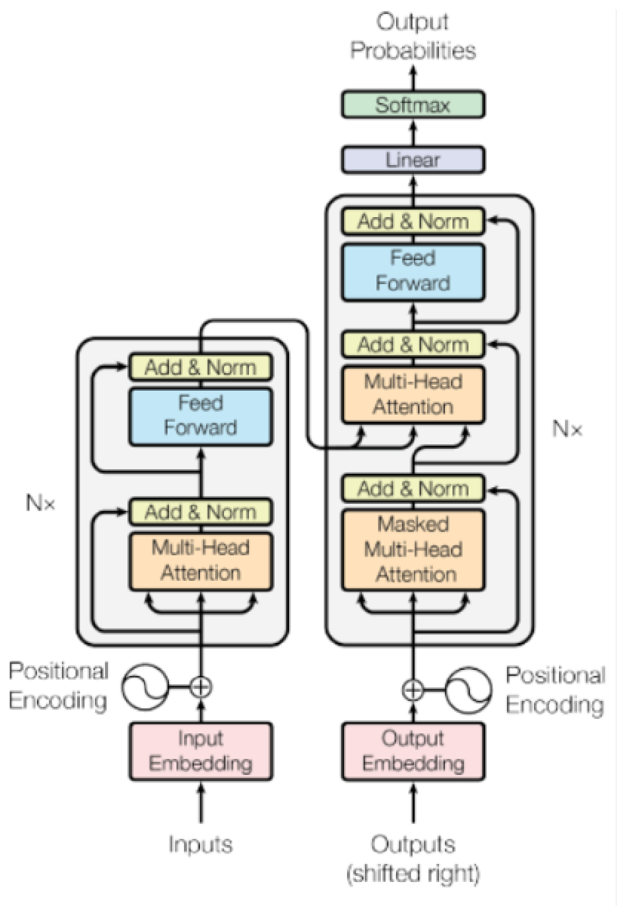
Transformers

BERT - Bidirectional Encoder Representation from Transformers; stacking of transformer encoders

