实验四：基于 GPLinker 的关系抽取实现

邓雅萱　学号：1120202004　教学班号：07112004

一、实验目的：关系抽取

关系抽取 (Relation Extraction, RE)，即从非结构化文本中抽取实体和对应关系，是自然语言处理中的基本任务。简而言之，给定一段文本，需要从中抽出（subject, predicate,object）三元组。

在论文《**TPLinker**：基于实体对链接的单阶段实体-关系联合抽取模型》[3] 一文中，作者提出了第一个真正意义上的单阶段联合抽取模型，在不受曝光误差的影响下成功解决了单一实体重叠、实体对重叠的问题，并在 NYT 和 WebNLG 两个关系抽取任务上均实现了当时的 SOTA 性能。在论文发布后，苏剑林借鉴其思想，利用自身曾经提出的 GlobalPointer 的 NER 模型，进一步推广出了 **GPLinker** 的关系抽取模型，使得模型较 TPLinker 具有更少的显存占用，并训练速度显著提升。

本实验参照 GPLinker 的部分 tensorflow 代码，对模型进行了 **pytorch 复现**，并在中文医疗信息处理挑战榜 CBLUE 中的 CMeIE 数据集上进行了实践。

1.1 任务难点

在抽取过程中，通常需要考虑下述三个问题：

(1) **交互缺失**：忽略了关系抽取和实体抽取的内在联系，导致实体抽取准确率高、关系抽取准确率低的问题。流水线方法中，先抽取实体再抽取关系的方式忽略了实体与关系之间在文本中的信息交互关系，产生了交互缺失问题。

(2) **实体重叠**：如何从嵌套的实体中提取出多种关系是一个重要问题。在传统的流水线（pipline）方法中，首先识别出文本中的实体，再对识别出的实体进行关系分类。因此，抽取出来的两个实体便只能对应一种关系，无法处理实体重叠的问题。具体而言，实体间的重叠关系主要分为以下三种情况（实例见图 1）：

- Normal：常规，即两个实体之间关系唯一且与其他实体不存在任何关系
- Single Entity Overlap：单一实体重叠，即一个实体与多个实体具有关系
- Entity Pair Overlap：实体对重叠，即同一对实体存在至少两种关系

(3) **曝光偏差**：来源于训练阶段与推理阶段的差异——训练时，模型接受真实的实体片段标签作为输入，实际推理时却以前一单元的输出作为当前单元的输入，当错误的输出成为下一单元的输入，训练与推理时的输入会在分布上存在偏差，会产生误差累计的问题，在一定程度上损害模型的性能。

	Texts	Triplets
Normal	[The United States] President [Trump] will meet [Xi Jinping], the president of [China].	(The United States, *president*, Trump) (China, *president*, Xi Jinping)
SEO	Two of them, [Jeff Francoeur] and [Brian McCann], are from [Atlanta].	(Jeff Francoeur, *live in*, **Atlanta**) (Brian McCann, *live in*, **Atlanta**)
EPO	**[Sacramento]** is the capital city of the U.S. state of **[California].**	(**California**, *contains*, **Sacramento**) (**California**, *capital city*, **Sacramento**)

图 1-1 实体重叠的例子

为了处理问题 1，联合抽取（joint extraction）的方法应运而生，它充分利用实体与关系之间的信息交互关系，在一个模型中同时对实体和关系进行统一的抽取。目前，改进的联合抽取方法已经可以较好地处理问题 2，却无法处理曝光偏差的问题：本质上，这些基于解码机制的模型在解码阶段仍需要分多个步骤对同一三元组中的实体、关系进行抽取。

单阶段联合抽取模型 TPLinker、GPLinker 在成功解决上述重叠关系的问题的同时，不产生曝光偏差，实现了较好的关系抽取效果。

1.2 实验方法

本实验主要通过以下步骤，实现了关系抽取的目标：

(1) 对 **CMeIE 数据集**原始数据进行预处理，提取出文本及对应关系的标签。

(2) 采用 **RoBETa 中文预训练模型**对句子进行编码，并采用 entity_labels、head_labels、tail_labels 分别存放实体头尾、实体对的 head 与实体对的 tail 的下标标签。

(3) 将`entity_labels`、`head_labels`、`tail_labels`的预测转化为 3 个 NER 问题：

- `pred_entity`：将 entity_labels 的预测转化为 2 个类型（subject、object）的 NER，识别实体的 head 和 tail。
- `pred_head`：将 head_labels 的预测转化为 m 个类型（关系的个数）的 NER，识别 subject 的 head 和 object 的 head。
- `pred_tail`：将 tail_labels 的预测转化为 m 个类型（关系的个数）的 NER，识别 subject 的 tail 和 object 的 tail。

