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



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

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

In [2]:
```
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  AdamW
from datasets import load_dataset, Dataset, Features, Sequence, ClassLabel, Value, Ar
import torch


from tqdm.notebook import tqdm
from datasets import load_dataset
```

In [4]:
```
import json
import random
# 步骤2：打开JSON文件
with open('Xfund/zh.train.json', 'r') as file:
    # 步骤3：读取文件内容
    json_data = file.read()

    # 步骤4：将JSON内容解析成Python对象
    data = json.loads(json_data)
```

In [5]:
```
documents=data["documents"]
```

In [6]:
```
bbox_list=[]
text_list=[]
label_list=[]
img_list=[]
```

```
id_list=[]

# 对于数据需要进行 正则化，使boarding box里面的四个坐标范围在0-1000之间

count=0
for i in documents:
    all_text=i["document"]
    id_list.append(str(count))
    subbbox_list=[]
    subtext_list=[]
    sublabel_list=[]
    img_name=i["img"]["fname"]
    img_name="Xfund/images/train_images/"+img_name
    img = Image.open(img_name).convert("RGB")


    original_height=img.height #3508
    original_width=img.width #2480

    target_height = 1000
    target_width = 762

    width_ratio = target_width / original_width
    height_ratio = target_height / original_height

    resized_img = img.resize((target_width, target_height))

    img_list.append(resized_img)
    for j in all_text:
        boarding_box=j["box"]

#         x1, y1, x2, y2 = boarding_box

#         # 转换为浮点数，防止整数除法
#         x1, y1, x2, y2 = float(x1), float(y1), float(x2), float(y2)
#         # 归一化处理
#         norm_x1 = (x1 / img_width)*1000
#         norm_y1 = (y1 / img_height)*1000
#         norm_x2 = (x2 / img_width)*1000
#         norm_y2 = (y2 / img_height)*1000

        resized_bbox = [
            int(boarding_box[0] * width_ratio),
            int(boarding_box[1] * height_ratio),
            int(boarding_box[2] * width_ratio),
            int(boarding_box[3] * height_ratio)
        ]



        text=j["text"]
        label=j["label"]

        if label=="other":
            sublabel_list.append(0)
        elif label=="header":
            sublabel_list.append(1)
        elif label=="question":
            sublabel_list.append(2)
        elif label=="answer":
            sublabel_list.append(3)


        subbbox_list.append(resized_bbox)
```

```
        subtext_list.append(text)


    bbox_list.append(subbbox_list)
    text_list.append(subtext_list)
    label_list.append(sublabel_list)
    count+=1

print(len(bbox_list),len(text_list),len(label_list),len(id_list),len(img_list))
```

```
149 149 149 149 149
```

In [7]:

```
import pandas as pd


dic={'id':id_list, 'tokens':text_list, 'bboxes':bbox_list, 'ner_tags':label_list, 'ima
train_dataset = Dataset.from_dict(dic)

print(train_dataset)
```

```
Dataset({
    features: ['id', 'tokens', 'bboxes', 'ner_tags', 'image'],
    num_rows: 149
})
```

In [8]:

```
with open('Xfund/zh.val.json', 'r') as file:
    # 步骤3：读取文件内容
    json_data = file.read()

    # 步骤4：将JSON内容解析成Python对象
    data = json.loads(json_data)

# 现在data就是一个Python字典或列表，具体取决于JSON文件的内容
documents=data["documents"]
```

