2023-2024 学年

知识工程专题实践报告

任课教师：吴天星

院　系 ____计软智学院____

专　业 ____人工智能____

组　别 ____第二组____

小组成员名单

知识工程专题实践报告

学号	姓名
58121124	张博彦
58121132	徐朔
58121207	舒劲菘
58121214	王璟琪
58121310	徐子健
58121316	何佳骏

一、 个人工作量

1、何佳骏：使用 doccano 进行训练集(Excel 中 10%数据，约 360 条文本)标签手动标注；训练 transformers 模型命名实体识别，初步提取出 Excel 中全部药名；但是错别字，别名混淆等严重降低准确率，所以对提取的药名和最后的测试数据进行数据清洗，统一名称，修改错别字，增加准确率。

2、 张博彦：对 NER 处理后的数据集进行再次处理，将用药信息重新与病人匹配，将无具体用药的病人进行剔除，同时对用药信息进行二次检查，确保通过 NER 提取出的药品信息正确率较高，能够用于后续训练；对徐朔构建知识图谱后导出的 excel 表格数据进行优化，形成 entities.csv,entities_attr.csv,relationships.csv 三份文件。

3、 徐朔：1.分割病人症状与所用药品的大段描述，将其转化为单个药品或症状的义项。并编写数据处理脚本，解决提取出的药品名称中的错别字与病人症状中本质相同的症状的不同表述造成的重复冗余问题、解决分隔符不同造成的分割失败问题、解决全半角标点不同造成的重复冗余问题；并根据内分泌降糖药物表格中提供的信息，合并同一药物的不同名字。2.提出本组知识图谱的结构，以病人为纽带，链接症状与药品。并利用精细处理过的数据在 Protege 中实际建立了知识图谱。

4、 王璟琪：1.分析已有三元组的可优化内容，与同学共同规划出选取前 100 种病和药、分组训练等创新思路。2.编写代码对数据集重新进行有效特征筛选、分两组、无关字符清洗、填充缺失值等处理，为模型提供可直接使用的输入，主要负责的是测试集的处理工作。3.课上负责后半部分的汇报演讲。

5、 舒劲菘：1.分析数据特征，商定数据处理方法和选定模型。2.负责去除部分无关特征，编写代码对训练数据和测试数据进行分组。3.使用编写好的模型进行训练，并预测测试数据的带药情况。4.处理测试数据预测结果和模型预测结果，计算准确率、召回率、F1，对结果进行可视化和评估，分析召回率较低的原因。

6、 徐子健：1.基于 Pytorch 框架搭建一个全连接神经网络(FCNN)模型，以整合后病人属性和病症诊断为总输入特征向量，以药物预测判断为输出标签向量，分别针对 AB 两组数据进行学习。2.针对项目情景：多组二分类问题，选用二进制交叉熵损失（BCELoss）和 Adam 优化器，并测试运行。3.为便于操作实现了模型封装，包括运行各功能的.bat 模块、模型训练过程和保存、训练日志输出和结果可视化等。

二、 实验目的

我们选择了实践专题的任务二——糖尿病知识图谱构建及用药推荐，需要根据原始数据集，进行知识图谱的构建。之后选择合适的统计模型进行训练，从而获得能够根据患者症状给出具体用药的模型。最后在给出的测试集上进行预测。

三、 实验过程

1、 使用 doccano 进行训练集(Excel 中 10%数据，约 360 条文本)标签手动标注，配置虚拟机环境，运行 doccano；训练 transformers 模型命名实体识别，初步提取出 Excel 中全部药名；但是错别字，别名混淆等严重降低准确率，所以对提取的药名和最后的测试数据进行数据清洗，python 对提取的药名进行去重然后导入到文件，人工对提取的问题实体统一名称，修改错别字，增加准确率。

2、 通过 NER 提取用药信息后，将病人的 id 等信息与提取后的用药信息进行一一对应，同时需要进行数据的检测，尽可能避免 NER 提取出的用药信息正确率较低，使得数据可靠性下降的问题。校验之后，由于数据集中存在部分病人未给出任何开药或仅给出了相关建议，因此需要，将该类病人剔除，不用作后续知识图谱构建与模型训练。