采用 3 个 **GlobalPointer** 构成 GPLinker 模型，对上述 3 个问题分别进行预测：

- 将经过 BERT 编码的向量序列放入线性层，得到每个类型的每个序列对应的 start 向量和 end 向量。
- 采用**旋转式位置编码（RoPE）**为 start 向量和 end 向量分别引入相对位置信息，采用 **multi-head attention** 的思想，计算每个类别的首、尾首尾的分数。
- 分数大于 0 的部分即为所预测结果。

(4) 采用**稀疏多标签交叉熵损失**，对预测结果分别计算损失函数，对结果求和取平均。

(5) 对损失函数求导，采用 **Adam** 优化器，引入权重衰减、预热学习率、梯度修剪等方法，对模型参数进行反向传播更新，并保存训练好的参数。

(6) 根据训练好的参数进行上述预测过程，对 GlobalPointer 的结果取交集得到 GPLinker 的预测结果。将`pred_entity`、`pred_head`、`pred_tail`与`entity_labels`、`head_labels`、`tail_labels`进行比较，计算训练集、验证集的 **F1-measure** 大小，选取最优的参数用于测试集的预测。

(7) 将步骤（6）保留的参数用于测试集，计算其 F1-measure。

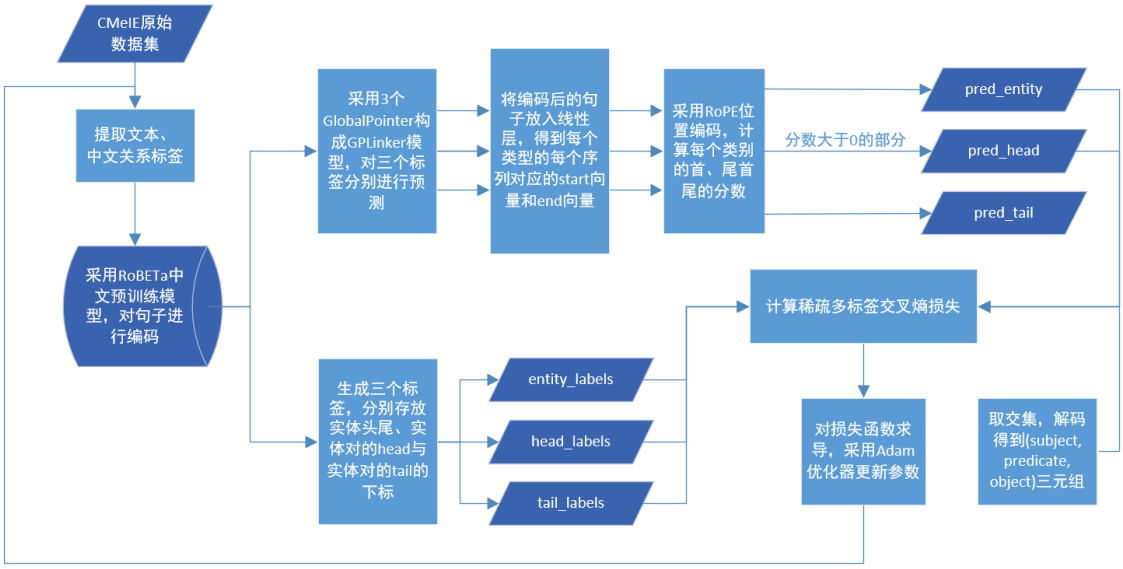

图 1-2 全过程流程图

目录

二、实验原理：GPLinker

2.1 基本思想

对于关系抽取问题，抽取三元组 (s, p, o)（即 **s**ubject,**p**redicate,**o**bject）的过程，在实现的时候可以转化为对**五元组**(s_h, s_t, p, o_h, o_t) 的抽取 [7]，其中，s_h, s_t 分别代表 s 的首、尾位置，o_h, o_t 分别代表 o 的首、尾位置。考虑采用如下方式构建关系抽取模型：

1. 设计五元组的打分函数 $S(s_h, s_t, p, o_h, o_t)$；
2. 训练时，使得标注的五元组 $S(s_h, s_t, p, o_h, o_t) > 0$，其余五元组 $S(s_h, s_t, p, o_h, o_t) < 0$；
3. 预测时，枚举所有可能的五元组，$S(s_h, s_t, p, o_h, o_t) > 0$ 的五元组即为抽取的目标。