In [9]:

```
bbox_list=[]
text_list=[]
label_list=[]
img_list=[]
id_list=[]

count=0
for i in documents:
    all_text=i["document"]
    id_list.append(str(count))
    subbbox_list=[]
    subtext_list=[]
    sublabel_list=[]
    img_name=i["img"]["fname"]

    img_name="Xfund/images/val_images/"+img_name

    img = Image.open(img_name).convert("RGB")

    original_height=img.height #3508
    original_width=img.width #2480

    target_height = 1000
    target_width = 762

    width_ratio = target_width / original_width
    height_ratio = target_height / original_height
```

```
    resized_img = img.resize((target_width, target_height))

    img_list.append(resized_img)

    for j in all_text:
        boarding_box=j["box"]

#         x1, y1, x2, y2 = boarding_box

#         # 转换为浮点数，防止整数除法
#         x1, y1, x2, y2 = float(x1), float(y1), float(x2), float(y2)
#         # 归一化处理
#         norm_x1 = (x1 / img_width)*1000
#         norm_y1 = (y1 / img_height)*1000
#         norm_x2 = (x2 / img_width)*1000
#         norm_y2 = (y2 / img_height)*1000

        resized_bbox = [
            int(boarding_box[0] * width_ratio),
            int(boarding_box[1] * height_ratio),
            int(boarding_box[2] * width_ratio),
            int(boarding_box[3] * height_ratio)
        ]


        text=j["text"]
        label=j["label"]

        if label=="other":
            sublabel_list.append(0)
        elif label=="header":
            sublabel_list.append(1)
        elif label=="question":
            sublabel_list.append(2)
        elif label=="answer":
            sublabel_list.append(3)

        subbbox_list.append(resized_bbox)
        subtext_list.append(text)


    bbox_list.append(subbbox_list)
    text_list.append(subtext_list)
    label_list.append(sublabel_list)
    count+=1

print(len(bbox_list),len(text_list),len(label_list),len(id_list),len(img_list))
```

```
50 50 50 50 50
```

In [10]:

```
dic={'id':id_list, 'tokens':text_list, 'bboxes':bbox_list, 'ner_tags':label_list, 'ima
val_dataset = Dataset.from_dict(dic)

print(val_dataset)
```

```
Dataset({
    features: ['id', 'tokens', 'bboxes', 'ner_tags', 'image'],
    num_rows: 50
})
```

In [11]:

```
from datasets import Dataset, DatasetDict
data_dict = DatasetDict({
```

```
    "train": train_dataset,
    "validation": val_dataset
})
```

In [12]:

```
print(data_dict)
print(data_dict["train"].features)
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'tokens', 'bboxes', 'ner_tags', 'image'],
        num_rows: 149
    })
    validation: Dataset({
        features: ['id', 'tokens', 'bboxes', 'ner_tags', 'image'],
        num_rows: 50
    })
})
{'id': Value(dtype='string', id=None), 'tokens': Sequence(feature=Value(dtype='strin
g', id=None), length=-1, id=None), 'bboxes': Sequence(feature=Sequence(feature=Value(d
type='int64', id=None), length=-1, id=None), length=-1, id=None), 'ner_tags': Sequence
(feature=Value(dtype='int64', id=None), length=-1, id=None), 'image': Image(decode=Tru
e, id=None)}
```

{'bboxes': Sequence(feature=Sequence(feature=Value(dtype='int64', id=None), length=-1, id=None), length=-1, id=None),

'id': Value(dtype='string', id=None),

'image': Image(decode=True, id=None),

'ner_tags': Sequence(feature=ClassLabel(num_classes=7, names=['O', 'B-HEADER', 'I-HEADER', 'B-QUESTION', 'I-QUESTION', 'B-ANSWER', 'I-ANSWER'], id=None), length=-1, id=None),

'tokens': Sequence(feature=Value(dtype='string', id=None), length=-1, id=None)}

In [13]:

```
print(data_dict["train"]["ner_tags"][0])
print(data_dict["train"]["id"])
print(data_dict["train"]["tokens"][0])
print(data_dict["train"]["bboxes"][0])
print(data_dict["train"]["image"][0])
```

```
[0, 2, 3, 1, 2, 0, 0, 3, 3, 0, 3, 3, 0, 0, 2, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 0, 3,
2, 3, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 0, 0, 3, 0, 2, 3, 3, 2, 2, 3, 0, 3, 2, 0, 2, 0,
0, 2, 0, 0, 3, 0, 2, 2, 2, 3, 3, 0, 3, 2, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 0, 3, 2, 3, 3,
3, 2, 3, 3, 2, 3, 3, 3, 3, 0, 0, 2, 3, 0, 0, 0, 0, 2, 3, 0, 0, 2, 2, 3, 3, 3, 3, 2, 3,
3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 2]
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15',
'16', '17', '18', '19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '3
0', '31', '32', '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43', '4
4', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', '5
8', '59', '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '7
2', '73', '74', '75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '8
6', '87', '88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '98', '99', '10
0', '101', '102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '11
2', '113', '114', '115', '116', '117', '118', '119', '120', '121', '122', '123', '12
4', '125', '126', '127', '128', '129', '130', '131', '132', '133', '134', '135', '13
6', '137', '138', '139', '140', '141', '142', '143', '144', '145', '146', '147', '14
8']
['汇丰晋信', '受理时间:', '2020.6.15', '开放式基金账户业务申请表一机构', '特别提示:',
'基金账号:', '(新开户免填)', '1.请在填表前仔细阅读有关基金的《基金合同》、最新《招募说
明书》及《开', '放式基金业务规则》。请用黑色或蓝黑色钢笔或签字笔填写,如有选择项,', '交
```

