Torch和torchvision需要根据detectron2 调整版本号 \ 我使用的是 PyTorch 1.10.0,Python 3.8(ubuntu20.04),Cuda 11.3

In [1]:

```
import torch
import torchvision
import cv2
#用于处理和评估COCO（Common Objects in Context）数据集
import pycocotools
#PyTorch的计算机视觉库，用于目标检测、实例分割和姿态估计等任务。
import detectron2
#用于使用Tesseract OCR引擎进行光学字符识别（OCR）.
import pytesseract
from PIL import Image, ImageDraw, ImageFont
from transformers import LayoutLMv2ForTokenClassification

print(torch.__version__)
print(torchvision.__version__)
print(cv2.__version__)
```

```
1.10.0+cu113
0.11.1+cu113
4.9.0
```

Layoutlmv2模型doc: https://huggingface.co/docs/transformers/model_doc/layoutlmv2

Layoutlmv2模型下载：https://huggingface.co/microsoft/layoutlmv2-base-uncased

Layoutlmv3中文模型doc: https://huggingface.co/docs/transformers/model_doc/layoutlmv3

Layoutlmv3中文模型下载：https://huggingface.co/microsoft/layoutlmv3-base-chinese

In [2]:

```
# AutoDL的学术加速器
import subprocess
import os
result = subprocess.run('bash -c "source /etc/network_turbo && env | grep proxy"', she
output = result.stdout
for line in output.splitlines():
    if '=' in line:
        var, value = line.split('=', 1)
        os.environ[var] = value
```

In [3]:

```
import numpy as np
from transformers import LayoutLMv2Processor, LayoutLMv2Tokenizer, LayoutLMv2ForToke
from datasets import load_dataset, Dataset, Features, Sequence, ClassLabel, Value, A
import torch
from PIL import Image, ImageDraw, ImageFont
from tqdm.notebook import tqdm
from datasets import load_dataset
```

Layoutlmv2部署：

Dataset 选用 funsd 数据集 https://guillaumejaume.github.io/FUNSD/

其中包含149训练集，50测试集

In [4]:

```
datasets = load_dataset("nielsr/funsd")
datasets
```

```
/root/miniconda3/lib/python3.8/site-packages/datasets/load.py:1454: FutureWarning: The
repository for nielsr/funsd contains custom code which must be executed to correctly l
oad the dataset. You can inspect the repository content at https://hf.co/datasets/niel
sr/funsd
You can avoid this message in future by passing the argument `trust_remote_code=True`.
Passing `trust_remote_code=True` will be mandatory to load this dataset from the next
major release of `datasets`.
  warnings.warn(
```

Out[4]:

```
DatasetDict({
    train: Dataset({
        features: ['id', 'words', 'bboxes', 'ner_tags', 'image_path'],
        num_rows: 149
    })
    test: Dataset({
        features: ['id', 'words', 'bboxes', 'ner_tags', 'image_path'],
        num_rows: 50
    })
})
```

In [5]:

```
print(datasets)
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'words', 'bboxes', 'ner_tags', 'image_path'],
        num_rows: 149
    })
    test: Dataset({
        features: ['id', 'words', 'bboxes', 'ner_tags', 'image_path'],
        num_rows: 50
    })
})
```

数据input类型： id： type:list，包含元素为string “0”, “1”, “2”....

Tokens： type:list of list, 大list里面的每个小list所包含的一个image所包含的所有text或者token。例子： ['R&D', ':', 'Suggestion:', 'Date:', 'Licensee', 'Yes', 'No', '597005708', 'R&D', 'QUALITY', 'IMPROVEMENT', 'SUGGESTION/', 'SOLUTION', 'FORM', 'Name']

Bboxes：type：list of list，大list里面的每个小list所包含的一个image所包含的所有text或者token的boarding box，一个boarding box 为list，包含四个数字，范围在0-1000之间，必须为long或者int。前两个数字代表token 左上角的x,y值,后两个数字代表token右下角的x,y值。

Ner_tags: type：list，包含所有token对应的label坐标，Classlabel为 ['O', 'B-HEADER', 'I-HEADER', 'B-QUESTION', 'I-QUESTION', 'B-ANSWER', 'I-ANSWER']。比如 token1对应的label为‘B-QUESTION’,拿list里面所包含的应该是 3，为label在classlabel里面的index数。为int。