然而，直接枚举所有的五元组是长度四次方级别的计算量，复杂度较高。因此，对打分函数进行分解，将计算量缩减至长度的平方级别：

$$S(s_h, s_t, p, o_h, o_t) = S(s_h, s_t) + S(o_h, o_t) + S(s_h, o_h|p) + S(s_t, o_t|p) \tag{2.1}$$

从直观的角度来理解，$S(s_h, s_t), S(o_h, o_t)$ 为 subject、object 的首尾打分，通过打分是否大于 0，可以判断该序列是否为 subject、object。$S(s_h, o_h|p), S(s_t, o_t|p)$ 则对应 predicate 的匹配，通过确定实体的头部（即 $S(s_h, o_h|p) > 0$），可以将抽取的实体与关系匹配起来。考虑嵌套实体的情况，需要增加对实体的尾部的匹配来确定实体，对应于 $S(s_h, o_h|p)$ 这一项。

因此，训练和预测过程可以转化为：

1. 训练时，标注的五元组 $S(s_h, s_t) > 0, S(o_h, o_t) > 0, S(s_h, o_h|p) > 0, S(s_t, o_t|p) > 0$，其余的五元组 $S(s_h, s_t) < 0, S(o_h, o_t) < 0, S(s_h, o_h|p) < 0, S(s_t, o_t|p) < 0$；
2. 预测时，枚举所有可能的五元组，将 $S(s_h, s_t) > 0, S(o_h, o_t) > 0, S(s_h, o_h|p) > 0, S(s_t, o_t|p) > 0$ 的部分分别进行输出，并将上述四个条件的交集作为最终的五元组输出。

在实现上，$S(s_h, s_t), S(o_h, o_t)$ 用于识别 subject、object 对应的实体，相当于识别 2 种类型的 NER 任务；$S(s_h, o_h|p)$ 用于识别 predicate 为 p 的 (s_h, o_h) 对，假设关系个数为 m 种，可以类似地转化为识别 m 种类型的 NER 任务，实践时只需去掉 $s_h \leq o_h$ 的约束；$S(s_t, o_t|p)$ 与 $S(s_h, o_h|p)$ 同理。

GlobalPointer [5] 是一个利用全局归一化的思路实现上述操作的模型，运用三个 GlobalPointer，可以实现关系抽取的目标。

2.2 GlobalPointer

假设文本序列长度为 n，则文本中有 $n(n+1)/2$ 个连续子序列。由于一段文本序列中可能存在多个实体，抽取实体的过程即为 "**n**$(\mathbf{n}+\mathbf{1})/\mathbf{2}$ **选 k**" 的多标签分类问题。假设实体类型共有 m 种，可以转换为**m 个** "**n**$(\mathbf{n}+\mathbf{1})/\mathbf{2}$ **选 k**" 的多标签分类问题。

2.2.1 数学形式

假设长度为 n 的输入句子 t，经过 BERT 编码后得到的向量序列为 $[h_1, h_2, \cdots, h_n]$，将每个 token 的编码放入线性层，得到每个类型的每个序列对应的 start 向量和 end 向量。

$$\begin{aligned} q_{i,p} &= W_{q,p} h_i + b_{q,p} \\ k_{i,p} &= W_{k,p} h_i + b_{k,p} \end{aligned} \tag{2.2}$$

其中，p 对应一个抽取类型。

因此，可以得出 m 个实体对分数序列。其中，类别为 p 的分数序列为：

$$[q_{1,p}, q_{2,p}, \cdots, q_{n,p}]$$
$$[k_{1,p}, k_{2,p}, \cdots, k_{n,p}]$$

对于不同的 $span[i:j]$，采用了 **Multi-Head Attention** 的思想类型 p 进行打分：

$$s_p(i,j) = q_{i,p}^T k_{j,p} \tag{2.3}$$

对应类型 p 所抽取的序列的首、尾分别为 i 和 j 的分数。

2.2.2 相对位置

由于式 (2.3) 没有显式包含相对位置信息，考虑采用**旋转式位置编码（RoPE）**[6]。该编码实质上引入了变换矩阵 R_i, R_j，对 $q_{i,p}, k_{j,p}$ 序列进行变换，得到包含位置信息的 $R_i q_{i,p}, R_j k_{j,p}$，其中，变换矩阵满足 $R_i^T R_j = R_{j-1}$。改动后的评分矩阵如下：

$$s_p(i,j) = (R_i q_{i,p})^T (R_j k_{j,p}) = q_{i,p}^T R_i^T R_j k_{j,p} = q_{i,p} R_{j-i} k_{j,p} \tag{2.4}$$

