Import Library

In [ ]:
```
from torch.utils.data import DataLoader
from torch.utils.data import DataLoader
import warnings,transformers,logging,torch
from transformers import TrainingArguments,Trainer
from transformers import AutoModelForSequenceClassification,AutoTokenizer
from datasets import load_dataset, Dataset, DatasetDict
import datasets
import pandas as pd
import numpy as np
```

```
/workspace/miniconda3/envs/nlp/lib/python3.7/site-packages/tqdm/auto.py:22: TqdmWarnin
g: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.re
adthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [ ]:
```
data = pd.read_csv('/workspace/xxl/train.csv')
data.head(5)
```

Out[ ]:

	discourse_id	essay_id	discourse_text	discourse_type	discourse_effectiveness
0	0013cc385424	007ACE74B050	Hi, i'm Isaac, i'm going to be writing about h...	Lead	Adequate
1	9704a709b505	007ACE74B050	On my perspective, I think that the face is a ...	Position	Adequate
2	c22adee811b6	007ACE74B050	I think that the face is a natural landform be...	Claim	Adequate
3	a10d361e54e4	007ACE74B050	If life was on Mars, we would know by now. The...	Evidence	Adequate
4	db3e453ec4e2	007ACE74B050	People thought that the face was formed by ali...	Counterclaim	Adequate

Load Model

Hugging Face Models：https://huggingface.co/models?sort=downloads&search=deberta

In [ ]:
```
tokz = AutoTokenizer.from_pretrained('microsoft/deberta-v3-small')
```

```
Special tokens have been added in the vocabulary, make sure the associated word embeddi
ngs are fine-tuned or trained.
/workspace/miniconda3/envs/nlp/lib/python3.7/site-packages/transformers/convert_slow_to
kenizer.py:435: UserWarning: The sentencepiece tokenizer that you are converting to a f
ast tokenizer uses the byte fallback option which is not implemented in the fast tokeni
zers. In practice this means that the fast version of the tokenizer can produce unknown
tokens whereas the sentencepiece version would have converted these unknown tokens into
a sequence of byte tokens matching the original piece of text.
  "The sentencepiece tokenizer that you are converting to a fast tokenizer uses the byt
e fallback option"
Special tokens have been added in the vocabulary, make sure the associated word embeddi
ngs are fine-tuned or trained.
```

In [ ]:
```
# 查看该模型的句间连接词是什么(这个地方理解可能有误)
sep = tokz.sep_token
sep
```

Out[ ]:
```
'[SEP]'
```

Data Processing

In [ ]:
```
# 将议论元素的类型和文本内容拼接在一起
data['inputs']= data.discourse_type + sep +data.discourse_text
data.head(5)
```

Out[ ]:

	discourse_id	essay_id	discourse_text	discourse_type	discourse_effectiveness	
0	0013cc385424	007ACE74B050	Hi, i'm Isaac, i'm going to be writing about h...	Lead	Adequate	Lead[SEP]Hi i'm going to
1	9704a709b505	007ACE74B050	On my perspective, I think that the face is a ...	Position	Adequate	Position[S perspective, I
2	c22adee811b6	007ACE74B050	I think that the face is a natural landform be...	Claim	Adequate	Claim[SEP]I the face is a
3	a10d361e54e4	007ACE74B050	If life was on Mars, we would know by now. The...	Evidence	Adequate	Evidence[SEP on Mars, we w
4	db3e453ec4e2	007ACE74B050	People thought that the face was formed by ali...	Counterclaim	Adequate	Counterclaim[S thought that

In [ ]:

```
# 将议论元素的有效性(Ineffective,Adequate,Effective)数字化，并将列标签名称改为'lable',
new_label = {"discourse_effectiveness": {"Ineffective": 0, "Adequate": 1, "Effective":
data = data.replace(new_label)
data = data.rename(columns = {"discourse_effectiveness": "label"})
data.head(5)
```

Out[ ]:

	discourse_id	essay_id	discourse_text	discourse_type	label	inputs
0	0013cc385424	007ACE74B050	Hi, i'm Isaac, i'm going to be writing about h...	Lead	1	Lead[SEP]Hi, i'm Isaac, i'm going to be writin...
1	9704a709b505	007ACE74B050	On my perspective, I think that the face is a ...	Position	1	Position[SEP]On my perspective, I think that t...
2	c22adee811b6	007ACE74B050	I think that the face is a natural landform be...	Claim	1	Claim[SEP]I think that the face is a natural l...
3	a10d361e54e4	007ACE74B050	If life was on Mars, we would know by now. The...	Evidence	1	Evidence[SEP]If life was on Mars, we would kno...
4	db3e453ec4e2	007ACE74B050	People thought that the face was formed by ali...	Counterclaim	1	Counterclaim[SEP]People thought that the face ...

In [ ]:

```
# 利用 dataset.Dataset() 将数据由 Dataframe 格式转换成 Dataset 格式，便于输入模型
data_1 = Dataset.from_pandas(data)
print(data_1)
# 查看一个样本数据
print(data_1[0])
```