GPT - Generator pretrained transformer; stacking of transformer decoders; left to right



```
pip install torch
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.0.1+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.12.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.11.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch) (3.25.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch) (16.0.5)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.2
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
```

```
pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting transformers
  Downloading transformers-4.29.2-py3-none-any.whl (7.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.1/7.1 MB 61.3 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.0)
Collecting huggingface-hub<1.0,>=0.14.1 (from transformers)
  Downloading huggingface_hub-0.14.1-py3-none-any.whl (224 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.5/224.5 kB 23.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2022
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.27.1)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 75.1 MB/s eta 0:00:00
```

```
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.65.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingfac
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->trans
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transfor
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->t
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.14.1 tokenizers-0.13.3 transformers-4.29.2
```

```
from transformers import BertForQuestionAnswering
from transformers import BertTokenizer
import torch
import numpy as np
```

```
model = BertForQuestionAnswering.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')
```

```
Downloading (…)lve/main/config.json: 100%                 443/443 [00:00<00:00, 9.52kB/s]

Downloading pytorch_model.bin: 100%                       1.34G/1.34G [00:09<00:00, 132MB/s]
```

```
tokenizer_for_bert = BertTokenizer.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')
```

```
Downloading (…)solve/main/vocab.txt: 100%                 232k/232k [00:00<00:00, 4.00MB/s]

Downloading (…)okenizer_config.json: 100%                 28.0/28.0 [00:00<00:00, 1.15kB/s]
```

```
def bert_question_answer(question, passage, max_len=500):

    """
    question: What is the name of my dog
    passage: I have a loving household of 5 members and a dog in my immediate family. The name of my dog is bruno. He is very playful.
    """

    #Tokenize input question and passage
    #Add special tokens - [CLS] and [SEP]
    input_ids = tokenizer_for_bert.encode (question, passage,  max_length= max_len, truncation=True)
    """
    [101, 2054, 2003, 1996, 2171, 1997, 2026, 3899, 102, 1045, 2031, 1037, 5386,
    6755, 1997, 1019, 3478, 1998, 1037, 3899, 1999, 2026, 7132, 2155, 1012, 1996, 2171,
    1997, 2026, 3899, 2003, 17263, 1012, 2002, 2003, 2200, 8382, 1012, 102]

     Please note that '[CLS]' is represented by the token ID 101, '[SEP]' is represented by the token ID 102,
     and other tokens represent different words or subword pieces in the original text.
    """

    #Getting number of tokens in 1st sentence (question) and 2nd sentence (passage that contains answer)
    sep_index = input_ids.index(102)
    len_question = sep_index + 1
    len_passage = len(input_ids)- len_question

    #Need to separate question and passage
    #Segment ids will be 0 for question and 1 for passage
    segment_ids =  [0]*len_question + [1]*(len_passage)
    """
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
    """

    #Converting token ids to tokens
    tokens = tokenizer_for_bert.convert_ids_to_tokens(input_ids)
    """
    ['[CLS]', 'what', 'is', 'the', 'name', 'of', 'my', 'dog', '[SEP]', 'i', 'have', 'a', 'loving', 'household', 'of',
    '5', 'members', 'and', 'a', 'dog', 'in', 'my', 'immediate', 'family', '.', 'the', 'name', 'of', 'my', 'dog', 'is',
    'bruno', '.', 'he', 'is', 'very', 'playful', '.', '[SEP]']

    """

    #Getting start and end scores for answer
    #Converting input arrays to torch tensors before passing to the model
    start_token_scores = model(torch.tensor([input_ids]), token_type_ids=torch.tensor([segment_ids]) )[0]
    end_token_scores = model(torch.tensor([input_ids]), token_type_ids=torch.tensor([segment_ids]) )[1]
    """
    tensor([[-5.9787, -3.0541, -7.7166, -5.9291, -6.8790, -7.2380, -1.8289, -8.1006,
         -5.9786, -3.9319, -5.6230, -4.1919, -7.2068, -6.7739, -2.3960, -5.9425,
         -5.6828, -8.7007, -4.2650, -8.0987, -8.0837, -7.1799, -7.7863, -5.1605,
         -8.2832, -5.1088, -8.1051, -5.3985, -6.7129, -1.4109, -3.2241,  1.5863,
         -4.9714, -4.1138, -5.9107, -5.9786]], grad_fn=<SqueezeBackward1>)