3、 经过前两步，我们已经得到了可以用于分析的病人数据信息。然而，直接分析这些大段的信息不太能帮助我们了解如何为糖尿病人开药，我们需要把这些信息分解为一个个单独的信息；类比于对文本做 parse 的过程，我们要得到一个个 token。在这里，我们的 token 就是单个的症状与药品的名称。因此，我们分割病人症状与所用药品的大段描述，将其转化为单个药品或症状的义项。

4、 然而，我们得到的这些数据并不能直接用来建立知识图谱，这是因为我们的原始数据并不那么“干净”，需要更精细化的处理。具体来说，我们通过编写脚本，解决了提取出的药品名称中的错别字与病人症状中本质相同的症状的不同表述造成的重复冗余问题、分隔符不同造成的分割失败问题、全半角标点不同造成的重复冗余问题；并根据内分泌降糖药物表格中提供的信息，合并同一药物的不同名字。这样，我们才得到了可堪一用的数据。

5、 我们的最终任务为已知糖尿病人的症状等信息，给出合理的处方。我们容易发现，这一问题中病人是联系症状与处方的纽带。因此，我们建立了（病人，具有症状，症状）、（病人，具有性别，性别）、（病人，使用药品，药品）等多种三元组。通过 Protege 软件，我们建立了对应的知识图谱并导出了对应的文件。

6、 虽然 protege 导出的数据格式已经包含所需的信息，但是存在格式问题，因此使用

python 进行手动优化，将导出后的 excel 表格处理成实验要求所需的 entities.csv,entities_attr.csv,relationships.csv 三份文件

7、 因存在不少样本诸多属性为 NA(缺省值)，我们将数据划分为 AB 组：A 组数据是缺省值较多的一组，删除缺省值较多的特征列；B 组数据是缺省值较少，缺省值换为均值。在预测阶段，我们采用全连接神经网络(FCNN)进行模型训练和结果预测，并对 AB 组数据分开进训练实现两组 FCNN 模型。对于 A 组，删去空值多于 80%的属性；此外我们也将其它非值的数据处理为值（存在 <0.2 等带有符号的值，将其分别处理为上限或下限）。对于图谱数据，因考虑病症和药品种类太多，过拟合风险太大，我们仅分别选取了出现率前 100 的药物和病症。在与图谱数据结合后，可以得到包含病名向量的输出矩阵。最终我们得到了 AB 两组的输入、输出矩阵，共计 8 个 csv 文件。