Image_path： type:list，包含所有image的路径。类型是string。

Layoutlmv3 image_path输入改为Image，type:list，包含所有image。类型需要用Image.open().convert("RGB")，对image进行保存。

In [6]:

```
labels = datasets['train'].features['ner_tags'].feature.names
print(labels)
```

```
['O', 'B-HEADER', 'I-HEADER', 'B-QUESTION', 'I-QUESTION', 'B-ANSWER', 'I-ANSWER']
```

In [7]:

```
id2label = {v: k for v, k in enumerate(labels)}
print(id2label)
```

```
label2id = {k: v for v, k in enumerate(labels)}
print(label2id)
```

```
{0: 'O', 1: 'B-HEADER', 2: 'I-HEADER', 3: 'B-QUESTION', 4: 'I-QUESTION', 5: 'B-ANSWE
R', 6: 'I-ANSWER'}
{'O': 0, 'B-HEADER': 1, 'I-HEADER': 2, 'B-QUESTION': 3, 'I-QUESTION': 4, 'B-ANSWER':
5, 'I-ANSWER': 6}
```

In [8]:

```
example = datasets["test"][0]
print(example.keys())
```

```
dict_keys(['id', 'words', 'bboxes', 'ner_tags', 'image_path'])
```

In [9]:

```
image = Image.open(example['image_path'])
image = image.convert("RGB")
```

文档图像分类: LayoutLMv2ForSequenceClassification，LayoutLMv3ForSequenceClassification

表格和收据的理解: LayoutLMv2ForTokenClassification，LayoutLMv3ForTokenClassification

文档视觉问: LayoutLMv2ForQuestionAnswering，LayoutLMv3ForQuestionAnswering

In [10]:

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
processor = LayoutLMv2Processor.from_pretrained("microsoft/layoutlmv2-base-uncased", r
model = LayoutLMv2ForTokenClassification.from_pretrained("microsoft/layoutlmv2-base-ur
model.to(device)
```

```
Some weights of LayoutLMv2ForTokenClassification were not initialized from the model c
heckpoint at microsoft/layoutlmv2-base-uncased and are newly initialized: ['classifie
r.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for pr
edictions and inference.
```

Out[10]:

```
LayoutLMv2ForTokenClassification(
  (layoutlmv2): LayoutLMv2Model(
    (embeddings): LayoutLMv2Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (x_position_embeddings): Embedding(1024, 128)
      (y_position_embeddings): Embedding(1024, 128)
      (h_position_embeddings): Embedding(1024, 128)
      (w_position_embeddings): Embedding(1024, 128)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (visual): LayoutLMv2VisualBackbone(
      (backbone): FPN(
        (fpn_lateral2): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
        (fpn_output2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
        (fpn_lateral3): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
        (fpn_output3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
        (fpn_lateral4): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1))
        (fpn_output4): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
        (fpn_lateral5): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1))
        (fpn_output5): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
        (top_block): LastLevelMaxPool()
        (bottom_up): ResNet(
          (stem): BasicStem(
            (conv1): Conv2d(
              3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False
              (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
            )
          )
          (res2): Sequential(
            (0): BottleneckBlock(
              (shortcut): Conv2d(
                64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
              (conv1): Conv2d(
                64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
              (conv2): Conv2d(
                256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=3
2, bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
              (conv3): Conv2d(
                256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
            )
            (1): BottleneckBlock(
              (conv1): Conv2d(
                256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
              (conv2): Conv2d(
                256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=3
2, bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
              (conv3): Conv2d(
                256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
            )
            (2): BottleneckBlock(
              (conv1): Conv2d(
                256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
              (conv2): Conv2d(
                256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=3
2, bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
              (conv3): Conv2d(
                256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
              )
            )
          )
          (res3): Sequential(
            (0): BottleneckBlock(
              (shortcut): Conv2d(
                256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv1): Conv2d(
                256, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv2): Conv2d(
                512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=3
2, bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv3): Conv2d(
                512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
            )
            (1): BottleneckBlock(
              (conv1): Conv2d(
                512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv2): Conv2d(
                512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=3
2, bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv3): Conv2d(
                512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
            )
            (2): BottleneckBlock(
              (conv1): Conv2d(
                512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv2): Conv2d(
                512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=3
2, bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv3): Conv2d(
                512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
            )
            (3): BottleneckBlock(
              (conv1): Conv2d(
                512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv2): Conv2d(
                512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=3
2, bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
              (conv3): Conv2d(
                512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
              )
            )
          )
          (res4): Sequential(
            (0): BottleneckBlock(
              (shortcut): Conv2d(
                512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv1): Conv2d(
                512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (1): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (2): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (3): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (4): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (5): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (6): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (7): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (8): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
```