    tensor([[-2.1025, -2.9121, -5.9192, -6.7459, -6.4667, -5.6418, -1.4504, -3.1943,
         -2.1024, -5.7470, -6.3381, -5.8520, -3.4871, -6.7667, -5.4711, -3.9885,
         -1.2502, -4.0869, -6.4930, -6.3751, -6.1309, -6.9721, -7.5558, -6.4056,
         -6.7456, -5.0527, -7.3854, -7.0440, -4.3720, -3.8936, -2.1085, -5.8211,
         -2.0906, -2.2184,  1.4268, -2.1026]], grad_fn=<SqueezeBackward1>)
    """

    #Converting scores tensors to numpy arrays
    start_token_scores = start_token_scores.detach().numpy().flatten()
    end_token_scores = end_token_scores.detach().numpy().flatten()
    """
    [-5.978666  -3.0541189 -7.7166095 -5.929051  -6.878973  -7.238004
    -1.8289301 -8.10058   -5.9786286 -3.9319289 -5.6229596 -4.191908
    -7.20684   -6.773916  -2.3959794 -5.942456  -5.6827617 -8.700695
    -4.265001  -8.09874   -8.083673  -7.179875  -7.7863474 -5.16046
    -8.283156  -5.108819  -8.1051235 -5.3984528 -6.7128663 -1.4108785
    -3.2240815  1.5863497 -4.9714    -4.113782  -5.9107194 -5.9786243]

    [-2.1025064 -2.912148  -5.9192414 -6.745929  -6.466673  -5.641759
    -1.4504088 -3.1943028 -2.1024144 -5.747039  -6.3380575 -5.852047
    -3.487066  -6.7667046 -5.471078  -3.9884708 -1.2501552 -4.0868535
    -6.4929943 -6.375147  -6.130891  -6.972091  -7.5557766 -6.405638
    -6.7455807 -5.0527067 -7.3854156 -7.043977  -4.37199   -3.8935976
    -2.1084964 -5.8210607 -2.0906193 -2.2184045  1.4268283 -2.1025767]
    """
    #Getting start and end index of answer based on highest scores
    answer_start_index = np.argmax(start_token_scores)
    answer_end_index = np.argmax(end_token_scores)
    """
    31
    31
    """

    #Getting scores for start and end token of the answer
    start_token_score = np.round(start_token_scores[answer_start_index], 2)
    end_token_score = np.round(end_token_scores[answer_end_index], 2)
    """
    6.64
    6.6
    """

    #Combining subwords starting with ## and get full words in output.
    #It is because tokenizer breaks words which are not in its vocab.
    answer = tokens[answer_start_index]
    for i in range(answer_start_index + 1, answer_end_index + 1):
        if tokens[i][0:2] == '##':
            answer += tokens[i][2:]
        else:
            answer += ' ' + tokens[i]

    # If the answer didn't find in the passage
    if ( answer_start_index == 0) or (start_token_score < 0 ) or  (answer == '[SEP]') or ( answer_end_index <  answer_start_index):
        answer = "Sorry!, I could not find an answer in the passage."

    return (answer_start_index, answer_end_index, start_token_score, end_token_score,  answer)

#Testing function
bert_question_answer("What is the name of my dog", "I have a loving household of 5 members and a dog in my immediate family. The name of my dog
```

```
(31, 31, 6.64, 6.6, 'bruno')
```

```
# Let me define one passage
passage = """Hello, I am Srishti Sharma. My name of my friend is Nirbhi. She is the daughter of Nirmala. I spend most of my time with mom and da
She always calls me by my nickname. Nirbhi calls me Doll. Except Sana, my other friend call me by my original name.
Bijay is also my friend. """

print (f'Length of the passage: {len(passage.split())} words')

question1 ="What is my name?"
print ('\nQuestion 1:\n', question1)
_, _ , _ , _, ans  = bert_question_answer( question1, passage)
print('\nAnswer from BERT: ', ans ,  '\n')


question2 ="Who is the mother of Nirbhi?"
print ('\nQuestion 2:\n', question2)
_, _ , _ , _, ans  = bert_question_answer( question2, passage)
print('\nAnswer from BERT: ', ans ,  '\n')
```

```
question3 ="With whom Srishti Sharma spend most of her time?"
print ('\nQuestion 3:\n', question3)
_, _ , _ , _, ans  = bert_question_answer( question3, passage)
print('\nAnswer from BERT: ', ans ,  '\n')

question4 ="Who is Srishti's other friend?"
print ('\nQuestion 4:\n', question4)
_, _ , _ , _, ans  = bert_question_answer( question4, passage)
print('\nAnswer from BERT: ', ans ,  '\n')
```

```
Length of the passage: 56 words

Question 1:
 What is my name?

Answer from BERT:  srishti sharma


Question 2:
 Who is the mother of Nirbhi?

Answer from BERT:  nirmala


Question 3:
 With whom Srishti Sharma spend most of her time?

Answer from BERT:  mom and dad


Question 4:
 Who is Srishti's other friend?

Answer from BERT:  sana
```

```
# Let me define another passage
passage= """NLP is a subfield of computer science and artificial intelligence concerned with interactions between
computers and human (natural) languages. It is used to apply machine learning algorithms to text and speech. For
example, we can use NLP to create systems like speech recognition, document summarization, machine translation, spam
detection, named entity recognition, question answering, autocomplete, predictive typing and so on. Nowadays, most of