易账号:', '请在口内划√,任何涂改请加盖公章或签字证明。', '2.请详细、准确、全面填写下列信息,以确保您的相关权益', '(新开户免填)', '以下表单中★为开户必填项', '▽业务类型', '□账户登记', '□客户资料变更(如需修改银行账户信息,请直接填写《投资者银行账户设置及变更申请表》)', '☑账户开户', '□交易密码重置', '□交易账户销户', '□基金账户销户', '▽机构基本信息', '秦皇岛旭森企业管理有限公司', '☑营业执照', '□社会团体证件', '□基金会证件', '□其他证件', '□行政机关证件', '*证件号码:', '91130301MAOFFMWL8N', '*证件有效期:2021年9', '*邮寄地址:', '省秦皇岛市市经济技术开发区永定河道9号', '邮编:', '*证件号码:', '李丛林', '*经办人姓名:', '*证件类型:', '*职务:', '*证件有效期:', '经办人手机:', '经办人联系电话:(日间)', '(晚间)', '18757832244@163.com', '传真:', '账单寄送方式:', '☑E-mail', '□手机短信', '*法定代表人(非法人机构为负责人)姓名:', '*证件类型:', '195658197506185119', '第一联注册登记人留存', '(只能选一项,后三项开通前均采用邮寄方式)', '*证件有效期:', '*控股股东或实际控制人名称:', '★投资资金是否为信托资产:', '信托委托人名称:', '联系方式:', '若是信托资产,', '联系方式:', '信托受益人名称:', '☑否', '\\', '▽开户预留银行账户信息(如需修改银行账户信息,请直接填写《投资者银行账户设置及变更申请表》)', '*户名:', '*银行账号:', '6215968321478528861', '中国银行(秦皇岛市开发区支行)', '第二联客户留存', '李丛林', '★开户银行属地:', '河北省', '省', '秦皇岛', '市', '*开户银行名称:', '▽机构其他身份信息', '*企业性质:', '□国有', '□集体', '☑私营', '□其他:', '□三资[中外合资、中外合作、外商独资]', '*行业类型:', '☑其他金融机构', '□非赢利机构', '□其他赢利机构_', '*注册资本:', '50', '万元', '▽声明', '本机构已仔细阅读了汇丰晋信基金管理有限公司旗下基金的《基金合同》、最新的《招募说明书》、《开放式基金业务规则》等文件及本表所有内容(包括背面', '条款),愿意接受上述文件中载明的全部条款。本机构保证本申请表所填信息及所提供的资料真实有效,自愿履行基金投资人的各项义务,并自行承担基金投资风', '险。本机构保证,用于证券投资基金投资和交易的资金来源合法,本机构进行证券投资基金投资和交易之行为符合中华人民共和国各项法律、法规和规定。本机', '构已阅读并充分了解《汇丰晋信基金管理有限公司投资人权益须知》。', '机构公章:', '法定代表人(负责人)签章:', '经办人签章:', '2020年6月15日', '销售网点:', '投资顾问:', '主管:', '活动来源:', '附件：', '张', '录入员:', '复核员:', '*申请机构名称:', '*证件类型:', '法定代表人', '身份证', '195658197506185119', '18757832244', 'E-mail:', '066004', '□邮寄', '李丛林', '2026年7月13日', '□是', '身份证', '*证件号码:', '2021年9月22日', '□传真', '2026年7月13日', '□保险机构', '李丛林', '日期:']