```
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (9): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (10): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (11): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (12): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (13): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (14): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (15): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (16): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (17): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (18): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (19): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (20): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (21): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
            (22): BottleneckBlock(
              (conv1): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv2): Conv2d(
                1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
              (conv3): Conv2d(
                1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
              )
            )
          )
          (res5): Sequential(
            (0): BottleneckBlock(
              (shortcut): Conv2d(
                1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
              (conv1): Conv2d(
                1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
              (conv2): Conv2d(
                2048, 2048, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
              (conv3): Conv2d(
                2048, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
            )
            (1): BottleneckBlock(
              (conv1): Conv2d(
                2048, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
              (conv2): Conv2d(
                2048, 2048, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
              (conv3): Conv2d(
                2048, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
            )
            (2): BottleneckBlock(
              (conv1): Conv2d(
                2048, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
              (conv2): Conv2d(
                2048, 2048, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=
32, bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
              (conv3): Conv2d(
                2048, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
                (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
              )
            )
          )
        )
      )
      (pool): AdaptiveAvgPool2d(output_size=[7, 7])
    )
    (visual_proj): Linear(in_features=256, out_features=768, bias=True)
    (visual_LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (visual_dropout): Dropout(p=0.1, inplace=False)
    (encoder): LayoutLMv2Encoder(
      (layer): ModuleList(
        (0): LayoutLMv2Layer(
          (attention): LayoutLMv2Attention(
            (self): LayoutLMv2SelfAttention(
              (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): LayoutLMv2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): LayoutLMv2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): LayoutLMv2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (1): LayoutLMv2Layer(
          (attention): LayoutLMv2Attention(
            (self): LayoutLMv2SelfAttention(
              (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): LayoutLMv2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): LayoutLMv2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): LayoutLMv2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (2): LayoutLMv2Layer(
          (attention): LayoutLMv2Attention(
            (self): LayoutLMv2SelfAttention(
              (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): LayoutLMv2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): LayoutLMv2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): LayoutLMv2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (3): LayoutLMv2Layer(
          (attention): LayoutLMv2Attention(
            (self): LayoutLMv2SelfAttention(
              (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): LayoutLMv2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): LayoutLMv2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): LayoutLMv2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (4): LayoutLMv2Layer(
          (attention): LayoutLMv2Attention(
            (self): LayoutLMv2SelfAttention(
              (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): LayoutLMv2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): LayoutLMv2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): LayoutLMv2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
```

```
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(5): LayoutLMv2Layer(
  (attention): LayoutLMv2Attention(
    (self): LayoutLMv2SelfAttention(
      (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): LayoutLMv2SelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): LayoutLMv2Intermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
    (intermediate_act_fn): GELUActivation()
  )
  (output): LayoutLMv2Output(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(6): LayoutLMv2Layer(
  (attention): LayoutLMv2Attention(
    (self): LayoutLMv2SelfAttention(
      (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): LayoutLMv2SelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): LayoutLMv2Intermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
    (intermediate_act_fn): GELUActivation()
  )
  (output): LayoutLMv2Output(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(7): LayoutLMv2Layer(
  (attention): LayoutLMv2Attention(
    (self): LayoutLMv2SelfAttention(
      (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): LayoutLMv2SelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): LayoutLMv2Intermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
    (intermediate_act_fn): GELUActivation()
  )
  (output): LayoutLMv2Output(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(8): LayoutLMv2Layer(
  (attention): LayoutLMv2Attention(
    (self): LayoutLMv2SelfAttention(
      (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): LayoutLMv2SelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): LayoutLMv2Intermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
    (intermediate_act_fn): GELUActivation()
  )
  (output): LayoutLMv2Output(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(9): LayoutLMv2Layer(
  (attention): LayoutLMv2Attention(
    (self): LayoutLMv2SelfAttention(
      (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): LayoutLMv2SelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): LayoutLMv2Intermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
    (intermediate_act_fn): GELUActivation()
  )
  (output): LayoutLMv2Output(
    (dense): Linear(in_features=3072, out_features=768, bias=True)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
(10): LayoutLMv2Layer(
  (attention): LayoutLMv2Attention(
    (self): LayoutLMv2SelfAttention(
      (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (output): LayoutLMv2SelfOutput(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (intermediate): LayoutLMv2Intermediate(
    (dense): Linear(in_features=768, out_features=3072, bias=True)
```