```
Dataset({
    features: ['discourse_id', 'essay_id', 'discourse_text', 'discourse_type', 'label',
'inputs'],
    num_rows: 36765
})
{'discourse_id': '0013cc385424', 'essay_id': '007ACE74B050', 'discourse_text': "Hi, i'm
Isaac, i'm going to be writing about how this face on Mars is a natural landform or if
there is life on Mars that made it. The story is about how NASA took a picture of Mars
and a face was seen on the planet. NASA doesn't know if the landform was created by lif
e on Mars, or if it is just a natural landform. ", 'discourse_type': 'Lead', 'label':
1, 'inputs': "Lead[SEP]Hi, i'm Isaac, i'm going to be writing about how this face on Ma
rs is a natural landform or if there is life on Mars that made it. The story is about h
ow NASA took a picture of Mars and a face was seen on the planet. NASA doesn't know if
the landform was created by life on Mars, or if it is just a natural landform. "}
```

In [ ]:
```
# 定义一个函数，为后面调用 dataset.map
# 这里的 tokz 是前面tokz = AutoTokenizer.from_pretrained('microsoft/deberta-v3-small')
# 是为了将内容分词，然后变成序列化的数字
def tok_func(x): return tokz(x["inputs"], truncation=True)
```

In [ ]:
```
# 举个例子，将 data_1 的第一个数据使用 tokenizer 变成序列化的数字
tok_func(data_1[0])
```

Out[ ]:
```
{'input_ids': [1, 8380, 2, 2684, 261, 584, 280, 358, 11759, 261, 584, 280, 358, 446, 26
4, 282, 898, 314, 361, 291, 812, 277, 7583, 269, 266, 1008, 1311, 9930, 289, 337, 343,
269, 432, 277, 7583, 272, 412, 278, 260, 279, 697, 269, 314, 361, 7658, 681, 266, 1355,
265, 7583, 263, 266, 812, 284, 757, 277, 262, 3568, 260, 7658, 702, 280, 297, 391, 337,
262, 1311, 9930, 284, 994, 293, 432, 277, 7583, 261, 289, 337, 278, 269, 348, 266, 100
8, 1311, 9930, 260, 2], 'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'attention_mask': [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, 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, 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, 1, 1, 1, 1, 1, 1]}
```

In [ ]:
```
# 删除不必要的列，只需要保留经过序列化的 'input_ids' 和 'label'列
#  'input_ids', 'token_type_ids', 'attention_mask' 是经过 tokenizer 后自动生成的列(具体
# map 函数就是传入一个函数参数，然后原来的数据集按照这个函数变化
inps = "discourse_text","discourse_type"
data_2 = data_1.map(tok_func, batched=True, remove_columns=inps+('inputs','discourse_i
data_2
```

```
100%|██████████| 37/37 [00:01<00:00, 19.07ba/s]
```

Out[ ]:
```
Dataset({
    features: ['label', 'input_ids', 'token_type_ids', 'attention_mask'],
    num_rows: 36765
})
```

Split Dataset

In [ ]:
```
# 选取随机种子，目的是为了每次跑这个代码时，产生的随机数是相同的，保持复现结果相同
# 打乱数据集
essay_ids = data.essay_id.unique()
np.random.seed(42)
np.random.shuffle(essay_ids)
essay_ids[:10]
```

Out[ ]:
```
array(['B5C606F0A883', 'FA4FE7706A1A', '37A77BEAD718', '0ED28D8A5EC4',
       'F25BA634ADDD', '1065173BBE31', '763EF698F56B', '1B976DF43007',
       '9922F6B9A55B', 'A187E6D70752'], dtype=object)
```

In [ ]:
```
# 确定测试集、训练集比例，制作训练集、测试集的索引
val_prop = 0.2
val_sz = int(len(essay_ids)*val_prop)
val_essay_ids = essay_ids[:val_sz]
is_val = np.isin(data.essay_id, val_essay_ids)
idxs = np.arange(len(data))
val_idxs = idxs[ is_val]
trn_idxs = idxs[~is_val]
print(len(val_idxs),len(trn_idxs))
print(val_idxs,trn_idxs)
```

```
7181 29584
[   53    54    55 ... 36762 36763 36764] [    0     1     2 ... 36757 36758 36759]
```

Get Finally Dataset

In [ ]:
```
data_3 = DatasetDict({"train":data_2.select(trn_idxs),
             "test": data_2.select(val_idxs)})
```

In [ ]:
```
data_3
```

Out[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['label', 'input_ids', 'token_type_ids', 'attention_mask'],
        num_rows: 29584
    })
    test: Dataset({
        features: ['label', 'input_ids', 'token_type_ids', 'attention_mask'],
        num_rows: 7181
    })
})
```

In [ ]:
```
type(data_3['train'][0]['input_ids'])
```

Out[ ]:
```
list
```

Set A Function

In [ ]:
```
# 上面这些步骤我们可以编写一个函数来统一调用
def get_data(data, train=True):
    ds = Dataset.from_pandas(data)
    to_remove = ['discourse_text','discourse_type','inputs','discourse_id','essay_id']
    data_out = ds.map(tok_func, batched=True, remove_columns=to_remove)
    if train:
        return DatasetDict({"train":data_out.select(trn_idxs), "test": data_out.select
    else:
        return data_out
```

In [ ]:
```
# 使用函数会和上面结果是一样的
data_out = get_data(data)
data_out
```

```
100%|██████████| 37/37 [00:01<00:00, 20.89ba/s]
```

Out[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['label', 'input_ids', 'token_type_ids', 'attention_mask'],
        num_rows: 29584
    })
    test: Dataset({
        features: ['label', 'input_ids', 'token_type_ids', 'attention_mask'],
        num_rows: 7181
    })
})
```