[[31, 32, 162, 49], [38, 76, 81, 86], [98, 68, 164, 81], [445, 62, 681, 79], [373, 91, 412, 102], [41, 103, 81, 113], [257, 103, 311, 113], [382, 102, 679, 113], [384, 114, 676, 125], [38, 129, 81, 140], [384, 125, 580, 136], [385, 136, 622, 147], [258, 129, 310, 140], [31, 162, 145, 174], [36, 197, 87, 208], [130, 217, 181, 229], [188, 218, 569, 231], [51, 216, 97, 229], [209, 237, 273, 247], [129, 236, 195, 247], [50, 236, 115, 248], [35, 258, 108, 271], [115, 266, 278, 283], [96, 291, 145, 303], [251, 292, 313, 302], [331, 292, 388, 303], [404, 293, 452, 303], [167, 292, 229, 302], [43, 309, 90, 319], [127, 309, 270, 320], [339, 308, 396, 321], [42, 325, 95, 336], [104, 323, 412, 338], [519, 326, 544, 337], [449, 342, 500, 354], [117, 338, 157, 351], [43, 342, 104, 353], [314, 341, 370, 354], [199, 340, 233, 354], [39, 357, 100, 370], [468, 373, 524, 389], [39, 375, 142, 387], [290, 376, 314, 387], [388, 390, 523, 404], [39, 393, 65, 402], [39, 410, 103, 420], [154, 410, 191, 420], [210, 409, 259, 421], [42, 424, 218, 437], [301, 425, 352, 437], [494, 425, 620, 437], [703, 415, 714, 535], [312, 409, 493, 423], [42, 442, 101, 454], [41, 459, 165, 471], [40, 476, 158, 488], [300, 477, 370, 488], [489, 477, 530, 489], [214, 476, 274, 488], [487, 494, 531, 505], [299, 494, 369, 505], [163, 510, 182, 521], [704, 539, 715, 552], [34, 537, 455, 550], [39, 559, 70, 569], [351, 559, 399, 570], [416, 554, 546, 565], [126, 573, 294, 587], [702, 561, 715, 648], [90, 552, 129, 564], [349, 582, 422, 593], [434, 576, 473, 589], [518, 581, 534, 592], [539, 575, 577, 588], [630, 585, 639, 593], [41, 582, 108, 593], [33, 597, 122, 609], [34, 616, 89, 626], [89, 617, 121, 627], [139, 617, 169, 626], [369, 616, 399, 627], [419, 617, 451, 627], [186, 616, 353, 627], [37, 633, 88, 644], [160, 634, 225, 645], [241, 634, 298, 644], [316, 633, 383, 645], [34, 652, 83, 662], [132, 649, 144, 659], [220, 652, 237, 663], [32, 678, 66, 688], [51, 695, 681, 708], [33, 710, 680, 723], [32, 726, 679, 738], [32, 740, 310, 752], [32, 775, 74, 787], [267, 776, 383, 788], [488, 776, 540, 789], [537, 829, 670, 843], [23, 867, 64, 877], [178, 866, 218, 876], [314, 866, 338, 876], [441, 867, 481, 876], [587, 867, 612, 877], [651, 868, 660, 877], [25, 915, 56, 926], [178, 914, 209, 925], [41, 273, 110, 286], [42, 291, 94, 304], [236, 340, 296, 355], [373, 340, 410, 355], [502, 341, 625, 354], [526, 371, 601, 383], [340, 392, 374, 403], [549, 326, 591, 338], [109, 411, 137, 420], [219, 422, 255, 435], [103, 440, 235, 453], [164, 477, 184, 488], [353, 425, 392, 439], [446, 427, 497, 440], [400, 311, 534, 323], [277, 409, 305, 421], [104, 355, 240, 372], [92, 633, 138, 646], [541, 777, 578, 791], [503, 832, 530, 846]]

<PIL.PngImagePlugin.PngImageFile image mode=RGB size=762x1000 at 0x7FBACB5DDD60>

In [14]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
from transformers import AutoProcessor, AutoModel,LayoutLMv3Tokenizer,XLMRobertaToker
from datasets import load_dataset

processor = AutoProcessor.from_pretrained("layoutlmv3-base-chinese", apply_ocr=False,
```

```
The tokenizer class you load from this checkpoint is not the same type as the class th
is function is called from. It may result in unexpected tokenization.
The tokenizer class you load from this checkpoint is 'XLMRobertaTokenizer'.
The class this function is called from is 'LayoutLMv3TokenizerFast'.
```

In [15]:
```
from datasets.features import ClassLabel

features = data_dict["train"].features
column_names = data_dict["train"].column_names
image_column_name = "image"
text_column_name = "tokens"
boxes_column_name = "bboxes"
label_column_name = "ner_tags"