us have smartphones that have speech recognition. These smartphones use NLP to understand what is said. Also, many
people use laptops which operating system has a built-in speech recognition. NLTK (Natural Language Toolkit) is a
leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces
to many corpora and lexical resources. Also, it contains a suite of text processing libraries for classification,
tokenization, stemming, tagging, parsing, and semantic reasoning. Best of all, NLTK is a free, open source,
community-driven project. We’ll use this toolkit to show some basics of the natural language processing field. For
the examples below, I’ll assume that we have imported the NLTK toolkit. We can do this like this: import nltk.
Sentence tokenization (also called sentence segmentation) is the problem of dividing a string of written language into
its component sentences. The idea here looks very simple. Word tokenization (also called word segmentation)
is the problem of dividing a string of written language into its component words. In English and many other languages
using some form of Latin alphabet, space is a good approximation of a word divider. However, we still can have problems
we only split by space to achieve the wanted results. Some English compound nouns are variably written and sometimes
they contain a space. In most cases, we use a library to achieve the wanted results, so again don’t worry too much
for the details. Stop words are words which are filtered out before or after processing of text. When applying machine
learning to text, these words can add a lot of noise. That’s why we want to remove these irrelevant words.
Stop words usually refer to the most common words such as “and”, “the”, “a” in a language, but there is no single
universal list of stopwords. The list of the stop words can change depending on your application. The NLTK tool has
a predefined list of stopwords that refers to the most common words. If you use it for your first time, you need to
download the stop words using this code: nltk.download(“stopwords”). Once we complete the downloading, we can load
the stopwords package from the nltk.corpus and use it to load the stop words."""

print (f'Length of the passage: {len(passage.split())} words')


question ="What is full form of NLTK"
print ('\nQuestion 1:\n', question)
_, _ , _ , _, ans  = bert_question_answer( question, passage)
print('\nAnswer from BERT: ', ans ,  '\n')

question ="What are stop words "
print ('\nQuestion 2:\n', question)
_, _ , _ , _, ans  = bert_question_answer( question, passage)
print('\nAnswer from BERT: ', ans ,  '\n')

question ="What is NLP "
print ('\nQuestion 3:\n', question)
_, _ , _ , _, ans  = bert_question_answer( question, passage)
print('\nAnswer from BERT: ', ans ,  '\n')

question ="How to download stop words from nltk"
print ('\nQuestion 4:\n', question)
print ('\nQuestion 4:\n', question)
_, _ , _ , _, ans  = bert_question_answer( question, passage)
print('\nAnswer from BERT: ', ans ,  '\n')

question ="What do smartphones use to understand speech recognition "
print ('\nQuestion 5:\n', question)
_, _ , _ , _, ans  = bert_question_answer( question, passage)
print('\nAnswer from BERT: ', ans ,  '\n')

question ="What is Computer vision"
print ('\nQuestion 6:\n', question)
_, _ , _ , _, ans  = bert_question_answer( question, passage)
print('\nAnswer from BERT: ', ans ,  '\n')

question ="What is supervised learning"
print ('\nQuestion 7:\n', question)
_, _ , _ , _, ans  = bert_question_answer( question, passage)
print('\nAnswer from BERT: ', ans ,  '\n')
```

```
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_
Length of the passage: 433 words

Question 1:
 What is full form of NLTK
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_

Answer from BERT:  natural language toolkit


Question 2:
 What are stop words
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_

Answer from BERT:  words which are filtered out before or after processing of text


Question 3:
 What is NLP
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_

Answer from BERT:  a subfield of computer science and artificial intelligence concerned with interactions between com


Question 4:
 How to download stop words from nltk
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_

Answer from BERT:  import nltk


Question 5:
 What do smartphones use to understand speech recognition
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_

Answer from BERT:  nlp


Question 6:
 What is Computer vision
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_

Answer from BERT:  artificial intelligence


Question 7:
 What is supervised learning

Answer from BERT:  Sorry!, I could not find an answer in the passage.
```

