their paper *Attention Is All You Need*, by introducing the Transformer architecture that works solely on attention mechanisms. The purpose of our work will be to create a new kind of Transformer architecture. Compare and contrast its performance against other architectures such as BERT, DistilBERT, ALBERT etc. using the SQuAD 2.0 Dataset. Through our research, we aim to produce a new architecture that has a better performance than existing models specifically for Question Answering based applications.

Chapter 1

Introduction

One of the most philosophical questions that anyone can encounter is *"Can an answer exist without a question?"*. While that question exists to be solved by philosophers and thinkers, we shall aim to address the *answer* that can be provided to a question in terms of the context that has been asked. While the field of Natural Language Processing(NLP) is vast and many unexplored areas exist, in this thesis the focus is entirely on developing a novel architecture that provides *highly contextual and relevant answers* to the questions asked of it.

1.1 Background Of The Study

The earliest examples of Question-Answering(QA) systems can be traced back to 1961, in the form of BASEBALL,(Green Jr et al., 1961), a system developed by Bert F. Green and team to answer basic questions posed using punch cards in ordinary English language about stored data on baseball games. The team implemented a dictionary structure to allow for answers to be looked up and printed. For example, the system could answer very direct questions such as: "Where did each team play on July 7?". A highly impressive feat for the early days of neural networks in the 1960s, the work done by Green paved the way for systems like LUNAR,(Woods and WA, 1977), answered geological questions for the rocks brought back from the Apollo moon missions. LUNAR, which was first presented in 1971, had an astounding 90% accuracy rate for questions that were asked by people who didn't know how the system worked! Question-Answering based systems have gained a lot of popularity, especially in the form of "chatbots". These systems depend highly on contextual understanding of the input, the training corpus and the question asked. They use this knowledge to output an answer that can help the user with whatever their query is. Recurrent neural networks and architectures based on them, have been able to provide great advancements in the field of Question-answering and chatbots in general. However, there is a behaviour of over-fitting and a lack of contextual understanding of the question. This, coupled with long training times and extremely complex mathematical model designs, have often kept the field of Natural Language Processing slightly obscured from the masses.

The task of Question-Answering is a machine comprehension problem. On digging deeper, once can classify it as a sequential problem too. One might wonder how it is a sequential problem? The answer to that is simple, traditionally QA systems work by "sequentially reading" the input and producing an output based on what they understand semantically or contextually. In a more mathematical context, we can say that the problem

we face is of *answer selection* and not *answer provision*. Through numerous studies that have been done, some of which we shall look into detail in chapter 2, we see that the problem is not how to provide an answer to a question, it is if the answer provided is the correct one or not in the *context* in which the question has been posed.

The research presented forth here is aimed at proposing a new variation to an already successful architecture in the field of NLP, namely *Transformers*, created by the team at Google (Vaswani et al., 2017). A critically assessed attempt is made at combining state of the art approaches that have been treated as separate elements so far and to show that the proposed improvements are at par, if not better than the current approach.

A Transformer is a form of transduction model that relies solely on self-attention to figure out how to represent its inputs and outputs. It does so without the use of any sequence aligned recurrent neural networks(RNNs) or convolutions.

The area of Natural language Processing has taken significant leaps in the last two decades. Work done towards improving the ability of machine learning models to first recognize words, then sentences, followed by contextual understanding has led to several interesting and novel approaches in the field. From early on neural networks to creating Long Short-Term Memory architectures (Schmidhuber and Hochreiter, 1997) by Sepp Hochreiter and Jurgen Schmidhuber in the mid-'90s that resolved the vanishing gradient problem of classical neural networks, we have come a long way.

The latest advancements in this field come from Google's research lab in the form of *Transformers*. We look into this in a bit more detail later in 2. However, no model can be successful without a good dataset to train on. This is where the SQuAD 2.0 dataset (Rajpurkar et al., 2018) comes in. This dataset is what forces the machine learning models to do contextual understanding. One might even say that it forces the models to "think" for themselves before answering a task.

1.2 Aims And Objectives

The main aim of this research is to propose an improvement to the transformer architecture that can perform better at Q&A using the SQuAD 2.0 dataset. This shall require understanding the nuances of various Transformer-type architectures like AlBert (Lan et al., 2019), RoBerta (Liu et al., 2019) and DistilBert (Sanh et al., 2019) which are variations of the BERT (Devlin et al., 2018) architecture.

To achieve the main aim of this research it is important to first outline a few steps that will be taken:

1. Obtain SQuAD2.0 dataset and prepare it for training
2. Implement the existing models such as AlBert, RoBerta and DistilBert that are available via libraries such as HuggingFace (Team) and PyTorch on the dataset
3. Obtain F1, Exact Match(EM), Training Loss and Validation Loss scores for existing models to treat them as our benchmark values
4. Identify drawbacks of the current architectures
5. Design our architecture and evaluate its performance against the provided benchmark scores
6. Fine-tune the architecture, re-evaluate and report improvements

7. Compare the results of our Transformer model with the benchmark scores.

1.3 Scope Of The Study

This research work is entirely focused on the area of Question-Answering based networks. This focus helps in significantly narrowing down the scope and critically assessing the problem that needs to be solved. The practical applications of the field of QA are endless. From chatbots to QA systems on domain-specific knowledge or conversational applications for voice assistants, a light, robust, highly accurate Transformer based architecture can help ease the daily lives of millions of users.

1.4 Significance Of The Study

The current state of the art systems, some of which have been highlighted in section 1.1, suffer from various drawbacks and even the best models have only been able to achieve an EM score of 90.871 with an F1 score of 93.183 on the SQuAD 2.0 leaderboard (Rajpurkar, 2021). Current models such as AlBert and DistilBert are highly useful, however, they aren't scalable entirely for QA applications. A narrowed focus on the specific field of Question-Answering can help eliminate certain issues with the current models such as contextual uderstanding and being able to get correct responses to questions.

1.5 Structure Of The Study

This report is divided into 6 main parts, each chapter is further broken down into sections and subsections to localize the reference and use it further in the global context of this research.

In chapter 1, an introduction to the concept of Question-Answering, a brief look at the problem, motivation, scope of this research and its significance have been provided. Chapter 1 lays the basis of the problem, takes the reader briefly back in time to where QA systems first started and how they have advanced today.

Chapter 2, looks in detail at the background studies that have been briefly mentioned in section 1.1. In this chapter critical analysis is conducted how some of the work that has been selected is relevant to what this research aims to highlight, it helps us understand the merits and shortcomings of current state of the art approaches. Detailed analysis of Long Short-Term Memory(LSTM) and Transformer architectures is written, with special attention to applications in the field of Question Answering. There is particular focus on the advancements and applications of existing architectures as well as acknowledgement of why LSTMs reigned supreme for almost 2 decades. Transformers provide a unique opportunity to perform highly accurate predictive tasks in NLP. The most key advances to both LSTMs and Transformers are their bi-directional applications which are also discussed. Finally, this chapter ends with a summary of the approaches and highlights certain drawbacks. These drawbacks are discussed in more detail in 4.

Next, chapter 3 proposes the research methods that have been implemented. It explains the various metrics that have been used and why they were chosen, specifically highlighting why Accuracy isn't a good measure for creating a successful model and highlighting the use of Precision, Recall and F1 as useful metrics for the experiments conducted

in this particular thesis, Exact Match scores are also introduced and discussed in this chapter. The chapter ends with a research flow diagram describing the prorcesses followed for the experiments and application stages of various methods in a visual manner.

Appendix A is where the reader can refer to the Gantt chart for the thesis work done and the updates to various stages.

Chapter 2

Literature Review

To be able to effectively deliver on the aims and objectives we have defined in Section 1.2, we must first critically analyse the work that has been done in the field of NLP and, in particular, the field of Question-Answering based networks. In this section, we will critically analyse a few key model architectures and how we reached Transformers, which form the base of our proposed architecture.

2.1 Long Short-Term Memory

In 1997 (Schmidhuber and Hochreiter, 1997) introduced to the world a gradient descent based model in the form of Long Short-Term Memory. They set out to solve the problem of the vanishing gradient which was often seen in "Back-Propagation Through Time"(BPPTs) based neural networks. Extensive studies done on this, some by Hochreiter himself, showed that the problem of vanishing gradients is a real one. It was also seen that in case of BPTTs, the error back-flow mechanisms would either blow up or also suffer from vanishing gradients leading to either oscillating weights or learning to bridge long time gaps would not work. To remediate this, the authors introduced the concept of a *constant* error flow through the internal states of special units.

Their paper, Long Short-Term Memory, (Schmidhuber and Hochreiter, 1997), highlights that the work previously done using various gradient-descent variants, time-delays, time constants, use of higher order units to influence connections when the network receives conflicting error signals, Kalman filters, Second order nets etc. still suffer from some of the same problems that we have earlier. In the case of Ring's approach (Ring, 1993), the network can be really fast but also suffers from memory constraints and lack of generalization. This happens because to bridge n time lags, the model adds n additional units.

It is essential for us to understand how LSTMs function as it forms the basis of Transformers to exist. Hochreiter and Scmidhuber designed a *memory cell* that allows for a constant error flow. This is achieved by introducing *constant error carousel*(CEC), a multiplicative input gate unit and a multiplicative output gate unit. These *multiplicative* units help shield the contents of the memory cell from irrelevant inputs. The CEC, input and output gates together form the *memory cell*, as seen in Fig. 2.1. The central feature for LSTM's memory cell is the CEC. The self-recurrent connection, where weight is 1.0, is what helps create a time-delayed feedback loop by 1.

The CEC uses gate units to circumvent the issue with conflicting weights in the input and output layers. The input layer has control over when to use and when to override the

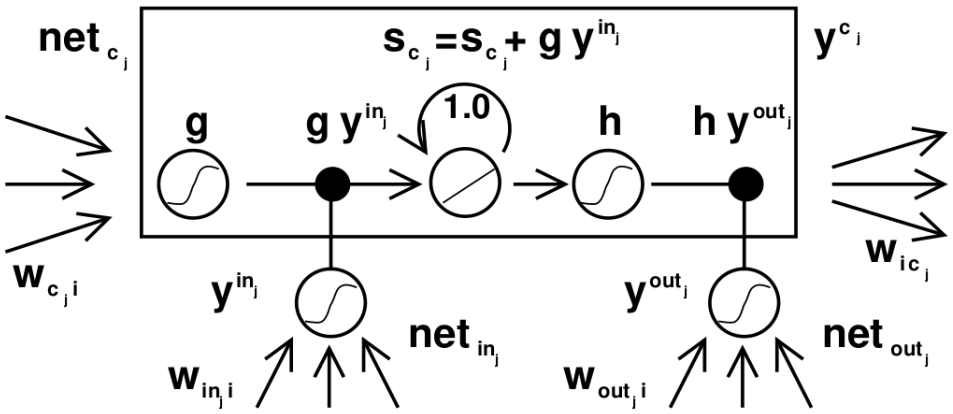


Figure 2.1: LSTM Memory cell with a Constant Error Carousel having fixed weight 1.0, (Schmidhuber and Hochreiter, 1997)

information in the CEC but has no control over the error signals that are in the memory cell. The output gate, however, uses a feedback mechanism to assess when to superimpose different error signals. It also falls on the output gate to *learn* which errors to trap in the CEC by appropriately scaling them. The gates have to learn when to release and trap errors therefore controlling the access to the CEC, helping maintain constant error flow. To mathematically understand the memory cell from fig. 2.1, Schmidhuber and Hochreiter proposed the following set of equations 2.1 - 2.3. More details on this can be found by referring to the appendix section A.1 in (Schmidhuber and Hochreiter, 1997).

$$y^{out_j}(t) = f_{out_j}\left(net_{out_j}(t)\right); y^{in_j}(t) = f_{in_j}\left(net_{in_j}(t)\right) \tag{2.1}$$

where

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1) \tag{2.2}$$

and

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1) \tag{2.3}$$

A generalised form of this equation can be represented as:

$$net_{c_j}(t) = \sum_u w_{c_j u} y^u(t-1) \tag{2.4}$$

At time t, c_j's output $y^{c_j}(t)$ is computed as:

$$y^{c_j}(t) = y^{out_j}(t) h(s_{c_j}(t)), \tag{2.5}$$

where the "internal state" $s_{c_j}(t)$ is:

$$s_{c_j}(0) = 0, s_{c_j}(t) = s_{c_j}(t-1) + y^{in_j}(t) g\left(net_{c_j}(t)\right) \text{ for } t > 0 \tag{2.6}$$

Where u may stand for input, output or gate units, memory cells or even conventional hidden units, if they are being used.

In fig. 2.2, we see the internal workings of how an LSTM network could look like.

A series of experiments were performed to prove that LSTMs are indeed "state-of-the-art" and provide an improvement over then existing RNN based architectures. Experiments around temporal order, which is defined as a series of events that happen in

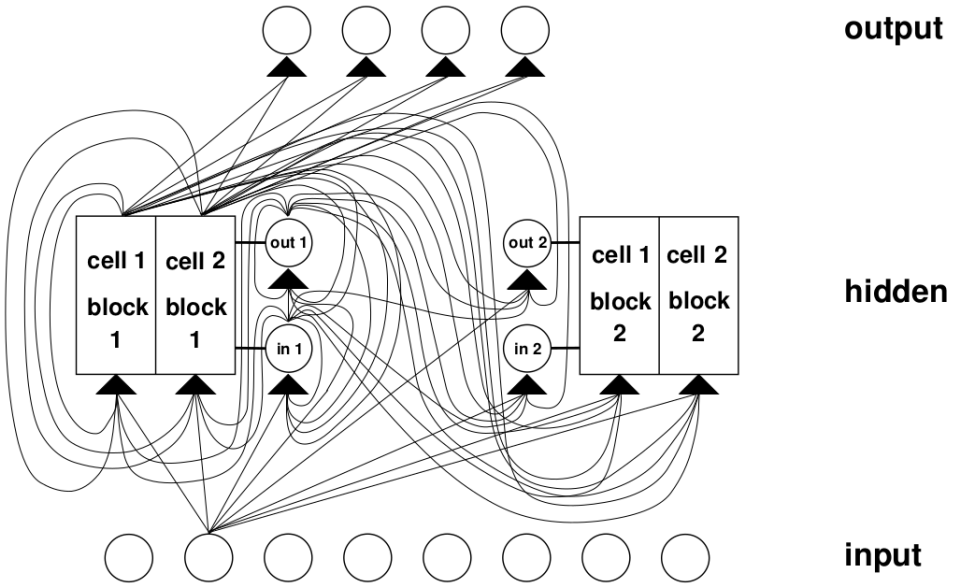


Figure 2.2: LSTM network with 8 input cells, 4 output cells and 2 memory cells of block size 2. Here $in1 \text{and} out1$ represent the input and output gates and $cell/block1$ is the first memory cell. The internal architecture of $cell/block1$ is similar to fig. 2.1. (Schmidhuber and Hochreiter, 1997)

a particular order, that were previously unsolved by RNNs, were successfully solved by LSTMs. This was possible because of how LSTMs address time lags vs. how RNNs suffer from vanishing gradients. Details of the analysis and experiments conducted are available in section 5 of the paper "Long Short-Term Memory" by (Schmidhuber and Hochreiter, 1997), with experimental summaries in tables 10 and 11 of the same. Being able to solve for temporal order based problems allowed LSTMs to be capable of solving various real world problems that we will discuss subsequently. Temporal orders are key in contextual understanding for NLP based tasks and especially in the case of QA based networks. Let us now look at some advantages and limitations of LSTMs, before moving on to understand how they have been applied in the field of QA.

Advantages of LSTMs

- LSTMs deal with long time lags by bridging the gaps using constant error flow in the memory cells.
- LSTMs respond well to generalization problems, even if the input sequences are widely separated.
- They work well with a variety of braod range hyper-parameters and don't usually require tuning.
- Their time and weight update complexity is the same as BPTT, ie. $O(1)$. The advantage for LSTMs however, is that it is local in both space and time.

Limitations of LSTMs

- LSTMs use memory cells which require an additional input and output unit. This *hidden unit* is replaced by 3 units in an LSTM architecture, compared to 9 in an RNN. A fully connected LSTM will have 3^2 connections however.
- LSTMs dont work well when they get the entire input in one go. The architecture is unable to generalize well by random weight guessing in this case.
- They do not have the ability to "natively" count discrete time steps which might be useful in some applications. This isn't a big issue for the case of QA based problems.

2.1.1 BiLSTMs

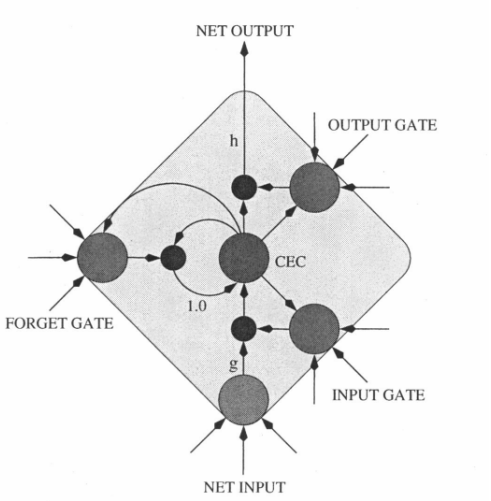


Figure 2.3: A memory cell for a biLSTM highlighting a *forget gate*. This forget gate helps in scaling the internal state of a memory cell, for example by resetting the state to 0. We can see that it is different to fig. 2.1, with the implementation of a connection to the forget gate, while still having an internal weight of 1. (Graves and Schmidhuber, 2005)

While LSTMs have found various applications across the field of machine and deep learning, QA seems to be one where being able to achieve consistent results of accuracy as well as having an exact match seems to elusive. Research carried out at IBM by Tan, Santos and co. looked at approaching this problem by using something called Bidirectional LSTMs or BiLSTM, (Graves and Schmidhuber, 2005). Introduced by Graves and Schmidhuber in their paper *Framewise Phoneme Classification with Bidirectional LSTM Networks*, the biLSTM implements a *forget gate* as seen in fig. 2.3. An advantage of biLSTMs over traditional LSTMs is that you feed the data twice, once from the beginning to the end and then from the end to the beginning. This helps the model contextualize and adjust weights better, therefore creating a more robust model that learns and performs better overall when compared to traditional LSTMs and RNNs.

BiLSTMs were introduced as an approach towards phoneme classification for speech recognition applications, however, in their paper "LSTM-BASED DEEP LEARNING MODELS FOR NON -FACTOID ANSWER SELECTION", (Tan et al., 2015) have shown that extending the biLSTM model 2 different approaches for QA has its advantages. One approach, using a more composite method of using CNNs for representing questions as well as answers. The other, by implementing an *attention mechanism* that helps in generating

	Questions	Answers	Question Word Count
Train	12887	18540	92095
Dev	1000	1454	7158
Test1	1800	2616	12893
Test2	1800	2593	12905

Table 2.1: InsuranceQA Corpus Details: Some questions can have multiple answers, therefore the number of answers is greater than the number of questions (Feng et al., 2015).

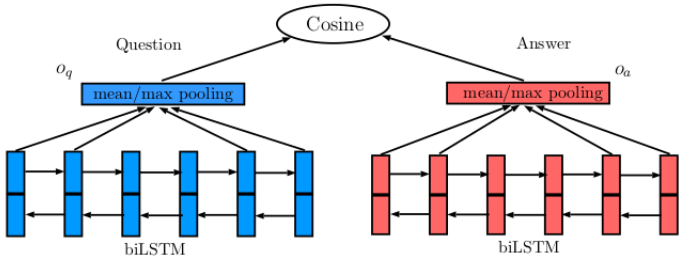


Figure 2.4: Basic QA-LSTM model depicting biLSTM implementation (Tan et al., 2015)

an answer representation according to the question context. Their framework is based on developing a biLSTM model for both questions and answers, applying a connective pooling layer and using a cosine similarity metric to calculate the degree of matching answers. They utilized the InsuranceQA(Feng et al., 2015) dataset that was created in 2015 and tested primarily on CNN based networks, details of the dataset can be found in table 2.1.

Pooling layers are known to suffer from an incapability to retain local linguistic information. To combat this the team proposed an additional CNN layer on top of the biLSTM layer. Additionally, they added a simple and efficient attention model to help their biLSTM model better distinguish between candidate answers according to the question context. This study is significant in the field of both deep learning based models and for the field of Question-Answering because it requires little or no feature engineering to achieve significant results. We will also see later in section 2.2, how biLSTMs are implemented with Transformers to produce an even better model. In their work (Tan et al., 2015) have shown that biLSTMs are capable of capitalizing long-range sequential context information. This is important as it is often seen that the answer is not directly semantically related to the question but more contextually related. Long-range exploitations help the model understand the question more holistically, therefore helping produce a more relevant answer.

BiLSTMs have an advantage over traditional LSTMs in that they utilize both previous and future contextual information by processing the input in both directions, forward and backward. This leads to the generation of two independent sequences of output vectors form LSTMs. The final output is a concatenation of the outputs from both directions ie. $h_t = \overrightarrow{h_t}||\overleftarrow{h_t}$. The **QA-LSTM** model proposed by (Tan et al., 2015) can be seen in fig. 2.4. We can see that it generates a distributed representation for both questions(blue) and answers(red) before moving on to perform pooling and cosine similarity operations. In fig. 2.5, it can be seen that the CNN layer has been implemented after the biLSTM layers and before the pooling layers. The pooling layers have been modified to be max-k pooling layers, similar to standard CNNs.

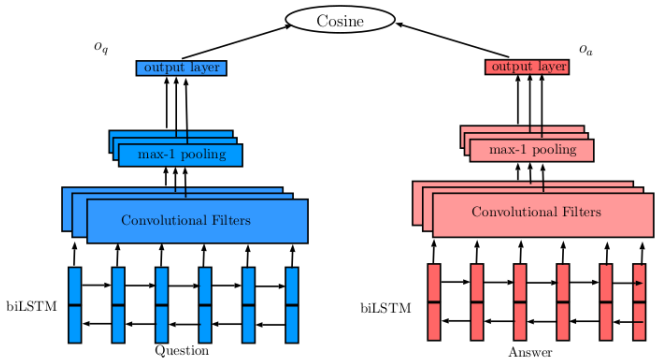


Figure 2.5: QA-LSTM with CNN layer (Tan et al., 2015)

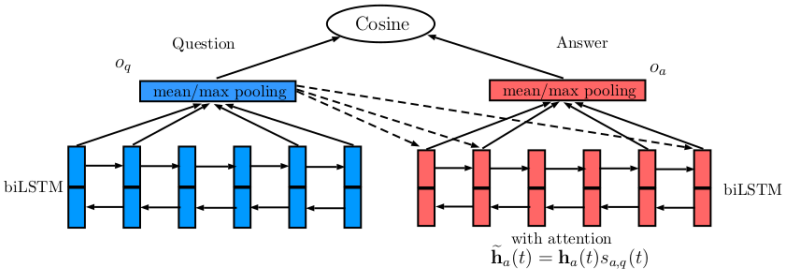


Figure 2.6: QA-LSTM with Attention layer (Tan et al., 2015)

The implementation of this convolution layer forces localized interactions between the inputs within a filter size of m. For each window of size m in biLSTM output vectors ie.

$$\mathbf{H}_m(t) = [\mathbf{h}(t), \mathbf{h(t+1)}, ..., \mathbf{h}(t+m-1)] \tag{2.7}$$

where t is a time step and the convolution filter will generate one value as follows

$$o_F(t) = tanh\left[\left(\sum_{i=0}^{m-1} \mathbf{h}(t+i)^T \mathbf{F}(i)\right) + b\right] \tag{2.8}$$

Using k-MaxPooling, the authors emphasized that a maximum of k values will be kept for one filter, thereby indicating the highest degree that a filter matches the input sequence. To end this approach, there are N parallel filters, with different parameter initialization which provide the CNN layer with N-dimension output vectors. This produces two output vectors of dimension kN for each question and answer. Their experiments intuitively took $k = 1$ as anything greater showed no improvements and it helped emphasize aspects of the answer such that this CNN based hybrid biLSTM could differentiate ground truths and incorrect answers efficiently.

The second approach adopted in this paper is actually inspired by another paper (Hermann et al., 2015) where attention based deep LSTM networks were used to enhance the capabilities of reading comprehension in LSTM based models. Here, in (Tan et al., 2015), a modification has been introduced in the form of attention-based connections along with biLSTM layers as depicted in fig. 2.6.

Before conducting a mean/average pooling operation on the output from every biLSTM, the output vectors are multiplied by a softmax weight that is determined by the

	Model	Validation	Test1	Test2
A	QA-LSTM basic-model(head/tail)	54.0	53.1	51.2
B	QA-LSTM basic-model(avg pooling)	58.5	58.2	54.0
C	QA-LSTM basic-model(max pooling)	64.3	63.1	58.0
D	QA-LSTM/CNN(fcount=1000)	65.5	65.9	62.3
E	QA-LSTM/CNN(fcount=2000)	64.8	66.8	62.6
F	QA-LSTM/CNN(fcount=4000)	66.2	64.6	62.2
G	QA-LSTM with attention (max pooling)	66.5	63.7	60.3
H	QA-LSTM with attention (avg pooling)	**68.4**	**68.1**	62.2
I	QA-LSTM/CNN (fcount=4000) with attention	67.2	65.7	**63.3**

Table 2.2: Experimental results from (Tan et al., 2015), highlighting how QA-LSTM with attention outperforms its various modifications

question embeddings from the biLSTM. It was shown that given an output vector from a biLSTM for an answer at time $t, \mathbf{h}_a(t)$ and the question embedding, o_q, the updated vector $\tilde{\mathbf{h}_a}(t)$ for each answer token can be given as:

$$\mathbf{m}_{a,q}(t) = tanh(\mathbf{W}_{am}\mathbf{h}_a(t) + \mathbf{W}_{qm}o_q) \tag{2.9}$$

$$s_{a,q}(t) \propto exp(\mathbf{w}_{ms}^T\mathbf{m}_{a,q}(t)) \tag{2.10}$$

$$\tilde{\mathbf{h}_a} = \mathbf{h}_a(t)s_{a,q}(t) \tag{2.11}$$

where $\mathbf{W}_{am}, \mathbf{W}_{qm}$ and $\mathbf{w}_m s$ are attention parameters. Attention parameters show somewhat similar behaviour to tf-idf where some words get more attention or weights associated to them. The main difference being that the attention mechanism adjusts its weights according to question information.

Critically, this attention based approach from (Tan et al., 2015) emphasizes on attention-driven representations and uses that to calculate the distances between questions and their respective answers. A combination approach with CNNs and attention based connections helps in localizing various internal connections, correctly scaling the errors and reducing unnecessary noise. From eq. 2.9-2.11, (Tan et al., 2015) we can mathematically prove how the average distances are computed. Experiments conducted on the InsuranceQA dataset showed that the *QA-LSTM/CNN with Attention* model can outperform the taken baseline models, which were based purely on QAs using CNNs.

An improvement to this approach was done in 2018 by (Brahma, 2018) who proposed *Suffix Bidirectional LSTM* or *SuBiLSTM* adding prefixes and suffixes to the inputs over both directions. There are a few obvious differences between the biLSTM and SuBiLSTM like the doubling of parameters, increased time complexity which happens over quadratic time for worst case scenarios compared to linear in biLSTMs. The author of this paper has shown that despite higher time complexity, there are significant gains in accuracy in text encoding and classification in their experiments, of particular note is the Paraphrase Detection experiment which uses GloVe embeddings and shows an 88.2% score in the test set, which is marginally higher than other models implemented. The work presented in this thesis will also implement paraphrase detection and GloVe embeddings to potentially improve model performance and answer detection.

2.1.2 LSTM And Question Answering

In 2016, The SQuAD dataset (Rajpurkar et al., 2016) was released, that generated quite a buzz in the field of QA based NLP tasks due to its nature of being extremely versatile and crowdsourced. We will take a deeper look at it in section **??**, where we discuss the various datasets in detail.

Using the SQuAD dataset (Wang and Jiang, 2016) developed a match-LSTM (Wang and Jiang, 2015) based model that they showed performed significantly better on the dataset than the work done by (Rajpurkar et al., 2016). Their work shows that using a match-LSTM approach combined with a Pointer Net model (Vinyals et al., 2015), that aids token predictions using only input sequences rather than using any form of large fixed vocabulary, therefore allowing for generation of answers with multiple tokens.

The approach by (Wang and Jiang, 2016) was tw-fold. The first approach implemented as a sequence model and the second as a boundary model, which was further extended with a search function.