# In the event the labels are not a `Sequence[ClassLabel]`, we will need to go through
# unique labels.
def get_label_list(labels):
    unique_labels = set()
    for label in labels:
        unique_labels = unique_labels | set(label)
    label_list = list(unique_labels)
    label_list.sort()
    return label_list

if isinstance(features[label_column_name].feature, ClassLabel):
    label_list = features[label_column_name].feature.names
    # No need to convert the labels since they are already ints.
    id2label = {k: v for k,v in enumerate(label_list)}
    label2id = {v: k for k,v in enumerate(label_list)}
else:
    label_list = get_label_list(data_dict["train"][label_column_name])
    id2label = {k: v for k,v in enumerate(label_list)}
    label2id = {v: k for k,v in enumerate(label_list)}
num_labels = len(label_list)
```

In [16]:
```
print(label_list)
```

```
[0, 1, 2, 3]
```

In [17]:
```
print(id2label)
```

```
{0: 0, 1: 1, 2: 2, 3: 3}
```

In [18]:
```
def prepare_examples(examples):
    images = examples[image_column_name]
    words = examples[text_column_name]
    boxes = examples[boxes_column_name]
    word_labels = examples[label_column_name]
    encoding = processor(images, words, boxes=boxes, word_labels=word_labels,
                         truncation=True, padding="max_length")
```

```
    return encoding
```

In [19]:
```
from datasets import Features, Sequence, ClassLabel, Value, Array2D, Array3D

# we need to define custom features for `set_format` (used later on) to work properly
features = Features({
    'pixel_values': Array3D(dtype="float32", shape=(3, 224, 224)),
    'input_ids': Sequence(feature=Value(dtype='int64')),
    'attention_mask': Sequence(Value(dtype='int64')),
    'bbox': Array2D(dtype="int64", shape=(512, 4)),
    'labels': Sequence(feature=Value(dtype='int64')),
})
```

In [20]:
```
def training_dataloader_from_dataset(dataset):
    encoded_data = dataset.map(
    prepare_examples,
    batched=True,
    remove_columns=column_names,
    features=features,)

    encoded_data.set_format(type='torch', device=device)
    dataloader = torch.utils.data.DataLoader(encoded_data, batch_size=2, shuffle=True
    batch = next(iter(dataloader))

    return dataloader

train_dataloader = training_dataloader_from_dataset(data_dict['train'])
valid_dataloader = training_dataloader_from_dataset(data_dict['validation'])
```

```
Map:   0%|          | 0/149 [00:00<?, ? examples/s]
Map:   0%|          | 0/50 [00:00<?, ? examples/s]
```

In [21]:
```
from transformers import LayoutLMv3ForTokenClassification

# number_labels 控制模型分类输出label个数
model = LayoutLMv3ForTokenClassification.from_pretrained("layoutlmv3-base-chinese", nu
model.to(device)
```

```
Some weights of LayoutLMv3ForTokenClassification were not initialized from the model c
heckpoint at layoutlmv3-base-chinese and are newly initialized: ['classifier.bias', 'c
lassifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for pr
edictions and inference.
```

Out[21]:
```
LayoutLMv3ForTokenClassification(
  (layoutlmv3): LayoutLMv3Model(
    (embeddings): LayoutLMv3TextEmbeddings(
      (word_embeddings): Embedding(250002, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (x_position_embeddings): Embedding(1024, 128)
      (y_position_embeddings): Embedding(1024, 128)
      (h_position_embeddings): Embedding(1024, 128)
      (w_position_embeddings): Embedding(1024, 128)
    )
    (patch_embed): LayoutLMv3PatchEmbeddings(
      (proj): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
    )
    (pos_drop): Dropout(p=0.0, inplace=False)
```