Question-Answering Application

```
#@title Question-Answering Application { vertical-output: true }
#@markdown ---
question= "Full form of BERT" #@param {type:"string"}
passage = "NLP stands for Natural Language Processing. NLP is an interdisciplinary subfield of linguistics, computer science, and artificial in
#@markdown ---

_, _ , _ , _, ans  = bert_question_answer( question, passage)
```

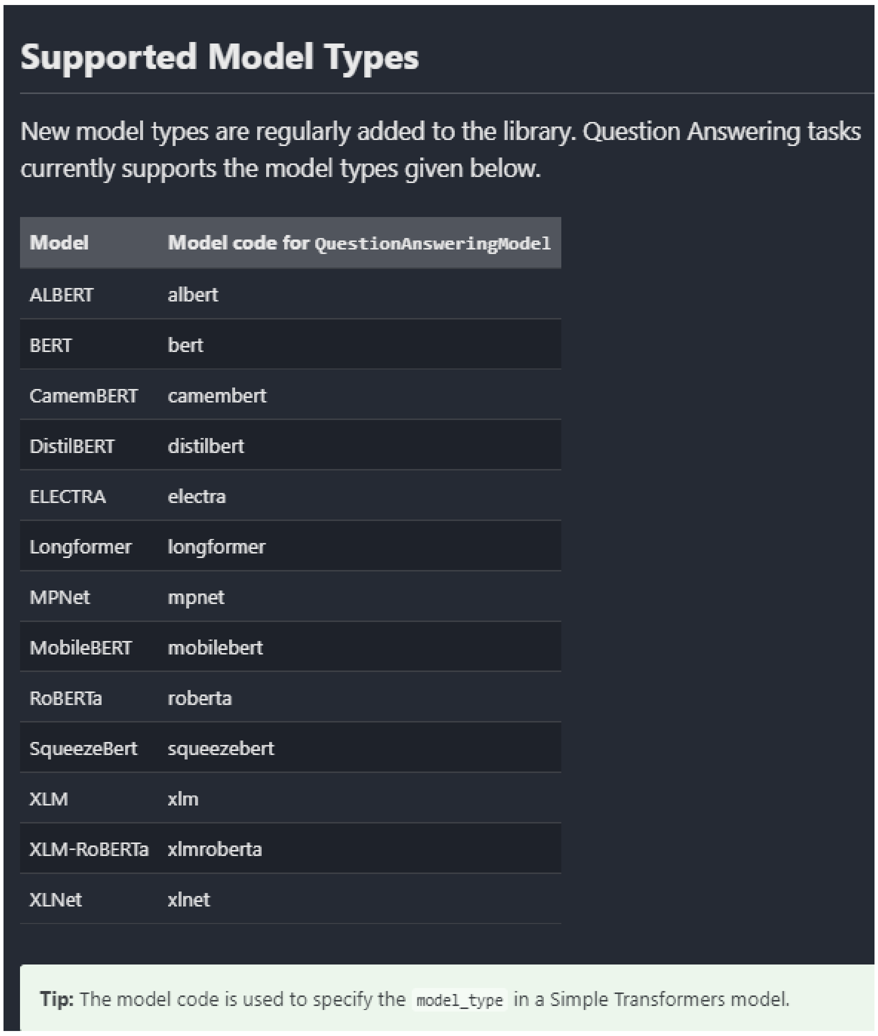
question: " Full form of BERT "