The match-LSTM model architecture consists of:

- **LSTM Preprocessing Layer**: As we can see from 2.7, there are 2 LSTM preporcessing layers. This layer is used to integrate contextual information directly into the representation of how each token in the passage and question is represented. This is based directly on the simple LSTM (Schmidhuber and Hochreiter, 1997) and not the BiLSTM.

- **Match-LSTM Layer**: Here, the model treats the question as a premise and the passage as hypothesis. The layer sequentially reads the passage and calculates an attention weight vector $\vec{\alpha}_i \in \mathbb{R}^Q$ as:

$$\begin{aligned} \overrightarrow{\mathbf{G}}_i &= \tanh\left(\mathbf{W}^{\mathrm{q}}\mathbf{H}^{\mathrm{q}} + \left(\mathbf{W}^{\mathrm{p}}\mathbf{h}_i^{\mathrm{p}} + \mathbf{W}^{\mathrm{r}}\overrightarrow{\mathbf{h}}_{i-1}^{\mathrm{r}} + \mathbf{b}^{\mathrm{p}}\right) \otimes \mathbf{e}_Q\right) \\ \vec{\alpha}_i &= \mathrm{softmax}\left(\mathbf{w}^\top \overrightarrow{\mathbf{G}}_i + b \otimes \mathbf{e}_Q\right) \end{aligned} \tag{2.12}$$

  where $\mathbf{W}^{\mathrm{q}}, \mathbf{W}^{\mathrm{p}}, \mathbf{W}^{\mathrm{r}} \in \mathbb{R}^{l \times l}, \mathbf{b}^{\mathrm{p}}, \mathbf{w} \in \mathbb{R}^l$ and $b \in \mathbb{R}$ are parameters to be learned, $\overrightarrow{\mathbf{h}}_{i-1}^{\mathrm{r}} \in \mathbb{R}^l$ is the hidden vector of the one-directional match-LSTM at position $i-1$, and the outer product $(\cdot \otimes \mathbf{e}_Q)$ produces a matrix or row vector by repeating the vector or scalar on the left for Q times. To simply summarise, the above equation 2.12 produces $\vec{\alpha}_{i,j}$ that gives us the degree of matching between token i in the passage with token j in the question. This is the forward pass representation of the mathc-LSTM, the backward pass, which generates encodings for each token in the passage is given by:

$$\begin{aligned} \overleftarrow{\mathbf{G}}_i &= \tanh\left(\mathbf{W}^{\mathrm{q}}\mathbf{H}^{\mathrm{q}} + \left(\mathbf{W}^{\mathrm{p}}\mathbf{h}_i^{\mathrm{p}} + \mathbf{W}^{\mathrm{r}}\overleftarrow{\mathbf{h}}_{i+1}^{\mathrm{r}} + \mathbf{b}^{\mathbf{p}}\right) \otimes \mathbf{e}_Q\right) \\ \overleftarrow{\alpha}_i &= \mathrm{softmax}\left(\mathbf{w}^\top \overleftarrow{\mathbf{G}}_i + b \otimes \mathbf{e}_Q\right) \end{aligned} \tag{2.13}$$

  The equations above, 2.12 and 2.13 are presented in (Wang and Jiang, 2016).

- **Answer Pointer Layer**: This layer, the Answer-Pointer layer(Ans-Ptr), introduced by (Vinyals et al., 2015). It uses the sequence $\mathbf{H}^r$ as input.

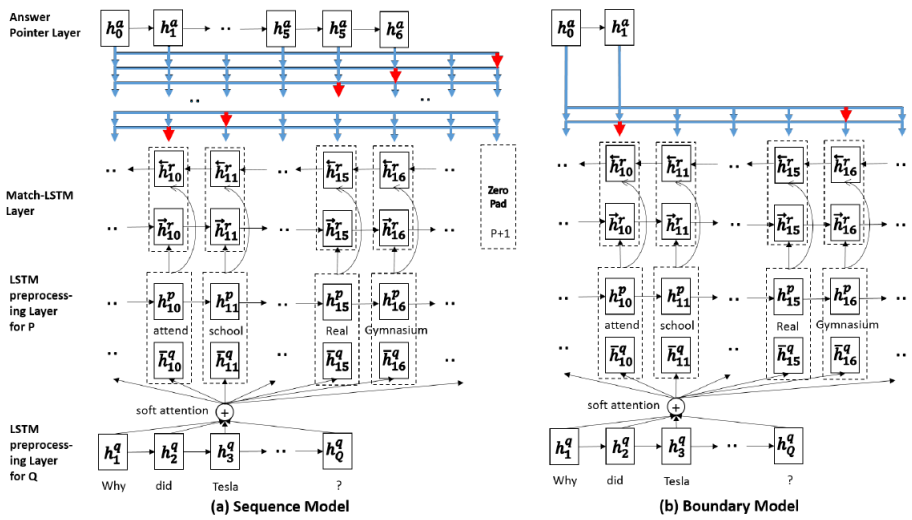


Figure 2.7: Overview of the 2 approaches by (Wang and Jiang, 2016), showing the Sequence Model on the left and the Boundary Model on the right.

It is important to highlight that the sequence model represents the answers as integers in a sequence that refer to the position of the selected token in the original passage. The Ans-Ptr layer models these sequentially as well. For the purposes of this thesis we will focus on the boundary model from the paper by (Wang and Jiang, 2016). As we can see from table 2.3, the results are significantly better for SQuAD 1.1 dataset. Further enhancements to this have been made and various other models have come out on top, however this paper pioneered the use of a bidirectional Ans-Ptr approach. Our research

	l	$\|\theta\|$	Exact Match		F1	
			Dev	Test	Dev	Test
Random Guess	–	0	1.1	1.3	4.1	4.3
Logistic Regression	–	–	40.0	40.4	51.0	51.0
DCR	–	–	62.5	62.5	71.2	71.0
Match-LSTM with Ans-Ptr (Sequence)	150	882 K	54.4	–	68.2	–
Match-LSTM with Ans-Ptr (Boundary)	150	882 K	61.1	–	71.2	–
Match-LSTM with Ans-Ptr (Boundary+Search)	150	882 K	63.0	–	72.7	–
Match-LSTM with Ans-Ptr (Boundary+Search)	300	3.2M	63.1	–	72.7	–
Match-LSTM with Ans-Ptr (Boundary+Search+b)	150	1.1M	63.4	–	73.0	–
Match-LSTM with Bi-Ans-Ptr (Boundary+Search+b)	150	1.4M	**64.1**	**64.7**	**73.9**	**73.7**
Match-LSTM with Ans-Ptr (Boundary+Search+en)	150	882 K	**67.6**	**67.9**	**76.8**	**77.0**

Table 2.3: Experiment results from (Wang and Jiang, 2016) that highlight various configurations implemented. The best being Match-LSTM with Ans-Ptr using a boundary, search and ensemble technique.

has shown that this approach hasn't yet been applied to the SQuAD 2.0 dataset. In chapter 3, section 3.3 we will hypothesize the implementation of this approach along with the use of Transformers, which are covered in the next section.

The final piece of research to look at in this section is by (Gennaro et al., 2020) titled *Intent Classification in Question-Answering Using LSTM Architectures*. The approach followed is similar to (Brahma, 2018), using GloVe embeddings, allowing for prefixes to be added and generating word closeness and using a supervised learning technique. In fig. 2.8 it can be observed how applying one-hot encoding, followed by GloVe embedding before reaching the LSTM layer and using a Softmax activation layer at one prediction

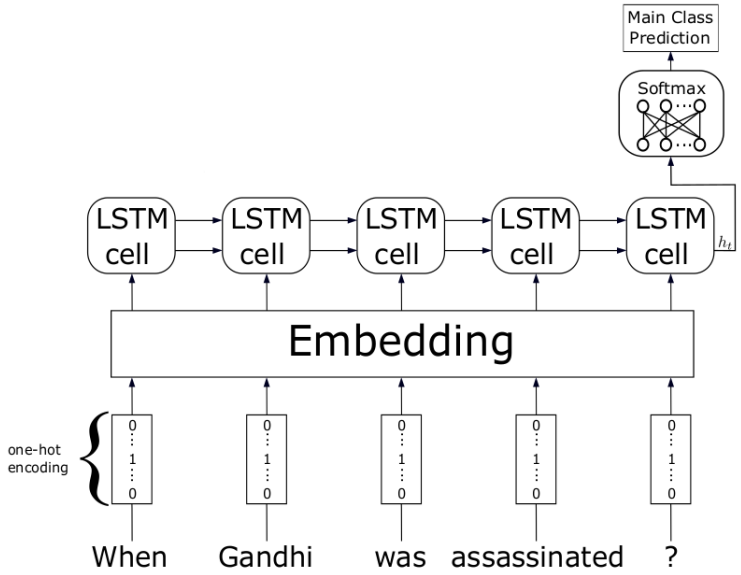


Figure 2.8: Basic classification model using LSTM (Gennaro et al., 2020)

classes allows the model to generate better predictions on the TREC (Dietz et al., 2017) dataset.

In fig. 2.9 we see the implementation of a padding layer at the end of each question and adding a subclass prediction strategy at the end of each prediction. This is done to gain better contextual understanding of the question. The subclass acts as a specialization class while the main class acts as a generalization class, therefore narrowing the context while maintaining global outlook. To verify their results (Gennaro et al., 2020) implemented

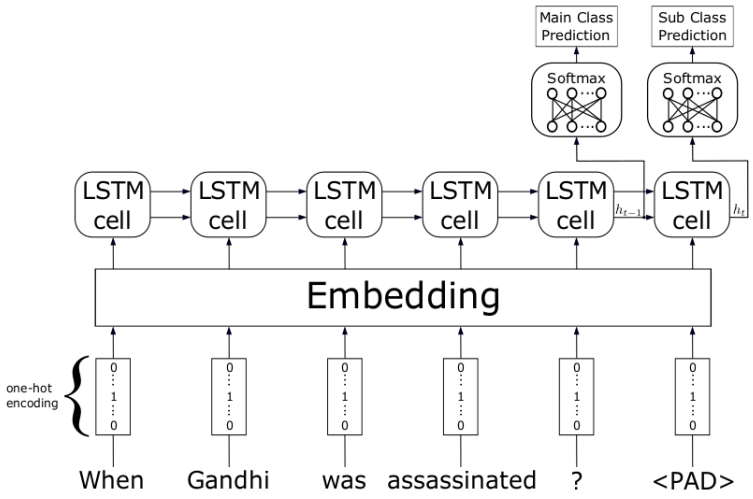


Figure 2.9: Basic classification model using RNNs (Gennaro et al., 2020)

a prototype responder by adding a biLSTM on top of the LSTM layer seen in figs. 2.8 and 2.9. The biLSTM model uses the predictions from the previous layer, both main and subclass predictions, and combines them to form the final output.

2.1.3 Critical Issues in LSTMs

From the literature reviewed so far, it has become apparent that LSTMs generate a view of the world in a very sequential manner. While successful in managing banishing gradients and generating better, contextually relevant results, LSTMs or even biLSTMs aren't entirely correct, with none of the models achieving over 90% accuracy across various datasets consistently. LSTMs suffer from another fatal flaw, their sequential nature makes it extremely difficult to parallelize the work and this leads to extended training times, out of memory errors etc.

2.2 Transformers

The previous section highlighted some key advancements in the field of machine comprehension and LSTMs, however, not a lot changed for almost two decades in the field of NLP based machine comprehension tasks architecturally. Since their introduction in 1997, LSTMs (Schmidhuber and Hochreiter, 1997) have been the base architecture for countless applications. However, as we noted in section 2.1.3, LSTMs have some critical drawbacks.

Transformers were introduced in late 2017 by (Vaswani et al., 2017) and have become a very reliable way to implement various NLP based tasks. The basic architecture for which can be seen in fig. 2.10. Recurrent models usually adopt a combination approach where

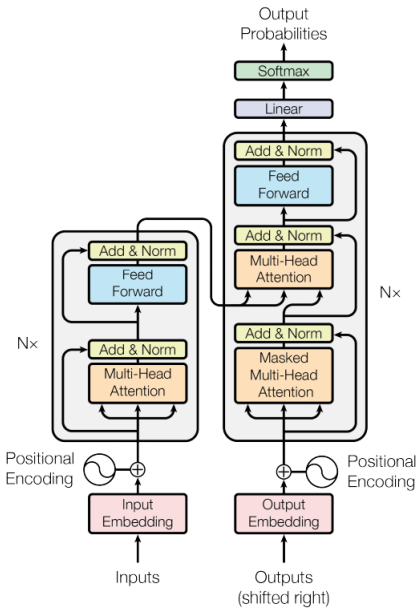


Figure 2.10: Transformer Architecture built by (Vaswani et al., 2017)

they take symbol position of the input and output positions along with the computation. This helps them align positions to steps in computation time and generate a sequence of hidden states h_t, which is a function of the previous hidden state h_{t-1} and the input function t. This inherently sequential nature of RNNs causes memory issues, leading to reduced batch sizes.

The architecture for a *Transformer* in this paper is outlined as having an encoder that maps input sequences to a continuous representation. The architecture can be seen in Fig. 2.10. This is then decoded into an output sequence of symbols one at a time. Each step

is auto-regressive, ie. it consumes the previously generated symbols as additional input when creating the next. This is similar to an ensemble model. Stacks of 6 encoder and 6 decoder layers is used.
Each encoder layer has 2 sub-layers of a multi-head self-attention and the other a simple, position-wise fully connected feed-forward network layer. The output from each encoder layer can be represented as $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layers in the model. Additionally, supporting residual connections is handled by producing outputs of dimension $d_{model} = 512$.

The decoder layer is similar to the encoder layer and has an additional 3rd sub-layer that performs multi-head attention over the output of the encoders. To ensure that the output layer isn't affected by the output of residual connections from the sub-layers, a normalization layer is implemented at the end. Masking the subsequent positions and offsetting the output by 1 position ensures that the predictions for position i can only depend on outputs from previous positions.

The *Transformer* architecture was the first one to implement a self-attention based transduction model to compute input and output representations without the use of convolution or sequence-aligned networks.

One of the key features in a Transformer architectures is *self-attention*, which is covered in-depth in section 2.2.1.

2.2.1 Self Attention

In their paper "Attention Is All You Need", (Vaswani et al., 2017) highlight that *self-attention* is a key piece of the puzzle. This is due to several reasons such as the ability to parallelize computation, reduce total complexity per layer and the reduced path lengths between various long-range dependencies in the network. It is easier to learn about these dependencies if the distance between them is shorter. Thus, the transformer also computes the length of the longest path between any two input-output positions.

The attention mechanism was designed by (Bahdanau et al., 2014) to solve a critical issue with encoder-decoder architecture dependent LSTM and RNN based models. As has been highlighted before, LSTMs suffer from an inability to retain long-range sequence dependencies. Using a (soft)-search approach allows the model to look for parts that are relevant to predicting an output sequence, rather than forming all the segments of a sentence produces better results than existing fixed-length encoder-decoder models. The (soft)-search is performed after a prediction is made. This prediction is based on a set of positions for the source sentence, where the target word has the highest concentration of context vectors associated with previously generated target words. Another distinguishing feature of the attention mechanism is that it does not try to encode an entire input sentence to a fixed-length vector. Rather, it encodes the input sentence into sequence vectors that can be adaptively decoded during translation. Thus freeing the model from having to squash a lot of information from the source sentence into a vector of fixed-length.

The work done in (Vaswani et al., 2017) modified that attention mechanism by mapping a query to a set of key-value pairs and developing *self-attention*. To understand self attention take an example sentence like:

"The lion was resting in the shade because it was tired after hunting."

While this is a perfectly normal sentence to a human being, a computer does not know

what "it" means and that "it" refers to the "lion" in the second half of this sentence. Self-attention allows a Transformer to understand that the word "it" refers to the "lion" in this context. In fig. 2.11 shows how (Vaswani et al., 2017) modified the attention mechanism.

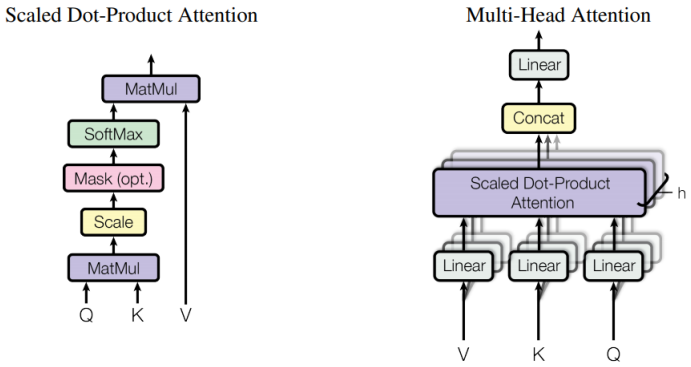


Figure 2.11: Scale Dot and Multi-Head Attention Models (Vaswani et al., 2017)

The "Scaled Dot-Product Attention" layer implemented in the transformer works by computing the dot products of all the input query with all the keys, divide it by the dimension of the keys and finally apply a softmax function to obtain the weights. This can be represented as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.14}$$

where K, V are the matrices containing the keys and values respectively, and Q is the set of queries. The dimension of the keys is given by d_k. The dot product is used because it is more efficient than the additive attention product and is faster as well (Vaswani et al., 2017).

Instead of using single attention, the transformer implements *multi-head* attention. Multi-head attention, which can be seen in fig. 2.11 on the right, allows for the model to simultaneously gather information from various different representations in different subspaces and positions.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= Concat(head_1, ..., head_h)W^O \\ \text{where head}_\text{i} = \ &\text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.15}$$

Matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{mod 1}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ represent the projection parameters. They also implemented $h = 8$ parallel attention/head layers. The reduced dimensionality of each head with $d_k = d_v = d_{\text{model}}\ /h = 64$ means that the total cost is similar to a fully connected, single-head attention layer. The evaluations performed on the Wall Street Journal dataset(Marcus et al., 1993), using 40k sentences, showed that even without task-specific tuning the model had better results with a fraction of the training cost.

The advantages of the Transformer architecture are clearly evident over the standard RNN based LSTMs and other architectures. Experiments on the WMT 2014 English to

German translation (Bojar et al., 2014) using the big transformer showed a significant jump in BLEU score 28.4 over various other seq2seq models and even combined ensemble models which earlier had a highest score of 26.36 using a ConvS2S Ensemble (Gehring et al., 2017).

2.2.2 Improvements of Transformer Architectures

Transformers have significant advantages as highlighted earlier. Improvements to the basic architecture are discussed below.

BERT

The paper on *Bidirectional Encoder Representations from Transformers* (BERT) (Devlin et al., 2018), introduces a new language model. This model is truly fascinating in many ways. First and foremost it is designed to pre-train deep bidirectional representations using unlabelled data. This is done by jointly conditioning context in all layers to the right and left. This pre-training allows the model to be fine-tuned simply using one additional output layer. These features make this model conceptually simple and very powerful empirically.

BERT employs language pre-training (Dai and Le, 2015, Peters et al., 2018, Radford et al., 2018) which has shown significant advantages in many applications e.g. paraphrasing, language level inference etc. These tasks aim to highlight the relationships between sentences through contextual understanding as well as by using tokenized outputs. BERT improves on the fine-tuning tasks that are important to performing

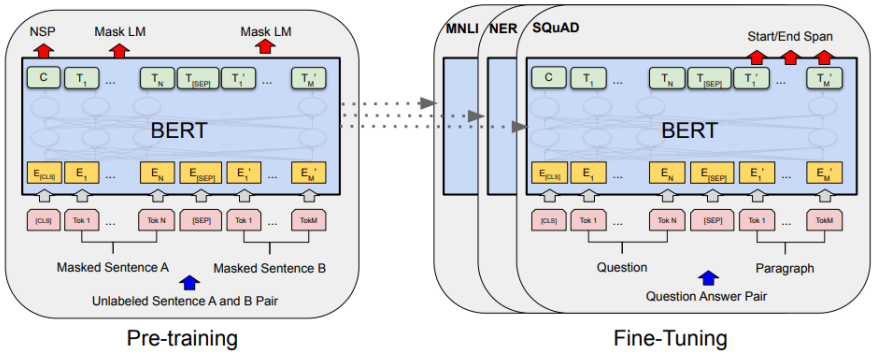


Figure 2.12: Pre-training and Fine Tuning procedures for BERT (Devlin et al., 2018)

BERT was tested on The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) which has a large number of diverse NLU tasks. BERT performed extremely well on the 11 NLP tasks that the authors ran it against. It showed an average accuracy improvement of 4.5% and 7% when compared to the previous state of the art models. Results of BERT and its significant gains make it one of the best candidate models for NLU tasks.

ALBERT

(Lan et al., 2019)

ROBERTA

(Liu et al., 2019)

DISTILBERT

(Sanh et al., 2019)

2.3 Summary

Chapter 3

Research Methodology

In this chapter

3.1 Data Selection

We have selected the Stanford Question Answering Dataset (SQuAD). This is as a reading comprehension dataset based on Wikipedia articles. It is based on questions posed by crowd-workers on a set of articles. The answer to every question is a segment of text or span, from the corresponding reading passage, or the question might be unanswerable (Rajpurkar et al., 2018).

The dataset consists of over 150,000 questions. Split into 100,000 answerable and 50,000+ unanswerable question, which were written to look similar to unanswerable questions. The challenge being that a model should be able to correctly answer the answerable questions and abstain from answering the unanswerable ones. The dataset is freely available as a part of the Transformers package in python or it can be downloaded from the SQuAD 2.0 website (Rajpurkar, 2021).

To effectively use this dataset for our purposes, let us first take a look at what its contents look like below.

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."*
Question: *Who was the Norse leader?*
Answer: *Rollo*

The answer to the aforementioned question is quite simple for humans to comprehend. The challenge is for computational systems is to contextualize this and make it machine-understandable so that the produced model can answer it correctly. The dataset consists of various kinds of English language examples like negation, antonyms, entity swaps, impossible conditions to answer, answerable, etc. making the dataset a well-balanced one. It is important to prepare this dataset appropriately

1. Data splitting into separate Question, Answer and Context lists.

2. Splitting the data into separate training and validation sets of question and answers using the 80/20 rule, also known as the Pareto principle. We will have 80% training data and 20% test data.

3. Tokenization of the split data to generate "context-question" pairs

4. Generating indexes for when an answer begins and ends in the dataset

5. Adding answer tokens based on their encoded positions

The SQuAD 2.0 Dataset (Rajpurkar et al., 2018), was developed with funding from Facebook to help address some major issues with existing datasets. Most datasets focus on questions that can be easily answered or use of automatically generated, unanswerable questions which are easily identifiable.
The SQuAD 2.0 dataset resolves this by combining the SQuAD dataset along with 50,000 crowd worker generated unanswerable questions. The key feature of these being that the unanswerable questions must look similar to answerable ones. For a model to be successful on this new dataset, it must be able to answer all possibly answerable questions as well as determine when no answers are provided for a question in the given paragraph and abstain from answering. A comparative study was done for a Natural Language Understanding(NLU) task that obtained an 86% score on SQuAD 1.1, only got 66% on the new 2.0 dataset. The dataset helps bridge the gap between true NLU and machine understanding by using the concept of Relevance. Through comparisons with various datasets such as RACE, MCTest, QASENT etc. they have identified the missing links like negative examples, antonyms and helped fill the gap. This dataset forces the models to understand whether a paragraph span has the answer to the question posed.

	SQuAD 1.1	SQuAD 2.0
Train		
Total Examples	87,599	130,319
Negative Examples	0	43,498
Total articles	442	442
Articles with negatives	0	285
Development		
Total Examples	10,570	11,873
Negative Examples	0	5,945
Total articles	48	35
Articles with negatives	0	35
Test		
Total Examples	9,533	8,862
Negative Examples	0	4,332
Total articles	46	28
Articles with negatives	0	28

Table 3.1: Comparison of SQuAD 2.0 to SQuAD 1.1(Rajpurkar et al., 2018).

Table 3.1 shows the difference between

3.2 Data Pre-processing And Transformation

Techniques used. (1) Average pooling; (2) max pooling; (3) the concatenation of the last vectors on both directions. Dropout operation is performed on the QA representations before cosine similarity matching. Use GloVe embeddings and suffix-prefix encoding in both directions of self attention pass to show improvements.

3.3 Research Hypothesis

The hypothesis of this research is that implementing a self attention based Transformer model with:

- glove embedding
- BERT base(Devlin et al., 2018)
- Answer pointer output layer(Wang and Jiang, 2016, Vinyals et al., 2015)

3.4 Model Evaluation Metrics

To successfully evaluate the model two metrics will be used. The *F1 - Score* and *Exact Match Score* will be used to measure the success of the model developed. While accuracy is a good score to measure how well a model is performing, the problem with accuracy is that it is prone to overfitting, and underfitting, which might lead to great accuracy scores but real world predictions wont be as accurate.

Accuracy also doesn't work because it is a simplistic probability metric that assumes both false positives and false negatives have an equal weight in predicting the outcome as shown in eq. 3.1.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + TrueNegative + FalseNegative} \tag{3.1}$$

Outlined below are some reasons why accuracy is not a good measure:

1. **Class Imbalance**
2. **Prone to Over and Under-fitting**

Before looking at what F1 and Exact Match scores are, one must be able to understand the terminology used there in.

Precision is the probability of identifying a correct outcome from all the possible outcomes.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3.2}$$

Recall

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3.3}$$

F1 Score

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.4}$$

Exact Match

3.5 Research Flow Diagram

To be able to achieve the objectives set out in the previous chapters, it is important to organize the research in such a way that it achieves its objective. This is represented in the research flow diagram 3.1. The steps identified in 3.1 encapsulate the broad format of this thesis and its various stages.

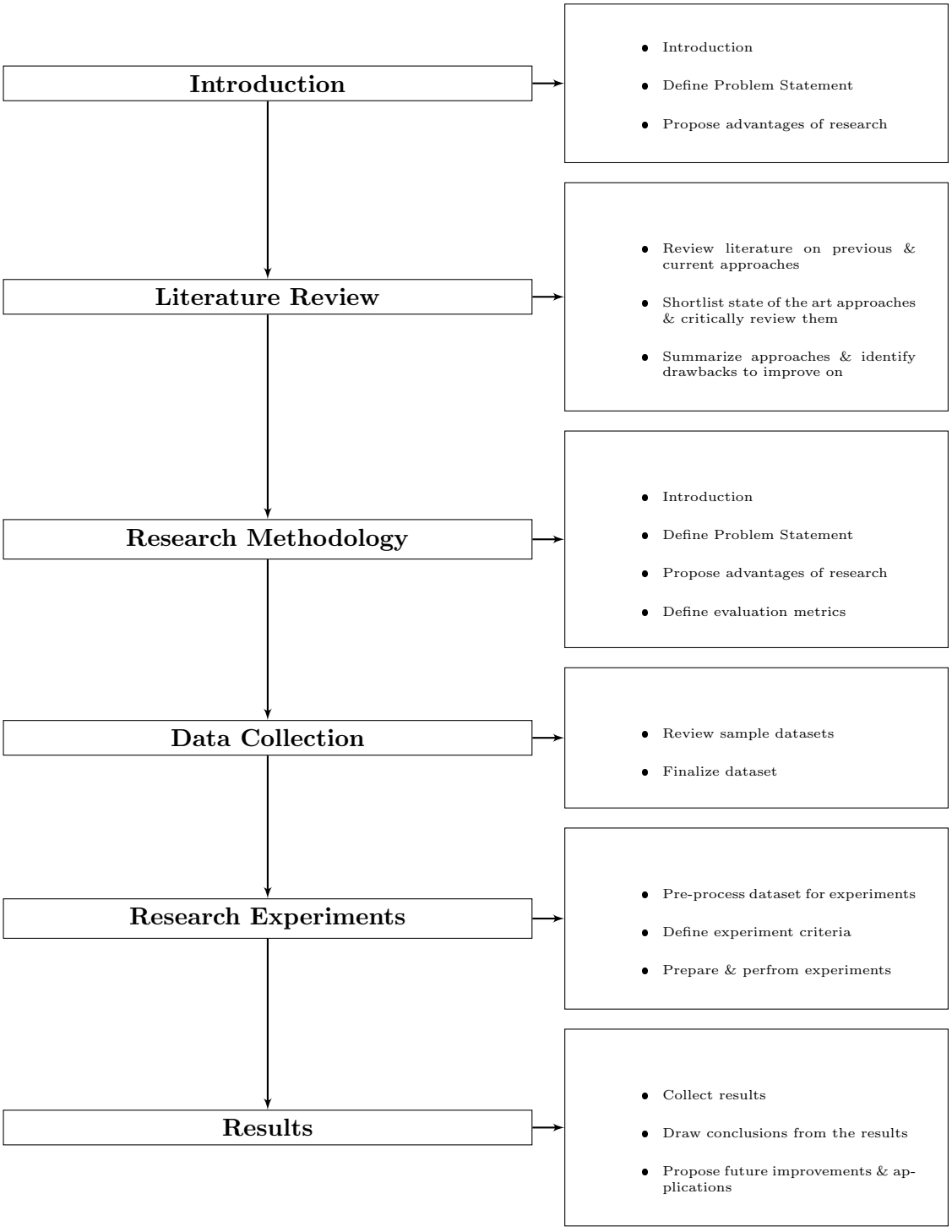


Figure 3.1: Research Flow Diagram

Chapter 4
Architecture Creation

In this cahpter we yaba daba do

4.1 Drawbacks Of Current Architectures

To aid in understanding the drawbacks of current architectures the author presents the table below

4.2 Proposed Architecture Improvements

4.3 Architecture Refinement

Chapter 5

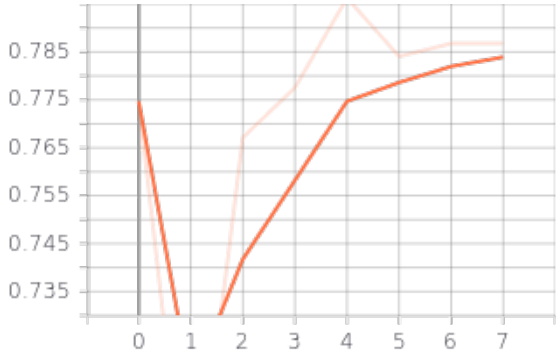
Results And Discussion

5.1 Existing Models And Benchmarks

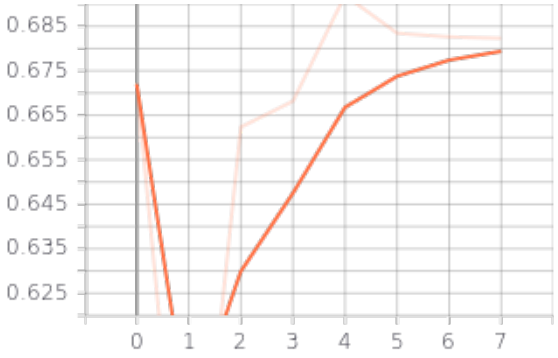
To be able to establish the improvements hypothesized in the previous section it is important to have a baseline metric. The ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019) and DistilBERT (Sanh et al., 2019) models are available pre-trained through the HuggingFace library (Team). The library also provides the SQuAD 2.0 Dataset (Rajpurkar et al., 2018) for use.