```
(LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
(dropout): Dropout(p=0.1, inplace=False)
(norm): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
(encoder): LayoutLMv3Encoder(
  (layer): ModuleList(
    (0): LayoutLMv3Layer(
      (attention): LayoutLMv3Attention(
        (self): LayoutLMv3SelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): LayoutLMv3SelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): LayoutLMv3Intermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): LayoutLMv3Output(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (1): LayoutLMv3Layer(
      (attention): LayoutLMv3Attention(
        (self): LayoutLMv3SelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): LayoutLMv3SelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): LayoutLMv3Intermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): LayoutLMv3Output(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (2): LayoutLMv3Layer(
      (attention): LayoutLMv3Attention(
        (self): LayoutLMv3SelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): LayoutLMv3SelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
```

```
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (3): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (4): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (5): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
```

```
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (6): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (7): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
```

```
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (8): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (9): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): LayoutLMv3Intermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): LayoutLMv3Output(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (10): LayoutLMv3Layer(
    (attention): LayoutLMv3Attention(
      (self): LayoutLMv3SelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): LayoutLMv3SelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
```

```
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): LayoutLMv3Intermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): LayoutLMv3Output(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (11): LayoutLMv3Layer(
        (attention): LayoutLMv3Attention(
          (self): LayoutLMv3SelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): LayoutLMv3SelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): LayoutLMv3Intermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): LayoutLMv3Output(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-05, 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)
  )
 )
 (dropout): Dropout(p=0.1, inplace=False)
 (classifier): Linear(in_features=768, out_features=4, bias=True)
)
```

In [22]:

```
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: FutureW
arning: This implementation of AdamW is deprecated and will be removed in a future ver
sion. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation
_warning=True` to disable this warning
  warnings.warn(
Epoch: 0
  0%|          | 0/75 [00:00<?, ?it/s]/root/miniconda3/lib/python3.8/site-packages/tra
nsformers/modeling_utils.py:993: FutureWarning: The `device` argument is deprecated an
d will be removed in v5 of Transformers.
  warnings.warn(
100%|██████████| 75/75 [00:46<00:00,  1.62it/s]
Training Loss: 61.237445175647736
100%|██████████| 25/25 [00:01<00:00, 24.87it/s]
Validation Loss: 6.685423314571381
Epoch: 1
100%|██████████| 75/75 [00:46<00:00,  1.62it/s]
Training Loss: 36.7808974981308
100%|██████████| 25/25 [00:00<00:00, 26.81it/s]
Validation Loss: 6.171919487416744
Epoch: 2
100%|██████████| 75/75 [00:46<00:00,  1.60it/s]
Training Loss: 25.54763574153185
100%|██████████| 25/25 [00:00<00:00, 27.11it/s]
Validation Loss: 5.038497112691402
Epoch: 3
100%|██████████| 75/75 [00:46<00:00,  1.61it/s]
Training Loss: 16.64214650541544
100%|██████████| 25/25 [00:00<00:00, 27.12it/s]
Validation Loss: 5.985387355089188
Epoch: 4
100%|██████████| 75/75 [00:46<00:00,  1.61it/s]
Training Loss: 13.154323246330023
100%|██████████| 25/25 [00:00<00:00, 27.18it/s]
Validation Loss: 4.813666954636574
Epoch: 5
100%|██████████| 75/75 [00:46<00:00,  1.61it/s]
Training Loss: 9.885523475706577
100%|██████████| 25/25 [00:00<00:00, 27.02it/s]
Validation Loss: 4.8512350507080555
Epoch: 6
100%|██████████| 75/75 [00:46<00:00,  1.61it/s]
Training Loss: 10.276367292739451
100%|██████████| 25/25 [00:00<00:00, 27.01it/s]
```

```
Validation Loss: 6.1577035784721375
Epoch: 7
100%|██████████| 75/75 [00:46<00:00,  1.61it/s]
Training Loss: 4.936611045850441
100%|██████████| 25/25 [00:00<00:00, 26.95it/s]
Validation Loss: 5.789982587099075
Epoch: 8
100%|██████████| 75/75 [00:46<00:00,  1.61it/s]
Training Loss: 3.0269110843073577
100%|██████████| 25/25 [00:00<00:00, 26.86it/s]
Validation Loss: 5.9046945348382
Epoch: 9
100%|██████████| 75/75 [00:46<00:00,  1.61it/s]
Training Loss: 2.672566587338224
100%|██████████| 25/25 [00:00<00:00, 26.94it/s]
Validation Loss: 6.5071463547647
```