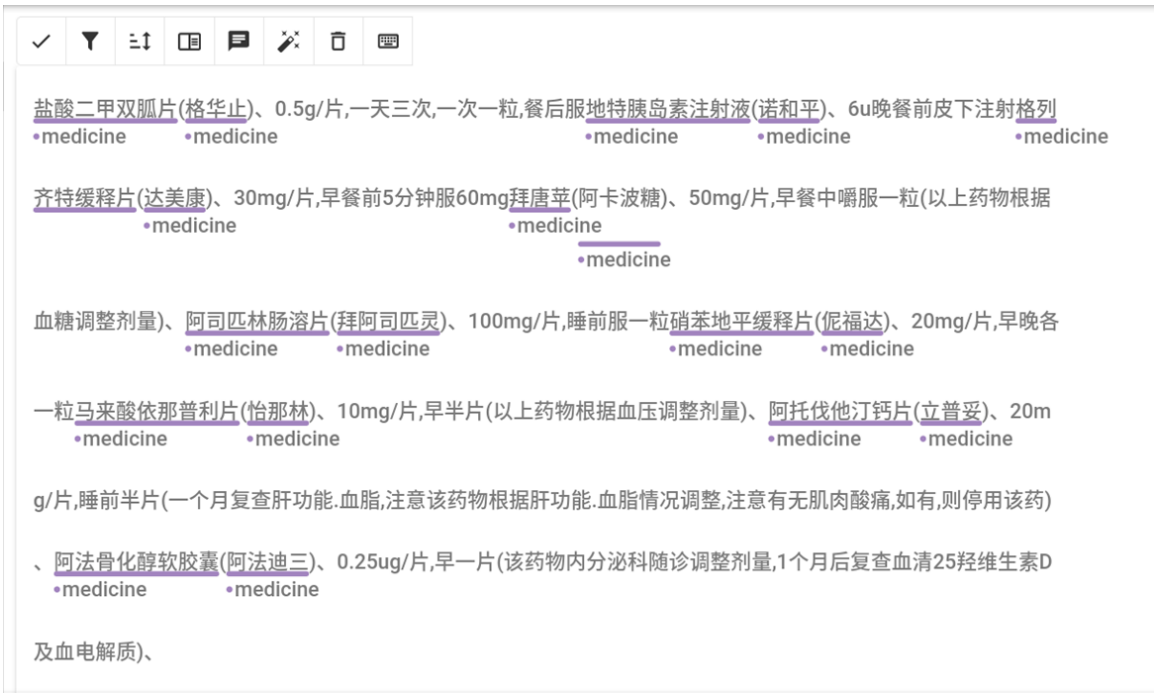
8、 在预测阶段，我们采用全连接神经网络(FCNN)进行模型训练和结果预测。我们基于 Pytorch 框架搭建全连接神经网络(FCNN)模型，针对 AB 两组数据分别训练并保存，检测时先对数据分组，然后使用对应的网络进行预测。模型的输入为病例特征，包括病人属性和图谱中获取的病症诊断；模型的输出位药品的判断标签向量。本项目并非“多分类问题”，而是多组药物的“二分类问题”，因此，本项目采用二进制交叉熵损失（Binary Cross Entropy Loss）作为损失函数。不同于常用的随机梯度下降算法(SGD)，本项目采用了 Adam 优化器，以实现学习率的动态调整、偏置矫正机制的引入和超参数的鲁棒选择。在训练过程中，我们对验证集进行动态划分，以降低过拟合、调整学习方向。除了 FCNN，我们亦实现了模型封装和各个模块的优化，以更好地进行训练、调整及结果保存。

9、 使用全连接神经网络进行训练，得到训练不同批次的模型，处理测试数据将其同样分为 A、B 两组。用得到的模型分别对测试数据进行预测得到一系列预测结果。

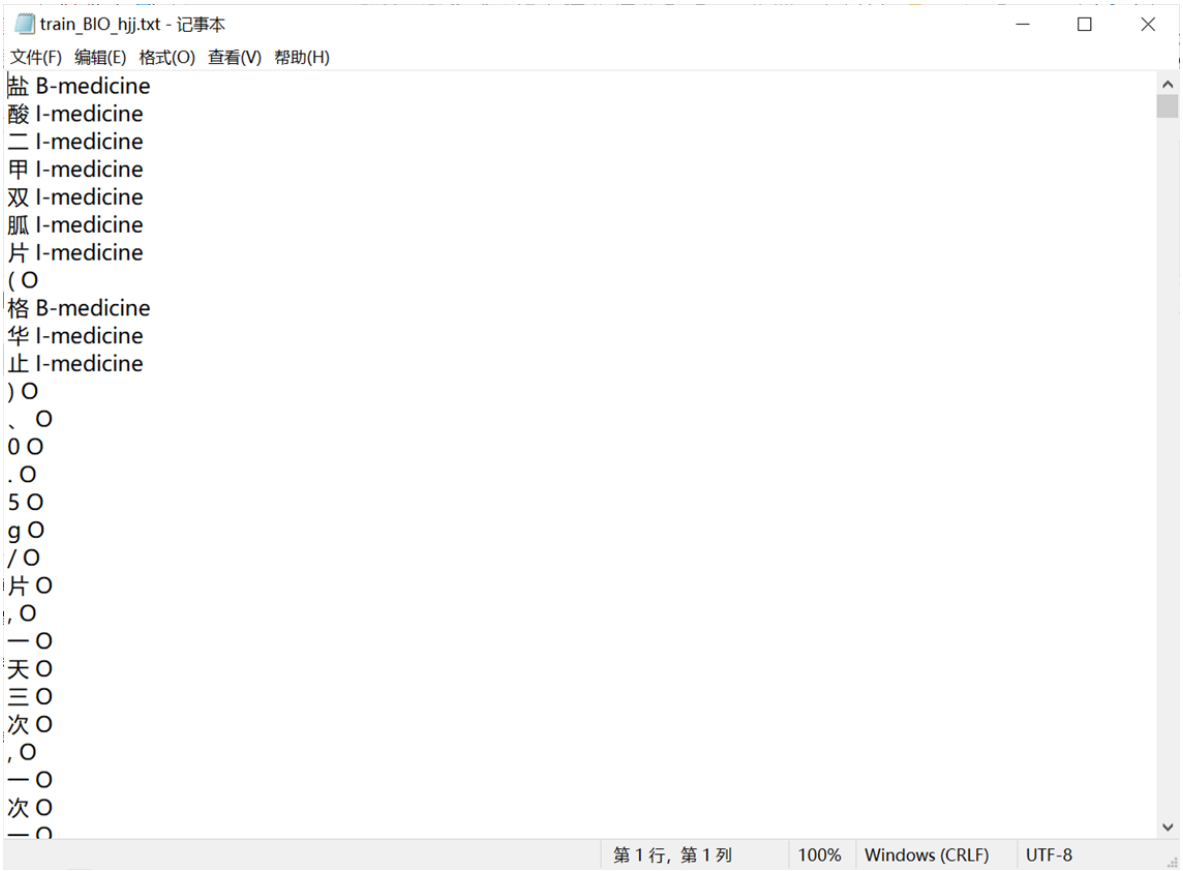
10、 对预测结果进行评估，计算准确率、召回率和 F1，发现当训练批次过多过拟合现象严重，故选取训练批次较少的结果进行可视化并进行分析。