Roberta Explanation

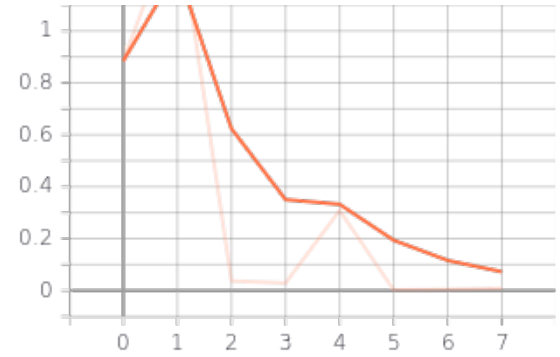
DistilBert Explaination



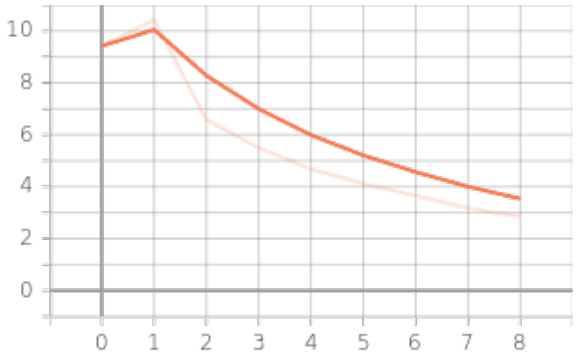
(a) AlBert Validation F1 Score



(b) AlBert Validation Exact Match

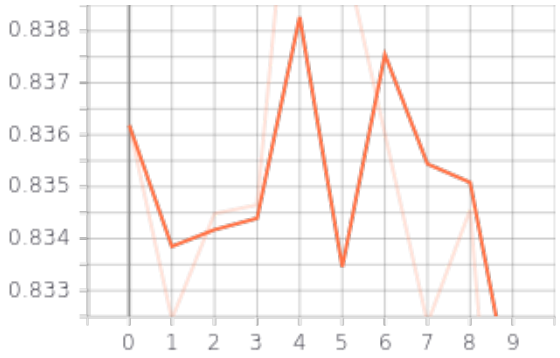


(c) AlBert Validation Loss

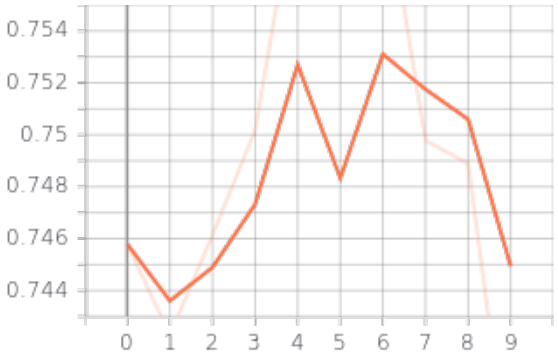


(d) AlBert Training Loss

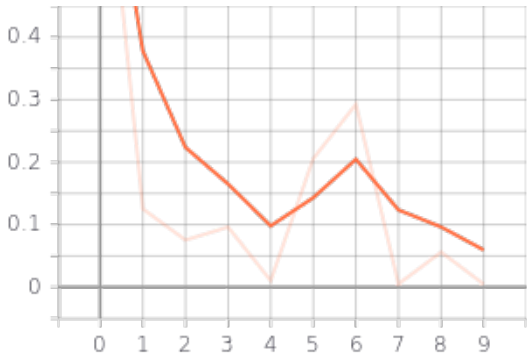
Figure 5.1: AlBert Model Benchmarks



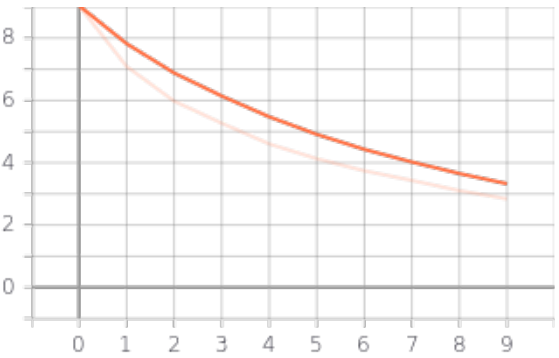
(a) RoBerta Validation F1 Score



(b) RoBerta Validation Exact Match

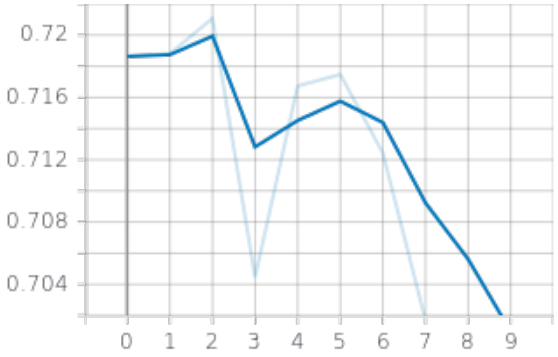


(c) RoBerta Validation Loss

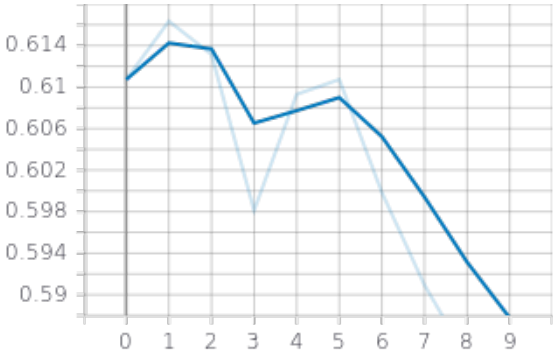


(d) RoBerta Training Loss

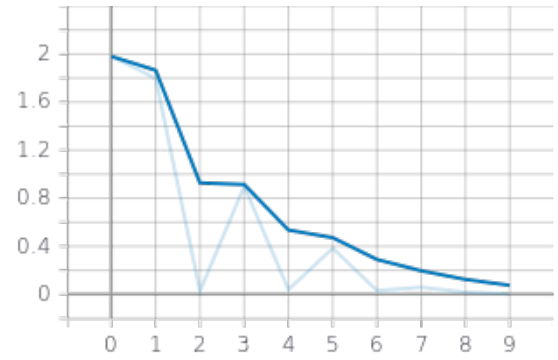
Figure 5.2: RoBerta Model Benchmarks



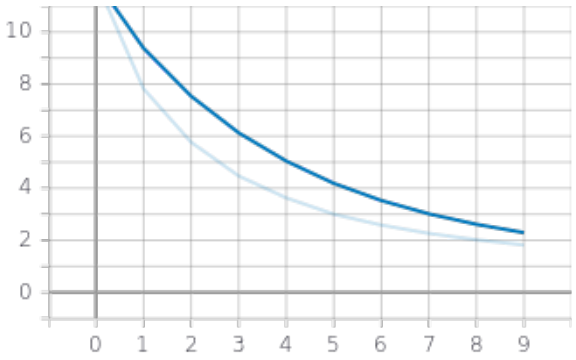
(a) Distilbert Validation F1 Score



(b) Distilbert Validation Exact Match



(c) Distilbert Validation Loss



(d) Distilbert Training Loss

Figure 5.3: Distilbert Model Benchmarks

Chapter 6
Conclusions And Recommendations

Through this work the author has attempted to highlight the advancements and propose a unique modification to the existing transformer architecture using an Answer-Pointer network.

6.1 Recommendations

This work provides a unique opportunity to introduce improvements to an already state of the art architecture by adding a layer of LSTM based pointers. Highlighted below are some recommendations, that the author believes, can improve the architecture further.

- **Increased Batch-Size**: The experiments presented in this work are limited by a relatively small batch size of 8. Systemic increase of batch size can allow for reduced training times, possibly better performance overall and an opportunity to test this model against various other NLP applications such as Sequence Classification, Token Classification, Sentence Predictions, etc.

- **Better Hardware**: One of the key limiting factors for the work presented in this thesis is the limiting amount of VRAM, The current hardware used by the author only had 8GB of VRAM available on the EVGA 2070 Super GPU, which was insufficient for the benchmark batch size of 8. Reducing the batch size any further led to increased training times per epoch, upwards of 3 hours per epoch at a batch size of 6. This problem was slightly remediated by the use of Google Colab Pro, which is a subscriptions service that allows you to run experiments using better GPUs such as Tesla P100 GPU with 16GB VRAM. This service is time limited and thus not suitable for running long experiments. To effectively quantify the proposed architecture in this thesis it is recommended that it be run for at least 10 epochs on a multi-gpu setup that allows for mini-batches. This approach is discussed below.

- **Mini-Batches Across Multiple GPUs**: Creating a multi-gpu setup, with 2 or more gpus, allows for accelerated training (Pal et al., 2019). A simple guide on how to set this up for PyTorch has been created on the Towards Data Science Medium page (Giacaglia, 2020) and even in the

Appendices

Chapter A
Gantt Chart

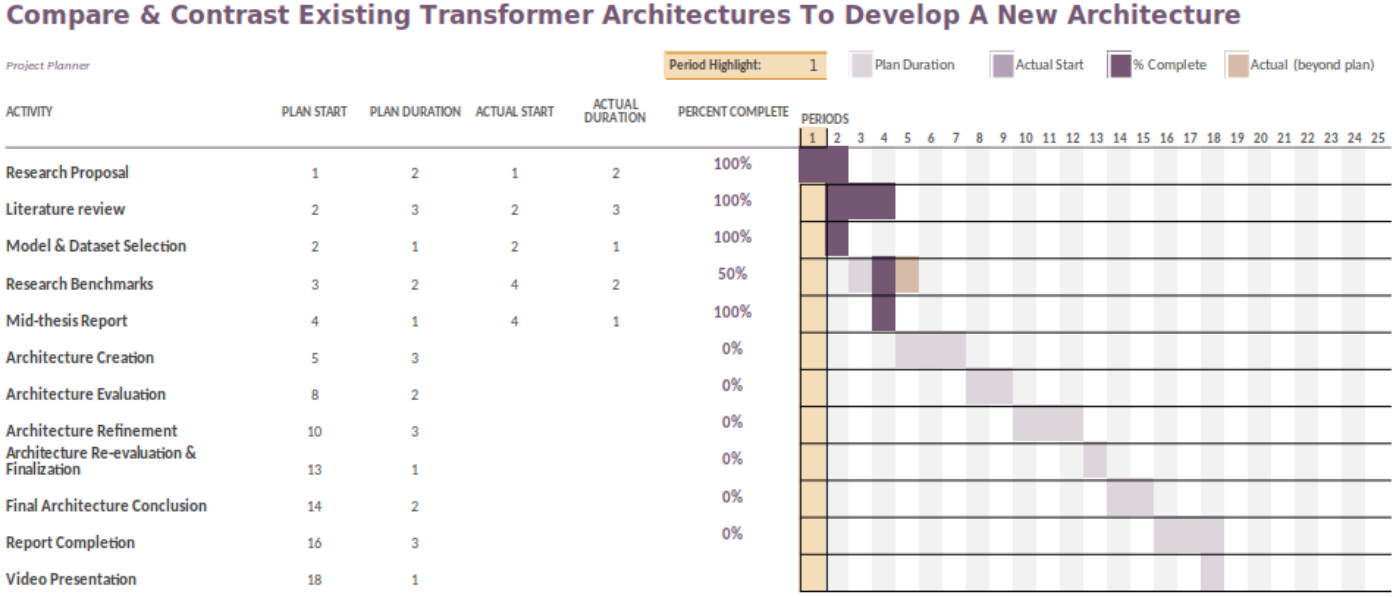


Figure A.1: Mid-Thesis report Gantt Chart

Chapter B
Research Proposal

Research Proposal

Nirbhay P. Tandon
MSc. Data Science
Research Title:
Develop New Transformer Architecture For
Question and Answering(QandA)

Abstract

Attention-based Transformer architectures have become the norm of current Natural Language Processing applications. Google began this trend back in 2017 with their paper *Attention Is All You Need*, by introducing the Transformer architecture that works solely on attention mechanisms. The purpose of our work will be to create a new kind of Transformer architecture. Compare and contrast its performance against other architectures such as BERT, DistilBERT, ALBERT etc. using the SQuAD 2.0 Dataset. Through our research, we aim to produce a new architecture that has a better performance than existing models specifically for Question Answering based applications.

Contents

List of Figures

1 Introduction

Question-Answering based systems have gained a lot of popularity, especially in the form of "chatbots". These systems depend highly on contextual understanding of the input, the training corpus and the question asked. They use this knowledge to output an answer that can help the user with whatever their query is. Recurrent neural networks and architectures based on them, have been able to provide great advancements in the field of Question-answering and chatbots in general. However, there is a behaviour of over-fitting and lack of contextual understanding of the question. This, coupled with long training times and extremely complex mathematical model designs, have often kept the field of Natural Language Processing slightly obscured from the masses.

We wish to change that. Through our work, we would like to aim at creating a sustainable, fast, easy to understand Transformer architecture for Question Answering. An advancement on the work done by the team at Google (Vaswani et al., 2017). To be able to do so, let us first understand what a *Transformer* is. A Transformer is a form of transduction model that relies solely on self-attention to figure out how to represent its inputs and outputs. It does so without the use of any sequence aligned recurrent neural networks(RNNs) or convolutions.

Through our research proposal we wish to highlight why we are going to be performing this research and putting in the effort to devise a new architecture. We have divided our research proposal into 7 sections. In Section 1, we shall take a look at briefly introducing the concept and why our work is necessary. Next, in Section 2, we outline the background work that has already been done in this field and how some of the papers relate to the work that has been done. We use this as an opportunity to highlight some of the shortcomings in current architectures and modelling techniques. In Section 3, we briefly outline the aim of our proposed research. In Section 4, we define in some detail the work that we will do to establish our research and how we plan to quantify the work that shall be done. Section 5, highlights our goal, which is to produce a new transformer architecture that performs better at Question Answering based tasks. In Section 6, we have outlined the minimum hardware requirements along with the resources available to the author that will be used to conduct this research. Finally, in Section 7, we submit a Gantt Chart to outline our plan against the number of weeks.

2 Background and Related Research

In this section, we shall highlight what has led us this far and some of the interesting challenges that it poses. In 2.1, we briefly look at the history of Natural Language Processing and how some of the challenges were addressed. In 2.2, we look at the latest research that has gone into creating the Transformer architecture, identify some of the common patterns and use that information to strategise our model in later sections.

2.1 Background

The area of Natural language Processing has taken significant leaps in the last two decades. Work done towards improving the ability of machine learning models to first recognize words, then sentences, followed by contextual understanding has led to several interesting and novel approaches in the field. From early on neural networks to creating Long Short-Term Memory architectures (Schmidhuber and Hochreiter, 1997) by Sepp Hochreiter and Jurgen Schmidhuber in the mid-'90s that resolved the vanishing gradient problem of classical neural networks, we have come a long way.

The latest advancements in this field come from Google's research lab in the form of *Transformers*. We look into this in a bit more detail later. However, no model can be successful without a good dataset to train on. This is where the SQuAD 2.0 dataset (Rajpurkar et al., 2018) comes in. This dataset is what forces the machine learning models to do contextual understanding. One might even say that it forces the models to "think" for themselves before answering a task.

Let us now look at some of the key research that has been done in this regard.

2.2 Related Research

Outlined below are some of the most important pieces of research that relate directly to the work done for Transformers and Q&A based systems.

1. The SQuAD 2.0 Dataset (Rajpurkar et al., 2018), was developed with funding from Facebook to help address some major issues with existing datasets. Most datasets focus on questions that can be easily answered or use of automatically generated, unanswerable questions which are easily identifiable.
   The SQuAD 2.0 dataset resolves this by combining the SQuAD dataset along with 50,000 crowd worker generated unanswerable questions.

The key feature of these being that the unanswerable questions must look similar to answerable ones. For a model to be successful on this new dataset, it must be able to answer all possibly answerable questions as well as determine when no answers are provided for a question in the given paragraph and abstain from answering. A comparative study was done for a Natural Language Understanding(NLU) task that obtained an 86% score on SQuAD 1.1, only got 66% on the new 2.0 dataset. The dataset helps bridge the gap between true NLU and machine understanding by using the concept of Relevance. Through comparisons with various datasets such as RACE, MCTest, QASENT etc. they have identified the missing links like negative examples, antonyms and helped fill the gap. This dataset forces the models to understand whether a paragraph span has the answer to the question posed.

2. Work done in the field of Long short-term and gated recurrent (Hochreiter et al., 2001) and (Zhou et al., 2016) neural networks, in particular, has been established as a state of the art approach in sequence modelling, transduction problems such as language modelling and machine translation. In their paper Attention Is All You Need,(Vaswani et al., 2017) the team set out to resolve problems in the parallelization and increased compute times of recurrent models. The inherently sequential nature of RNNs causes issues in memory constraints, leading to reduced batch sizes.

   The architecture for a *Transformer* in this paper is outlined as having

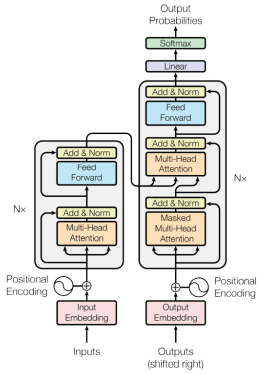


Figure 1: Transformer Architecture built by (Vaswani et al., 2017)

an encoder that maps input sequences to a continuous representation. The architecture can be seen above in Figure 1. This is then decoded into an output sequence of symbols one at a time. Each step is auto-regressive, i.e. it consumes the previously generated symbols as additional input when creating the next. This is similar to an ensemble model. Stacks of 6 encoder layers and 6 decoder layers is used.
The encoder layers each have 2 sub-layers of a multi-head self-attention and the other a simple, position-wise fully connected feed-forward network layer.
The decoder layer is similar to the encoder layer and has an additional 3rd sub-layer that performs multi-head attention over the output of the encoders. There is also normalization and the outputs are prevented from attending to subsequent positions.
The attention mechanism can be described as mapping a query to a set of key-value pairs. This can be seen from Figure 2, below.
The evaluations performed on the Wall Street Journal dataset(Marcus et al., 1993), using 40k sentences, showed that even without task-specific tuning the model had better results with a fraction of the training cost.

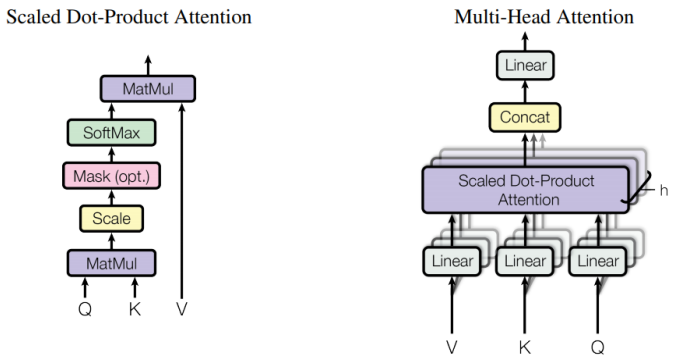


Figure 2: Scale Dot and Multihead Attention
Models Vaswani et al. (2017)

3. The paper on BERT, which is *Bidirectional Encoder Representations from Transformers* (Devlin et al., 2018), introduces a new language model. This model is truly fascinating in many ways. First and fore-

most it is designed to pre-train deep bidirectional representations using unlabelled data. This is done by jointly conditioning context in all layers to the right and left. This pre-training allows the model to be fine-tuned simply using one additional output layer. These features make this model conceptually simple and very powerful empirically.

BERT employs language pre-training (Dai and Le, 2015) which has shown significant advantages in many applications e.g. paraphrasing, language level inference etc. These tasks aim to highlight the relationships between sentences through contextual understanding as well as by using tokenized outputs.

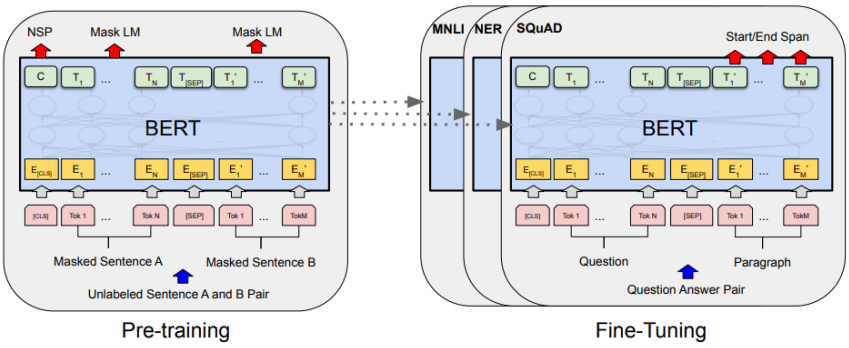


Figure 3: Pre-training and Fine Tuning procedures for BERT (Devlin et al., 2018)

BERT was tested on The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) which has a large number of diverse NLU tasks. BERT performed extremely well on the 11 NLP tasks that the authors ran it against. it showed an average accuracy improvement of 4.5% and 7% when compared to the previous state of the art models. Results of BERT and its significant gains make it one of the best candidate models for NLU tasks.

Several advancements have been made to the BERT model to make it fast, better at understanding and even performing application-specific tasks such as Q&A.

The SQuAD 2.0 dataset has inspired an LSTM based FastQA (Weissenborn et al., 2017) model architecture. This architecture takes cues from the work done by Hochreiter and Schmidhuber in their work on Long Short-Term Memory Architecture (Hochreiter et al., 2001), to create a model specifically for end-to-end question answering systems.

A common theme with BERT is that it takes a long time to train. Especially in the BERT based RoBERTa architecture Liu et al. (2019). RoBERTa takes on average 4-5 times more time to train than BERT, however, it also shows a maximum of 20% improvement over BERT, depending on the application. ALBERT, which is A Liter BERT, Lan et al. (2019) was developed specifically to deal with memory limitations and reduce training times. ALBERT's XXL implementation has been documented to perform better in models trained on the BOOKCORPUS and Wikipedia ones by at least 2% across multiple applications. In a smaller amount of time.

However, none of these model architectures has been able to address all the problems and serve as more generic solutions to multiple end-to-end sequence encoding problems across various applications.

Our work is primarily focused on building a fast model that allows for higher accuracy in responses specifically with Q&A systems.

3 Aims and Objectives

The main aim of this research is to propose a new transformer architecture that can perform better at Q&A using the SQuAD 2.0 dataset. We shall:

1. Implement the existing models that are available via libraries such as HuggingFace (Team), PyTorch and Tensorflow on the dataset
2. Obtain F1, validation, etc. scores for existing models and treat them as our benchmark scores
3. Identify drawbacks of the current architectures
4. Design our architecture and evaluate its performance
5. Fine-tune the architecture, re-evaluate and report improvements
6. Compare the results of our Transformer model with the benchmark scores.

4 Research Methodology

To implement this research we shall break the project down into 5 phases. These are outlined below.

4.1 Research Dataset

We have selected the Stanford Question Answering Dataset (SQuAD). This is as a reading comprehension dataset based on Wikipedia articles. It is based on questions posed by crowd-workers on a set of articles. The answer to every question is a segment of text or span, from the corresponding reading passage, or the question might be unanswerable (Rajpurkar et al., 2018).

The dataset consists of over 150,000 questions. Split into 100,000 answerable and 50,000+ unanswerable question, which were written to look similar to unanswerable questions. The challenge being that a model should be able to correctly answer the answerable questions and abstain from answering the unanswerable ones. The dataset is freely available as a part of the Transformers package in python or it can be downloaded from the SQuAD 2.0 website (Rajpurkar).

To effectively use this dataset for our purposes, let us first take a look at what its contents look like below.

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."*
Question: *Who was the Norse leader?*
Answer: *Rollo*

The answer to the aforementioned question is quite simple for humans to comprehend. The challenge is for us to contextualize this and make it machine-understandable so that our model can answer it correctly.

The dataset consists of various kinds of English language examples like negation, antonyms, entity swaps, impossible conditions to answer, answerable, etc. making the dataset a well-balanced one.

To use this dataset correctly we shall perform the following pre-processing steps on it:

1. Data splitting into separate Question, Answer and Context lists.
2. Splitting the data into separate training and validation sets of question and answers using the 80/20 rule, also known as the Pareto principle. We will have 80% training data and 20% test data.
3. Tokenization of the split data to generate "context-question" pairs
4. Generating indexes for when an answer begins and ends in the dataset
5. Adding answer tokens based on their encoded positions

4.2 Research Benchmarks

Here we shall focus on obtaining benchmark scores for the shortlisted architectures i.e BERT (Devlin et al., 2018), DistilBERT (Sanh et al., 2019) and ALBERT Lan et al. (2019), on the above dataset.

We shall use the F1, Exact Match(EM), Recall and Training Time scores to create a benchmark to compare our architecture against. The Exact Match score will help us identify how many questions were 100% correctly answered by each model.

4.3 Architecture Creation

In this section we will:

1. Mathematically model a new transformer architecture
2. Code the architecture
3. Run sample dataset to identify base benchmarks
4. Run the SQuAD 2.0 dataset to obtain 1st pass performance benchmarks
5. Document architecture performance, identify pros and cons

4.4 Architecture Refinement

In this phase, we will focus on:

1. Reviewing the results from the previous section
2. Identifying the areas of improvement

3. Hypothesise the improvements and implement them in the architecture
4. Run the SQuAD 2.0 dataset to obtain new performance benchmarks
5. Document architecture performance, identify pros and cons

4.5 Model Evaluation

The training shall be carried out using the train-test loss plot to identify the optimal number of epochs for which our model needs to be run. This will also be done for the selected model architectures.

The main parameters we will use for model evaluation are F1, Exact Match(EM), Recall and Training Time.

These metrics will help us reiterate and quantify correctly if our model has improved performance or not.

5 Expected Outcomes

We expect that our created model is at-par, if not better, at performing Q&A than existing models.