从而显式地为打分 $s_p(i,j)$ 注入相对位置信息。对于 $R_i q_{i,p}$ 的具体计算方式，在下一小节中展开详细阐述。

2.2.3 RoPE 编码

对于位置 m，设位置向量的维度为 d 的形式如下：

$$\begin{pmatrix} \cos m\theta_0 & -\sin m\theta_0 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_0 & \cos m\theta_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_1 & -\sin m\theta 1 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta 1 & \cos m\theta 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2-1} & -\sin m\theta_{d/2-1} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2-1} & \cos m\theta_{d/2-1} \end{pmatrix} \tag{2.5}$$

由于 R_i 较为稀疏，为了提高效率，在计算 $R_i q_i$ 时采用如下方法进行实现：

$$R_i q_i = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_{d-2} \\ q_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_0 \\ \cos m\theta_0 \\ \cos m\theta_1 \\ \cos m\theta_1 \\ \vdots \\ \cos m\theta_{d/2-1} \\ \cos m\theta_{d/2-1} \end{pmatrix} + \begin{pmatrix} -q_1 \\ q_0 \\ -q_3 \\ q_2 \\ \vdots \\ -q_{d-1} \\ q_{d-2} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_0 \\ \sin m\theta_0 \\ \sin m\theta_1 \\ \sin m\theta_1 \\ \vdots \\ \sin m\theta_{d/2-1} \\ \sin m\theta_{d/2-1} \end{pmatrix} \tag{2.6}$$

对于 θ_i，参考 Google 在《Attention is all you need》[2] 中提出的 **Sinusoidal 位置编码**：

$$\begin{cases} \boldsymbol{p}_{m,2i} = \sin\left(m/10000^{2i/d}\right) \\ \boldsymbol{p}_{m,2i+1} = \cos\left(m/10000^{2i/d}\right) \end{cases} \tag{2.7}$$

其中，$p_{m,2i}, p_{m,2i+1}$ 分别是位置 m 的编码向量的第 $2i, 2i+1$ 个分量，d 是位置向量的维度。

类似地，对于式（2.6），选取 $\theta_i = 1000^{-2i/d}$，代入式（2.4），即可得到 RoPE 编码的生成方式。随着相对距离的变化，该种选取方式可以使得内积呈现衰减趋势。

2.3 稀疏多标签交叉熵损失

针对 GlobalPointer 的**m 个 “n$(n+1)/2$ 选 k”** 的多标签分类问题的设计，本节将单标签分类的交叉熵损失函数推广至多标签的情形，并针对抽取过程中正例远大于负例的情况，推广至更优的**稀疏多标签交叉熵损失** [4]。

2.3.1 从单标签分类到多标签分类

对于单标签分类的交叉熵损失函数，可以进行如下形式的变化：

$$-\log\frac{e^{s_t}}{\sum_{i=1}^{n}e^{s_i}}=-\log\frac{1}{\sum_{i=1}^{n}e^{s_i-s_t}}=\log\sum_{i=1}^{n}e^{s_i-s_t}=\log(1+\sum_{i=1,i\neq t}^{n}e^{s_i-s_t}) \tag{2.8}$$

通过 max 的光滑近似，可以对上式进行进一步的简化。

$$\log\left(\sum_{i=1}^{n}e^{s_i-s_t}\right)\approx\max\begin{pmatrix}0\\ s_1-s_t\\ \vdots\\ s_{t-1}-s_t\\ s_{t+1}-s_t\\ \vdots\\ s_n-s_t\end{pmatrix} \tag{2.9}$$

证明 当 $x\geq 0, y\geq 0$ 时，

$$\max(x,y)=\frac{1}{2}(|x+y|+|x-y|)$$

为了对 max 函数进行近似表示，可以对绝对值函数进行近似表示。绝对值函数的导数如下：

$$f'(x)=\begin{cases}1, x>0\\ -1, x<0\end{cases} \tag{2.10}$$

通过单位阶跃函数来对其进行近似：

$$\theta(x)=\begin{cases}1, x>0\\ 0, x<0\end{cases} \tag{2.11}$$

$$f'(x)=2\theta(x)-1 \tag{2.12}$$

在物理中，通常采用下式对 $\theta(x)$ 进行近似：

$$\theta(x)=\lim_{k\to+\infty}\frac{1}{1+e^{-kx}} \tag{2.13}$$

将式（2.13）带入式（2.12），积分可得：

$$|x|=\lim_{k\to+\infty}\frac{1}{k}\ln(e^{kx}+e^{-kx}) \tag{2.14}$$

从而得到 max 函数的积分公式：

$$\max(x,y)=\lim_{k\to+\infty}\frac{1}{2k}\ln(e^{2kx}+e^{-2kx}+e^{2ky}+e^{-2ky}) \tag{2.15}$$