passage: " NLP stands for Natural Language Process "

```
#@markdown Answer:
print(ans)
```

Answer:

```
bi - directional encoder representation
```



Supported Model Types

New model types are regularly added to the library. Question Answering tasks currently supports the model types given below.

Model	Model code for `QuestionAnsweringModel`
ALBERT	albert
BERT	bert
CamemBERT	camembert
DistilBERT	distilbert
ELECTRA	electra
Longformer	longformer
MPNet	mpnet
MobileBERT	mobilebert
RoBERTa	roberta
SqueezeBert	squeezebert
XLM	xlm
XLM-RoBERTa	xlmroberta
XLNet	xlnet

Tip: The model code is used to specify the `model_type` in a Simple Transformers model.

```
pip install simpletransformers
```

```
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-au
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata>=1.4-
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich>
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.1
Requirement already satisfied: pytz-deprecation-shim in /usr/local/lib/python3.10/dist-packages (from tzlocal>=1.1
Requirement already satisfied: decorator>=3.4.0 in /usr/local/lib/python3.10/dist-packages (from validators>=0.2->
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->GitPython!=3.1.29,>=1.0.0->wandb>=0.10.32->simpletransformers)
  Downloading smmap-5.0.0-py3-none-any.whl (24 kB)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py<3.0.0,>=
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-module
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>
Requirement already satisfied: tzdata in /usr/local/lib/python3.10/dist-packages (from pytz-deprecation-shim->tzlo
Building wheels for collected packages: seqeval, validators, pathtools
  Building wheel for seqeval (setup.py) ... done
  Created wheel for seqeval: filename=seqeval-1.2.2-py3-none-any.whl size=16165 sha256=f7fbf49c7a59a929b4c7cd41f2c
  Stored in directory: /root/.cache/pip/wheels/1a/67/4a/ad4082dd7dfc30f2abfe4d80a2ed5926a506eb8a972b4767fa
  Building wheel for validators (setup.py) ... done
  Created wheel for validators: filename=validators-0.20.0-py3-none-any.whl size=19579 sha256=d07f508d8d1e0b27eb18
  Stored in directory: /root/.cache/pip/wheels/f2/ed/dd/d3a556ad245ef9dc570c6bcd2f22886d17b0b408dd3bbb9ac3
  Building wheel for pathtools (setup.py) ... done
  Created wheel for pathtools: filename=pathtools-0.1.2-py3-none-any.whl size=8791 sha256=cd42fd158f63c916cc85c870
  Stored in directory: /root/.cache/pip/wheels/e7/f3/22/152153d6eb222ee7a56ff8617d80ee5207207a8c00a7aab794
Successfully built seqeval validators pathtools
Installing collected packages: tokenizers, sentencepiece, pathtools, xxhash, watchdog, validators, smmap, setproct
Successfully installed GitPython-3.1.31 aiohttp-3.8.4 aiosignal-1.3.1 async-timeout-4.0.2 blinker-1.6.2 datasets-2
```

```
import json
with open(r"train.json", "r") as read_file:
    train = json.load(read_file)
```

```
train
```

```
[{'context': 'Mistborn is a series of epic fantasy novels written by American author Brandon Sanderson.',
  'qas': [{'id': '00001',
    'is_impossible': False,
    'question': 'Who is the author of the Mistborn series?',
    'answers': [{'text': 'Brandon Sanderson', 'answer_start': 71}]}]},
 {'context': 'The first series, published between 2006 and 2008, consists of The Final Empire,The Well of Ascension,
and The Hero of Ages.',
  'qas': [{'id': '00002',
    'is_impossible': False,
    'question': 'When was the series published?',
    'answers': [{'text': 'between 2006 and 2008', 'answer_start': 28}]},
   {'id': '00003',
    'is_impossible': False,
    'question': 'What are the three books in the series?',
    'answers': [{'text': 'The Final Empire, The Well of Ascension, and The Hero of Ages',
      'answer_start': 63}]},
   {'id': '00004',
    'is_impossible': True,
    'question': 'Who is the main character in the series?',
    'answers': []}]}]
```

```
with open(r"test.json", "r") as read_file:
    test = json.load(read_file)
```

```
test
```

```
[{'context': 'The series primarily takes place in a region called the Final Empire on a world called Scadrial, where
the sun and sky are red, vegetation is brown, and the ground is constantly being covered under black volcanic
ashfalls.',
  'qas': [{'id': '00001',
    'is_impossible': False,
    'question': 'Where does the series take place?',
    'answers': [{'text': 'region called the Final Empire', 'answer_start': 38},
     {'text': 'world called Scadrial', 'answer_start': 74}]}]},
 {'context': '"Mistings" have only one of the many Allomantic powers, while "Mistborns" have all the powers.',
  'qas': [{'id': '00002',
    'is_impossible': False,
    'question': 'How many powers does a Misting possess?',
    'answers': [{'text': 'one', 'answer_start': 21}]},
   {'id': '00003',
    'is_impossible': True,
    'question': 'What are Allomantic powers?',
    'answers': []}]}]
```

```
with open(r"predictions.json", "r") as read_file:
    predictions = json.load(read_file)
```

```
predictions
```

```
[{'context': 'Vin is a Mistborn of great power and skill.',
  'qas': [{'question': "What is Vin's speciality?", 'id': '0'}]}]
```

```
import logging

from simpletransformers.question_answering import QuestionAnsweringModel, QuestionAnsweringArgs
```

```
model_type="bert"
model_name= "bert-base-cased"
if model_type == "bert":
    model_name = "bert-base-cased"

elif model_type == "roberta":
    model_name = "roberta-base"

elif model_type == "distilbert":
    model_name = "distilbert-base-cased"

elif model_type == "distilroberta":
    model_type = "roberta"
    model_name = "distilroberta-base"

elif model_type == "electra-base":
    model_type = "electra"
    model_name = "google/electra-base-discriminator"

elif model_type == "electra-small":
    model_type = "electra"
    model_name = "google/electra-small-discriminator"

elif model_type == "xlnet":
    model_name = "xlnet-base-cased"
```

```
# Configure the model
model_args = QuestionAnsweringArgs()
model_args.train_batch_size = 16
model_args.evaluate_during_training = True
model_args.n_best_size=3
model_args.num_train_epochs=10
```

```
pip install wandb
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: wandb in /usr/local/lib/python3.10/dist-packages (0.15.3)
Requirement already satisfied: Click!=8.0.0,>=7.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (8.1.3)
Requirement already satisfied: GitPython!=3.1.29,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (3.1
Requirement already satisfied: requests<3,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (2.27.1)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (5.9.5)
Requirement already satisfied: sentry-sdk>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (1.24.0)
Requirement already satisfied: docker-pycreds>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (0.4.0)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from wandb) (6.0)
Requirement already satisfied: pathtools in /usr/local/lib/python3.10/dist-packages (from wandb) (0.1.2)
Requirement already satisfied: setproctitle in /usr/local/lib/python3.10/dist-packages (from wandb) (1.3.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from wandb) (67.7.2)
Requirement already satisfied: appdirs>=1.4.3 in /usr/local/lib/python3.10/dist-packages (from wandb) (1.4.4)
Requirement already satisfied: protobuf!=4.21.0,<5,>=3.19.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from docker-pycreds>=0.4.0->wan
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-packages (from GitPython!=3.1.29,>=1
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests<3,
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wand
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from gitdb<5,>=4.0.1->GitP
```

```
### Advanced Methodology
train_args = {
    "reprocess_input_data": True,
    "overwrite_output_dir": True,
    "use_cached_eval_features": True,
    "output_dir": f"outputs/{model_type}",
    "best_model_dir": f"outputs/{model_type}/best_model",
    "evaluate_during_training": True,
    "max_seq_length": 128,
    "num_train_epochs": 20,
    "evaluate_during_training_steps": 1000,
    "wandb_project": "Question Answer Application",
    "wandb_kwargs": {"name": model_name},
    "save_model_every_epoch": False,
    "save_eval_checkpoints": False,
    "n_best_size":3,
    # "use_early_stopping": True,
    # "early_stopping_metric": "mcc",
    # "n_gpu": 2,
    # "manual_seed": 4,
    # "use_multiprocessing": False,
    "train_batch_size": 128,
    "eval_batch_size": 64,
    # "config": {
    #     "output_hidden_states": True
    # }
}
```