6 Requirements and Resources

To successfully deliver on our research we will be utilizing the following hardware:

- EVGA GeForce RTX 2070 SUPER KO GAMING, 08G-P4-2072-KR, 8GB GDDR6, Dual Fans(Evga). This graphics card is based on the Nvidia "Turing" architecture and has 2560 CuDA cores.
- Intel 10700 processor. 8 cores, 16 threads, 16M cache(Intel).
- VENGEANCE® LPX 8GB (1 x 8GB) DDR4 DRAM 2400MHz C14 Memory Kit - Black(Corsair). 8GB x 4, 32 GB total.
- Ubuntu 20.04 Operating System
- We will also be using the latest versions of the following packages: Pandas, NumPy, SciPy, Transformers by HuggingFace, Matplotlib, Tensorflow and PyTorch. In case there are compatibility issues the appropriate versions will be mentioned. We will also mention any other packages that might be required in the course of the research.

The above hardware is available to the author and any changes to the same will be notified/highlighted in the subsequent reports.

7 Research Plan

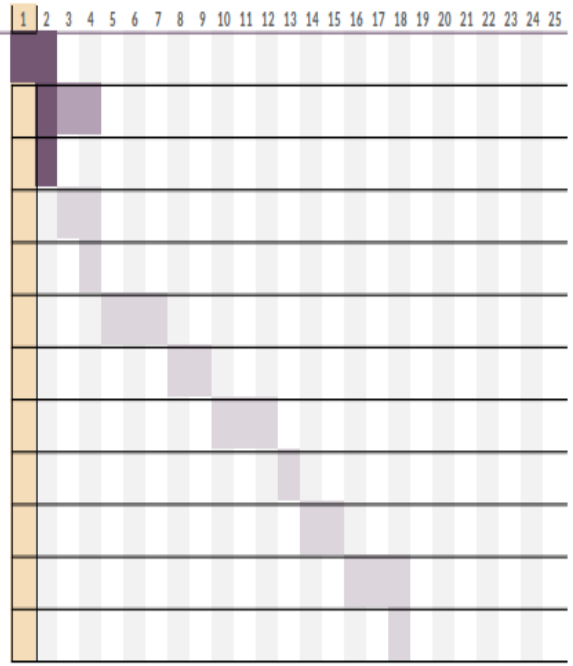
Shown on the next page is the Gantt Chart highlighting the research stages and timelines.

Compare & Contrast Existing Transformer Architectures To Develop A New Architecture

Project Planner

Period Highlight: 1 | Plan Duration | Actual Start | % Complete | Actual (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Research Proposal	1	2	1	2	100%
Literature review	2	3	2	3	50%
Model & Dataset Selection	2	1	2	1	100%
Research Benchmarks	3	2			0%
Mid-thesis Report	4	1			0%
Architecture Creation	5	3			0%
Architecture Evaluation	8	2			0%
Architecture Refinement	10	3			0%
Architecture Re-evaluation & Finalization	13	1			0%
Final Architecture Conclusion	14	2			0%
Report Completion	16	3			0%
Video Presentation	18	1			



Chapter C

Thesis Code

This chapter contains the code for the various benchmarks and the 2 custom models implemented in this thesis.

C.1 Benchmarks

This section contains the code for various benchmark models that have been used to compare the Custom Bert Model against.

AlBert Benchmark

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)
```

```
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.

[ ]:
```
## 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_…
```

[ ]:
```
## 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

[ ]:
```
## 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

[ ]:
```
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()
```

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

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

[3]:
```
# 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"
```

[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]:
```
''
```

[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'
```

[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
```

[ ]:
```
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"
```

[ ]:
```
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
```

[ ]:
```
question_answer(test_question, test_context, model_loaded)
```

[ ]:
```
## 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

RoBerta Benchmark

RoBerta_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 8.0 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 42.9 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)
Collecting sacremoses
  Downloading sacremoses-0.0.45-py3-none-any.whl (895 kB)
     |                 | 895 kB 56.4 MB/s
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from transformers) (4.6.1)
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 80.9 MB/s
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from transformers) (21.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from transformers) (3.0.12)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from transformers) (2.23.0)
Collecting huggingface-hub==0.0.12
  Downloading huggingface_hub-0.0.12-py3-none-any.whl (37 kB)
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: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->transformers) (2.10)
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: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (2021.5.30)
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)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (1.15.0)
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 7.7 MB/s
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.96
Collecting wandb
  Downloading wandb-0.11.1-py2.py3-none-any.whl (1.7 MB)
```

```
     |                 | 1.7 MB 8.0 MB/s
Collecting sentry-sdk>=1.0.0
  Downloading sentry_sdk-1.3.1-py2.py3-none-any.whl (133 kB)
     |                 | 133 kB 68.2 MB/s
Collecting graphql-core>=2.3.0
  Downloading graphql_core-3.1.5-py3-none-any.whl (188 kB)
     |                 | 188 kB 72.6 MB/s
Requirement already satisfied: python-dateutil>=2.6.1 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.8.1)
Requirement already satisfied: Click!=8.0.0,>=7.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (7.1.2)
Collecting GitPython>=1.0.0
  Downloading GitPython-3.1.18-py3-none-any.whl (170 kB)
     |                 | 170 kB 59.8 MB/s
Requirement already satisfied: promise<3,>=2.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.3)
Requirement already satisfied: protobuf>=3.12.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (3.17.3)
Requirement already satisfied: six>=1.13.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (1.15.0)
Collecting configparser>=3.8.1
  Downloading configparser-5.0.2-py3-none-any.whl (19 kB)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages
(from wandb) (5.4.1)
Collecting pathtools
  Downloading pathtools-0.1.2.tar.gz (11 kB)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (5.4.8)
Requirement already satisfied: requests<3,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.23.0)
Collecting shortuuid>=0.5.0
  Downloading shortuuid-1.0.1-py3-none-any.whl (7.5 kB)
Collecting docker-pycreds>=0.4.0
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
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 gitdb<5,>=4.0.1
  Downloading gitdb-4.0.7-py3-none-any.whl (63 kB)
     |                 | 63 kB 2.2 MB/s
Collecting smmap<5,>=3.0.1
  Downloading smmap-4.0.0-py2.py3-none-any.whl (24 kB)
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)
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<3,>=2.0.0->wandb) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests<3,>=2.0.0->wandb) (2.10)
```

```
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)
Building wheels for collected packages: pathtools
  Building wheel for pathtools (setup.py) … done
  Created wheel for pathtools: filename=pathtools-0.1.2-py3-none-any.whl
size=8806
sha256=9752bd4db52087efbb75e5b119bc27d70672a725c0cf11dce86417f9e47ba840
  Stored in directory: /root/.cache/pip/wheels/3e/31/09/fa59cef12cdcfecc627b3d24
273699f390e71828921b2cbba2
Successfully built pathtools
Installing collected packages: smmap, gitdb, shortuuid, sentry-sdk, pathtools,
graphql-core, GitPython, docker-pycreds, configparser, wandb
Successfully installed GitPython-3.1.18 configparser-5.0.2 docker-pycreds-0.4.0
gitdb-4.0.7 graphql-core-3.1.5 pathtools-0.1.2 sentry-sdk-1.3.1 shortuuid-1.0.1
smmap-4.0.0 wandb-0.11.1
```

[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, RobertaForQuestionAnswering,␣
 ↪RobertaTokenizerFast
import string, re
```

```
2021-08-03 00:46:44.039651: 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 00:46:44.039674: 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.

[ ]:
```
torch.cuda.is_available()
```

[ ]:
```
True
```

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
```

```
--2021-08-02 08:56:00--  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   142MB/s    in 0.3s

2021-08-02 08:56:01 (142 MB/s) - 'squad/train-v2.0.json' saved
[42123633/42123633]

--2021-08-02 08:56:01--  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 0.1s

2021-08-02 08:56:01 (38.2 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 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 = RobertaTokenizerFast.from_pretrained("roberta-base")

## 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=898823.0,␣
 ↪style=ProgressStyle(descripti…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=456318.0,␣
 ↪style=ProgressStyle(descripti…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=1355863.
 ↪0, style=ProgressStyle(descript…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=481.0,␣
 ↪style=ProgressStyle(description_…
```

[ ]:
```
## 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)
```

2.0.1 Observations

In this section we have successfully taken the split datasets and:

- Added end index postions which helps us identify the correct end values for an answer in a pa
  text piece.
- Tokenized our data using DistilBert
- Added token positions to the start and end of the answers based on their encoded positions.

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

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

4 4. Model Building & Training

[ ]:
```
model = RobertaForQuestionAnswering.from_pretrained("roberta-base")
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=501200538.
↪0, style=ProgressStyle(descri…


Some weights of the model checkpoint at roberta-base were not used when
initializing RobertaForQuestionAnswering: ['lm_head.layer_norm.weight',
'lm_head.dense.weight', 'lm_head.dense.bias', 'lm_head.decoder.weight',
'lm_head.bias', 'lm_head.layer_norm.bias']
- This IS expected if you are initializing RobertaForQuestionAnswering 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 RobertaForQuestionAnswering from
the checkpoint of a model that you expect to be exactly identical (initializing
a BertForSequenceClassification model from a BertForSequenceClassification
model).
Some weights of RobertaForQuestionAnswering were not initialized from the model
checkpoint at roberta-base and are newly initialized: ['qa_outputs.bias',
'qa_outputs.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[ ]:
```
!nvidia-smi
```

```
Mon Aug  2 08:56:43 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.42.01    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE…  Off   | 00000000:00:04.0 Off |                    0 |
| N/A   38C    P0    28W / 250W |      2MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

```
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

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

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))
```

[ ]:
```
# 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
```

4.0.1 Observations

Above we have created a few functions that will help us with validation better.

The normalize text function allows us to create a uniform text format. It removes punctuations, fixes whitespaces, removes articles and converts everything to lower text. This helps in ensuring that the input and outputs match properly.

The calculate stats function returns the exact match and F1 scores for each item we train on. This helps us identiy how well our model is performing.

[28]:
```
# 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: 9.0432
```

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

```
Val Loss: 0.7995, EM: 0.7458, F1: 0.8357
Epoch 1/9
----------
```

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

```
Training Loss: 7.0914
```

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

```
Val Loss: 0.1252, EM: 0.7423, F1: 0.8319
Epoch 2/9
```

```
----------
```

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

```
Training Loss: 5.9771
```

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

```
Val Loss: 0.0753, EM: 0.7461, F1: 0.8340
Epoch 3/9
----------
```

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

```
Training Loss: 5.2691
```

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

```
Val Loss: 0.0962, EM: 0.7502, F1: 0.8342
Epoch 4/9
----------
```

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

```
Training Loss: 4.6029
```

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

```
Val Loss: 0.0104, EM: 0.7597, F1: 0.8428
Epoch 5/9
----------
```

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

```
Training Loss: 4.1284
```

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

```
Val Loss: 0.2058, EM: 0.7423, F1: 0.8263
Epoch 6/9
----------
```

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

```
Training Loss: 3.7402
```

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

```
Val Loss: 0.2926, EM: 0.7599, F1: 0.8429
Epoch 7/9
----------
```

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

```
Training Loss: 3.4308
```

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

```
Val Loss: 0.0051, EM: 0.7498, F1: 0.8319
Epoch 8/9
----------
```

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

```
Training Loss: 3.1041
```

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

```
Val Loss: 0.0561, EM: 0.7489, F1: 0.8341
Epoch 9/9
----------
```

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

```
Training Loss: 2.8382
```

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

```
Val Loss: 0.0048, EM: 0.7365, F1: 0.8241
```

[29]:
```
# We save our model so that it can be reused later

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

[2]:
```
# Generate a Tensorboard

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

```
<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

[31]:
```
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"
```

[32]:
```
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
```

[33]:
```
question_answer(test_question, test_context, model)
```

[33]:
```
' Rollo'
```

[34]:
```
## Checking the response is the same as we had in the validation set.
question_answer(val_questions[0], val_contexts[0], model)
```

[34]:
```
' France'
```

5.0.1 Observations

We can see that our model is performing correctly. As an aditional step below we will load the model again and try to predict the answers for the same questions as we did above.

[35]:
```
model_loaded = torch.load('./robertaModel.pt')
tokenizer = RobertaTokenizerFast.from_pretrained('roberta-base')
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

[36]:
```
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"
```

[37]:
```
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
```

[38]:
```
question_answer(test_question, test_context, model_loaded)
```

[38]:
```
' Rollo'
```

[39]:
```
## 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"""
```

[40]:
```
question_answer(question, context, model_loaded)
```

[40]:
```
' exploding gradient problem'
```

6 END OF FILE

DistilBert Benchmark

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]:
```
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]:

C.2 Custom Bert Model Code

Custom_Bert_Model_AdaFactor_3e

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

[1]:
```
!pip install torch
!pip install transformers
!pip install sentencepiece
!pip install wandb
!pip install allennlp
```

```
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 14.5 MB/s
Collecting pyyaml>=5.1
  Downloading PyYAML-5.4.1-cp37-cp37m-manylinux1_x86_64.whl (636 kB)
     |                 | 636 kB 62.0 MB/s
Collecting sacremoses
  Downloading sacremoses-0.0.45-py3-none-any.whl (895 kB)
     |                 | 895 kB 65.3 MB/s
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from transformers) (4.6.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from transformers) (1.19.5)
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 60.9 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)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from transformers) (21.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from transformers) (2.23.0)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-
packages (from transformers) (4.41.1)
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: 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: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
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)
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 14.1 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 12.4 MB/s
Requirement already satisfied: requests<3,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.23.0)
Collecting pathtools
  Downloading pathtools-0.1.2.tar.gz (11 kB)
Requirement already satisfied: Click!=8.0.0,>=7.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (7.1.2)
Collecting GitPython>=1.0.0
  Downloading GitPython-3.1.18-py3-none-any.whl (170 kB)
     |                | 170 kB 57.3 MB/s
Collecting docker-pycreds>=0.4.0
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
Collecting sentry-sdk>=1.0.0
  Downloading sentry_sdk-1.3.1-py2.py3-none-any.whl (133 kB)
     |                | 133 kB 64.1 MB/s
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)
Collecting shortuuid>=0.5.0
  Downloading shortuuid-1.0.1-py3-none-any.whl (7.5 kB)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (5.4.8)
Collecting subprocess32>=3.5.3
  Downloading subprocess32-3.5.4.tar.gz (97 kB)
     |                | 97 kB 8.3 MB/s
Collecting configparser>=3.8.1
  Downloading configparser-5.0.2-py3-none-any.whl (19 kB)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages
(from wandb) (5.4.1)
Collecting urllib3>=1.26.5
  Downloading urllib3-1.26.6-py2.py3-none-any.whl (138 kB)
     |                | 138 kB 63.2 MB/s
Requirement already satisfied: protobuf>=3.12.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (3.17.3)
Collecting gitdb<5,>=4.0.1
  Downloading gitdb-4.0.7-py3-none-any.whl (63 kB)
     |                | 63 kB 2.3 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)
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: idna<4,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.0.0->wandb) (2.10)
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=70d7aed3fc55a42b15e484748f907db663e583983339fe47e42bc0db2fcaa294
  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=8c21a1962a7cafa21a3a05a2a757cbbb749eb8532645cbc2c8624143b551cbec
  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
Collecting allennlp
  Downloading allennlp-2.6.0-py3-none-any.whl (689 kB)
     |                  | 689 kB 14.6 MB/s
```

```
Requirement already satisfied: wandb<0.12.0,>=0.10.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.11.2)
Collecting transformers<4.9,>=4.1
  Downloading transformers-4.8.2-py3-none-any.whl (2.5 MB)
     |                | 2.5 MB 24.6 MB/s
Requirement already satisfied: sentencepiece in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.1.96)
Collecting overrides==3.1.0
  Downloading overrides-3.1.0.tar.gz (11 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from allennlp) (1.19.5)
Requirement already satisfied: requests>=2.18 in /usr/local/lib/python3.7/dist-
packages (from allennlp) (2.26.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-
packages (from allennlp) (0.22.2.post1)
Requirement already satisfied: lmdb in /usr/local/lib/python3.7/dist-packages
(from allennlp) (0.99)
Requirement already satisfied: huggingface-hub>=0.0.8 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.0.12)
Requirement already satisfied: filelock<3.1,>=3.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (3.0.12)
Requirement already satisfied: termcolor==1.1.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (1.1.0)
Collecting google-cloud-storage<1.42.0,>=1.38.0
  Downloading google_cloud_storage-1.41.1-py2.py3-none-any.whl (105 kB)
     |                | 105 kB 58.6 MB/s
Requirement already satisfied: torchvision<0.11.0,>=0.8.1 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.10.0+cu102)
Collecting boto3<2.0,>=1.14
  Downloading boto3-1.18.16-py3-none-any.whl (131 kB)
     |                | 131 kB 63.0 MB/s
Collecting checklist==0.0.11
  Downloading checklist-0.0.11.tar.gz (12.1 MB)
     |                | 12.1 MB 24.8 MB/s
Collecting jsonnet>=0.10.0
  Downloading jsonnet-0.17.0.tar.gz (259 kB)
     |                | 259 kB 50.1 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-
packages (from allennlp) (1.4.1)
Collecting tensorboardX>=1.2
  Downloading tensorboardX-2.4-py2.py3-none-any.whl (124 kB)
     |                | 124 kB 61.1 MB/s
Requirement already satisfied: more-itertools in
/usr/local/lib/python3.7/dist-packages (from allennlp) (8.8.0)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages
(from allennlp) (3.1.0)
Requirement already satisfied: spacy<3.1,>=2.1.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (2.2.4)
```

```
Requirement already satisfied: pytest in /usr/local/lib/python3.7/dist-packages
(from allennlp) (3.6.4)
Requirement already satisfied: torch<1.10.0,>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (1.9.0+cu102)
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages
(from allennlp) (3.2.5)
Requirement already satisfied: tqdm>=4.19 in /usr/local/lib/python3.7/dist-
packages (from allennlp) (4.41.1)
Collecting munch>=2.5
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: dill>=0.3.1 in /usr/local/lib/python3.7/dist-
packages (from checklist==0.0.11->allennlp) (0.3.4)
Requirement already satisfied: jupyter>=1.0 in /usr/local/lib/python3.7/dist-
packages (from checklist==0.0.11->allennlp) (1.0.0)
Requirement already satisfied: ipywidgets>=7.5 in /usr/local/lib/python3.7/dist-
packages (from checklist==0.0.11->allennlp) (7.6.3)
Collecting patternfork-nosql
  Downloading patternfork_nosql-3.6.tar.gz (22.3 MB)
     |                  | 22.3 MB 1.3 MB/s
Collecting iso-639
  Downloading iso-639-0.4.5.tar.gz (167 kB)
     |                  | 167 kB 64.3 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting s3transfer<0.6.0,>=0.5.0
  Downloading s3transfer-0.5.0-py3-none-any.whl (79 kB)
     |                  | 79 kB 10.1 MB/s
Collecting botocore<1.22.0,>=1.21.16
  Downloading botocore-1.21.16-py3-none-any.whl (7.8 MB)
     |                  | 7.8 MB 49.4 MB/s
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/usr/local/lib/python3.7/dist-packages (from
botocore<1.22.0,>=1.21.16->boto3<2.0,>=1.14->allennlp) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in
/usr/local/lib/python3.7/dist-packages (from
botocore<1.22.0,>=1.21.16->boto3<2.0,>=1.14->allennlp) (1.26.6)
Collecting google-resumable-media<3.0dev,>=1.3.0
  Downloading google_resumable_media-1.3.3-py2.py3-none-any.whl (75 kB)
     |                  | 75 kB 6.2 MB/s
Requirement already satisfied: google-auth<3.0dev,>=1.24.0 in
/usr/local/lib/python3.7/dist-packages (from google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (1.32.1)
Collecting google-cloud-core<3.0dev,>=1.6.0
  Downloading google_cloud_core-1.7.2-py2.py3-none-any.whl (28 kB)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (0.2.8)
Requirement already satisfied: setuptools>=40.3.0 in
```

```
/usr/local/lib/python3.7/dist-packages (from google-
auth<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (57.2.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-
packages (from google-auth<3.0dev,>=1.24.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (1.15.0)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (4.2.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-
packages (from google-auth<3.0dev,>=1.24.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (4.7.2)
Requirement already satisfied: google-api-core<2.0.0dev,>=1.21.0 in
/usr/local/lib/python3.7/dist-packages (from google-cloud-
core<3.0dev,>=1.6.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (1.26.3)
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.7/dist-
packages (from google-api-core<2.0.0dev,>=1.21.0->google-cloud-
core<3.0dev,>=1.6.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (21.0)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-
core<2.0.0dev,>=1.21.0->google-cloud-core<3.0dev,>=1.6.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (1.53.0)
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages
(from google-api-core<2.0.0dev,>=1.21.0->google-cloud-
core<3.0dev,>=1.6.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (2018.9)
Requirement already satisfied: protobuf>=3.12.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-
core<2.0.0dev,>=1.21.0->google-cloud-core<3.0dev,>=1.6.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (3.17.3)
Collecting google-crc32c<2.0dev,>=1.0
  Downloading google_crc32c-1.1.2-cp37-cp37m-manylinux2014_x86_64.whl (38 kB)
Requirement already satisfied: cffi>=1.0.0 in /usr/local/lib/python3.7/dist-
packages (from google-crc32c<2.0dev,>=1.0->google-resumable-
media<3.0dev,>=1.3.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (1.14.6)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-
packages (from cffi>=1.0.0->google-crc32c<2.0dev,>=1.0->google-resumable-
media<3.0dev,>=1.3.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (2.20)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from huggingface-hub>=0.0.8->allennlp)
(4.6.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from huggingface-hub>=0.0.8->allennlp)
(3.7.4.3)
Requirement already satisfied: traitlets>=4.3.1 in
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (5.0.5)
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.7/dist-
packages (from ipywidgets>=7.5->checklist==0.0.11->allennlp) (5.5.0)
Requirement already satisfied: ipykernel>=4.5.1 in
```

```
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (4.10.1)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (1.0.0)
Requirement already satisfied: widgetsnbextension~=3.5.0 in
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (3.5.1)
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.7/dist-
packages (from ipywidgets>=7.5->checklist==0.0.11->allennlp) (5.1.3)
Requirement already satisfied: tornado>=4.0 in /usr/local/lib/python3.7/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(5.1.1)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.7/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(5.3.5)
Requirement already satisfied: simplegeneric>0.8 in
/usr/local/lib/python3.7/dist-packages (from
ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (0.8.1)
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages
(from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (4.8.0)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(0.7.5)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in
/usr/local/lib/python3.7/dist-packages (from
ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (1.0.18)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(2.6.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(4.4.2)
Requirement already satisfied: notebook in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.3.1)
Requirement already satisfied: qtconsole in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.1.1)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.6.1)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.2.0)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-
packages (from nbformat>=4.2.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(4.7.1)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/usr/local/lib/python3.7/dist-packages (from
nbformat>=4.2.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (2.6.0)
Requirement already satisfied: ipython-genutils in
```

```
/usr/local/lib/python3.7/dist-packages (from
nbformat>=4.2.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (0.2.0)
Requirement already satisfied: pyparsing>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging>=14.3->google-api-
core<2.0.0dev,>=1.21.0->google-cloud-core<3.0dev,>=1.6.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (2.4.7)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages
(from prompt-toolkit<2.0.0,>=1.0.4->ipython>=4.0.0->ipywidgets>=7.5->checklist==
0.0.11->allennlp) (0.2.5)
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<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (0.4.8)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests>=2.18->allennlp) (2.10)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.18->allennlp) (2.0.2)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.18->allennlp)
(2021.5.30)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.0.5)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(3.0.5)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.0.0)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.1.3)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(0.4.1)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.0.5)
Requirement already satisfied: thinc==7.4.0 in /usr/local/lib/python3.7/dist-
packages (from spacy<3.1,>=2.1.0->allennlp) (7.4.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(2.0.5)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(0.8.2)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->huggingface-hub>=0.0.8->allennlp) (3.5.0)
Requirement already satisfied: pillow>=5.3.0 in /usr/local/lib/python3.7/dist-
```

```
packages (from torchvision<0.11.0,>=0.8.1->allennlp) (7.1.2)
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in
/usr/local/lib/python3.7/dist-packages (from transformers<4.9,>=4.1->allennlp)
(0.10.3)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers<4.9,>=4.1->allennlp)
(2019.12.20)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages
(from transformers<4.9,>=4.1->allennlp) (5.4.1)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/dist-
packages (from transformers<4.9,>=4.1->allennlp) (0.0.45)
Requirement already satisfied: GitPython>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(3.1.18)
Requirement already satisfied: docker-pycreds>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(0.4.0)
Requirement already satisfied: configparser>=3.8.1 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(5.0.2)
Requirement already satisfied: Click!=8.0.0,>=7.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(7.1.2)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.7/dist-
packages (from wandb<0.12.0,>=0.10.0->allennlp) (5.4.8)
Requirement already satisfied: sentry-sdk>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(1.3.1)
Requirement already satisfied: subprocess32>=3.5.3 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(3.5.4)
Requirement already satisfied: pathtools in /usr/local/lib/python3.7/dist-
packages (from wandb<0.12.0,>=0.10.0->allennlp) (0.1.2)
Requirement already satisfied: promise<3,>=2.0 in /usr/local/lib/python3.7/dist-
packages (from wandb<0.12.0,>=0.10.0->allennlp) (2.3)
Requirement already satisfied: shortuuid>=0.5.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(1.0.1)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.7/dist-
packages (from GitPython>=1.0.0->wandb<0.12.0,>=0.10.0->allennlp) (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<0.12.0,>=0.10.0->allennlp) (4.0.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages
(from notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (2.11.3)
Requirement already satisfied: terminado>=0.8.1 in
/usr/local/lib/python3.7/dist-packages (from
notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (0.10.1)
```

```
Requirement already satisfied: Send2Trash in /usr/local/lib/python3.7/dist-
packages (from notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (1.7.1)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.7/dist-
packages (from jupyter-
client->ipykernel>=4.5.1->ipywidgets>=7.5->checklist==0.0.11->allennlp) (22.1.0)
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.7/dist-
packages (from
terminado>=0.8.1->notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (0.7.0)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-
packages (from h5py->allennlp) (1.5.2)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.7/dist-packages (from
jinja2->notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (2.0.1)
Requirement already satisfied: testpath in /usr/local/lib/python3.7/dist-
packages (from nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.5.0)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.7/dist-packages (from
nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.3)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.7/dist-packages (from
nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (1.4.3)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.7/dist-
packages (from nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.7.1)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.7/dist-packages (from
nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.8.4)
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages
(from nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (3.3.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-
packages (from bleach->nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp)
(0.5.1)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages
(from patternfork-nosql->checklist==0.0.11->allennlp) (0.16.0)
Collecting backports.csv
  Downloading backports.csv-1.0.7-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-
packages (from patternfork-nosql->checklist==0.0.11->allennlp) (4.6.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages
(from patternfork-nosql->checklist==0.0.11->allennlp) (4.2.6)
Collecting feedparser
  Downloading feedparser-6.0.8-py3-none-any.whl (81 kB)
     |                 | 81 kB 11.0 MB/s
Collecting pdfminer.six
  Downloading pdfminer.six-20201018-py3-none-any.whl (5.6 MB)
     |                 | 5.6 MB 58.6 MB/s
Collecting python-docx
  Downloading python-docx-0.8.11.tar.gz (5.6 MB)
     |                 | 5.6 MB 56.5 MB/s
```

```
Collecting cherrypy
  Downloading CherryPy-18.6.1-py2.py3-none-any.whl (419 kB)
     |                  | 419 kB 48.9 MB/s
Collecting cheroot>=8.2.1
  Downloading cheroot-8.5.2-py2.py3-none-any.whl (97 kB)
     |                  | 97 kB 8.1 MB/s
Collecting jaraco.collections
  Downloading jaraco.collections-3.3.0-py3-none-any.whl (9.9 kB)
Collecting portend>=2.1.1
  Downloading portend-2.7.1-py3-none-any.whl (5.3 kB)
Collecting zc.lockfile
  Downloading zc.lockfile-2.0-py2.py3-none-any.whl (9.7 kB)
Collecting jaraco.functools
  Downloading jaraco.functools-3.3.0-py3-none-any.whl (6.8 kB)
Collecting tempora>=1.8
  Downloading tempora-4.1.1-py3-none-any.whl (15 kB)
Collecting sgmllib3k
  Downloading sgmllib3k-1.0.0.tar.gz (5.8 kB)
Collecting jaraco.classes
  Downloading jaraco.classes-3.2.1-py3-none-any.whl (5.6 kB)
Collecting jaraco.text
  Downloading jaraco.text-3.5.1-py3-none-any.whl (8.1 kB)
Requirement already satisfied: sortedcontainers in
/usr/local/lib/python3.7/dist-packages (from pdfminer.six->patternfork-
nosql->checklist==0.0.11->allennlp) (2.4.0)
Requirement already satisfied: chardet in /usr/local/lib/python3.7/dist-packages
(from pdfminer.six->patternfork-nosql->checklist==0.0.11->allennlp) (3.0.4)
Collecting cryptography
  Downloading cryptography-3.4.7-cp36-abi3-manylinux2014_x86_64.whl (3.2 MB)
     |                  | 3.2 MB 51.1 MB/s
Requirement already satisfied: pluggy<0.8,>=0.5 in
/usr/local/lib/python3.7/dist-packages (from pytest->allennlp) (0.7.1)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/dist-
packages (from pytest->allennlp) (21.2.0)
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.7/dist-
packages (from pytest->allennlp) (1.10.0)
Requirement already satisfied: atomicwrites>=1.0 in
/usr/local/lib/python3.7/dist-packages (from pytest->allennlp) (1.4.0)
Requirement already satisfied: qtpy in /usr/local/lib/python3.7/dist-packages
(from qtconsole->jupyter>=1.0->checklist==0.0.11->allennlp) (1.9.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers<4.9,>=4.1->allennlp) (1.0.1)
Building wheels for collected packages: checklist, overrides, jsonnet, iso-639,
patternfork-nosql, python-docx, sgmllib3k
  Building wheel for checklist (setup.py) … done
  Created wheel for checklist: filename=checklist-0.0.11-py3-none-any.whl
size=12165634
sha256=765491d4ae3fdb36a72cffb0317e3c14cb54120981e7c6aaa2b7fd718d96f021
```

```
  Stored in directory: /root/.cache/pip/wheels/6a/8a/07/6446879be434879c27671c83
443727d74cecf6b630c8a24d03
  Building wheel for overrides (setup.py) … done
  Created wheel for overrides: filename=overrides-3.1.0-py3-none-any.whl
size=10188
sha256=06783c7e5040b140fa1cc683e6902114f7b7b4d3380e4c0530985c00f776b3cf
  Stored in directory: /root/.cache/pip/wheels/3a/0d/38/01a9bc6e20dcfaf0a6a7b552
d03137558ba1c38aea47644682
  Building wheel for jsonnet (setup.py) … done
  Created wheel for jsonnet: filename=jsonnet-0.17.0-cp37-cp37m-linux_x86_64.whl
size=3388670
sha256=8ece4c7815c8e829ab607002b8e0d44a03d98863b79c7bfe7242f820414e5e8b
  Stored in directory: /root/.cache/pip/wheels/1c/28/7e/287c6b19f7161bb03c6986a3
c46b51d0d7d9a1805346634e3a
  Building wheel for iso-639 (setup.py) … done
  Created wheel for iso-639: filename=iso_639-0.4.5-py3-none-any.whl size=169062
sha256=30ca3fb80b4ab22e735af18fa8931eb178760cbd1700f666e1183f940040e5c3
  Stored in directory: /root/.cache/pip/wheels/47/60/19/6d020fc92138ed1b113a1827
1e83ea4b5525fe770cb45b9a2e
  Building wheel for patternfork-nosql (setup.py) … done
  Created wheel for patternfork-nosql: filename=patternfork_nosql-3.6-py3-none-
any.whl size=22332805
sha256=b3dc7f7300dc7e6cb36391018606333f8f7a514020a4c21f7a674dd5f2956c5c
  Stored in directory: /root/.cache/pip/wheels/97/72/8f/5305fe28168f93b658da9ed4
33b9a1d3ec90594faa0c9aaf4b
  Building wheel for python-docx (setup.py) … done
  Created wheel for python-docx: filename=python_docx-0.8.11-py3-none-any.whl
size=184507
sha256=a4dc3df40300d8d001d44921b3e020ad4a17abc84bfae6bafd12f1112a0e6c98
  Stored in directory: /root/.cache/pip/wheels/f6/6f/b9/d798122a8b55b74ad30b5f52
b01482169b445fbb84a11797a6
  Building wheel for sgmllib3k (setup.py) … done
  Created wheel for sgmllib3k: filename=sgmllib3k-1.0.0-py3-none-any.whl
size=6065
sha256=02b9f0bb6671ed289da558267d4a1e93a0b7ea629b8e23df3358871de3f5f79b
  Stored in directory: /root/.cache/pip/wheels/73/ad/a4/0dff4a6ef231fc0dfa12ffba
c2a36cebfdddfe059f50e019aa
Successfully built checklist overrides jsonnet iso-639 patternfork-nosql python-
docx sgmllib3k
Installing collected packages: jaraco.functools, tempora, jaraco.text,
jaraco.classes, zc.lockfile, sgmllib3k, portend, jmespath, jaraco.collections,
cryptography, cheroot, python-docx, pdfminer.six, google-crc32c, feedparser,
cherrypy, botocore, backports.csv, transformers, s3transfer, patternfork-nosql,
munch, iso-639, google-resumable-media, google-cloud-core, tensorboardX,
overrides, jsonnet, google-cloud-storage, checklist, boto3, allennlp
  Attempting uninstall: transformers
    Found existing installation: transformers 4.9.1
    Uninstalling transformers-4.9.1:
```

```
    Successfully uninstalled transformers-4.9.1
  Attempting uninstall: google-resumable-media
    Found existing installation: google-resumable-media 0.4.1
    Uninstalling google-resumable-media-0.4.1:
      Successfully uninstalled google-resumable-media-0.4.1
  Attempting uninstall: google-cloud-core
    Found existing installation: google-cloud-core 1.0.3
    Uninstalling google-cloud-core-1.0.3:
      Successfully uninstalled google-cloud-core-1.0.3
  Attempting uninstall: google-cloud-storage
    Found existing installation: google-cloud-storage 1.18.1
    Uninstalling google-cloud-storage-1.18.1:
      Successfully uninstalled google-cloud-storage-1.18.1
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-cloud-bigquery 1.21.0 requires google-resumable-
media!=0.4.0,<0.5.0dev,>=0.3.1, but you have google-resumable-media 1.3.3 which
is incompatible.
Successfully installed allennlp-2.6.0 backports.csv-1.0.7 boto3-1.18.16
botocore-1.21.16 checklist-0.0.11 cheroot-8.5.2 cherrypy-18.6.1
cryptography-3.4.7 feedparser-6.0.8 google-cloud-core-1.7.2 google-cloud-
storage-1.41.1 google-crc32c-1.1.2 google-resumable-media-1.3.3 iso-639-0.4.5
jaraco.classes-3.2.1 jaraco.collections-3.3.0 jaraco.functools-3.3.0
jaraco.text-3.5.1 jmespath-0.10.0 jsonnet-0.17.0 munch-2.5.0 overrides-3.1.0
patternfork-nosql-3.6 pdfminer.six-20201018 portend-2.7.1 python-docx-0.8.11
s3transfer-0.5.0 sgmllib3k-1.0.0 tempora-4.1.1 tensorboardX-2.4
transformers-4.8.2 zc.lockfile-2.0
```