由于 $x\geq 0, y\geq 0$，e^{-2kx} 和 e^{-2ky} 均趋近于 0，可以对式（2.15）进一步简化：

$$\max(x,y)=\lim_{k\to+\infty}\frac{1}{k}\ln(e^{kx}+e^{ky}) \tag{2.16}$$

不难发现，上述化简过程对任意实数 x、y 均成立，将其推广至多变量的形式：

$$\max(x_1,x_2,\cdots,x_n)=\lim_{k\to+\infty}\frac{1}{k}\ln\sum_{i=1}^{n}e^{x_i} \tag{2.17}$$

对于模型的训练，可以假设 $k=1$，将 k 融入模型自身的训练中，得到下述近似：

$$\max(x_1,x_2,\cdots,x_n)\approx\ln\sum_{i=1}^{n}e^{x_i} \tag{2.18}$$

式（2.9）得证。

通过上述证明过程可以发现，式（2.8）的单标签分类损失函数在本质上实现了“**每个目标类得分大于每个非目标类得分**”的效果。将单标签分类推广至多标签分类的情形，可以得到下式：

$$\log(1+\sum_{i\in\Omega_{neg},j\in\Omega_{pos}}e^{s_i-s_j})=\log(1+\sum_{i\in\Omega_{neg}}e^{s_i}\sum_{j\in\Omega_{pos}}e^{-s_j}) \tag{2.19}$$

其中，Ω_{pos}、Ω_{neg} 分别表示样本的正负类别。通过在 log 函数内加入 $e^{s_i-s_j}$，以实现 $s_i<s_j$ 的目的。

在 CVPR2020《Circle Loss: A Unified Perspective of Pair Similarity Optimization》[1] 一文中，作者给出了一个损失函数的统一计算形式。

$$\begin{aligned}L_{\text{uni}}&=\log\left(1+\sum_{i=1}^{K}\sum_{j=1}^{L}e^{\gamma\left(s_n^j-s_p^i+m\right)}\right)\\&=\log\left(1+\sum_{i=1}^{K}e^{\gamma\left(s_n^j+m\right)}\sum_{j=1}^{L}e^{-\gamma s_p^i}\right)\end{aligned} \tag{2.20}$$

可以发现，式（2.20）与式（2.19）具有相同的形式，从而从另一个角度证明该损失函数的正确性。

对于 GPLinker，我们希望目标类的分数都大于 0，非目标类的分数都小于 0。因此，可以对式（2.19）多标签分类的损失函数进行进一步修正：

$$\begin{aligned}&\log\left(1+\sum_{i\in\Omega_{neg},j\in\Omega_{pos}}e^{s_i-s_j}+\sum_{i\in\Omega_{neg}}e^{s_i-0}+\sum_{j\in\Omega_{pos}}e^{0-s_j}\right)\\&=\log\left(1+\sum_{i\in\Omega_{neg}}e^{s_i}\right)+\log\left(1+\sum_{j\in\Omega_{pos}}e^{-s_j}\right)\end{aligned}\tag{2.21}$$

2.3.2 从多标签分类到稀疏的多标签分类

在 GPLinker 的设计下，正负例比例极其不平衡。因此，可以使用 Ω_{pos} 和 $A=\Omega_{pos}\cup\Omega_{neg}$ 来实现式（），只根据正例的下标进行计算，以减少标签矩阵的尺寸，提高训练速度。对于负例，可以采用如下实现方式：

$$\begin{aligned}&\log\left(1+\sum_{i\in\Omega_{neg}}e^{s_i}\right)=\log\left(1+\sum_{i\in A}e^{s_i}-\sum_{i\in\Omega_{pos}}e^{s_i}\right)\\&=\log(1+\sum_{i\in A}e_i^s)-\log\left(1-(\sum_{i\in\Omega_{pos}}e^{s_i})/(1+\sum_{i\in A}e^{s_i})\right)\end{aligned}\tag{2.22}$$