```
            (intermediate_act_fn): GELUActivation()
          )
          (output): LayoutLMv2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (11): LayoutLMv2Layer(
          (attention): LayoutLMv2Attention(
            (self): LayoutLMv2SelfAttention(
              (qkv_linear): Linear(in_features=768, out_features=2304, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): LayoutLMv2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): LayoutLMv2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): LayoutLMv2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
      (rel_pos_bias): Linear(in_features=32, out_features=12, bias=False)
      (rel_pos_x_bias): Linear(in_features=64, out_features=12, bias=False)
      (rel_pos_y_bias): Linear(in_features=64, out_features=12, bias=False)
    )
    (pooler): LayoutLMv2Pooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=7, bias=True)
)
```

定义了两个主要功能：encode_training_example 和 training_dataloader_from_dataset，目的是为训练和验证集创建数据加载器（DataLoader）。

encode_training_example 函数负责编码单个训练样例。它做了以下几件事：

从路径读取图像并转换为 RGB 格式。 获取样例中的词语、边界框（bbox）和词性标签（ner_tags）。 使用处理器（processor）对图像、词语、边界框和词性标签进行编码，填充到最大长度并进行截断（padding/truncation）。 training_features 定义了经过编码后数据的特征结构，包括图像特征、输入ID、注意力掩码、类型ID、边界框坐标和标签。

training_dataloader_from_dataset 函数负责从给定的训练或测试数据集中创建一个 PyTorch 数据加载器：

首先，使用 map 方法将整个数据集应用 encode_training_example 函数，批量处理数据，并移除不需要的列。 接着，将编码后的数据集设置为 PyTorch 兼容的格式，并放到指定的设备上。 最后，创建一个 PyTorch DataLoader，设置批大小为8，启用数据洗牌，并返回这个数据加载器。

在主程序中，它创建了训练集和验证集（此处称为测试集）的数据加载器，并分别存储在train_dataloader 和 valid_dataloader 中。这两个数据加载器将在训练循环中用于加载批次数据。

In [11]:

```
def encode_training_example(examples):
    images = [Image.open(path).convert("RGB") for path in examples['image_path']]
    words = examples['words']
    boxes = examples['bboxes']
    word_labels = examples['ner_tags']

    encoded_inputs = processor(
        images, words, boxes=boxes, word_labels=word_labels, padding="max_length", tru
    )

    return encoded_inputs

training_features = Features({
    'image': Array3D(dtype="int64", shape=(3, 224, 224)),
    'input_ids': Sequence(feature=Value(dtype='int64')),
    'attention_mask': Sequence(Value(dtype='int64')),
    'token_type_ids': Sequence(Value(dtype='int64')),
    'bbox': Array2D(dtype="int64", shape=(512, 4)),
    'labels': Sequence(ClassLabel(names=labels)),
})

def training_dataloader_from_dataset(dataset):
    encoded_data = dataset.map(
        encode_training_example, batched=True, remove_columns=datasets['train'].colum
        features=training_features
    )
    encoded_data.set_format(type='torch', device=device)
    dataloader = torch.utils.data.DataLoader(encoded_data, batch_size=8, shuffle=True
    batch = next(iter(dataloader))

    return dataloader
train_dataloader = training_dataloader_from_dataset(datasets['train'])
valid_dataloader = training_dataloader_from_dataset(datasets['test'])
```

In [12]:

```
print(len(train_dataloader),len(valid_dataloader))
```

```
19 7
```

可以给每个批次包含x个样本，在每个训练周期开始前都会对数据进行随机打乱。

AdamW优化器对一个Layoutlmv2模型进行训练。模型正在经历多个训练周期 对于训练数据集：

使用train_dataloader加载批次数据。 将批次数据输入模型，model(**batch)会根据模型的需求从批次数据字典中提取必要的键值对（如input_ids, attention_mask, 等）。 计算批次的损失值（loss）。 将损失累加到训练总损失中。 损失反向传播以更新模型参数。 执行优化器的step()方法来应用梯度更新。 清空梯度缓冲区，即调用optimizer.zero_grad()。 训练周期结束后，打印出训练损失的平均值。

对于验证数据集：

类似地，使用valid_dataloader加载验证批次数据。 计算每个批次的损失值并累加到验证总损失中。 训练周期结束后，打印出验证损失的平均值

In [13]:

```
from transformers import AdamW
from tqdm import tqdm
```

```
optimizer = AdamW(model.parameters(), lr=5e-5)
num_epochs = 10


for epoch in range(num_epochs):
    print("Epoch:", epoch)
    training_loss = 0.0
    model.train()
    for batch in tqdm(train_dataloader):
        outputs = model(**batch)
        loss = outputs.loss

        training_loss += loss.item()

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

    print("Training Loss:", training_loss / batch["input_ids"].shape[0])

    validation_loss = 0.0
    for batch in tqdm(valid_dataloader):
        outputs = model(**batch)
        loss = outputs.loss

        validation_loss += loss.item()

    print("Validation Loss:", validation_loss / batch["input_ids"].shape[0])
```

```
/root/miniconda3/lib/python3.8/site-packages/transformers/optimization.py:429: FutureWarning: This implementation of AdamW is deprecated and will be removed in a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to disable this warning
  warnings.warn(
Epoch: 0
100%|██████████| 19/19 [00:32<00:00,  1.70s/it]
Training Loss: 6.665440273284912
100%|██████████| 7/7 [00:00<00:00,  9.42it/s]
Validation Loss: 5.309442400932312
Epoch: 1
100%|██████████| 19/19 [00:32<00:00,  1.69s/it]
Training Loss: 5.0834424018859865
100%|██████████| 7/7 [00:00<00:00,  9.50it/s]
Validation Loss: 4.145301401615143
Epoch: 2
100%|██████████| 19/19 [00:32<00:00,  1.69s/it]
Training Loss: 4.069354045391083
100%|██████████| 7/7 [00:00<00:00,  9.51it/s]
Validation Loss: 3.6265477538108826
Epoch: 3
100%|██████████| 19/19 [00:32<00:00,  1.70s/it]
Training Loss: 3.5799248456954955
100%|██████████| 7/7 [00:00<00:00,  9.49it/s]
Validation Loss: 3.3737061619758606
Epoch: 4
100%|██████████| 19/19 [00:32<00:00,  1.70s/it]
Training Loss: 3.3894755125045775
100%|██████████| 7/7 [00:00<00:00,  9.54it/s]
Validation Loss: 3.2641759514808655
Epoch: 5
```

```
100%|██████████| 19/19 [00:32<00:00,  1.71s/it]
Training Loss: 3.0248158812522887
100%|██████████| 7/7 [00:00<00:00,  9.46it/s]
Validation Loss: 3.0857855677604675
Epoch: 6
100%|██████████| 19/19 [00:32<00:00,  1.72s/it]
Training Loss: 2.766488194465637
100%|██████████| 7/7 [00:00<00:00,  9.53it/s]
Validation Loss: 3.2941944003105164
Epoch: 7
100%|██████████| 19/19 [00:32<00:00,  1.69s/it]
Training Loss: 2.5503732740879057
100%|██████████| 7/7 [00:00<00:00,  9.52it/s]
Validation Loss: 2.9178306758403778
Epoch: 8
100%|██████████| 19/19 [00:32<00:00,  1.69s/it]
Training Loss: 2.348145592212677
100%|██████████| 7/7 [00:00<00:00,  9.48it/s]
Validation Loss: 2.7878752052783966
Epoch: 9
100%|██████████| 19/19 [00:32<00:00,  1.71s/it]
Training Loss: 2.1548160195350645
100%|██████████| 7/7 [00:00<00:00,  9.50it/s]
Validation Loss: 2.7746661007404327
```

In [14]:

```
model.save_pretrained('saved_model/')
```

数据预处理： 训练的时候每个token长度不超过512，多于512会被截断，这里和bert一致。每个token之间用 token连接即可。

使用LayoutLM处理元素过多的图像时，将图像四等分是一种可行的策略，尤其是当图像太大以至于模型难以一次性处理全部内容时。这种方法可以将大图像拆分成较小的部分，便于模型逐个处理，随后将各个部分的输出结果整合起来

预测：

训练结束之后保存模型，然后用自己的测试文件，进行pdf转image，再对于每一页的image进行一个上下四等分切分，进行预测和画框，最后再将四等分的图片重新合成一份。