Initializing the model

```
model = QuestionAnsweringModel(
    model_type,model_name, args=train_args)
# use_cuda=False
```

```
Some weights of the model checkpoint at bert-base-cased were not used when initializing BertForQuestionAnswering: ['c
- This IS expected if you are initializing BertForQuestionAnswering from the checkpoint of a model trained on another
- This IS NOT expected if you are initializing BertForQuestionAnswering from the checkpoint of a model that you expec
Some weights of BertForQuestionAnswering were not initialized from the model checkpoint at bert-base-cased and are nev
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
# Train the model
model.train_model(train, eval_data=test)
```

```
convert squad examples to features:   0%|          | 0/4 [00:00<?, ?it/s]Could not find answer: 'The Final Empire,The
convert squad examples to features: 100%|██████████| 4/4 [00:00<00:00, 353.44it/s]
add example index and unique id: 100%|██████████| 4/4 [00:00<00:00, 9857.35it/s]
Epoch 20 of 20: 100%                                                    20/20 [00:18<00:00, 5.39it/s]
```

Finishing last run (ID:g4fnc4fe) before initializing another...
Waiting for W&B process to finish... **(success).**

0.001 MB of 0.001 MB uploaded (0.000 MB deduped)

Run history:

correct	
eval_loss	
global_step	
incorrect	
similar	
train_loss	

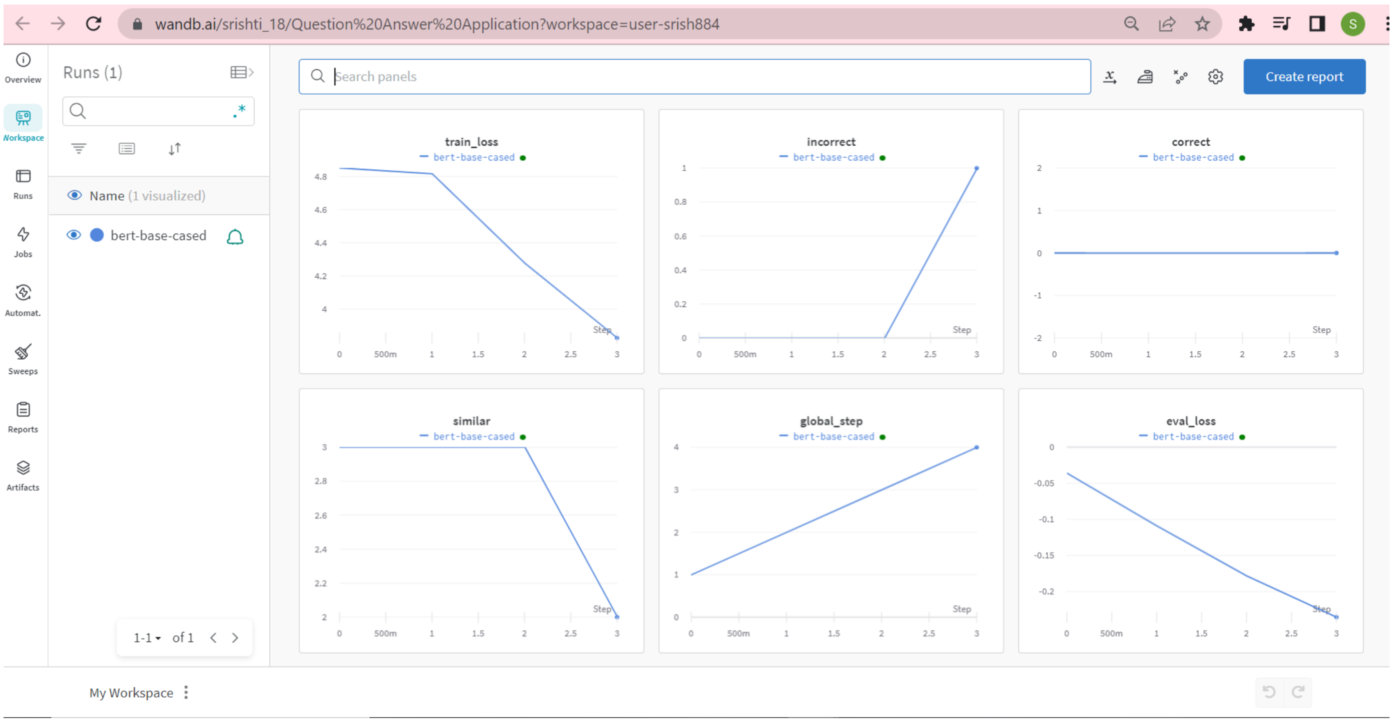
Run summary:

correct	0
eval_loss	-0.43433
global_step	10
incorrect	1
similar	2
train_loss	1.82129