设 $a=\log(1+\sum_{i\in A}e^{s_i}), b=\log(\sum_{i\in\Omega_{pos}}e^{s_i})$，上式可以采用如下方法对 a,b 进行表示：

$$\log(1+\sum_{i\in\Omega_{neg}}e^{s_i})=a+\log(1-e^{b-a})\tag{2.23}$$

通过式（2.23），即可得到负例损失函数更优的计算方式，正例的损失函数保持不变，得到稀疏多标签交叉熵损失的计算公式（a,b 的含义同上）：

$$L=\log\left(1+\sum_{j\in\Omega_{pos}}e^{-s_j}\right)+a+\log(1-e^{b-a})\tag{2.24}$$

由于正例个数无法确定，可以采用零填充的方式，使得每个样本的标签矩阵大小一致，并在上述损失函数计算时进行 mask 处理。

三、实验内容：基于 GPLinker 的关系抽取实现

3.0.1 代码各文件说明

(1) 模型配置：`config.ini`文件

包括数据集与预训练模型的路径、训练时的参数设置，方便了参数的管理，其中：

- [paths]：预训练模型 RoBERTa、数据集的位置
- [paras]：训练时的参数，其中，`head_size`对应 Multi-Head Attention 中 head 的维度

(2) 数据集存储：`datasets`文件夹

本实验采用医学实体关系抽取 CMeIE 数据集，以 json 文件的形式存储，包含以下文件：

- 53_schema.json: SPO 关系约束表
- CMeIE_train.json: 训练集
- CMeIE_dev.json: 验证集
- CMeIE_test.json: 测试集

(3) 数据预处理：`utils/data_loader.py`文件

该部分将数据集数据进行编码，并将数据、标签转化为可用于 GPLinker 模型所需要的形式。其中，`data_generator`类提供了 pytorch 的 DataLoader 在训练、推理时的数据生成方式。

(4) GPLinker 模型：`GPlinker.py`文件

采用 pytorch 的神经网络库，实现了关系抽取 GPLinker 模型，并规定了损失函数的计算方式。

(5) 优化方法：`utils/bert_optimization.py`文件

采用 HuggingFace 针对 Bert 的 Adam 优化器，即文件中的 BertAdam 函数，并在训练过程中采用权重衰减、学习率预热、梯度剪裁等方式。

(6) 模型参数存储：`result/GPLinker_para.pth`文件

训练结束后，存储模型的所有参数，用于评估模式下的关系抽取预测。

(7) 训练与评估：`main.ipynb`文件

完成 GPLinker 的正反向传播，存储各项参数，并将其用于模型的评估。**相关说明已呈现于 jupyter notebook 中**，在此不再赘述。

3.1 数据预处理

本实验采用医学实体关系抽取 CMeIE 数据集，为了更好地查看标签，将原始数据的验证集随机划分验证集和测试集。每条数据由 text 和 spo_list 组成，分别为中文文本与对应的关系三元组标签，数据集信息如下：

表 3-1 模型数据信息

类型	训练集	验证集	测试集
句子数量	14339	2008	1577
关系三元组数量	44006	6020	4641

对于中文文本，采用 **RoBERTa 的中文预训练模型**对文本进行编码，得到每个句子的编码`input_ids`和对应的`attention_mask`。

对于标签，需要将原始数据集 (subject, predicate,object) 的三元组形式转化为 GPLinker 模型所能处理的形式，最终转化为以下三项：

(1) `entity_labels`：形状为 $[2 \times m \times 2]$，`entity[0]`为 subject，`entity[1]`为 object
(2) `head_labels`：形状为 [关系数 $\times n \times 2$]，存储 subject 和 object 的 head 下标
(3) `tail_labels`：形状为 [关系数 $\times p \times 2$]，存储 subject 和 object 的 tail 下标

其中，m, n, p 为相关标签 padding 后的长度。该部分的代码实现，见`data_loader.py`文件

3.2 GPLinker 模型实现

`GPlinker.py`文件实现了模型的基本架构：

- `GlobalPointer`类：GPLinker 模型的组成单位
- `sparse_multilabel_categorical_crossentropy`函数：GloabalPointer 的损失函数——稀疏多标签交叉熵损失的计算
- `GPLinker`类：GPLinker 模型的实现

下面，对各部分进行简要介绍。

3.2.1 基本单位——GlobalPointer 的实现

在 GlobalPointer 类中，模型的输入为 RoBERTa 编码后的句子及对应的`attention mask`，目的为抽取该句子 m 种类型的实体，输出不同类型实体的 start 和 end 在该句子中的下标。

对于 m 种预测类型，将编码后的句子输入该模型，可以得到每个类型在句子的每个位置作为 start、end 的分数向量。接着，引入 **RoPE 编码**，根据式 (2.4)，为分数向量引入位置信息。在计算 RoPE 的时，借助了 Sinusoidal 位置编码的思想（在`sinusoidal_position_embedding`函数中实现）。对引入位置信息后的分数向量，利用 **multi-head attention** 的思想计算内积，得到每个类型每对 $(start, end)$ 下标对应的评分矩阵。