[2]:
```
import torch
import transformers as tfs
import numpy as np
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 BertTokenizerFast,BertPreTrainedModel, BertConfig,␣
↪BertModel, BertGenerationEncoder
from transformers import Adafactor, DistilBertTokenizerFast, DistilBertConfig
from transformers.modeling_outputs import (
    BaseModelOutput,
    BaseModelOutputWithPastAndCrossAttentions,
    BaseModelOutputWithPoolingAndCrossAttentions,
    CausalLMOutputWithCrossAttentions,
```

```
    MaskedLMOutput,
    MultipleChoiceModelOutput,
    NextSentencePredictorOutput,
    QuestionAnsweringModelOutput,
    SequenceClassifierOutput,
    TokenClassifierOutput,
)
import string, re
import torch.nn as nn
from allennlp.nn.util import masked_log_softmax, masked_max

import math

from transformers.activations import gelu
from transformers.deepspeed import is_deepspeed_zero3_enabled
from transformers.file_utils import (
    add_code_sample_docstrings,
    add_start_docstrings,
    add_start_docstrings_to_model_forward,
    replace_return_docstrings,
)

from transformers.modeling_utils import (
    PreTrainedModel,
    apply_chunking_to_forward,
    find_pruneable_heads_and_indices,
    prune_linear_layer,
)
from transformers.utils import logging
import torch.nn.functional as F
from torch.nn import Parameter
```

[3]:
```
torch.cuda.is_available()
```

[3]:
```
True
```

1 Utility functions for Metrics evaluation

[4]:
```
# 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))))
```

2 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.

2.1 Download SQuAD 2.0 Data

[5]:
```
# 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))
```

[6]:
```
# 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)
```

[7]:
```
# 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
```

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").

[8]:
```
# ## 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
```

```
--2021-08-08 11:25:46--  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   128MB/s    in 0.3s
```

```
2021-08-08 11:25:47 (128 MB/s) - ‘squad/train-v2.0.json’ saved
[42123633/42123633]

--2021-08-08 11:25:47--  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 0.06s

2021-08-08 11:25:47 (69.0 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.

[9]:
```
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.

[10]:
```
len(train_contexts)
```

[10]:
```
86821
```

[11]:
```
train_contexts[0]
```

[11]:
```
'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".'
```

[12]:
```
train_questions[0]
```

[12]:
```
'When did Beyonce start becoming popular?'
```

[13]:
```
train_answers[0]
```

[13]:
```
{'answer_start': 269, 'text': 'in the late 1990s'}
```

[14]:
```
len(train_qac)
```

[14]:
```
86821
```

[15]:
```
train_qac[0]
```

[15]:
```
{'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

[16]:
```
len(val_contexts)
```

[16]:
```
20302
```

[17]:
```
val_contexts[0]
```

[17]:
```
'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.'
```

[18]:
```
val_questions[0]
```

[18]:
```
'In what country is Normandy located?'
```

[19]:
```
val_answers[0]
```

[19]:
```
{'answer_start': 159, 'text': 'France'}
```

[20]:
```
val_qac[0]
```

[20]:
```
{'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?'}
```

2.2 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

3 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.

[21]:
```
## 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)
```

4 Creating Custom BERT Model

[22]:
```
class Encoder(nn.Module):
    """
    Encoder class for Pointer-Net
    """

    def __init__(self, embedding_dim,
                 hidden_dim,
                 n_layers,
                 dropout,
                 bidir):
        """
        Initiate Encoder

        :param Tensor embedding_dim: Number of embbeding channels
        :param int hidden_dim: Number of hidden units for the LSTM
        :param int n_layers: Number of layers for LSTMs
        :param float dropout: Float between 0-1
        :param bool bidir: Bidirectional
        """

        super(Encoder, self).__init__()
        self.hidden_dim = hidden_dim//2 if bidir else hidden_dim
        self.n_layers = n_layers*2 if bidir else n_layers
        self.bidir = bidir
        self.lstm = nn.LSTM(embedding_dim,
                            self.hidden_dim,
                            n_layers,
                            dropout=dropout,
                            bidirectional=bidir)

        # Used for propagating .cuda() command
        self.h0 = Parameter(torch.zeros(1), requires_grad=False)
        self.c0 = Parameter(torch.zeros(1), requires_grad=False)

    def forward(self, embedded_inputs,
                hidden):
        """
        Encoder - Forward-pass

        :param Tensor embedded_inputs: Embedded inputs of Pointer-Net
        :param Tensor hidden: Initiated hidden units for the LSTMs (h, c)
        :return: LSTMs outputs and hidden units (h, c)
        """

        embedded_inputs = embedded_inputs.permute(1, 0, 2)
```

```
        outputs, hidden = self.lstm(embedded_inputs, hidden)

        return outputs.permute(1, 0, 2), hidden

    def init_hidden(self, embedded_inputs):
        """
        Initiate hidden units

        :param Tensor embedded_inputs: The embedded input of Pointer-NEt
        :return: Initiated hidden units for the LSTMs (h, c)
        """

        batch_size = embedded_inputs.size(0)

        # Reshaping (Expanding)
        h0 = self.h0.unsqueeze(0).unsqueeze(0).repeat(self.n_layers,
                                                      batch_size,
                                                      self.hidden_dim)
        c0 = self.h0.unsqueeze(0).unsqueeze(0).repeat(self.n_layers,
                                                      batch_size,
                                                      self.hidden_dim)

        return h0, c0


class Attention(nn.Module):
    """
    Attention model for Pointer-Net
    """

    def __init__(self, input_dim,
                 hidden_dim):
        """
        Initiate Attention

        :param int input_dim: Input's diamention
        :param int hidden_dim: Number of hidden units in the attention
        """

        super(Attention, self).__init__()

        self.input_dim = input_dim
        self.hidden_dim = hidden_dim

        self.input_linear = nn.Linear(input_dim, hidden_dim)
        self.context_linear = nn.Conv1d(input_dim, hidden_dim, 1, 1)
```

```
        self.V = Parameter(torch.FloatTensor(hidden_dim), requires_grad=True)
        self._inf = Parameter(torch.FloatTensor([float('-inf')]),␣
↪requires_grad=False)
        self.tanh = nn.Tanh()
        self.softmax = nn.Softmax()

        # Initialize vector V
        nn.init.uniform(self.V, -1, 1)

    def forward(self, input,
                context,
                mask):
        """
        Attention - Forward-pass

        :param Tensor input: Hidden state h
        :param Tensor context: Attention context
        :param ByteTensor mask: Selection mask
        :return: tuple of - (Attentioned hidden state, Alphas)
        """

        # (batch, hidden_dim, seq_len)
        inp = self.input_linear(input).unsqueeze(2).expand(-1, -1, context.
↪size(1))

        # (batch, hidden_dim, seq_len)
        context = context.permute(0, 2, 1)
        ctx = self.context_linear(context)

        # (batch, 1, hidden_dim)
        V = self.V.unsqueeze(0).expand(context.size(0), -1).unsqueeze(1)

        # (batch, seq_len)
        att = torch.bmm(V, self.tanh(inp + ctx)).squeeze(1)
        if len(att[mask]) > 0:
            att[mask] = self.inf[mask]
        alpha = self.softmax(att)

        hidden_state = torch.bmm(ctx, alpha.unsqueeze(2)).squeeze(2)

        return hidden_state, alpha

    def init_inf(self, mask_size):
        self.inf = self._inf.unsqueeze(1).expand(*mask_size)


class Decoder(nn.Module):
```

```
    """
    Decoder model for Pointer-Net
    """

    def __init__(self, embedding_dim,
                 hidden_dim):
        """
        Initiate Decoder

        :param int embedding_dim: Number of embeddings in Pointer-Net
        :param int hidden_dim: Number of hidden units for the decoder's RNN
        """

        super(Decoder, self).__init__()
        self.embedding_dim = embedding_dim
        self.hidden_dim = hidden_dim

        self.input_to_hidden = nn.Linear(embedding_dim, 4 * hidden_dim)
        self.hidden_to_hidden = nn.Linear(hidden_dim, 4 * hidden_dim)
        self.hidden_out = nn.Linear(hidden_dim * 2, hidden_dim)
        self.att = Attention(hidden_dim, hidden_dim)

        # Used for propagating .cuda() command
        self.mask = Parameter(torch.ones(1), requires_grad=False)
        self.runner = Parameter(torch.zeros(1), requires_grad=False)

    def forward(self, embedded_inputs,
                decoder_input,
                hidden,
                context):
        """
        Decoder - Forward-pass

        :param Tensor embedded_inputs: Embedded inputs of Pointer-Net
        :param Tensor decoder_input: First decoder's input
        :param Tensor hidden: First decoder's hidden states
        :param Tensor context: Encoder's outputs
        :return: (Output probabilities, Pointers indices), last hidden state
        """

        batch_size = embedded_inputs.size(0)
        input_length = embedded_inputs.size(1)

        # (batch, seq_len)
        mask = self.mask.repeat(input_length).unsqueeze(0).repeat(batch_size, 1)
        self.att.init_inf(mask.size())
```

```
        # Generating arang(input_length), broadcasted across batch_size
        runner = self.runner.repeat(input_length)
        for i in range(input_length):
            runner.data[i] = i
        runner = runner.unsqueeze(0).expand(batch_size, -1).long()

        outputs = []
        pointers = []

        def step(x, hidden):
            """
            Recurrence step function

            :param Tensor x: Input at time t
            :param tuple(Tensor, Tensor) hidden: Hidden states at time t-1
            :return: Hidden states at time t (h, c), Attention probabilities␣
↪(Alpha)
            """

            # Regular LSTM
            h, c = hidden

            gates = self.input_to_hidden(x) + self.hidden_to_hidden(h)
            input, forget, cell, out = gates.chunk(4, 1)

            input = torch.sigmoid(input)
            forget = torch.sigmoid(forget)
            cell = torch.tanh(cell)
            out = torch.sigmoid(out)

            c_t = (forget * c) + (input * cell)
            h_t = out * torch.tanh(c_t)

            # Attention section
            hidden_t, output = self.att(h_t, context, torch.eq(mask, 0))
            hidden_t = torch.tanh(self.hidden_out(torch.cat((hidden_t, h_t),␣
↪1)))

            return hidden_t, c_t, output

        # Recurrence loop
        for _ in range(input_length):
            h_t, c_t, outs = step(decoder_input, hidden)
            hidden = (h_t, c_t)

            # Masking selected inputs
            masked_outs = outs * mask
```

```
            # Get maximum probabilities and indices
            max_probs, indices = masked_outs.max(1)
            one_hot_pointers = (runner == indices.unsqueeze(1).expand(-1, outs.
 ↪size()[1])).float()

            # Update mask to ignore seen indices
            mask  = mask * (1 - one_hot_pointers)

            # Get embedded inputs by max indices
            embedding_mask = one_hot_pointers.unsqueeze(2).expand(-1, -1, self.
 ↪embedding_dim).byte()
            decoder_input = embedded_inputs[embedding_mask.data].
 ↪view(batch_size, self.embedding_dim)

            outputs.append(outs.unsqueeze(0))
            pointers.append(indices.unsqueeze(1))

        outputs = torch.cat(outputs).permute(1, 0, 2)
        pointers = torch.cat(pointers, 1)

        return (outputs, pointers), hidden


class PointerNet(nn.Module):
    """
    Pointer-Net
    """

    def __init__(self, embedding_dim,
                 hidden_dim,
                 lstm_layers,
                 dropout,
                 bidir=False):
        """
        Initiate Pointer-Net

        :param int embedding_dim: Number of embbeding channels
        :param int hidden_dim: Encoders hidden units
        :param int lstm_layers: Number of layers for LSTMs
        :param float dropout: Float between 0-1
        :param bool bidir: Bidirectional
        """

        super(PointerNet, self).__init__()
        self.embedding_dim = embedding_dim
        self.bidir = bidir
```

```
        self.embedding = nn.Linear(2, embedding_dim)
        self.encoder = Encoder(embedding_dim,
                               hidden_dim,
                               lstm_layers,
                               dropout,
                               bidir)
        self.decoder = Decoder(embedding_dim, hidden_dim)
        self.decoder_input0 = Parameter(torch.FloatTensor(embedding_dim),␣
↪requires_grad=False)

        # Initialize decoder_input0
        nn.init.uniform(self.decoder_input0, -1, 1)

    def forward(self, inputs):
        """
        PointerNet - Forward-pass

        :param Tensor inputs: Input sequence
        :return: Pointers probabilities and indices
        """

        batch_size = inputs.size(0)
        input_length = inputs.size(1)

        decoder_input0 = self.decoder_input0.unsqueeze(0).expand(batch_size, -1)

        inputs = inputs.view(batch_size * input_length, -1)
        embedded_inputs = self.embedding(inputs).view(batch_size, input_length,␣
↪-1)

        encoder_hidden0 = self.encoder.init_hidden(embedded_inputs)
        encoder_outputs, encoder_hidden = self.encoder(embedded_inputs,
                                                       encoder_hidden0)
        if self.bidir:
            decoder_hidden0 = (torch.cat(encoder_hidden[0][-2:], dim=-1),
                               torch.cat(encoder_hidden[1][-2:], dim=-1))
        else:
            decoder_hidden0 = (encoder_hidden[0][-1],
                               encoder_hidden[1][-1])
        (outputs, pointers), decoder_hidden = self.decoder(embedded_inputs,
                                                           decoder_input0,
                                                           decoder_hidden0,
                                                           encoder_outputs)

        return  outputs, pointers
```

[23]:
```
# coding=utf-8
# Copyright 2019-present, the HuggingFace Inc. team, The Google AI Language␣
 ↪Team and Facebook, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
"""
 PyTorch DistilBERT model adapted in part from Facebook, Inc XLM model (https://
 ↪github.com/facebookresearch/XLM) and in
 part from HuggingFace PyTorch version of Google AI Bert model (https://github.
 ↪com/google-research/bert)
"""

logger = logging.get_logger(__name__)
_CHECKPOINT_FOR_DOC = "distilbert-base-uncased"
_CONFIG_FOR_DOC = "DistilBertConfig"
_TOKENIZER_FOR_DOC = "DistilBertTokenizer"

DISTILBERT_PRETRAINED_MODEL_ARCHIVE_LIST = [
    "distilbert-base-uncased",
    "distilbert-base-uncased-distilled-squad",
    "distilbert-base-cased",
    "distilbert-base-cased-distilled-squad",
    "distilbert-base-german-cased",
    "distilbert-base-multilingual-cased",
    "distilbert-base-uncased-finetuned-sst-2-english",
    # See all DistilBERT models at https://huggingface.co/models?
 ↪filter=distilbert
]


# UTILS AND BUILDING BLOCKS OF THE ARCHITECTURE #


def create_sinusoidal_embeddings(n_pos, dim, out):
    position_enc = np.array([[pos / np.power(10000, 2 * (j // 2) / dim) for j␣
 ↪in range(dim)] for pos in range(n_pos)])
    out.requires_grad = False
```

```
    out[:, 0::2] = torch.FloatTensor(np.sin(position_enc[:, 0::2]))
    out[:, 1::2] = torch.FloatTensor(np.cos(position_enc[:, 1::2]))
    out.detach_()


class Embeddings(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.word_embeddings = nn.Embedding(config.vocab_size, config.dim,␣
↪padding_idx=config.pad_token_id)
        self.position_embeddings = nn.Embedding(config.max_position_embeddings,␣
↪config.dim)
        if config.sinusoidal_pos_embds:

            if is_deepspeed_zero3_enabled():
                import deepspeed

                with deepspeed.zero.GatheredParameters(self.position_embeddings.
↪weight, modifier_rank=0):
                    if torch.distributed.get_rank() == 0:
                        create_sinusoidal_embeddings(
                            n_pos=config.max_position_embeddings, dim=config.
↪dim, out=self.position_embeddings.weight
                        )
            else:
                create_sinusoidal_embeddings(
                    n_pos=config.max_position_embeddings, dim=config.dim,␣
↪out=self.position_embeddings.weight
                )

        self.LayerNorm = nn.LayerNorm(config.dim, eps=1e-12)
        self.dropout = nn.Dropout(config.dropout)

    def forward(self, input_ids):
        """
        Parameters:
            input_ids: torch.tensor(bs, max_seq_length) The token ids to embed.

        Returns: torch.tensor(bs, max_seq_length, dim) The embedded tokens␣
↪(plus position embeddings, no token_type
        embeddings)
        """
        seq_length = input_ids.size(1)
        position_ids = torch.arange(seq_length, dtype=torch.long,␣
↪device=input_ids.device)  # (max_seq_length)
```

```
        position_ids = position_ids.unsqueeze(0).expand_as(input_ids)  # (bs,␣
↪max_seq_length)

        word_embeddings = self.word_embeddings(input_ids)  # (bs,␣
↪max_seq_length, dim)
        position_embeddings = self.position_embeddings(position_ids)  # (bs,␣
↪max_seq_length, dim)

        embeddings = word_embeddings + position_embeddings  # (bs,␣
↪max_seq_length, dim)
        embeddings = self.LayerNorm(embeddings)  # (bs, max_seq_length, dim)
        embeddings = self.dropout(embeddings)  # (bs, max_seq_length, dim)
        return embeddings


class MultiHeadSelfAttention(nn.Module):
    def __init__(self, config):
        super().__init__()

        self.n_heads = config.n_heads
        self.dim = config.dim
        self.dropout = nn.Dropout(p=config.attention_dropout)

        assert self.dim % self.n_heads == 0

        self.q_lin = nn.Linear(in_features=config.dim, out_features=config.dim)
        self.k_lin = nn.Linear(in_features=config.dim, out_features=config.dim)
        self.v_lin = nn.Linear(in_features=config.dim, out_features=config.dim)
        self.out_lin = nn.Linear(in_features=config.dim, out_features=config.
↪dim)

        self.pruned_heads = set()

    def prune_heads(self, heads):
        attention_head_size = self.dim // self.n_heads
        if len(heads) == 0:
            return
        heads, index = find_pruneable_heads_and_indices(heads, self.n_heads,␣
↪attention_head_size, self.pruned_heads)
        # Prune linear layers
        self.q_lin = prune_linear_layer(self.q_lin, index)
        self.k_lin = prune_linear_layer(self.k_lin, index)
        self.v_lin = prune_linear_layer(self.v_lin, index)
        self.out_lin = prune_linear_layer(self.out_lin, index, dim=1)
        # Update hyper params
        self.n_heads = self.n_heads - len(heads)
```

```
        self.dim = attention_head_size * self.n_heads
        self.pruned_heads = self.pruned_heads.union(heads)

    def forward(self, query, key, value, mask, head_mask=None,␣
↪output_attentions=False):
        """
        Parameters:
            query: torch.tensor(bs, seq_length, dim)
            key: torch.tensor(bs, seq_length, dim)
            value: torch.tensor(bs, seq_length, dim)
            mask: torch.tensor(bs, seq_length)

        Returns:
            weights: torch.tensor(bs, n_heads, seq_length, seq_length)␣
↪Attention weights context: torch.tensor(bs,
            seq_length, dim) Contextualized layer. Optional: only if␣
↪`output_attentions=True`
        """
        bs, q_length, dim = query.size()
        k_length = key.size(1)
        # assert dim == self.dim, f'Dimensions do not match: {dim} input vs␣
↪{self.dim} configured'
        # assert key.size() == value.size()

        dim_per_head = self.dim // self.n_heads

        mask_reshp = (bs, 1, 1, k_length)

        def shape(x):
            """separate heads"""
            return x.view(bs, -1, self.n_heads, dim_per_head).transpose(1, 2)

        def unshape(x):
            """group heads"""
            return x.transpose(1, 2).contiguous().view(bs, -1, self.n_heads *␣
↪dim_per_head)

        q = shape(self.q_lin(query))  # (bs, n_heads, q_length, dim_per_head)
        k = shape(self.k_lin(key))  # (bs, n_heads, k_length, dim_per_head)
        v = shape(self.v_lin(value))  # (bs, n_heads, k_length, dim_per_head)

        q = q / math.sqrt(dim_per_head)  # (bs, n_heads, q_length, dim_per_head)
        scores = torch.matmul(q, k.transpose(2, 3))  # (bs, n_heads, q_length,␣
↪k_length)
        mask = (mask == 0).view(mask_reshp).expand_as(scores)  # (bs, n_heads,␣
↪q_length, k_length)
```

```
        scores.masked_fill_(mask, -float("inf"))  # (bs, n_heads, q_length,␣
↪k_length)

        weights = nn.Softmax(dim=-1)(scores)  # (bs, n_heads, q_length,␣
↪k_length)
        weights = self.dropout(weights)  # (bs, n_heads, q_length, k_length)

        # Mask heads if we want to
        if head_mask is not None:
            weights = weights * head_mask

        context = torch.matmul(weights, v)  # (bs, n_heads, q_length,␣
↪dim_per_head)
        context = unshape(context)  # (bs, q_length, dim)
        context = self.out_lin(context)  # (bs, q_length, dim)

        if output_attentions:
            return (context, weights)
        else:
            return (context,)


class FFN(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.dropout = nn.Dropout(p=config.dropout)
        self.chunk_size_feed_forward = config.chunk_size_feed_forward
        self.seq_len_dim = 1
        self.lin1 = nn.Linear(in_features=config.dim, out_features=config.
↪hidden_dim)
        self.lin2 = nn.Linear(in_features=config.hidden_dim,␣
↪out_features=config.dim)
        assert config.activation in ["relu", "gelu"], f"activation ({config.
↪activation}) must be in ['relu', 'gelu']"
        self.activation = gelu if config.activation == "gelu" else nn.ReLU()

    def forward(self, input):
        return apply_chunking_to_forward(self.ff_chunk, self.
↪chunk_size_feed_forward, self.seq_len_dim, input)

    def ff_chunk(self, input):
        x = self.lin1(input)
        x = self.activation(x)
        x = self.lin2(x)
        x = self.dropout(x)
        return x
```

```python
class TransformerBlock(nn.Module):
    def __init__(self, config):
        super().__init__()

        assert config.dim % config.n_heads == 0

        self.attention = MultiHeadSelfAttention(config)
        self.sa_layer_norm = nn.LayerNorm(normalized_shape=config.dim,␣
↪eps=1e-12)

        self.ffn = FFN(config)
        self.output_layer_norm = nn.LayerNorm(normalized_shape=config.dim,␣
↪eps=1e-12)

    def forward(self, x, attn_mask=None, head_mask=None,␣
↪output_attentions=False):
        """
        Parameters:
            x: torch.tensor(bs, seq_length, dim)
            attn_mask: torch.tensor(bs, seq_length)

        Returns:
            sa_weights: torch.tensor(bs, n_heads, seq_length, seq_length) The␣
↪attention weights ffn_output:
            torch.tensor(bs, seq_length, dim) The output of the transformer␣
↪block contextualization.
        """
        # Self-Attention
        sa_output = self.attention(
            query=x,
            key=x,
            value=x,
            mask=attn_mask,
            head_mask=head_mask,
            output_attentions=output_attentions,
        )
        if output_attentions:
            sa_output, sa_weights = sa_output  # (bs, seq_length, dim), (bs,␣
↪n_heads, seq_length, seq_length)
        else:  # To handle these `output_attentions` or `output_hidden_states`␣
↪cases returning tuples
            assert type(sa_output) == tuple
            sa_output = sa_output[0]
        sa_output = self.sa_layer_norm(sa_output + x)  # (bs, seq_length, dim)
```

```
        # Feed Forward Network
        ffn_output = self.ffn(sa_output)  # (bs, seq_length, dim)
        ffn_output = self.output_layer_norm(ffn_output + sa_output)  # (bs,␣
↪seq_length, dim)

        output = (ffn_output,)
        if output_attentions:
            output = (sa_weights,) + output
        return output


class Transformer(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.n_layers = config.n_layers
        self.layer = nn.ModuleList([TransformerBlock(config) for _ in␣
↪range(config.n_layers)])

    def forward(
        self, x, attn_mask=None, head_mask=None, output_attentions=False,␣
↪output_hidden_states=False, return_dict=None
    ):  # docstyle-ignore
        """
        Parameters:
            x: torch.tensor(bs, seq_length, dim) Input sequence embedded.
            attn_mask: torch.tensor(bs, seq_length) Attention mask on the␣
↪sequence.