四、 实验结果

1、 (1)手动标注结果切片



(2) BIO 标签结果



(3) 数据清洗结果前后对比

拜糖苹	亚莫利	拜阿司匹林	怡开	洛丁新
安博维	波依定	拜阿司匹林	格华止	捷诺维
格华止	拜糖苹	瑞易宁	诺和平	立普妥
拜复乐	拜糖平	诺和锐	依姆多	立普妥
格华止	甲钴胺片	开瑞坦		
利加隆	万爽力	迈之灵	阿法迪三	怡开
亚莫利	拜糖平	捷诺维	二甲双胍	代文
诺和锐				
拜糖苹	诺和龙	捷诺维	络活喜	可多华
格华止	诺和锐	立普妥	唐林	甲钴胺片
拜糖苹	格华止	阿法迪三	拜阿司匹林	怡开
拜糖平	亚莫利	二甲双胍	立普妥	拜阿司匹林
诺和锐	诺和平	拜糖苹	代文	波依定
拜糖苹	二甲双胍	阿法迪三	代文	再宁平
诺和锐	来得时	弥可保	立普妥	波利维
格华止	瑞易宁	代文	怡开	立普妥
拜糖平	优泌乐	立普妥	弥可保	贝前列腺素钠
诺和锐	诺和平	哈乐	倍他乐克	倪福达

拜糖平	亚莫利	拜阿司匹林	怡开	盐酸贝那普利
安博维	波依定	拜阿司匹林	格华止	捷诺维
格华止	拜糖平	格列吡嗪	诺和平	立普妥
拜复乐	拜糖平	诺和锐	依姆多	立普妥
格华止	甲钴胺片	氯雷他定片		
利加隆	万爽力	迈之灵	阿法迪三	怡开
亚莫利	拜糖平	捷诺维	盐酸二甲双胍片	代文
诺和锐				
拜糖平	诺和龙	捷诺维	络活喜	可多华
格华止	诺和锐	立普妥	唐林	甲钴胺片
拜糖平	格华止	阿法迪三	拜阿司匹林	怡开
拜糖平	亚莫利	盐酸二甲双胍片	立普妥	拜阿司匹林
诺和锐	诺和平	拜糖平	代文	波依定