由于输入句子在编码时为了保持句子长度一致，引入了`attention mask`，在计算时需要排除其影响；对于 start 一定在 end 之前的实体抽取，可以采用下三角遮罩，排除 start 在 end 之后的情况。排除特殊情况之后，每个类型对应的评分矩阵中大于 0 的位置即为所求的的 $(start, end)$ 对应的坐标。

3.2.2 损失函数——稀疏多标签交叉熵损失的计算

`sparse_multilabel_categorical_crossentropy`函数实现了损失函数的计算，以对模型的各项参数进行更新。函数的输入`y_pred`为 GlobalPointer 预测得到的评分矩阵，`y_true`为 GlobalPointer 的标签（start 和 end 的下标），`mask_zero`默认为真，对应`y_true`标签中采用零填充的方式进行对齐；函数的输出即为损失函数的值。

损失函数的计算公式为：

$$\log\left(1+\sum_{i\in\Omega_{neg}} e^{s_i}\right)+a+\log(1-e^{b-a}) \tag{3.1}$$

其中，$A=\Omega_{pos}\cup\Omega_{neg}, a=\log(1+\sum_{i\in A}e^{s_i}), b=\log(\sum_{i\in\Omega_{pos}}e^{s_i})$。

输入后，对`y_pred`进行展平处理，并将`y_true`中的 $(start, end)$ 对应的下标对应到展平后的 y_pred 位置。需要注意的是，`y_pred[0]`不存放有效数值。由于`y\_true`中采用了零填充的方式，若存放有效数值会导致 start 和 end 均为 0 的标签、为了对齐标签使用的零填充两种情况对应于同一个位置，导致损失函数的错误计算。同时，对于`y_pred[0]`的具体数值，在计算公式的不同部分时设置为 $+\infty$ 或 $-\infty$，以免在求 e 指数影响计算结果。

3.2.3 GPLinker 的实现

GPLinker 由三个 GlobalPointer 构成，分别进行如下预测：

1. `pred_entity`：将`entity_labels`的预测转化为 2 个类型（subject 和 object）的 NER，识别的 $(start, end)$ 对应实体的 head 和 tail。
2. `pred_head`：将`head_labels`的预测转化为 m 个类型（关系的个数）的 NER，识别的 $(start, end)$ 对应 subject 的 head 和 object 的 head。
3. `pred_tail`：将`tail_labels`的预测转化为 m 个类型（关系的个数）的 NER，识别的 $(start, end)$ 对应 subject 的 tail 和 object 的 tail。

在 GlobalPointer 实现的时候，需要注意的是，在预测`pred_entity`时需要下三角遮罩与 RoPE 编码，而在预测`pred_head`、`pred_tail`时不需要：subject 的 head 可能在 object 之后，$(subjecthead, objecthead)$ 的抽取与二者间下标的相对距离大小关系较小。

3.3 准确程度评价

采用 $F1-measure$ 的方法对结果进行评价，以较为全面地反映模型预测的准确程度。

首先，引入混淆矩阵，其中，列对应预测值，行对应实际值：

表 3-2 混淆矩阵

	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

计算**查准率**，即模型所有 (subject, predicate, object) 的预测中正确的比例，其计算公式如下：

$$Precision = \frac{TP}{FP + TP} \tag{3.2}$$

计算**查全率**，即模型所有 (subject, predicate, object) 的标签中被正确预测的比例，计算公式如下：

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

取查准率和查全率的调和平均数，得到评估指标 **F1 − measure**：

$$F1 - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3.4}$$

四、实验结果：loss 和 F1-measure 呈现

在RTX3090上训练 20 个 epoch（约 3 个小时）后，得到最终的训练结果。训练与测试过程已在main.ipynb中详细阐述，在此不再赘述。

模型的损失函数下降过程如下图所示，可以发现在 1 个 epoch 后，损失函数下降速度放缓并逐渐趋近于 0。训练结束时，损失函数的大小约为 0.07，已经在训练集上达到了较好的训练效果。训练集、验证集、测试集的结果如下：

表 4-1 训练结果

数据集	F1-measure	Precision	Recall
训练集	97.5%	97.3%	97.6%
验证集	62.8%	65.1%	60.6%
测试集	63.7%	66.1%	61.5%