        Returns:
            hidden_state: torch.tensor(bs, seq_length, dim) Sequence of hidden␣
↪states in the last (top)
            layer all_hidden_states: Tuple[torch.tensor(bs, seq_length, dim)]
                Tuple of length n_layers with the hidden states from each layer.
                Optional: only if output_hidden_states=True
            all_attentions: Tuple[torch.tensor(bs, n_heads, seq_length,␣
↪seq_length)]
                Tuple of length n_layers with the attention weights from each␣
↪layer
                Optional: only if output_attentions=True
        """
        all_hidden_states = () if output_hidden_states else None
        all_attentions = () if output_attentions else None

        hidden_state = x
        for i, layer_module in enumerate(self.layer):
```

```
            if output_hidden_states:
                all_hidden_states = all_hidden_states + (hidden_state,)

            layer_outputs = layer_module(
                x=hidden_state, attn_mask=attn_mask, head_mask=head_mask[i],␣
↪output_attentions=output_attentions
            )
            hidden_state = layer_outputs[-1]

            if output_attentions:
                assert len(layer_outputs) == 2
                attentions = layer_outputs[0]
                all_attentions = all_attentions + (attentions,)
            else:
                assert len(layer_outputs) == 1

        # Add last layer
        if output_hidden_states:
            all_hidden_states = all_hidden_states + (hidden_state,)

        if not return_dict:
            return tuple(v for v in [hidden_state, all_hidden_states,␣
↪all_attentions] if v is not None)
        return BaseModelOutput(
            last_hidden_state=hidden_state, hidden_states=all_hidden_states,␣
↪attentions=all_attentions
        )


# INTERFACE FOR ENCODER AND TASK SPECIFIC MODEL #
class DistilBertPreTrainedModel(PreTrainedModel):
    """
    An abstract class to handle weights initialization and a simple interface␣
↪for downloading and loading pretrained
    models.
    """

    config_class = DistilBertConfig
    load_tf_weights = None
    base_model_prefix = "distilbert"

    def _init_weights(self, module):
        """Initialize the weights."""
        if isinstance(module, nn.Linear):
            # Slightly different from the TF version which uses␣
↪truncated_normal for initialization
            # cf https://github.com/pytorch/pytorch/pull/5617
```

```
            module.weight.data.normal_(mean=0.0, std=self.config.
 ↪initializer_range)
            if module.bias is not None:
                module.bias.data.zero_()
        elif isinstance(module, nn.Embedding):
            module.weight.data.normal_(mean=0.0, std=self.config.
 ↪initializer_range)
            if module.padding_idx is not None:
                module.weight.data[module.padding_idx].zero_()
        elif isinstance(module, nn.LayerNorm):
            module.bias.data.zero_()
            module.weight.data.fill_(1.0)


DISTILBERT_START_DOCSTRING = r"""

    This model inherits from :class:`~transformers.PreTrainedModel`. Check the␣
 ↪superclass documentation for the generic
    methods the library implements for all its model (such as downloading or␣
 ↪saving, resizing the input embeddings,
    pruning heads etc.)

    This model is also a PyTorch `torch.nn.Module <https://pytorch.org/docs/
 ↪stable/nn.html#torch.nn.Module>`__
    subclass. Use it as a regular PyTorch Module and refer to the PyTorch␣
 ↪documentation for all matter related to
    general usage and behavior.

    Parameters:
        config (:class:`~transformers.DistilBertConfig`): Model configuration␣
 ↪class with all the parameters of the model.
            Initializing with a config file does not load the weights␣
 ↪associated with the model, only the
            configuration. Check out the :meth:`~transformers.PreTrainedModel.
 ↪from_pretrained` method to load the model
            weights.
"""

DISTILBERT_INPUTS_DOCSTRING = r"""
    Args:
        input_ids (:obj:`torch.LongTensor` of shape :obj:`({0})`):
            Indices of input sequence tokens in the vocabulary.

            Indices can be obtained using :class:`~transformers.
 ↪DistilBertTokenizer`. See
```

```
            :meth:`transformers.PreTrainedTokenizer.encode` and :meth:
↪`transformers.PreTrainedTokenizer.__call__` for
            details.

            `What are input IDs? <../glossary.html#input-ids>`__
        attention_mask (:obj:`torch.FloatTensor` of shape :obj:`({0})`,␣
↪`optional`):
            Mask to avoid performing attention on padding token indices. Mask␣
↪values selected in ``[0, 1]``:

            - 1 for tokens that are **not masked**,
            - 0 for tokens that are **masked**.

            `What are attention masks? <../glossary.html#attention-mask>`__
        head_mask (:obj:`torch.FloatTensor` of shape :obj:`(num_heads,)` or :
↪obj:`(num_layers, num_heads)`, `optional`):
            Mask to nullify selected heads of the self-attention modules. Mask␣
↪values selected in ``[0, 1]``:

            - 1 indicates the head is **not masked**,
            - 0 indicates the head is **masked**.

        inputs_embeds (:obj:`torch.FloatTensor` of shape :obj:`({0},␣
↪hidden_size)`, `optional`):
            Optionally, instead of passing :obj:`input_ids` you can choose to␣
↪directly pass an embedded representation.
            This is useful if you want more control over how to convert :obj:
↪`input_ids` indices into associated
            vectors than the model's internal embedding lookup matrix.
        output_attentions (:obj:`bool`, `optional`):
            Whether or not to return the attentions tensors of all attention␣
↪layers. See ``attentions`` under returned
            tensors for more detail.
        output_hidden_states (:obj:`bool`, `optional`):
            Whether or not to return the hidden states of all layers. See␣
↪``hidden_states`` under returned tensors for
            more detail.
        return_dict (:obj:`bool`, `optional`):
            Whether or not to return a :class:`~transformers.file_utils.
↪ModelOutput` instead of a plain tuple.
"""



class DistilBertModel(DistilBertPreTrainedModel):
    def __init__(self, config):
```

```
        super().__init__(config)

        self.hidden_size = config.hidden_dim
        self.num_layers = 1
        self.num_directions = 2 #because we are implementing it in␣
↪bidirectional lstm mode

        self.embeddings = Embeddings(config)  # Embeddings
        self.transformer = Transformer(config)  # Encoder
        self.pointer = PointerNet(1,8,1,0.0,False)

        self.init_weights()

    def get_input_embeddings(self):
        return self.embeddings.word_embeddings

    def set_input_embeddings(self, new_embeddings):
        self.embeddings.word_embeddings = new_embeddings

    def _prune_heads(self, heads_to_prune):
        """
        Prunes heads of the model. heads_to_prune: dict of {layer_num: list of␣
↪heads to prune in this layer} See base
        class PreTrainedModel
        """
        for layer, heads in heads_to_prune.items():
            self.transformer.layer[layer].attention.prune_heads(heads)

    def forward(
        self,
        input_ids=None,
        attention_mask=None,
        head_mask=None,
        inputs_embeds=None,
        output_attentions=None,
        output_hidden_states=None,
        return_dict=None,
    ):
        output_attentions = output_attentions if output_attentions is not None␣
↪else self.config.output_attentions
        output_hidden_states = (
            output_hidden_states if output_hidden_states is not None else self.
↪config.output_hidden_states
        )
        return_dict = return_dict if return_dict is not None else self.config.
↪use_return_dict
```

```
        if input_ids is not None and inputs_embeds is not None:
            raise ValueError("You cannot specify both input_ids and␣
↪inputs_embeds at the same time")
        elif input_ids is not None:
            input_shape = input_ids.size()
        elif inputs_embeds is not None:
            input_shape = inputs_embeds.size()[:-1]
        else:
            raise ValueError("You have to specify either input_ids or␣
↪inputs_embeds")

        device = input_ids.device if input_ids is not None else inputs_embeds.
↪device

        if attention_mask is None:
            attention_mask = torch.ones(input_shape, device=device)  # (bs,␣
↪seq_length)

        # Prepare head mask if needed
        head_mask = self.get_head_mask(head_mask, self.config.num_hidden_layers)

        if inputs_embeds is None:
            inputs_embeds = self.embeddings(input_ids)  # (bs, seq_length, dim)


        inputs_embed, ots=self.pointer(inputs_embeds.type(torch.float))
        return self.transformer(
            x=inputs_embeds,
            attn_mask=attention_mask,
            head_mask=head_mask,
            output_attentions=output_attentions,
            output_hidden_states=output_hidden_states,
            return_dict=return_dict,
        )


class DistilBertForQuestionAnsweringC(DistilBertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)

        self.distilbert = DistilBertModel(config)
        self.qa_outputs = nn.Linear(config.dim, config.num_labels)
        assert config.num_labels == 2
        self.dropout = nn.Dropout(config.qa_dropout)

        self.init_weights()
```

```
    def forward(
        self,
        input_ids=None,
        attention_mask=None,
        head_mask=None,
        inputs_embeds=None,
        start_positions=None,
        end_positions=None,
        output_attentions=None,
        output_hidden_states=None,
        return_dict=None,
    ):
        r"""
        start_positions (:obj:`torch.LongTensor` of shape :obj:`(batch_size,)`,␣
↪`optional`):
            Labels for position (index) of the start of the labelled span for␣
↪computing the token classification loss.
            Positions are clamped to the length of the sequence (:obj:
↪`sequence_length`). Position outside of the
            sequence are not taken into account for computing the loss.
        end_positions (:obj:`torch.LongTensor` of shape :obj:`(batch_size,)`,␣
↪`optional`):
            Labels for position (index) of the end of the labelled span for␣
↪computing the token classification loss.
            Positions are clamped to the length of the sequence (:obj:
↪`sequence_length`). Position outside of the
            sequence are not taken into account for computing the loss.
        """
        return_dict = return_dict if return_dict is not None else self.config.
↪use_return_dict


        distilbert_output = self.distilbert(
            input_ids=input_ids,
            attention_mask=attention_mask,
            head_mask=head_mask,
            inputs_embeds=inputs_embeds,
            output_attentions=output_attentions,
            output_hidden_states=output_hidden_states,
            return_dict=return_dict,
        )


        hidden_states = distilbert_output[0]  # (bs, max_query_len, dim)

        hidden_states = self.dropout(hidden_states)  # (bs, max_query_len, dim)
```

```
        logits = self.qa_outputs(hidden_states)  # (bs, max_query_len, 2)
        start_logits, end_logits = logits.split(1, dim=-1)
        start_logits = start_logits.squeeze(-1).contiguous()  # (bs,␣
↪max_query_len)
        end_logits = end_logits.squeeze(-1).contiguous()  # (bs, max_query_len)

        total_loss = None
        if start_positions is not None and end_positions is not None:
            # If we are on multi-GPU, split add a dimension
            if len(start_positions.size()) > 1:
                start_positions = start_positions.squeeze(-1)
            if len(end_positions.size()) > 1:
                end_positions = end_positions.squeeze(-1)
            # sometimes the start/end positions are outside our model inputs,␣
↪we ignore these terms
            ignored_index = start_logits.size(1)
            start_positions = start_positions.clamp(0, ignored_index)
            end_positions = end_positions.clamp(0, ignored_index)

            loss_fct = nn.CrossEntropyLoss(ignore_index=ignored_index)
            start_loss = loss_fct(start_logits, start_positions)
            end_loss = loss_fct(end_logits, end_positions)
            total_loss = (start_loss + end_loss) / 2

        if not return_dict:
            output = (start_logits, end_logits) + distilbert_output[1:]
            return ((total_loss,) + output) if total_loss is not None else␣
↪output

        return QuestionAnsweringModelOutput(
            loss=total_loss,
            start_logits=start_logits,
            end_logits=end_logits,
            hidden_states=distilbert_output.hidden_states,
            attentions=distilbert_output.attentions,
        )
```

[24]:
```
## 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)
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=231508.0,␣
↪style=ProgressStyle(descripti…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=466062.0,␣
↪style=ProgressStyle(descripti…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=28.0,␣
↪style=ProgressStyle(description_w…
```

observation

[25]:
```
## 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)
```

5 3. Train & Validation Dataset Creation

[26]:
```
## 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)
```

5.0.1 Observations

6 4. Model Building & Training

[27]:
```
model =DistilBertForQuestionAnsweringC.
 ↪from_pretrained('distilbert-base-uncased')
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=442.0,␣
 ↪style=ProgressStyle(description_…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=267967963.
 ↪0, style=ProgressStyle(descri…
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:99: UserWarning:
nn.init.uniform is now deprecated in favor of nn.init.uniform_.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:284: UserWarning:
nn.init.uniform is now deprecated in favor of nn.init.uniform_.
Some weights of the model checkpoint at distilbert-base-uncased were not used
when initializing DistilBertForQuestionAnsweringC: ['vocab_transform.weight',
'vocab_layer_norm.weight', 'vocab_transform.bias', 'vocab_layer_norm.bias',
'vocab_projector.bias', 'vocab_projector.weight']
- This IS expected if you are initializing DistilBertForQuestionAnsweringC 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 DistilBertForQuestionAnsweringC
from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
Some weights of DistilBertForQuestionAnsweringC were not initialized from the
model checkpoint at distilbert-base-uncased and are newly initialized:
['distilbert.pointer.decoder_input0',
'distilbert.pointer.decoder.att.input_linear.bias',
'distilbert.pointer.encoder.h0', 'distilbert.pointer.decoder.att.V',
'distilbert.pointer.decoder.hidden_to_hidden.weight',
'distilbert.pointer.decoder.att._inf',
'distilbert.pointer.decoder.hidden_out.weight',
'distilbert.pointer.decoder.input_to_hidden.bias',
```

```
'distilbert.pointer.encoder.lstm.weight_hh_l0',
'distilbert.pointer.encoder.lstm.bias_hh_l0',
'distilbert.pointer.decoder.hidden_out.bias',
'distilbert.pointer.embedding.weight', 'qa_outputs.weight',
'distilbert.pointer.decoder.input_to_hidden.weight',
'distilbert.pointer.decoder.runner', 'qa_outputs.bias',
'distilbert.pointer.decoder.att.context_linear.weight',
'distilbert.pointer.decoder.att.context_linear.bias',
'distilbert.pointer.encoder.lstm.bias_ih_l0',
'distilbert.pointer.decoder.att.input_linear.weight',
'distilbert.pointer.encoder.c0',
'distilbert.pointer.decoder.hidden_to_hidden.bias',
'distilbert.pointer.encoder.lstm.weight_ih_l0',
'distilbert.pointer.decoder.mask', 'distilbert.pointer.embedding.bias']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

6.0.1 Observations

[28]:
```
# 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 = Adafactor(model.parameters(),scale_parameter=False,␣
 ↪relative_step=False, lr=3e-5)
```

6.0.2 Observations

[29]:
```
torch.cuda.empty_cache()
```

6.0.3 Observations

[30]:
```
# Train for the model, perform validation on it per epoch and generate files␣
 ↪for a tensorboard
num_epochs = 2

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 0/1
----------
```

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:127: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:240: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

```
Training Loss: 11.1297
```

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

```
Val Loss: 2.0633, EM: 0.6457, F1: 0.7424
Epoch 1/1
----------
```

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

```
Training Loss: 6.9106
```

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

```
Val Loss: 1.2249, EM: 0.6577, F1: 0.7556
```

6.0.4 Observations

[31]:
```
# We save our model so that it can be reused later

torch.save(model,'./customBertmodelAdafactor3e.pt')
```

[32]:
```
# Generate a Tensorboard

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

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

6.0.5 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.

7 5. Running The Model

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

[33]:
```
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"
```

[34]:
```
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
```

[35]:
```
question_answer(test_question, test_context, model)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:127: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:240: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[35]:
```
'rollo'
```

[36]:
```
question_answer(val_questions[0], val_contexts[0], model)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:127: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:240: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[36]:
```
'france'
```

8 Running The Model

[37]:
```
model_loaded = torch.load('./customBertmodelAdafactor3e.pt')
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

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"
```

[38]:
```
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
```

[39]:
```
question_answer(test_question, test_context, model_loaded)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:127: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:240: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[39]:
```
'rollo'
```

[40]:
```
## 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"""
```

[41]:
```
question_answer(question, context, model_loaded)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:127: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:240: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
```

```
    /pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[41]: `'exploding gradient problem'`

Custom_Bert_Model_AdamW_5e

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

[1]:
```
!pip install torch
!pip install transformers
!pip install sentencepiece
!pip install wandb
!pip install allennlp
```

```
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 14.0 MB/s
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.7/dist-packages (from transformers) (1.19.5)
Collecting huggingface-hub==0.0.12
  Downloading huggingface_hub-0.0.12-py3-none-any.whl (37 kB)
Collecting pyyaml>=5.1
  Downloading PyYAML-5.4.1-cp37-cp37m-manylinux1_x86_64.whl (636 kB)
     |                | 636 kB 76.8 MB/s
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from transformers) (21.0)
Collecting sacremoses
  Downloading sacremoses-0.0.45-py3-none-any.whl (895 kB)
     |                | 895 kB 87.5 MB/s
Requirement already satisfied: importlib-metadata in
```

```
/usr/local/lib/python3.7/dist-packages (from transformers) (4.6.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from transformers) (2.23.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-
packages (from transformers) (4.41.1)
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 72.5 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from transformers) (3.0.12)
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: 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: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
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: 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)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (1.15.0)
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 13.6 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 12.5 MB/s
Collecting configparser>=3.8.1
  Downloading configparser-5.0.2-py3-none-any.whl (19 kB)
Requirement already satisfied: six>=1.13.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (1.15.0)
Collecting docker-pycreds>=0.4.0
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
Requirement already satisfied: promise<3,>=2.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (2.3)
Requirement already satisfied: python-dateutil>=2.6.1 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.8.1)
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: requests<3,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (2.23.0)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.7/dist-
packages (from wandb) (5.4.8)
Collecting pathtools
  Downloading pathtools-0.1.2.tar.gz (11 kB)
Collecting urllib3>=1.26.5
  Downloading urllib3-1.26.6-py2.py3-none-any.whl (138 kB)
     |                 | 138 kB 85.2 MB/s
Collecting subprocess32>=3.5.3
  Downloading subprocess32-3.5.4.tar.gz (97 kB)
     |                 | 97 kB 7.6 MB/s
Collecting shortuuid>=0.5.0
  Downloading shortuuid-1.0.1-py3-none-any.whl (7.5 kB)
Collecting sentry-sdk>=1.0.0
  Downloading sentry_sdk-1.3.1-py2.py3-none-any.whl (133 kB)
     |                 | 133 kB 79.9 MB/s
Requirement already satisfied: protobuf>=3.12.0 in
/usr/local/lib/python3.7/dist-packages (from wandb) (3.17.3)
Collecting GitPython>=1.0.0
  Downloading GitPython-3.1.18-py3-none-any.whl (170 kB)
     |                 | 170 kB 77.0 MB/s
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-
packages (from wandb) (5.4.1)
Collecting gitdb<5,>=4.0.1
  Downloading gitdb-4.0.7-py3-none-any.whl (63 kB)
     |                 | 63 kB 2.3 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: 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 746 kB/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=9ab75fdd6fcc3e6ed9b69e28bfaad3e92f4c38a6488bfcfb279ecfe1f6bd0e31
  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=e2f8826b6bb483b9c6e0c2e24e3091618064548dad67e3017de07c5cf4d15705
  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
Collecting allennlp
  Downloading allennlp-2.6.0-py3-none-any.whl (689 kB)
     |                  | 689 kB 14.2 MB/s
```

```
Requirement already satisfied: torch<1.10.0,>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (1.9.0+cu102)
Requirement already satisfied: pytest in /usr/local/lib/python3.7/dist-packages
(from allennlp) (3.6.4)
Requirement already satisfied: spacy<3.1,>=2.1.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (2.2.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from allennlp) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from allennlp) (1.19.5)
Requirement already satisfied: huggingface-hub>=0.0.8 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.0.12)
Collecting boto3<2.0,>=1.14
  Downloading boto3-1.18.16-py3-none-any.whl (131 kB)
     |                  | 131 kB 25.7 MB/s
Requirement already satisfied: requests>=2.18 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (2.26.0)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-
packages (from allennlp) (8.8.0)
Collecting tensorboardX>=1.2
  Downloading tensorboardX-2.4-py2.py3-none-any.whl (124 kB)
     |                  | 124 kB 32.4 MB/s
Collecting transformers<4.9,>=4.1
  Downloading transformers-4.8.2-py3-none-any.whl (2.5 MB)
     |                  | 2.5 MB 33.1 MB/s
Requirement already satisfied: termcolor==1.1.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (1.1.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages
(from allennlp) (3.2.5)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages
(from allennlp) (3.1.0)
Requirement already satisfied: filelock<3.1,>=3.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (3.0.12)
Requirement already satisfied: wandb<0.12.0,>=0.10.0 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.11.2)
Requirement already satisfied: lmdb in /usr/local/lib/python3.7/dist-packages
(from allennlp) (0.99)
Requirement already satisfied: torchvision<0.11.0,>=0.8.1 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.10.0+cu102)
Collecting overrides==3.1.0
  Downloading overrides-3.1.0.tar.gz (11 kB)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-
packages (from allennlp) (0.22.2.post1)
Collecting jsonnet>=0.10.0
  Downloading jsonnet-0.17.0.tar.gz (259 kB)
     |                  | 259 kB 52.8 MB/s
Collecting google-cloud-storage<1.42.0,>=1.38.0
  Downloading google_cloud_storage-1.41.1-py2.py3-none-any.whl (105 kB)
```

```
     |                | 105 kB 66.6 MB/s
Requirement already satisfied: tqdm>=4.19 in
/usr/local/lib/python3.7/dist-packages (from allennlp) (4.41.1)
Collecting checklist==0.0.11
  Downloading checklist-0.0.11.tar.gz (12.1 MB)
     |                | 12.1 MB 44.0 MB/s
Requirement already satisfied: sentencepiece in
/usr/local/lib/python3.7/dist-packages (from allennlp) (0.1.96)
Collecting munch>=2.5
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: dill>=0.3.1 in /usr/local/lib/python3.7/dist-
packages (from checklist==0.0.11->allennlp) (0.3.4)
Requirement already satisfied: jupyter>=1.0 in /usr/local/lib/python3.7/dist-
packages (from checklist==0.0.11->allennlp) (1.0.0)
Requirement already satisfied: ipywidgets>=7.5 in /usr/local/lib/python3.7/dist-
packages (from checklist==0.0.11->allennlp) (7.6.3)
Collecting patternfork-nosql
  Downloading patternfork_nosql-3.6.tar.gz (22.3 MB)
     |                | 22.3 MB 1.3 MB/s
Collecting iso-639
  Downloading iso-639-0.4.5.tar.gz (167 kB)
     |                | 167 kB 67.4 MB/s
Collecting botocore<1.22.0,>=1.21.16
  Downloading botocore-1.21.16-py3-none-any.whl (7.8 MB)
     |                | 7.8 MB 40.8 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting s3transfer<0.6.0,>=0.5.0
  Downloading s3transfer-0.5.0-py3-none-any.whl (79 kB)
     |                | 79 kB 10.1 MB/s
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/usr/local/lib/python3.7/dist-packages (from
botocore<1.22.0,>=1.21.16->boto3<2.0,>=1.14->allennlp) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in
/usr/local/lib/python3.7/dist-packages (from
botocore<1.22.0,>=1.21.16->boto3<2.0,>=1.14->allennlp) (1.26.6)
Collecting google-resumable-media<3.0dev,>=1.3.0
  Downloading google_resumable_media-1.3.3-py2.py3-none-any.whl (75 kB)
     |                | 75 kB 6.2 MB/s
Collecting google-cloud-core<3.0dev,>=1.6.0
  Downloading google_cloud_core-1.7.2-py2.py3-none-any.whl (28 kB)
Requirement already satisfied: google-auth<3.0dev,>=1.24.0 in
/usr/local/lib/python3.7/dist-packages (from google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (1.32.1)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-
packages (from google-auth<3.0dev,>=1.24.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
```

```
/usr/local/lib/python3.7/dist-packages (from google-
auth<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (4.2.2)
Requirement already satisfied: setuptools>=40.3.0 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (57.2.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-
packages (from google-auth<3.0dev,>=1.24.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (1.15.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (0.2.8)
Requirement already satisfied: google-api-core<2.0.0dev,>=1.21.0 in
/usr/local/lib/python3.7/dist-packages (from google-cloud-
core<3.0dev,>=1.6.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (1.26.3)
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages
(from google-api-core<2.0.0dev,>=1.21.0->google-cloud-
core<3.0dev,>=1.6.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (2018.9)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-
core<2.0.0dev,>=1.21.0->google-cloud-core<3.0dev,>=1.6.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (1.53.0)
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.7/dist-
packages (from google-api-core<2.0.0dev,>=1.21.0->google-cloud-
core<3.0dev,>=1.6.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (21.0)
Requirement already satisfied: protobuf>=3.12.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-
core<2.0.0dev,>=1.21.0->google-cloud-core<3.0dev,>=1.6.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (3.17.3)
Collecting google-crc32c<2.0dev,>=1.0
  Downloading google_crc32c-1.1.2-cp37-cp37m-manylinux2014_x86_64.whl (38 kB)
Requirement already satisfied: cffi>=1.0.0 in /usr/local/lib/python3.7/dist-
packages (from google-crc32c<2.0dev,>=1.0->google-resumable-
media<3.0dev,>=1.3.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (1.14.6)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-
packages (from cffi>=1.0.0->google-crc32c<2.0dev,>=1.0->google-resumable-
media<3.0dev,>=1.3.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (2.20)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from huggingface-hub>=0.0.8->allennlp)
(4.6.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from huggingface-hub>=0.0.8->allennlp)
(3.7.4.3)
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.7/dist-
packages (from ipywidgets>=7.5->checklist==0.0.11->allennlp) (5.5.0)
Requirement already satisfied: widgetsnbextension~=3.5.0 in
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (3.5.1)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in
```