(4) 各组数据处理

我们从 A 组中删去了如下特征：

'尿素氮 数值', '尿微量白蛋白 数值', '促甲状腺素受体抗体 数值', '脂蛋白(a) 数值', '胰岛素-空腹 数值', '胰岛素-餐后30 数值', '胰岛素-餐后60 数值', '胰岛素-餐后120 数值', 'C-肽-餐后30 数值', 'C-肽-餐后60 数值', '糖化血红蛋白 数值', '谷丙转氨酶 数值', '碱性磷酸酶 数值','谷酰转肽酶 数值', '总胆红素 数值', '直接胆红素 数值', '总胆汁酸 数值', '肌酐 数值', '尿酸 数值', '甘油三酯 数值', '胆固醇 数值', 'H-胆固醇 数值', 'L-胆固醇 数值', '载脂蛋白A Ⅰ 数值', '载脂蛋白B 数值','促甲状腺激素 数值', '游离三碘甲状腺原氨酸 数值', '游离甲状腺素 数值', '甲状腺球蛋白抗体 数值','抗甲状腺过氧化酶抗体 数值', '孕酮 数值', '雌二醇 数值', '泌乳素 数值', '总睾酮 数值', '硫酸去氢表雄酮 数值', '性激素结合蛋白 数值', '血清骨钙素测定 数值', '血清I型胶原羟末端肽β特殊序列测定 数值', '血清总I型胶原氨基末端肽测定 数值','25-羟基维生素D 数值', '葡萄糖(餐后0.5h) 数值', '葡萄糖(餐后1h) 数值', '葡萄糖(餐后2h) 数值','eGFR(MDRD) 数值', '甲胎蛋白 数值', '癌胚抗原 数值', '糖类抗原125 数值', '糖类抗原15-3 数值','糖类抗原19-9 数值', '糖类抗原72-4 数值', '糖类抗原242 数值', '非小细胞肺癌相关抗原21-1 数值','神经元特异性烯醇化酶 数值', '游离前列腺特异性抗原 数值', '总前列腺特异性抗原 数值', '铁蛋白 数值','抗谷氨酸脱羧酶抗体(GAD-Ab) 数值', '胰岛细胞抗体 数值', '抗胰岛素抗体(IAA) 数值'。

(5) 数据分析与优化

部分处理完毕的输入矩阵：

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	0	1	2	3	4	5	6	7	8	9	10	11
2	0	1	1	24.91	137	71	83	73	69	9.6528571	13.03	24.58	54.635625
3	1	2	1	24	147	84	88	69	59	9.6528571	13.03	24.58	54.635625
4	2	13	1	27.92	145	62	112	89	69	9.6528571	13.03	24.58	54.635625
5	3	26	1	23.46	139	78	93	68	48	9.6528571	13.03	24.58	54.635625
6	4	56	1	27.04	139	96	99.8	55	44	9.6528571	13.03	24.58	54.635625
7	5	61	2	21.59	138	78	83	76	66	9.6528571	13.03	24.58	54.635625
8	6	63	2	22.55	130	80	80	57	54	9.6528571	13.03	24.58	54.635625
9	7	66	1	18.51	119	73	68	54	53	9.6528571	13.03	24.58	6.6
10	8	75	1	20.8	115	50	82	50	36	9.6528571	13.03	24.58	54.635625
11	9	85	1	26.1	150	103	88	58	43	9.6528571	13.03	24.58	54.635625
12	10	175	2	22.31	98	62	84	47	33	9.6528571	13.03	24.58	54.635625
13	11	186	1	26.58	141	80	93	21	13	9.6528571	13.03	24.58	54.635625
14	12	190	1	24.19	121	79	96	64	52	9.6528571	13.03	24.58	54.635625
15	13	191	1	21.46	131	74	95	62	42	9.6528571	13.03	24.58	54.635625
16	14	192	1	26.57	177	89	98	53	52	9.6528571	13.03	24.58	54.635625
17	15	193	2	20.4	147	90	81	62	48	9.6528571	13.03	24.58	54.635625
18	16	194	1	21.45	130.54878	75.939024	88.928767	47	37	9.6528571	13.03	24.58	54.635625
19	17	201	1	28.4	131	86	101	49	42	9.6528571	13.03	24.58	54.635625
20	18	202	1	27.7	130	70	88	53	50	9.6528571	13.03	24.58	54.635625
21	19	212	1	23.11	170	71	88	75	33	9.6528571	13.03	24.58	54.635625
22	20	239	2	20	136	60	78	62	52	9.6528571	13.03	24.58	54.635625
23	21	240	1	28.72	131	89	102	49	35	9.6528571	13.03	24.58	54.635625
24	22	257	2	21.11	112	58	82	67	57	10.9	13.03	24.58	132.7
25	23	263	2	24	125	99	81	53	41	9.6528571	13.03	24.58	54.635625
26	24	270	2	23.7	105	60	85	56	48	9.6528571	13.03	24.58	54.635625
27	25	273	1	20.6	120	61	73	38	32	9.6528571	13.03	24.58	54.635625
28	26	275	2	23.23	128	70	88.928767	59	38	9.6528571	13.03	24.58	54.635625
29	27	281	1	26.76	110	52	90	54	39	9.6528571	13.03	24.58	54.635625

