Importing Libraries

In [1]:

```
import os
import json
import gc
import pickle

import numpy as np
import pandas as pd
import tensorflow as tf
from tqdm import tqdm_notebook as tqdm
from tensorflow.keras.models import Model
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import text, sequence

import warnings
warnings.filterwarnings("ignore")
```

Setting up Hugging Face BERT

In [ ]:

```
from transformers import BertTokenizer, TFBertForQuestionAnswering

modelName = 'bert-large-uncased-whole-word-masking-finetuned-squad' # https://huggingface.co/transformers/pretrained_models.html

tokenizer = BertTokenizer.from_pretrained(modelName)
model = TFBertForQuestionAnswering.from_pretrained(modelName)
```

Loading Test Data and Previous Submission Data

In [15]:

```
test_path = '../input/tensorflow2-question-answering/simplified-nq-test.jsonl'
test = pd.read_json(test_path, orient='records', lines=True, dtype={'example_id':np.dtype('object')})
submission = pd.read_csv("../input/baseline-lstm/submission.csv")
```

In [5]:

```
submission.head()
```

Out[5]:

	example_id	PredictionString
0	-1011141123527297803_long	931:1088
1	-1011141123527297803_short	931:1088
2	-1028916936938579349_long	781:923
3	-1028916936938579349_short	781:923
4	-1055197305756217938_long	741:998

- **Note that every short answer till now is equal to the long answer.**
- **I predict short answer using that long answer as the text to the question.**
- **i.e my short answer is a subset to the long answer.**

Predict

In [3]:

```
def bert_predict_short_answer(q, t, base):
    """
    Predict the answer tokens for the given question and text.
    parameters:
        question: question
        text: corresponding text
    returns:
        predicted answer tokens
    """
    try:
        input_text =  q + " [SEP] " + t
        input_ids = tokenizer.encode(input_text)
        input = tf.constant(input_ids)[None, :]  # Batch size 1
        token_type_ids = [0 if i <= input_ids.index(102) else 1 for i in range(len(input
_ids))]

        startScores, endScores =model(input, token_type_ids = tf.convert_to_tensor([toke
n_type_ids]))
        startIdx = tf.math.argmax(startScores[0],0).numpy()
        endIdx = tf.math.argmax(endScores[0],0).numpy()+1
        # print(startIdx,endIdx)
        # input_tokens = tokenizer.convert_ids_to_tokens(input_ids)
        # print(" ".join(input_tokens[startIdx:endIdx]))
        return str(startIdx+base) +':' + str(endIdx+base)
    except:
        return np.nan


def predict(submission, test):
    """
    Modifies the short answer in the submission file.
    Using that long answer as the Text to the Question, my BERT model will predict a subs
et of indies that i will consider my short answer.
    Parameters:
        submission: submission file to be modified
        test: The test file
    Returns:
        Returns Modified submission file.
    """
    short = '_short'
    for i in tqdm(range(len(submission))):
        if submission.iloc[i]['example_id'].endswith(short):
            id = submission.iloc[i]['example_id'][:-6]
            token = submission.iloc[i]['PredictionString']

            if isinstance(token, str): # https://www.geeksforgeeks.org/python-check-if-a
-variable-is-string/
                # sample dataframe corresponding to the id
                sample_df = test[test['example_id'] == id]
                # Text of the sample df
                text = sample_df.iloc[0]['document_text'].split()
                # Corresponding Question
                question = sample_df.iloc[0]['question_text']

                # start: the token before ":", end: the token after ":"
                index = token.index(':')
                start = int(token[:index])
                end = int(token[index+1:])

                # text corresponds to the long answer
                text = " ".join(text[start:end])

                # Tokens for the short answer.
                token = bert_predict_short_answer(question, text, base = start)
```

```
            else:
                # No long answer.
                token = np.nan
            submission.iloc[i]['PredictionString'] = str(token)
    return submission
```

In [16]:

```
submission = predict(submission, test)
```

In [8]:

```
submission.head()
```

Out[8]:

	example_id	PredictionString
0	-1011141123527297803_long	931:1088
1	-1011141123527297803_short	932:941
2	-1028916936938579349_long	781:923
3	-1028916936938579349_short	797:818
4	-1055197305756217938_long	741:998

As we can see, the short answer is modified by our BERT model.

Submission

In [13]:

```
final_submission = (
    submission.drop(columns='PredictionString').merge(submission, on=['example_id'], how
='left')
)
final_submission.to_csv("submission.csv", index=False)
```

In [14]:

```
final_submission.head()
```

Out[14]:

	example_id	PredictionString
0	-1011141123527297803_long	NaN
1	-1011141123527297803_short	NaN
2	-1028916936938579349_long	NaN
3	-1028916936938579349_short	NaN
4	-1055197305756217938_long	NaN

In [ ]:

```

```