```
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (1.0.0)
Requirement already satisfied: traitlets>=4.3.1 in
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (5.0.5)
Requirement already satisfied: ipykernel>=4.5.1 in
/usr/local/lib/python3.7/dist-packages (from
ipywidgets>=7.5->checklist==0.0.11->allennlp) (4.10.1)
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.7/dist-
packages (from ipywidgets>=7.5->checklist==0.0.11->allennlp) (5.1.3)
Requirement already satisfied: tornado>=4.0 in /usr/local/lib/python3.7/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(5.1.1)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.7/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(5.3.5)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(2.6.1)
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages
(from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (4.8.0)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(4.4.2)
Requirement already satisfied: simplegeneric>0.8 in
/usr/local/lib/python3.7/dist-packages (from
ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (0.8.1)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(0.7.5)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in
/usr/local/lib/python3.7/dist-packages (from
ipython>=4.0.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (1.0.18)
Requirement already satisfied: qtconsole in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.1.1)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.2.0)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.6.1)
Requirement already satisfied: notebook in /usr/local/lib/python3.7/dist-
packages (from jupyter>=1.0->checklist==0.0.11->allennlp) (5.3.1)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/usr/local/lib/python3.7/dist-packages (from
nbformat>=4.2.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (2.6.0)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-
packages (from nbformat>=4.2.0->ipywidgets>=7.5->checklist==0.0.11->allennlp)
(4.7.1)
Requirement already satisfied: ipython-genutils in
```

```
/usr/local/lib/python3.7/dist-packages (from
nbformat>=4.2.0->ipywidgets>=7.5->checklist==0.0.11->allennlp) (0.2.0)
Requirement already satisfied: pyparsing>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging>=14.3->google-api-
core<2.0.0dev,>=1.21.0->google-cloud-core<3.0dev,>=1.6.0->google-cloud-
storage<1.42.0,>=1.38.0->allennlp) (2.4.7)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages
(from prompt-toolkit<2.0.0,>=1.0.4->ipython>=4.0.0->ipywidgets>=7.5->checklist==
0.0.11->allennlp) (0.2.5)
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<3.0dev,>=1.24.0->google-cloud-storage<1.42.0,>=1.38.0->allennlp) (0.4.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.18->allennlp) (2.0.2)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.18->allennlp)
(2021.5.30)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests>=2.18->allennlp) (2.10)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(0.4.1)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(0.8.2)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.0.0)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.0.5)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.1.3)
Requirement already satisfied: thinc==7.4.0 in /usr/local/lib/python3.7/dist-
packages (from spacy<3.1,>=2.1.0->allennlp) (7.4.0)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(3.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(1.0.5)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from spacy<3.1,>=2.1.0->allennlp)
(2.0.5)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->huggingface-hub>=0.0.8->allennlp) (3.5.0)
Requirement already satisfied: pillow>=5.3.0 in /usr/local/lib/python3.7/dist-
```

```
packages (from torchvision<0.11.0,>=0.8.1->allennlp) (7.1.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages
(from transformers<4.9,>=4.1->allennlp) (5.4.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers<4.9,>=4.1->allennlp)
(2019.12.20)
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in
/usr/local/lib/python3.7/dist-packages (from transformers<4.9,>=4.1->allennlp)
(0.10.3)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/dist-
packages (from transformers<4.9,>=4.1->allennlp) (0.0.45)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.7/dist-
packages (from wandb<0.12.0,>=0.10.0->allennlp) (5.4.8)
Requirement already satisfied: shortuuid>=0.5.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(1.0.1)
Requirement already satisfied: GitPython>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(3.1.18)
Requirement already satisfied: sentry-sdk>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(1.3.1)
Requirement already satisfied: Click!=8.0.0,>=7.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(7.1.2)
Requirement already satisfied: docker-pycreds>=0.4.0 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(0.4.0)
Requirement already satisfied: subprocess32>=3.5.3 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(3.5.4)
Requirement already satisfied: pathtools in /usr/local/lib/python3.7/dist-
packages (from wandb<0.12.0,>=0.10.0->allennlp) (0.1.2)
Requirement already satisfied: configparser>=3.8.1 in
/usr/local/lib/python3.7/dist-packages (from wandb<0.12.0,>=0.10.0->allennlp)
(5.0.2)
Requirement already satisfied: promise<3,>=2.0 in /usr/local/lib/python3.7/dist-
packages (from wandb<0.12.0,>=0.10.0->allennlp) (2.3)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.7/dist-
packages (from GitPython>=1.0.0->wandb<0.12.0,>=0.10.0->allennlp) (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<0.12.0,>=0.10.0->allennlp) (4.0.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages
(from notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (2.11.3)
Requirement already satisfied: terminado>=0.8.1 in
/usr/local/lib/python3.7/dist-packages (from
notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (0.10.1)
```

```
Requirement already satisfied: Send2Trash in /usr/local/lib/python3.7/dist-
packages (from notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (1.7.1)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.7/dist-
packages (from jupyter-
client->ipykernel>=4.5.1->ipywidgets>=7.5->checklist==0.0.11->allennlp) (22.1.0)
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.7/dist-
packages (from
terminado>=0.8.1->notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (0.7.0)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-
packages (from h5py->allennlp) (1.5.2)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.7/dist-packages (from
jinja2->notebook->jupyter>=1.0->checklist==0.0.11->allennlp) (2.0.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.7/dist-packages (from
nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.3)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.7/dist-packages (from
nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (1.4.3)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.7/dist-packages (from
nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.8.4)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.7/dist-
packages (from nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.7.1)
Requirement already satisfied: testpath in /usr/local/lib/python3.7/dist-
packages (from nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (0.5.0)
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages
(from nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp) (3.3.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-
packages (from bleach->nbconvert->jupyter>=1.0->checklist==0.0.11->allennlp)
(0.5.1)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages
(from patternfork-nosql->checklist==0.0.11->allennlp) (0.16.0)
Collecting backports.csv
  Downloading backports.csv-1.0.7-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-
packages (from patternfork-nosql->checklist==0.0.11->allennlp) (4.6.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages
(from patternfork-nosql->checklist==0.0.11->allennlp) (4.2.6)
Collecting feedparser
  Downloading feedparser-6.0.8-py3-none-any.whl (81 kB)
     |                | 81 kB 12.4 MB/s
Collecting pdfminer.six
  Downloading pdfminer.six-20201018-py3-none-any.whl (5.6 MB)
     |                | 5.6 MB 57.1 MB/s
Collecting python-docx
  Downloading python-docx-0.8.11.tar.gz (5.6 MB)
     |                | 5.6 MB 18.6 MB/s
```

```
Collecting cherrypy
  Downloading CherryPy-18.6.1-py2.py3-none-any.whl (419 kB)
     |                  | 419 kB 66.2 MB/s
Collecting zc.lockfile
  Downloading zc.lockfile-2.0-py2.py3-none-any.whl (9.7 kB)
Collecting cheroot>=8.2.1
  Downloading cheroot-8.5.2-py2.py3-none-any.whl (97 kB)
     |                  | 97 kB 8.0 MB/s
Collecting jaraco.collections
  Downloading jaraco.collections-3.3.0-py3-none-any.whl (9.9 kB)
Collecting portend>=2.1.1
  Downloading portend-2.7.1-py3-none-any.whl (5.3 kB)
Collecting jaraco.functools
  Downloading jaraco.functools-3.3.0-py3-none-any.whl (6.8 kB)
Collecting tempora>=1.8
  Downloading tempora-4.1.1-py3-none-any.whl (15 kB)
Collecting sgmllib3k
  Downloading sgmllib3k-1.0.0.tar.gz (5.8 kB)
Collecting jaraco.text
  Downloading jaraco.text-3.5.1-py3-none-any.whl (8.1 kB)
Collecting jaraco.classes
  Downloading jaraco.classes-3.2.1-py3-none-any.whl (5.6 kB)
Requirement already satisfied: sortedcontainers in
/usr/local/lib/python3.7/dist-packages (from pdfminer.six->patternfork-
nosql->checklist==0.0.11->allennlp) (2.4.0)
Requirement already satisfied: chardet in /usr/local/lib/python3.7/dist-packages
(from pdfminer.six->patternfork-nosql->checklist==0.0.11->allennlp) (3.0.4)
Collecting cryptography
  Downloading cryptography-3.4.7-cp36-abi3-manylinux2014_x86_64.whl (3.2 MB)
     |                  | 3.2 MB 43.7 MB/s
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.7/dist-
packages (from pytest->allennlp) (1.10.0)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/dist-
packages (from pytest->allennlp) (21.2.0)
Requirement already satisfied: pluggy<0.8,>=0.5 in
/usr/local/lib/python3.7/dist-packages (from pytest->allennlp) (0.7.1)
Requirement already satisfied: atomicwrites>=1.0 in
/usr/local/lib/python3.7/dist-packages (from pytest->allennlp) (1.4.0)
Requirement already satisfied: qtpy in /usr/local/lib/python3.7/dist-packages
(from qtconsole->jupyter>=1.0->checklist==0.0.11->allennlp) (1.9.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers<4.9,>=4.1->allennlp) (1.0.1)
Building wheels for collected packages: checklist, overrides, jsonnet, iso-639,
patternfork-nosql, python-docx, sgmllib3k
  Building wheel for checklist (setup.py) … done
  Created wheel for checklist: filename=checklist-0.0.11-py3-none-any.whl
size=12165634
sha256=80e83845cfb61d75f92499c768afe0a6f58539e4e8b3660fbb3ded8943210243
```

```
  Stored in directory: /root/.cache/pip/wheels/6a/8a/07/6446879be434879c27671c83
443727d74cecf6b630c8a24d03
  Building wheel for overrides (setup.py) … done
  Created wheel for overrides: filename=overrides-3.1.0-py3-none-any.whl
size=10188
sha256=d7d55543a00d1868f974b922ef7aaf44cbbedba5aa13db38c2ebafc0f5acd06d
  Stored in directory: /root/.cache/pip/wheels/3a/0d/38/01a9bc6e20dcfaf0a6a7b552
d03137558ba1c38aea47644682
  Building wheel for jsonnet (setup.py) … done
  Created wheel for jsonnet: filename=jsonnet-0.17.0-cp37-cp37m-linux_x86_64.whl
size=3388684
sha256=eec2a97041ae5341ebb2fcb43c772dff3ebdea9d694eb9cbe01a1aee4fe41dde
  Stored in directory: /root/.cache/pip/wheels/1c/28/7e/287c6b19f7161bb03c6986a3
c46b51d0d7d9a1805346634e3a
  Building wheel for iso-639 (setup.py) … done
  Created wheel for iso-639: filename=iso_639-0.4.5-py3-none-any.whl size=169062
sha256=11f7a6b046ea7cdeec6f243cc2e84f33e54b10636db09eca4b0c3f38d364b290
  Stored in directory: /root/.cache/pip/wheels/47/60/19/6d020fc92138ed1b113a1827
1e83ea4b5525fe770cb45b9a2e
  Building wheel for patternfork-nosql (setup.py) … done
  Created wheel for patternfork-nosql: filename=patternfork_nosql-3.6-py3-none-
any.whl size=22332805
sha256=a368093aba218492fa7aa3edc61815bb405f731cee3e123f506062e004dc6820
  Stored in directory: /root/.cache/pip/wheels/97/72/8f/5305fe28168f93b658da9ed4
33b9a1d3ec90594faa0c9aaf4b
  Building wheel for python-docx (setup.py) … done
  Created wheel for python-docx: filename=python_docx-0.8.11-py3-none-any.whl
size=184507
sha256=15e7ea363995f375b03e19810a06133acf4313873963a1a35d781e2d756a719b
  Stored in directory: /root/.cache/pip/wheels/f6/6f/b9/d798122a8b55b74ad30b5f52
b01482169b445fbb84a11797a6
  Building wheel for sgmllib3k (setup.py) … done
  Created wheel for sgmllib3k: filename=sgmllib3k-1.0.0-py3-none-any.whl
size=6065
sha256=165d114c48385bffc85121f5a2ebb6d402c492083decd77bee5729fc4331523e
  Stored in directory: /root/.cache/pip/wheels/73/ad/a4/0dff4a6ef231fc0dfa12ffba
c2a36cebfdddfe059f50e019aa
Successfully built checklist overrides jsonnet iso-639 patternfork-nosql python-
docx sgmllib3k
Installing collected packages: jaraco.functools, tempora, jaraco.text,
jaraco.classes, zc.lockfile, sgmllib3k, portend, jmespath, jaraco.collections,
cryptography, cheroot, python-docx, pdfminer.six, google-crc32c, feedparser,
cherrypy, botocore, backports.csv, transformers, s3transfer, patternfork-nosql,
munch, iso-639, google-resumable-media, google-cloud-core, tensorboardX,
overrides, jsonnet, google-cloud-storage, checklist, boto3, allennlp
  Attempting uninstall: transformers
    Found existing installation: transformers 4.9.1
    Uninstalling transformers-4.9.1:
```

```
    Successfully uninstalled transformers-4.9.1
  Attempting uninstall: google-resumable-media
    Found existing installation: google-resumable-media 0.4.1
    Uninstalling google-resumable-media-0.4.1:
      Successfully uninstalled google-resumable-media-0.4.1
  Attempting uninstall: google-cloud-core
    Found existing installation: google-cloud-core 1.0.3
    Uninstalling google-cloud-core-1.0.3:
      Successfully uninstalled google-cloud-core-1.0.3
  Attempting uninstall: google-cloud-storage
    Found existing installation: google-cloud-storage 1.18.1
    Uninstalling google-cloud-storage-1.18.1:
      Successfully uninstalled google-cloud-storage-1.18.1
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-cloud-bigquery 1.21.0 requires google-resumable-
media!=0.4.0,<0.5.0dev,>=0.3.1, but you have google-resumable-media 1.3.3 which
is incompatible.
Successfully installed allennlp-2.6.0 backports.csv-1.0.7 boto3-1.18.16
botocore-1.21.16 checklist-0.0.11 cheroot-8.5.2 cherrypy-18.6.1
cryptography-3.4.7 feedparser-6.0.8 google-cloud-core-1.7.2 google-cloud-
storage-1.41.1 google-crc32c-1.1.2 google-resumable-media-1.3.3 iso-639-0.4.5
jaraco.classes-3.2.1 jaraco.collections-3.3.0 jaraco.functools-3.3.0
jaraco.text-3.5.1 jmespath-0.10.0 jsonnet-0.17.0 munch-2.5.0 overrides-3.1.0
patternfork-nosql-3.6 pdfminer.six-20201018 portend-2.7.1 python-docx-0.8.11
s3transfer-0.5.0 sgmllib3k-1.0.0 tempora-4.1.1 tensorboardX-2.4
transformers-4.8.2 zc.lockfile-2.0
```

[2]:
```python
import torch
import transformers as tfs
import numpy as np
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 BertTokenizerFast,BertPreTrainedModel, BertConfig,␣
↪BertModel, BertGenerationEncoder
from transformers import AdamW, DistilBertTokenizerFast, DistilBertConfig
from transformers.modeling_outputs import (
    BaseModelOutput,
    BaseModelOutputWithPastAndCrossAttentions,
    BaseModelOutputWithPoolingAndCrossAttentions,
    CausalLMOutputWithCrossAttentions,
```

```
    MaskedLMOutput,
    MultipleChoiceModelOutput,
    NextSentencePredictorOutput,
    QuestionAnsweringModelOutput,
    SequenceClassifierOutput,
    TokenClassifierOutput,
)
import string, re
import torch.nn as nn
from allennlp.nn.util import masked_log_softmax, masked_max

import math

from transformers.activations import gelu
from transformers.deepspeed import is_deepspeed_zero3_enabled
from transformers.file_utils import (
    add_code_sample_docstrings,
    add_start_docstrings,
    add_start_docstrings_to_model_forward,
    replace_return_docstrings,
)

from transformers.modeling_utils import (
    PreTrainedModel,
    apply_chunking_to_forward,
    find_pruneable_heads_and_indices,
    prune_linear_layer,
)
from transformers.utils import logging
import torch.nn.functional as F
```

[3]:
```
torch.cuda.is_available()
```

[3]: True

1 Utility functions for Metrics evaluation

[4]:
```
# 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))))
```

2 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.

2.1 Download SQuAD 2.0 Data

[5]:
```
# 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))
```

[6]:
```
# 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)
```

[7]:
```
# 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
```

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").

[8]:
```
# ## 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
```

```
--2021-08-08 10:59:57--  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   129MB/s    in 0.3s
```

```
2021-08-08 10:59:58 (129 MB/s) - ‘squad/train-v2.0.json’ saved
[42123633/42123633]

--2021-08-08 10:59:59--  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 0.03s

2021-08-08 10:59:59 (140 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.

[9]:
```
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.

[10]:
```
len(train_contexts)
```

[10]:
```
86821
```

[11]:
```
train_contexts[0]
```

[11]:
```
'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".'
```

[12]:
```
train_questions[0]
```

[12]:
```
'When did Beyonce start becoming popular?'
```

[13]:
```
train_answers[0]
```

[13]:
```
{'answer_start': 269, 'text': 'in the late 1990s'}
```

[14]:
```
len(train_qac)
```

[14]:
```
86821
```

[15]:
```
train_qac[0]
```

[15]:
```
{'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

[16]:
```
len(val_contexts)
```

[16]:
```
20302
```

[17]:
```
val_contexts[0]
```

[17]:
```
'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.'
```

[18]:
```
val_questions[0]
```

[18]:
```
'In what country is Normandy located?'
```

[19]:
```
val_answers[0]
```

[19]:
```
{'answer_start': 159, 'text': 'France'}
```

[20]:
```
val_qac[0]
```

[20]:
```
{'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?'}
```

2.2 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

3 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.

[21]:
```
## 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)
```

4 Creating Custom BERT Model

[22]:
```
import torch
import torch.nn as nn
from torch.nn import Parameter



class Encoder(nn.Module):
    """
    Encoder class for Pointer-Net
    """

    def __init__(self, embedding_dim,
                 hidden_dim,
                 n_layers,
                 dropout,
                 bidir):
        """
        Initiate Encoder

        :param Tensor embedding_dim: Number of embbeding channels
        :param int hidden_dim: Number of hidden units for the LSTM
        :param int n_layers: Number of layers for LSTMs
        :param float dropout: Float between 0-1
        :param bool bidir: Bidirectional
        """

        super(Encoder, self).__init__()
        self.hidden_dim = hidden_dim//2 if bidir else hidden_dim
        self.n_layers = n_layers*2 if bidir else n_layers
        self.bidir = bidir
        self.lstm = nn.LSTM(embedding_dim,
                            self.hidden_dim,
                            n_layers,
                            dropout=dropout,
                            bidirectional=bidir)

        # Used for propagating .cuda() command
        self.h0 = Parameter(torch.zeros(1), requires_grad=False)
        self.c0 = Parameter(torch.zeros(1), requires_grad=False)

    def forward(self, embedded_inputs,
                hidden):
        """
        Encoder - Forward-pass
```

```
        :param Tensor embedded_inputs: Embedded inputs of Pointer-Net
        :param Tensor hidden: Initiated hidden units for the LSTMs (h, c)
        :return: LSTMs outputs and hidden units (h, c)
        """

        embedded_inputs = embedded_inputs.permute(1, 0, 2)

        outputs, hidden = self.lstm(embedded_inputs, hidden)

        return outputs.permute(1, 0, 2), hidden

    def init_hidden(self, embedded_inputs):
        """
        Initiate hidden units

        :param Tensor embedded_inputs: The embedded input of Pointer-NEt
        :return: Initiated hidden units for the LSTMs (h, c)
        """

        batch_size = embedded_inputs.size(0)

        # Reshaping (Expanding)
        h0 = self.h0.unsqueeze(0).unsqueeze(0).repeat(self.n_layers,
                                                      batch_size,
                                                      self.hidden_dim)
        c0 = self.h0.unsqueeze(0).unsqueeze(0).repeat(self.n_layers,
                                                      batch_size,
                                                      self.hidden_dim)

        return h0, c0


class Attention(nn.Module):
    """
    Attention model for Pointer-Net
    """

    def __init__(self, input_dim,
                 hidden_dim):
        """
        Initiate Attention

        :param int input_dim: Input's diamention
        :param int hidden_dim: Number of hidden units in the attention
        """

        super(Attention, self).__init__()
```

```
        self.input_dim = input_dim
        self.hidden_dim = hidden_dim

        self.input_linear = nn.Linear(input_dim, hidden_dim)
        self.context_linear = nn.Conv1d(input_dim, hidden_dim, 1, 1)
        self.V = Parameter(torch.FloatTensor(hidden_dim), requires_grad=True)
        self._inf = Parameter(torch.FloatTensor([float('-inf')]),␣
↪requires_grad=False)
        self.tanh = nn.Tanh()
        self.softmax = nn.Softmax()

        # Initialize vector V
        nn.init.uniform(self.V, -1, 1)

    def forward(self, input,
                context,
                mask):
        """
        Attention - Forward-pass

        :param Tensor input: Hidden state h
        :param Tensor context: Attention context
        :param ByteTensor mask: Selection mask
        :return: tuple of - (Attentioned hidden state, Alphas)
        """

        # (batch, hidden_dim, seq_len)
        inp = self.input_linear(input).unsqueeze(2).expand(-1, -1, context.
↪size(1))

        # (batch, hidden_dim, seq_len)
        context = context.permute(0, 2, 1)
        ctx = self.context_linear(context)

        # (batch, 1, hidden_dim)
        V = self.V.unsqueeze(0).expand(context.size(0), -1).unsqueeze(1)

        # (batch, seq_len)
        att = torch.bmm(V, self.tanh(inp + ctx)).squeeze(1)
        if len(att[mask]) > 0:
            att[mask] = self.inf[mask]
        alpha = self.softmax(att)

        hidden_state = torch.bmm(ctx, alpha.unsqueeze(2)).squeeze(2)

        return hidden_state, alpha
```

```
    def init_inf(self, mask_size):
        self.inf = self._inf.unsqueeze(1).expand(*mask_size)


class Decoder(nn.Module):
    """
    Decoder model for Pointer-Net
    """

    def __init__(self, embedding_dim,
                 hidden_dim):
        """
        Initiate Decoder

        :param int embedding_dim: Number of embeddings in Pointer-Net
        :param int hidden_dim: Number of hidden units for the decoder's RNN
        """

        super(Decoder, self).__init__()
        self.embedding_dim = embedding_dim
        self.hidden_dim = hidden_dim

        self.input_to_hidden = nn.Linear(embedding_dim, 4 * hidden_dim)
        self.hidden_to_hidden = nn.Linear(hidden_dim, 4 * hidden_dim)
        self.hidden_out = nn.Linear(hidden_dim * 2, hidden_dim)
        self.att = Attention(hidden_dim, hidden_dim)

        # Used for propagating .cuda() command
        self.mask = Parameter(torch.ones(1), requires_grad=False)
        self.runner = Parameter(torch.zeros(1), requires_grad=False)

    def forward(self, embedded_inputs,
                decoder_input,
                hidden,
                context):
        """
        Decoder - Forward-pass

        :param Tensor embedded_inputs: Embedded inputs of Pointer-Net
        :param Tensor decoder_input: First decoder's input
        :param Tensor hidden: First decoder's hidden states
        :param Tensor context: Encoder's outputs
        :return: (Output probabilities, Pointers indices), last hidden state
        """

        batch_size = embedded_inputs.size(0)
```

```
        input_length = embedded_inputs.size(1)

        # (batch, seq_len)
        mask = self.mask.repeat(input_length).unsqueeze(0).repeat(batch_size, 1)
        self.att.init_inf(mask.size())

        # Generating arang(input_length), broadcasted across batch_size
        runner = self.runner.repeat(input_length)
        for i in range(input_length):
            runner.data[i] = i
        runner = runner.unsqueeze(0).expand(batch_size, -1).long()

        outputs = []
        pointers = []

        def step(x, hidden):
            """
            Recurrence step function

            :param Tensor x: Input at time t
            :param tuple(Tensor, Tensor) hidden: Hidden states at time t-1
            :return: Hidden states at time t (h, c), Attention probabilities␣
↪(Alpha)
            """

            # Regular LSTM
            h, c = hidden

            gates = self.input_to_hidden(x) + self.hidden_to_hidden(h)
            input, forget, cell, out = gates.chunk(4, 1)

            input = torch.sigmoid(input)
            forget = torch.sigmoid(forget)
            cell = torch.tanh(cell)
            out = torch.sigmoid(out)

            c_t = (forget * c) + (input * cell)
            h_t = out * torch.tanh(c_t)

            # Attention section
            hidden_t, output = self.att(h_t, context, torch.eq(mask, 0))
            hidden_t = torch.tanh(self.hidden_out(torch.cat((hidden_t, h_t),␣
↪1)))

            return hidden_t, c_t, output

        # Recurrence loop
```

```
        for _ in range(input_length):
            h_t, c_t, outs = step(decoder_input, hidden)
            hidden = (h_t, c_t)

            # Masking selected inputs
            masked_outs = outs * mask

            # Get maximum probabilities and indices
            max_probs, indices = masked_outs.max(1)
            one_hot_pointers = (runner == indices.unsqueeze(1).expand(-1, outs.
↪size()[1])).float()

            # Update mask to ignore seen indices
            mask  = mask * (1 - one_hot_pointers)

            # Get embedded inputs by max indices
            embedding_mask = one_hot_pointers.unsqueeze(2).expand(-1, -1, self.
↪embedding_dim).byte()
            decoder_input = embedded_inputs[embedding_mask.data].
↪view(batch_size, self.embedding_dim)

            outputs.append(outs.unsqueeze(0))
            pointers.append(indices.unsqueeze(1))

        outputs = torch.cat(outputs).permute(1, 0, 2)
        pointers = torch.cat(pointers, 1)

        return (outputs, pointers), hidden


class PointerNet(nn.Module):
    """
    Pointer-Net
    """

    def __init__(self, embedding_dim,
                 hidden_dim,
                 lstm_layers,
                 dropout,
                 bidir=False):
        """
        Initiate Pointer-Net

        :param int embedding_dim: Number of embbeding channels
        :param int hidden_dim: Encoders hidden units
        :param int lstm_layers: Number of layers for LSTMs
        :param float dropout: Float between 0-1
```

```
        :param bool bidir: Bidirectional
        """

        super(PointerNet, self).__init__()
        self.embedding_dim = embedding_dim
        self.bidir = bidir
        self.embedding = nn.Linear(2, embedding_dim)
        self.encoder = Encoder(embedding_dim,
                               hidden_dim,
                               lstm_layers,
                               dropout,
                               bidir)
        self.decoder = Decoder(embedding_dim, hidden_dim)
        self.decoder_input0 = Parameter(torch.FloatTensor(embedding_dim),␣
↪requires_grad=False)

        # Initialize decoder_input0
        nn.init.uniform(self.decoder_input0, -1, 1)

    def forward(self, inputs):
        """
        PointerNet - Forward-pass

        :param Tensor inputs: Input sequence
        :return: Pointers probabilities and indices
        """

        batch_size = inputs.size(0)
        input_length = inputs.size(1)

        decoder_input0 = self.decoder_input0.unsqueeze(0).expand(batch_size, -1)

        inputs = inputs.view(batch_size * input_length, -1)
        embedded_inputs = self.embedding(inputs).view(batch_size, input_length,␣
↪-1)

        encoder_hidden0 = self.encoder.init_hidden(embedded_inputs)
        encoder_outputs, encoder_hidden = self.encoder(embedded_inputs,
                                                       encoder_hidden0)
        if self.bidir:
            decoder_hidden0 = (torch.cat(encoder_hidden[0][-2:], dim=-1),
                               torch.cat(encoder_hidden[1][-2:], dim=-1))
        else:
            decoder_hidden0 = (encoder_hidden[0][-1],
                               encoder_hidden[1][-1])
        (outputs, pointers), decoder_hidden = self.decoder(embedded_inputs,
                                                           decoder_input0,
```

```
                                                        decoder_hidden0,
                                                        encoder_outputs)

        return  outputs, pointers
```