(6) FCNN 代码实现

在 models.py 中定义的 FCNN 类

```
class FCNN(nn.Module):
    def __init__(self,fetlen,lablen,cuda=True):
        super(FCNN, self).__init__()
        self.fetlen = fetlen
        self.lablen = lablen
        self.cuda = cuda
        self.fc1 = nn.Linear(self.fetlen, out_features=256)
        self.fc2 = nn.Linear(in_features=256, out_features=512)
        self.fc3 = nn.Linear(in_features=512, out_features=128)
        self.fc4 = nn.Linear(in_features=128, out_features=64)
        self.fc5 = nn.Linear(in_features=64, out_features=32)
        self.fc6 = nn.Linear(in_features=32,out_features=self.lablen)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = torch.relu(self.fc3(x))
        x = torch.relu(self.fc4(x))
        x = torch.relu(self.fc5(x))
        x = self.fc6(x)
        return x
```

动态划分数据集

```python
# 动态划分验证集
if (epoch + 1) % (validate_every * redivide_every) == 0:
    combined_X = torch.cat( tensors: (X_train, X_val),  dim: 0)
    combined_y = torch.cat( tensors: (y_train, y_val),  dim: 0)
    combined_X = combined_X.numpy()
    combined_y = combined_y.numpy()

    timestamp = int(datetime.datetime.now().timestamp())
    new_train_X, new_val_X, new_train_Y, new_val_Y = train_test_split( *arrays: combined_X, combined_y, tes
                                                                       random_state=timestamp)
    X_train = torch.from_numpy(new_train_X)
    X_val = torch.from_numpy(new_val_X)
    y_train = torch.from_numpy(new_train_Y)
    y_val = torch.from_numpy(new_val_Y)
```

验证函数&评估指标的计算

```python
# 验证函
1 个用法
def validate(model, criterion, X_val, y_val, threshold):
    model.eval()
    with torch.no_grad():
        outputs = model(X_val)
        loss = criterion(outputs, y_val.squeeze().float())

        # 根据阈值确定预测选择的药品
        predicted_labels = torch.gt(outputs, threshold).int()

        # 计算准确率、召回率和F1分数
        true_positives = torch.sum(predicted_labels * y_val).item()
        predicted_positives = torch.sum(predicted_labels).item()
        actual_positives = torch.sum(y_val).item()

        precision = true_positives / predicted_positives if predicted_positives > 0 else 0
        recall = true_positives / actual_positives if actual_positives > 0 else 0
        f1 = 2 * precision * recall / (precision + recall) if precision + recall > 0 else 0

    return loss.item(), precision, recall, f1
```

训练过程可视化

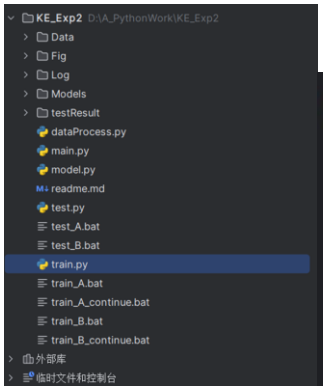
```python
plt.figure(figsize=(12, 6))

# 绘制验证损失
plt.subplot( *args: 1, 2, 1)
plt.plot( *args: range(0, num_epochs, validate_every), val_losses, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title(f'Validation Loss of {args.Group}')
plt.legend()

# 绘制准确率、召回率和F1分数
plt.subplot( *args: 1, 2, 2)
plt.plot( *args: range(0, num_epochs, validate_every), precisions, label='Precision')
plt.plot( *args: range(0, num_epochs, validate_every), recalls, label='Recall')
plt.plot( *args: range(0, num_epochs, validate_every), f1_scores, label='F1 Score')
plt.xlabel('Epoch')
plt.ylabel('Score')
plt.title(f'Metrics of {args.Group}')
plt.legend()

# 添加epoch数和日期到图像
plt.text(num_epochs - 5,  y: 0.9,  s: f'lr: {args.lr}', ha='right',zorder=10)
plt.text(num_epochs - 5,  y: 0.85,  s: f'Epochs: {num_epochs}', ha='right',zorder=10)
plt.text(num_epochs - 5,  y: 0.80,  s: f'Date: {current_time}', ha='right',zorder=10)
```