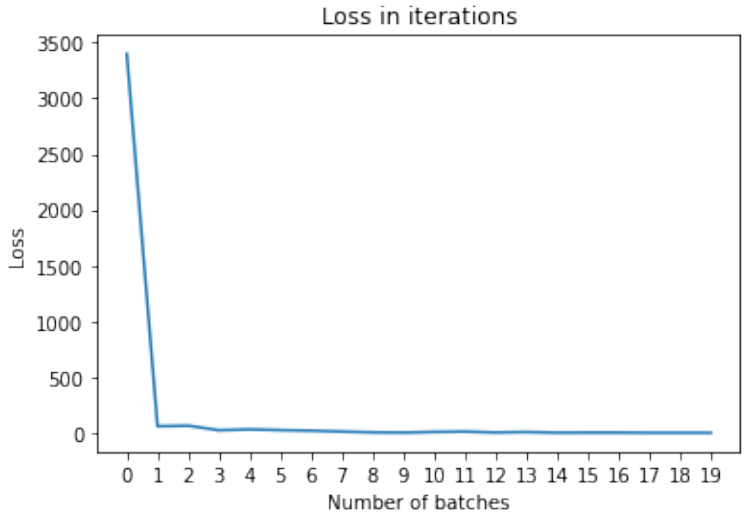

图 4-1 loss 下降的结果

分析结果可以发现，模型在验证集和测试集上的效果显著差于训练集。模型在训练集上准确率超过 90%，而在验证集、测试集上只能达到 60% 左右，存在较为严重的过拟合问题。在该数据集的 F1-measure 排行榜上，最高也不超过 65%。应当意识到，尽管 TPLinker、GPLinker 在 WEBNLG 等英文数据集上的预测准确程度已超过 90%，在中文数据集上的关系抽取仍有很大的进步空间。

对于模型可能的改进方案，考虑引入正则化项，以避免参数过大，抑制过拟合。

五、实验心得：挑战与收获

在文献阅读的作业中，我学习了单阶段实体-关系联合抽取模型 TPLinker。由于现有关系抽取的实践多针对于英文数据集，对中文数据集的实践相对较少，因此决定在中文数据集上进行一些尝试。在准备复现相关代码时，我发现了 GPLinker 模型，它相较 TPLinker 具有更少的显存占用，在训练速度显著提升（约 3 倍）的同时又具有较高的准确率，因此产生了在 CMeIE 数据集运用 GPLinker 进行关系抽取实践的动机。

在本次实验中，我完成了关系抽取任务的全过程。在实践过程中，我遇到了不少困难：

- 模型的理解：GPLinker 虽然借鉴了 TPLinker 的思想，但在模型的具体结构上与 TPLinker 具有显著的区别，同时该模型的具体内容均发布在苏剑林的多篇个人博客上，没有一篇完整、系统的描述模型的论文，在推导过程中包含大量数学公式，因此，前期花了大量时间在模型的理解上。经过对文章的反复阅读与思考后，对模型架构最终有了较好的理解。

- 代码实现：采用 pytorch 框架完成了从数据预处理、模型定义、训练再到预测、评估的全过程。限于个人能力无法独立完成模型全过程的代码，而网上 GPLinker 的开源的代码较为零散且注释较少，在阅读理解过程中遇到了不少困难。通过对代码的仔细阅读，甚至对大块代码的逐行调试，我理解并实现了模型的架构、训练的过程，并独立完成了预测、评价等部分，并尝试了 jupyter notebook 的方式更清楚地完成模型的训练与结果的呈现。

当然，过程中也收获了很多，只有在独立完成对一个新模型的理解后才克服了自身对文献阅读的畏惧心理，在个人完成一个较大项目的时候才发现阅读没有注释的代码、独立完成一个工程并没有想象中那么难。在对模型复现的过程中，不但个人的理解能力与代码的实践能力得到了提升，而且知识工程任务构建的思维方法被加深，为后续在相关领域进行分析、设计提供了思考方向。

参考文献

[1] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6398–6407, 2020.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[3] Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. Tplinker: Single-stage joint extraction of entities and relations through token pair linking. *arXiv preprint arXiv:2010.13415*, 2020.

[4] 苏剑林. 将“softmax+ 交叉熵”推广到多标签分类问题, Apr 2020.

[5] 苏剑林. Globalpointer：用统一的方式处理嵌套和非嵌套 ner, May 2021.

[6] 苏剑林. Transformer 升级之路：2、博采众长的旋转式位置编码, Mar 2021.

[7] 苏剑林. Gplinker：基于 globalpointer 的实体关系联合抽取, Jan 2022.