[23]:
```
# coding=utf-8
# Copyright 2019-present, the HuggingFace Inc. team, The Google AI Language␣
 ↪Team and Facebook, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
"""
 PyTorch DistilBERT model adapted in part from Facebook, Inc XLM model (https://
 ↪github.com/facebookresearch/XLM) and in
 part from HuggingFace PyTorch version of Google AI Bert model (https://github.
 ↪com/google-research/bert)
"""

logger = logging.get_logger(__name__)
_CHECKPOINT_FOR_DOC = "distilbert-base-uncased"
_CONFIG_FOR_DOC = "DistilBertConfig"
_TOKENIZER_FOR_DOC = "DistilBertTokenizer"

DISTILBERT_PRETRAINED_MODEL_ARCHIVE_LIST = [
    "distilbert-base-uncased",
    "distilbert-base-uncased-distilled-squad",
    "distilbert-base-cased",
    "distilbert-base-cased-distilled-squad",
    "distilbert-base-german-cased",
    "distilbert-base-multilingual-cased",
    "distilbert-base-uncased-finetuned-sst-2-english",
    # See all DistilBERT models at https://huggingface.co/models?
 ↪filter=distilbert
]


# UTILS AND BUILDING BLOCKS OF THE ARCHITECTURE #
```

```
def create_sinusoidal_embeddings(n_pos, dim, out):
    position_enc = np.array([[pos / np.power(10000, 2 * (j // 2) / dim) for j␣
↪in range(dim)] for pos in range(n_pos)])
    out.requires_grad = False
    out[:, 0::2] = torch.FloatTensor(np.sin(position_enc[:, 0::2]))
    out[:, 1::2] = torch.FloatTensor(np.cos(position_enc[:, 1::2]))
    out.detach_()


class Embeddings(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.word_embeddings = nn.Embedding(config.vocab_size, config.dim,␣
↪padding_idx=config.pad_token_id)
        self.position_embeddings = nn.Embedding(config.max_position_embeddings,␣
↪config.dim)
        if config.sinusoidal_pos_embds:

            if is_deepspeed_zero3_enabled():
                import deepspeed

                with deepspeed.zero.GatheredParameters(self.position_embeddings.
↪weight, modifier_rank=0):
                    if torch.distributed.get_rank() == 0:
                        create_sinusoidal_embeddings(
                            n_pos=config.max_position_embeddings, dim=config.
↪dim, out=self.position_embeddings.weight
                        )
            else:
                create_sinusoidal_embeddings(
                    n_pos=config.max_position_embeddings, dim=config.dim,␣
↪out=self.position_embeddings.weight
                )

        self.LayerNorm = nn.LayerNorm(config.dim, eps=1e-12)
        self.dropout = nn.Dropout(config.dropout)

    def forward(self, input_ids):
        """
        Parameters:
            input_ids: torch.tensor(bs, max_seq_length) The token ids to embed.

        Returns: torch.tensor(bs, max_seq_length, dim) The embedded tokens␣
↪(plus position embeddings, no token_type
        embeddings)
```

```
        """
        seq_length = input_ids.size(1)
        position_ids = torch.arange(seq_length, dtype=torch.long,␣
↪device=input_ids.device)  # (max_seq_length)
        position_ids = position_ids.unsqueeze(0).expand_as(input_ids)  # (bs,␣
↪max_seq_length)

        word_embeddings = self.word_embeddings(input_ids)  # (bs,␣
↪max_seq_length, dim)
        position_embeddings = self.position_embeddings(position_ids)  # (bs,␣
↪max_seq_length, dim)

        embeddings = word_embeddings + position_embeddings  # (bs,␣
↪max_seq_length, dim)
        embeddings = self.LayerNorm(embeddings)  # (bs, max_seq_length, dim)
        embeddings = self.dropout(embeddings)  # (bs, max_seq_length, dim)
        return embeddings


class MultiHeadSelfAttention(nn.Module):
    def __init__(self, config):
        super().__init__()

        self.n_heads = config.n_heads
        self.dim = config.dim
        self.dropout = nn.Dropout(p=config.attention_dropout)

        assert self.dim % self.n_heads == 0

        self.q_lin = nn.Linear(in_features=config.dim, out_features=config.dim)
        self.k_lin = nn.Linear(in_features=config.dim, out_features=config.dim)
        self.v_lin = nn.Linear(in_features=config.dim, out_features=config.dim)
        self.out_lin = nn.Linear(in_features=config.dim, out_features=config.
↪dim)

        self.pruned_heads = set()

    def prune_heads(self, heads):
        attention_head_size = self.dim // self.n_heads
        if len(heads) == 0:
            return
        heads, index = find_pruneable_heads_and_indices(heads, self.n_heads,␣
↪attention_head_size, self.pruned_heads)
        # Prune linear layers
        self.q_lin = prune_linear_layer(self.q_lin, index)
        self.k_lin = prune_linear_layer(self.k_lin, index)
```

```
        self.v_lin = prune_linear_layer(self.v_lin, index)
        self.out_lin = prune_linear_layer(self.out_lin, index, dim=1)
        # Update hyper params
        self.n_heads = self.n_heads - len(heads)
        self.dim = attention_head_size * self.n_heads
        self.pruned_heads = self.pruned_heads.union(heads)

    def forward(self, query, key, value, mask, head_mask=None,␣
↪output_attentions=False):
        """
        Parameters:
            query: torch.tensor(bs, seq_length, dim)
            key: torch.tensor(bs, seq_length, dim)
            value: torch.tensor(bs, seq_length, dim)
            mask: torch.tensor(bs, seq_length)

        Returns:
            weights: torch.tensor(bs, n_heads, seq_length, seq_length)␣
↪Attention weights context: torch.tensor(bs,
            seq_length, dim) Contextualized layer. Optional: only if␣
↪`output_attentions=True`
        """
        bs, q_length, dim = query.size()
        k_length = key.size(1)
        # assert dim == self.dim, f'Dimensions do not match: {dim} input vs␣
↪{self.dim} configured'
        # assert key.size() == value.size()

        dim_per_head = self.dim // self.n_heads

        mask_reshp = (bs, 1, 1, k_length)

        def shape(x):
            """separate heads"""
            return x.view(bs, -1, self.n_heads, dim_per_head).transpose(1, 2)

        def unshape(x):
            """group heads"""
            return x.transpose(1, 2).contiguous().view(bs, -1, self.n_heads *␣
↪dim_per_head)

        q = shape(self.q_lin(query))  # (bs, n_heads, q_length, dim_per_head)
        k = shape(self.k_lin(key))  # (bs, n_heads, k_length, dim_per_head)
        v = shape(self.v_lin(value))  # (bs, n_heads, k_length, dim_per_head)

        q = q / math.sqrt(dim_per_head)  # (bs, n_heads, q_length, dim_per_head)
```

```
        scores = torch.matmul(q, k.transpose(2, 3))  # (bs, n_heads, q_length,␣
↪k_length)
        mask = (mask == 0).view(mask_reshp).expand_as(scores)  # (bs, n_heads,␣
↪q_length, k_length)
        scores.masked_fill_(mask, -float("inf"))  # (bs, n_heads, q_length,␣
↪k_length)

        weights = nn.Softmax(dim=-1)(scores)  # (bs, n_heads, q_length,␣
↪k_length)
        weights = self.dropout(weights)  # (bs, n_heads, q_length, k_length)

        # Mask heads if we want to
        if head_mask is not None:
            weights = weights * head_mask

        context = torch.matmul(weights, v)  # (bs, n_heads, q_length,␣
↪dim_per_head)
        context = unshape(context)  # (bs, q_length, dim)
        context = self.out_lin(context)  # (bs, q_length, dim)

        if output_attentions:
            return (context, weights)
        else:
            return (context,)


class FFN(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.dropout = nn.Dropout(p=config.dropout)
        self.chunk_size_feed_forward = config.chunk_size_feed_forward
        self.seq_len_dim = 1
        self.lin1 = nn.Linear(in_features=config.dim, out_features=config.
↪hidden_dim)
        self.lin2 = nn.Linear(in_features=config.hidden_dim,␣
↪out_features=config.dim)
        assert config.activation in ["relu", "gelu"], f"activation ({config.
↪activation}) must be in ['relu', 'gelu']"
        self.activation = gelu if config.activation == "gelu" else nn.ReLU()

    def forward(self, input):
        return apply_chunking_to_forward(self.ff_chunk, self.
↪chunk_size_feed_forward, self.seq_len_dim, input)

    def ff_chunk(self, input):
        x = self.lin1(input)
```

```
        x = self.activation(x)
        x = self.lin2(x)
        x = self.dropout(x)
        return x


class TransformerBlock(nn.Module):
    def __init__(self, config):
        super().__init__()

        assert config.dim % config.n_heads == 0

        self.attention = MultiHeadSelfAttention(config)
        self.sa_layer_norm = nn.LayerNorm(normalized_shape=config.dim,␣
↪eps=1e-12)

        self.ffn = FFN(config)
        self.output_layer_norm = nn.LayerNorm(normalized_shape=config.dim,␣
↪eps=1e-12)

    def forward(self, x, attn_mask=None, head_mask=None,␣
↪output_attentions=False):
        """
        Parameters:
            x: torch.tensor(bs, seq_length, dim)
            attn_mask: torch.tensor(bs, seq_length)

        Returns:
            sa_weights: torch.tensor(bs, n_heads, seq_length, seq_length) The␣
↪attention weights ffn_output:
            torch.tensor(bs, seq_length, dim) The output of the transformer␣
↪block contextualization.
        """
        # Self-Attention
        sa_output = self.attention(
            query=x,
            key=x,
            value=x,
            mask=attn_mask,
            head_mask=head_mask,
            output_attentions=output_attentions,
        )
        if output_attentions:
            sa_output, sa_weights = sa_output  # (bs, seq_length, dim), (bs,␣
↪n_heads, seq_length, seq_length)
```

```
        else:  # To handle these `output_attentions` or `output_hidden_states` cases returning tuples
            assert type(sa_output) == tuple
            sa_output = sa_output[0]
        sa_output = self.sa_layer_norm(sa_output + x)  # (bs, seq_length, dim)

        # Feed Forward Network
        ffn_output = self.ffn(sa_output)  # (bs, seq_length, dim)
        ffn_output = self.output_layer_norm(ffn_output + sa_output)  # (bs, seq_length, dim)

        output = (ffn_output,)
        if output_attentions:
            output = (sa_weights,) + output
        return output


class Transformer(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.n_layers = config.n_layers
        self.layer = nn.ModuleList([TransformerBlock(config) for _ in range(config.n_layers)])

    def forward(
        self, x, attn_mask=None, head_mask=None, output_attentions=False, output_hidden_states=False, return_dict=None
    ):  # docstyle-ignore
        """
        Parameters:
            x: torch.tensor(bs, seq_length, dim) Input sequence embedded.
            attn_mask: torch.tensor(bs, seq_length) Attention mask on the sequence.

        Returns:
            hidden_state: torch.tensor(bs, seq_length, dim) Sequence of hidden states in the last (top)
            layer all_hidden_states: Tuple[torch.tensor(bs, seq_length, dim)]
                Tuple of length n_layers with the hidden states from each layer.
                Optional: only if output_hidden_states=True
            all_attentions: Tuple[torch.tensor(bs, n_heads, seq_length, seq_length)]
                Tuple of length n_layers with the attention weights from each layer
                Optional: only if output_attentions=True
        """
```

```
        all_hidden_states = () if output_hidden_states else None
        all_attentions = () if output_attentions else None

        hidden_state = x
        for i, layer_module in enumerate(self.layer):
            if output_hidden_states:
                all_hidden_states = all_hidden_states + (hidden_state,)

            layer_outputs = layer_module(
                x=hidden_state, attn_mask=attn_mask, head_mask=head_mask[i],␣
↪output_attentions=output_attentions
            )
            hidden_state = layer_outputs[-1]

            if output_attentions:
                assert len(layer_outputs) == 2
                attentions = layer_outputs[0]
                all_attentions = all_attentions + (attentions,)
            else:
                assert len(layer_outputs) == 1

        # Add last layer
        if output_hidden_states:
            all_hidden_states = all_hidden_states + (hidden_state,)

        if not return_dict:
            return tuple(v for v in [hidden_state, all_hidden_states,␣
↪all_attentions] if v is not None)
        return BaseModelOutput(
            last_hidden_state=hidden_state, hidden_states=all_hidden_states,␣
↪attentions=all_attentions
        )


# INTERFACE FOR ENCODER AND TASK SPECIFIC MODEL #
class DistilBertPreTrainedModel(PreTrainedModel):
    """
    An abstract class to handle weights initialization and a simple interface␣
↪for downloading and loading pretrained
    models.
    """

    config_class = DistilBertConfig
    load_tf_weights = None
    base_model_prefix = "distilbert"

    def _init_weights(self, module):
```

```
        """Initialize the weights."""
        if isinstance(module, nn.Linear):
            # Slightly different from the TF version which uses␣
↪truncated_normal for initialization
            # cf https://github.com/pytorch/pytorch/pull/5617
            module.weight.data.normal_(mean=0.0, std=self.config.
↪initializer_range)
            if module.bias is not None:
                module.bias.data.zero_()
        elif isinstance(module, nn.Embedding):
            module.weight.data.normal_(mean=0.0, std=self.config.
↪initializer_range)
            if module.padding_idx is not None:
                module.weight.data[module.padding_idx].zero_()
        elif isinstance(module, nn.LayerNorm):
            module.bias.data.zero_()
            module.weight.data.fill_(1.0)


DISTILBERT_START_DOCSTRING = r"""

    This model inherits from :class:`~transformers.PreTrainedModel`. Check the␣
↪superclass documentation for the generic
    methods the library implements for all its model (such as downloading or␣
↪saving, resizing the input embeddings,
    pruning heads etc.)

    This model is also a PyTorch `torch.nn.Module <https://pytorch.org/docs/
↪stable/nn.html#torch.nn.Module>`__
    subclass. Use it as a regular PyTorch Module and refer to the PyTorch␣
↪documentation for all matter related to
    general usage and behavior.

    Parameters:
        config (:class:`~transformers.DistilBertConfig`): Model configuration␣
↪class with all the parameters of the model.
            Initializing with a config file does not load the weights␣
↪associated with the model, only the
            configuration. Check out the :meth:`~transformers.PreTrainedModel.
↪from_pretrained` method to load the model
            weights.
"""

DISTILBERT_INPUTS_DOCSTRING = r"""
    Args:
        input_ids (:obj:`torch.LongTensor` of shape :obj:`({0})`):
```

```
            Indices of input sequence tokens in the vocabulary.

            Indices can be obtained using :class:`~transformers.
↪DistilBertTokenizer`. See
            :meth:`transformers.PreTrainedTokenizer.encode` and :meth:
↪`transformers.PreTrainedTokenizer.__call__` for
            details.

            `What are input IDs? <../glossary.html#input-ids>`__
        attention_mask (:obj:`torch.FloatTensor` of shape :obj:`({0})`,␣
↪`optional`):
            Mask to avoid performing attention on padding token indices. Mask␣
↪values selected in ``[0, 1]``:

            - 1 for tokens that are **not masked**,
            - 0 for tokens that are **masked**.

            `What are attention masks? <../glossary.html#attention-mask>`__
        head_mask (:obj:`torch.FloatTensor` of shape :obj:`(num_heads,)` or :
↪obj:`(num_layers, num_heads)`, `optional`):
            Mask to nullify selected heads of the self-attention modules. Mask␣
↪values selected in ``[0, 1]``:

            - 1 indicates the head is **not masked**,
            - 0 indicates the head is **masked**.

        inputs_embeds (:obj:`torch.FloatTensor` of shape :obj:`({0},␣
↪hidden_size)`, `optional`):
            Optionally, instead of passing :obj:`input_ids` you can choose to␣
↪directly pass an embedded representation.
            This is useful if you want more control over how to convert :obj:
↪`input_ids` indices into associated
            vectors than the model's internal embedding lookup matrix.
        output_attentions (:obj:`bool`, `optional`):
            Whether or not to return the attentions tensors of all attention␣
↪layers. See ``attentions`` under returned
            tensors for more detail.
        output_hidden_states (:obj:`bool`, `optional`):
            Whether or not to return the hidden states of all layers. See␣
↪``hidden_states`` under returned tensors for
            more detail.
        return_dict (:obj:`bool`, `optional`):
            Whether or not to return a :class:`~transformers.file_utils.
↪ModelOutput` instead of a plain tuple.
"""
```

```
class DistilBertModel(DistilBertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)

        self.hidden_size = config.hidden_dim
        self.num_layers = 1
        self.num_directions = 2 #because we are implementing it in␣
↪bidirectional lstm mode

        self.embeddings = Embeddings(config)  # Embeddings
        self.transformer = Transformer(config)  # Encoder
        self.pointer = PointerNet(1,8,1,0.0,False)

        self.init_weights()

    def get_input_embeddings(self):
        return self.embeddings.word_embeddings

    def set_input_embeddings(self, new_embeddings):
        self.embeddings.word_embeddings = new_embeddings

    def _prune_heads(self, heads_to_prune):
        """
        Prunes heads of the model. heads_to_prune: dict of {layer_num: list of␣
↪heads to prune in this layer} See base
        class PreTrainedModel
        """
        for layer, heads in heads_to_prune.items():
            self.transformer.layer[layer].attention.prune_heads(heads)

    def forward(
        self,
        input_ids=None,
        attention_mask=None,
        head_mask=None,
        inputs_embeds=None,
        output_attentions=None,
        output_hidden_states=None,
        return_dict=None,
    ):
        output_attentions = output_attentions if output_attentions is not None␣
↪else self.config.output_attentions
        output_hidden_states = (
            output_hidden_states if output_hidden_states is not None else self.
↪config.output_hidden_states
```

```
        )
        return_dict = return_dict if return_dict is not None else self.config.
↪use_return_dict

        if input_ids is not None and inputs_embeds is not None:
            raise ValueError("You cannot specify both input_ids and␣
↪inputs_embeds at the same time")
        elif input_ids is not None:
            input_shape = input_ids.size()
        elif inputs_embeds is not None:
            input_shape = inputs_embeds.size()[:-1]
        else:
            raise ValueError("You have to specify either input_ids or␣
↪inputs_embeds")

        device = input_ids.device if input_ids is not None else inputs_embeds.
↪device

        if attention_mask is None:
            attention_mask = torch.ones(input_shape, device=device)  # (bs,␣
↪seq_length)

        # Prepare head mask if needed
        head_mask = self.get_head_mask(head_mask, self.config.num_hidden_layers)

        if inputs_embeds is None:
            inputs_embeds = self.embeddings(input_ids)  # (bs, seq_length, dim)


        inputs_embed, ots=self.pointer(inputs_embeds.type(torch.float))
        return self.transformer(
            x=inputs_embeds,
            attn_mask=attention_mask,
            head_mask=head_mask,
            output_attentions=output_attentions,
            output_hidden_states=output_hidden_states,
            return_dict=return_dict,
        )


class DistilBertForQuestionAnsweringC(DistilBertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)

        self.distilbert = DistilBertModel(config)
        self.qa_outputs = nn.Linear(config.dim, config.num_labels)
        assert config.num_labels == 2
```

```
        self.dropout = nn.Dropout(config.qa_dropout)

        self.init_weights()

    def forward(
        self,
        input_ids=None,
        attention_mask=None,
        head_mask=None,
        inputs_embeds=None,
        start_positions=None,
        end_positions=None,
        output_attentions=None,
        output_hidden_states=None,
        return_dict=None,
    ):
        r"""
        start_positions (:obj:`torch.LongTensor` of shape :obj:`(batch_size,)`,␣
↪`optional`):
            Labels for position (index) of the start of the labelled span for␣
↪computing the token classification loss.
            Positions are clamped to the length of the sequence (:obj:
↪`sequence_length`). Position outside of the
            sequence are not taken into account for computing the loss.
        end_positions (:obj:`torch.LongTensor` of shape :obj:`(batch_size,)`,␣
↪`optional`):
            Labels for position (index) of the end of the labelled span for␣
↪computing the token classification loss.
            Positions are clamped to the length of the sequence (:obj:
↪`sequence_length`). Position outside of the
            sequence are not taken into account for computing the loss.
        """
        return_dict = return_dict if return_dict is not None else self.config.
↪use_return_dict


        distilbert_output = self.distilbert(
            input_ids=input_ids,
            attention_mask=attention_mask,
            head_mask=head_mask,
            inputs_embeds=inputs_embeds,
            output_attentions=output_attentions,
            output_hidden_states=output_hidden_states,
            return_dict=return_dict,
        )
```

```
        hidden_states = distilbert_output[0]  # (bs, max_query_len, dim)

        hidden_states = self.dropout(hidden_states)  # (bs, max_query_len, dim)
        logits = self.qa_outputs(hidden_states)  # (bs, max_query_len, 2)
        start_logits, end_logits = logits.split(1, dim=-1)
        start_logits = start_logits.squeeze(-1).contiguous()  # (bs,␣
↪max_query_len)
        end_logits = end_logits.squeeze(-1).contiguous()  # (bs, max_query_len)

        total_loss = None
        if start_positions is not None and end_positions is not None:
            # If we are on multi-GPU, split add a dimension
            if len(start_positions.size()) > 1:
                start_positions = start_positions.squeeze(-1)
            if len(end_positions.size()) > 1:
                end_positions = end_positions.squeeze(-1)
            # sometimes the start/end positions are outside our model inputs,␣
↪we ignore these terms
            ignored_index = start_logits.size(1)
            start_positions = start_positions.clamp(0, ignored_index)
            end_positions = end_positions.clamp(0, ignored_index)

            loss_fct = nn.CrossEntropyLoss(ignore_index=ignored_index)
            start_loss = loss_fct(start_logits, start_positions)
            end_loss = loss_fct(end_logits, end_positions)
            total_loss = (start_loss + end_loss) / 2

        if not return_dict:
            output = (start_logits, end_logits) + distilbert_output[1:]
            return ((total_loss,) + output) if total_loss is not None else␣
↪output

        return QuestionAnsweringModelOutput(
            loss=total_loss,
            start_logits=start_logits,
            end_logits=end_logits,
            hidden_states=distilbert_output.hidden_states,
            attentions=distilbert_output.attentions,
        )
```

[24]:
```
## 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)
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=231508.0,␣
↪style=ProgressStyle(descripti…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=466062.0,␣
↪style=ProgressStyle(descripti…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=28.0,␣
↪style=ProgressStyle(description_w…
```

observation

[25]:
```
## 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)
```

5 3. Train & Validation Dataset Creation

[26]:
```
## 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)
```

5.0.1 Observations

6 4. Model Building & Training

[27]:
```
model =DistilBertForQuestionAnsweringC.
 ↪from_pretrained('distilbert-base-uncased')
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=442.0,␣
 ↪style=ProgressStyle(description_…
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=267967963.
 ↪0, style=ProgressStyle(descri…
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:105: UserWarning:
nn.init.uniform is now deprecated in favor of nn.init.uniform_.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:290: UserWarning:
nn.init.uniform is now deprecated in favor of nn.init.uniform_.
Some weights of the model checkpoint at distilbert-base-uncased were not used
when initializing DistilBertForQuestionAnsweringC: ['vocab_layer_norm.weight',
'vocab_projector.weight', 'vocab_transform.weight', 'vocab_projector.bias',
'vocab_layer_norm.bias', 'vocab_transform.bias']
- This IS expected if you are initializing DistilBertForQuestionAnsweringC 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 DistilBertForQuestionAnsweringC
from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
Some weights of DistilBertForQuestionAnsweringC were not initialized from the
model checkpoint at distilbert-base-uncased and are newly initialized:
['distilbert.pointer.decoder.att.context_linear.weight', 'qa_outputs.weight',
'distilbert.pointer.decoder.att._inf',
```

```
'distilbert.pointer.decoder.hidden_to_hidden.bias',
'distilbert.pointer.decoder.hidden_to_hidden.weight',
'distilbert.pointer.decoder.input_to_hidden.weight',
'distilbert.pointer.decoder.att.V', 'distilbert.pointer.encoder.c0',
'distilbert.pointer.decoder.runner', 'distilbert.pointer.decoder_input0',
'distilbert.pointer.encoder.lstm.bias_ih_l0',
'distilbert.pointer.encoder.lstm.weight_hh_l0',
'distilbert.pointer.encoder.lstm.weight_ih_l0', 'distilbert.pointer.encoder.h0',
'distilbert.pointer.encoder.lstm.bias_hh_l0',
'distilbert.pointer.decoder.att.context_linear.bias',
'distilbert.pointer.decoder.hidden_out.weight',
'distilbert.pointer.embedding.weight', 'qa_outputs.bias',
'distilbert.pointer.decoder.att.input_linear.weight',
'distilbert.pointer.decoder.att.input_linear.bias',
'distilbert.pointer.decoder.hidden_out.bias',
'distilbert.pointer.decoder.input_to_hidden.bias',
'distilbert.pointer.embedding.bias', 'distilbert.pointer.decoder.mask']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

6.0.1 Observations

[28]:
```
# 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)
```

6.0.2 Observations

[29]:
```
torch.cuda.empty_cache()
```

6.0.3 Observations

[30]:
```
# Train for the model, perform validation on it per epoch and generate files␣
 ↪for a tensorboard
num_epochs = 2

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 0/1
----------
```

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:133: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:246: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

```
Training Loss: 11.7764
```

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

```
Val Loss: 1.6985, EM: 0.6186, F1: 0.7256
Epoch 1/1
----------
```

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

```
Training Loss: 7.7792
```

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

```
Val Loss: 0.1955, EM: 0.6313, F1: 0.7316
```

6.0.4 Observations

[31]:
```
# We save our model so that it can be reused later

torch.save(model,'./customBertmodelAdamW5e.pt')
```

[32]:
```
# Generate a Tensorboard

%load_ext tensorboard
```

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

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

6.0.5 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.

7 5. Running The Model

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

[33]:
```
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"
```

[34]:
```
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
```

[35]:
```
question_answer(test_question, test_context, model)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:133: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:246: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[35]:
```
'rollo'
```

[36]:
```
question_answer(val_questions[0], val_contexts[0], model)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:133: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:246: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[36]:
```
'france'
```

8 Running The Model

[37]:
```
model_loaded = torch.load('./customBertmodelAdamW5e.pt')
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

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"
```

[38]:
```
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
```

[39]:
```
question_answer(test_question, test_context, model_loaded)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:133: UserWarning:
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:246: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[39]:
```
'rollo'
```

[40]:
```
## 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"""
```

[41]:
```
question_answer(question, context, model_loaded)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:133: UserWarning:
```

```
Implicit dimension choice for softmax has been deprecated. Change the call to
include dim=X as an argument.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:246: UserWarning:
indexing with dtype torch.uint8 is now deprecated, please use a dtype torch.bool
instead. (Triggered internally at
/pytorch/aten/src/ATen/native/IndexingUtils.h:30.)
```

[41]: 'infinity'

References

Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural Comput*, 9 (8):1735–1780, 1997.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. 4:2047–2052 vol. 4, 2005. doi: 10.1109/IJCNN.2005.1556215.

Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108, 2015. URL `http://arxiv.org/abs/1511.04108`.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016. URL `http://arxiv.org/abs/1608.07905`.

Giovanni Di Gennaro, Amedeo Buonanno, Antonio Di Girolamo, Armando Ospedale, and Francesco A. N. Palmieri. Intent classification in question-answering using LSTM architectures. *CoRR*, abs/2001.09330, 2020. URL `https://arxiv.org/abs/2001.09330`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. *CoRR*, abs/1508.01585, 2015. URL `http://arxiv.org/abs/1508.01585`.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL `http://arxiv.org/abs/1806.03822`.

Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224, 1961.

William A Woods and WOODS WA. Lunar rocks in natural english: Explorations in natural language question answering. 1977.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL `http://arxiv.org/abs/1910.01108`.

The Hugging Face Team. Transformers. URL `https://huggingface.co/transformers/`. Accessed: 2021-04-16.

Pranav Rajpurkar. Squad2.0, 2021. URL `https://rajpurkar.github.io/SQuAD-explorer/`. Accessed: 2021-04-16.

Mark B Ring. Learning sequential tasks by incrementally adding higher orders. *Advances in neural information processing systems*, pages 115–115, 1993.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015. URL `http://arxiv.org/abs/1506.03340`.

Siddhartha Brahma. Suffix bidirectional long short-term memory. *CoRR*, abs/1805.07340, 2018. URL `http://arxiv.org/abs/1805.07340`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL `http://arxiv.org/abs/1606.05250`.

Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849, 2015. URL `http://arxiv.org/abs/1512.08849`.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *arXiv preprint arXiv:1506.03134*, 2015.

Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. Trec complex answer retrieval overview. In *TREC*, 2017.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3302. URL `https://www.aclweb.org/anthology/W14-3302`.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017. URL `http://arxiv.org/abs/1705.03122`.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *arXiv preprint arXiv:1511.01432*, 2015.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

Wei Wang, Ming Yan, and Chen Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *arXiv preprint arXiv:1811.11934*, 2018.

Saptadeep Pal, Eiman Ebrahimi, Arslan Zulfiqar, Yaosheng Fu, Victor Zhang, Szymon Migacz, David W. Nellans, and Puneet Gupta. Optimizing multi-gpu parallelization strategies for deep learning training. *CoRR*, abs/1907.13257, 2019. URL `http://arxiv.org/abs/1907.13257`.

Giuliano Giacaglia. How to scale training on multiple gpus, Dec 2020. URL `https://towardsdatascience.com/how-to-scale-training-on-multiple-gpus-dae1041f49d2`.