项目目录结构和执行脚本

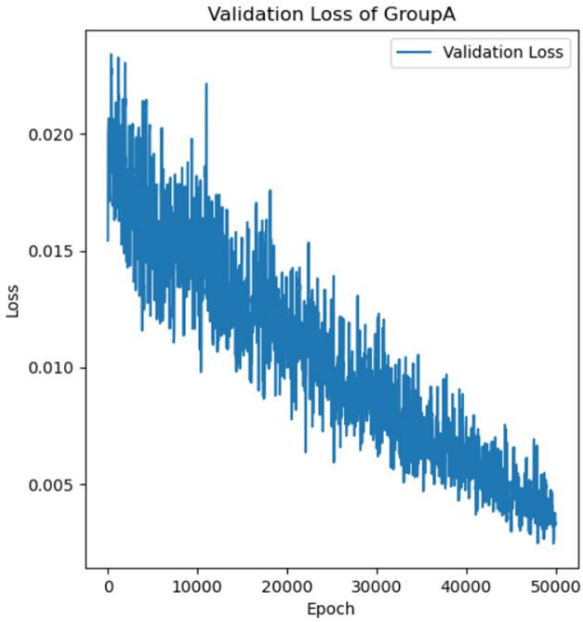


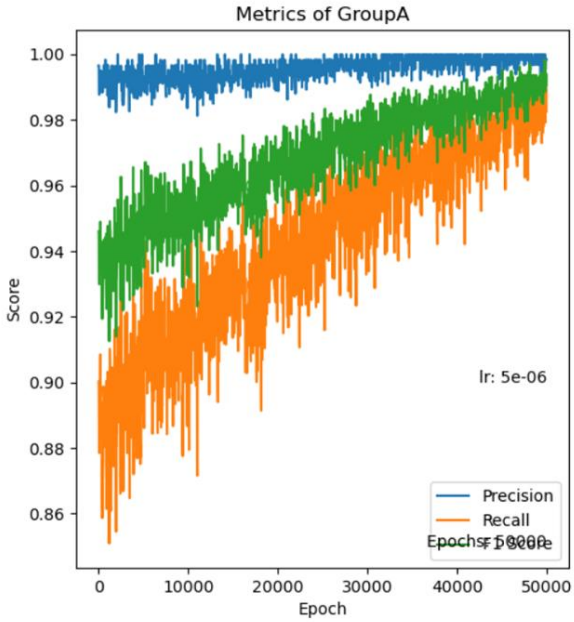
文件功能

- **train_A.bat**是训练A组模型并保存;
- **train_B.bat**是训练B组模型并保存;
- **train_A_continue.bat**是加载之前训练的A组模型继续训练并保存;
- **train_B_continue.bat**是加载之前训练的B组模型继续训练并保存;
- **test_A.bat**用于测试A组数据并保存测试结果;
- **test_B.bat**用于测试A组数据并保存测试结果。

./Data是数据文件，train是训练数据，test是测试数据，训练数据会在训练过程中自动划分训练集和验证集;
./Fig是训练结果可视化，文件名后缀是日期时间;
./Models是加载的训练权重，如想加载不同的模型，用记事本打开bat或者用pycharm改对应的bat文件参数就可以了;
./testResult是测试结果，这里的测试集是伪测试集，从训练集中抠出来的。