View run **bert-base-cased** at: https://wandb.ai/srishti_18/Question%20Answer%20Application/runs/g4fnc4fe
Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)
Find logs at: `./wandb/run-20230525_094553-g4fnc4fe/logs`
Successfully finished last run (ID:g4fnc4fe). Initializing new run:
Tracking run with wandb version 0.15.3
Run data is saved locally in `/content/wandb/run-20230525_100355-folvkuwi`
Syncing run **bert-base-cased** to Weights & Biases (docs)
View project at https://wandb.ai/srishti_18/Question%20Answer%20Application
View run at https://wandb.ai/srishti_18/Question%20Answer%20Application/runs/folvkuwi

```
Epochs 0/20. Running Loss: 4.9486: 100%                          1/1 [00:00<00:00, 6.81it/s]
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:139: UserWarning: Detected call of `lr_scheduler.
  warnings.warn("Detected call of `lr_scheduler.step()` before `optimizer.step()`. "
Running Evaluation: 100%                                 1/1 [00:00<00:00, 23.20it/s]
Epochs 1/20. Running Loss: 5.0254: 100%                          1/1 [00:00<00:00, 9.11it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 18.42it/s]
Epochs 2/20. Running Loss: 4.6686: 100%                          1/1 [00:00<00:00, 8.54it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 15.98it/s]
Epochs 3/20. Running Loss: 4.2119: 100%                          1/1 [00:00<00:00, 10.30it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 17.75it/s]
Epochs 4/20. Running Loss: 3.6374: 100%                          1/1 [00:00<00:00, 10.22it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 16.28it/s]
Epochs 5/20. Running Loss: 2.9554: 100%                          1/1 [00:00<00:00, 9.76it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 17.59it/s]
Epochs 6/20. Running Loss: 2.5465: 100%                          1/1 [00:00<00:00, 10.07it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 16.96it/s]
Epochs 7/20. Running Loss: 2.0086: 100%                          1/1 [00:00<00:00, 10.10it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 20.44it/s]
Epochs 8/20. Running Loss: 1.7493: 100%                          1/1 [00:00<00:00, 9.84it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 17.58it/s]
Epochs 9/20. Running Loss: 1.4548: 100%                          1/1 [00:00<00:00, 10.23it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 18.69it/s]
Epochs 10/20. Running Loss: 1.2037: 100%                          1/1 [00:00<00:00, 8.62it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 18.37it/s]
Epochs 11/20. Running Loss: 1.1586: 100%                          1/1 [00:00<00:00, 9.44it/s]
Running Evaluation: 100%                                 1/1 [00:00<00:00, 20.33it/s]
Epochs 12/20. Running Loss: 0.9927: 100%                          1/1 [00:00<00:00, 9.09it/s]
```

```
Running Evaluation: 100%                                        1/1 [00:00<00:00, 17.01it/s]
```



```
–,
```

```
#Evaluate the model
results,texts=model.eval_model(test)
```

```
Running Evaluation: 100%                                        1/1 [00:00<00:00, 19.28it/s]
    9,
```

```
results
```

```
{'correct': 1, 'similar': 2, 'incorrect': 0, 'eval_loss': -2.12109375}
   14,
```

```
texts
```

```
{'correct_text': {'00003': ''},
 'similar_text': {'00001': {'truth': 'region called the Final Empire',
   'predicted': '',
   'question': 'Where does the series take place?'},
  '00002': {'truth': 'one',
   'predicted': '',
   'question': 'How many powers does a Misting possess?'}},
 'incorrect_text': {}}
   4.6686201095581055,
```

```
# Make predictions with the model
to_predict = [
    {
        "context": "Vin is a Mistborn of great power and skill.",
        "qas": [
            {
                "question": "What is Vin's speciality?",
                "id": "0",
            }
        ],
    }
]
```

```
answers, probabilities = model.predict(to_predict)

print(answers)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 119.02it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 3146.51it/s]
Running Prediction: 100%                                        1/1 [00:00<00:00, 13.26it/s]
[{'id': '0', 'answer': ['', 'born of great power and']}]
   -1.50078125,
   -1.4970703125,
```