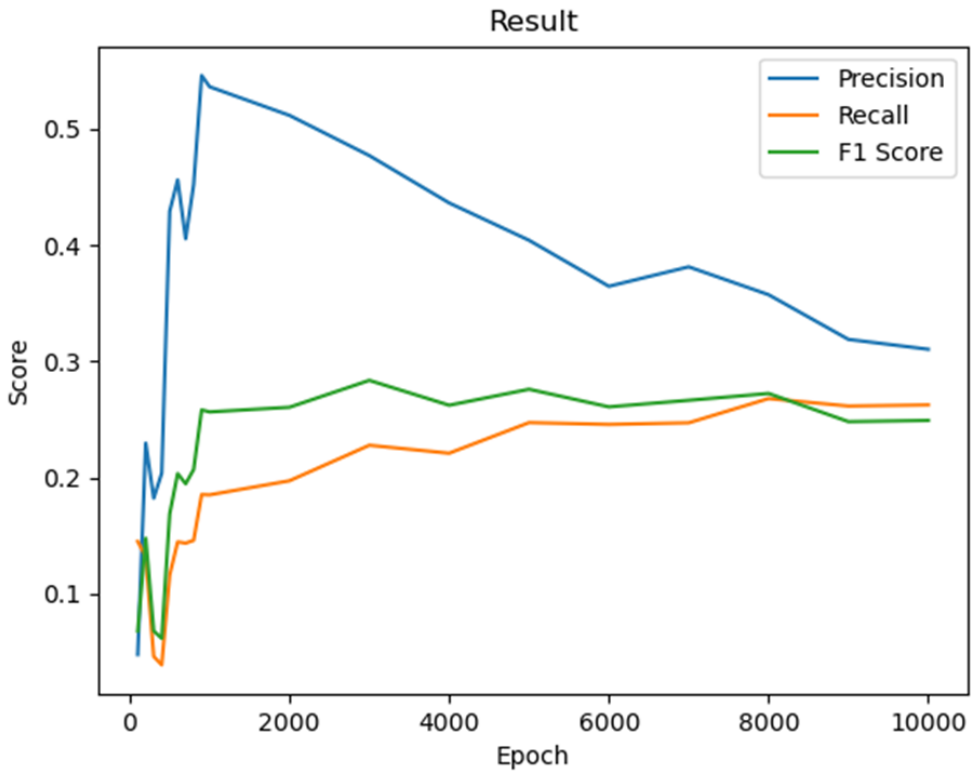
初期训练收敛结果





（7） 测试结果：

训练不同的批次的模型对带药预测的准确率，召回率，F1。



五、 实验中遇到的问题

1、 NER 提取准确率不高：因为 doccano 手动标注标签数量太少，导致 transfomers 模型准确率不高，而且切片错位情况非常频繁，也并不会识别别名和本名，导致最后预测准确率因为这些因素变低。

2、 模型预测召回率较低：经过训练，在最优时，准确率有 50%左右，但召回率较低，只有 20%左右，经过分析可能原因如下：（1）在训练和预测中我们只选取了出现频率最多的前 100 种药，存在一定缺失；（2）我们组的知识图谱中提取的药名纯在部分遗漏；（3）我们使用的全连接模型比较简单，对药的选择上没有学习到不同药之间的特征区别；（4）数据集中数据风格迥异，对同一种情况可能有多种带药方案，方案并不固定，而不同医生的给药风格不一样，对训练产生了影响；（5）数据集中数据的特征过多，且在处理中还添加了出现频率最多的前 100 种病，而数据集较小，这可能导致无关特征对训练结果产生影响。

六、 实验总结

本次知识工程专题实践中，我们选择了糖尿病知识图谱构建及用药推荐这一专题，在实践过程中，我们根据个人所擅长的不同方向及相关能力，将任务分为两大板块进行分工合作；同时在实践过程中，由于数据来源于真实生活中，所以数据集与之前机器学习，知识工程课程中所使用的数据集有较大差别，在专题实践中，数据处理和模型训练需要投入较大的时间，也遇到了不少的挑战和问题。在此情况下，我们小组成员之间，积极沟通和讨论，提出不同的解决方法，最后成功且高效的解决了相关问题，最后取得了较好的实践结果。