利用 PyTorch 实现基于 LSTM+CRF 的命名实体识别模型

梨花先雪

摘要

长短时记忆（LSTM）与条件随机场（CRF）的结合在序列标注任务中表现出色，尤其在自然语言处理（NLP）领域的命名实体识别（NER）、分词、词性标注等任务中取得了显著的效果。本文主要探讨 LSTM+CRF 结构的原理、特点及应用，通过分析其时间依赖性建模能力和标签序列的全局最优性，揭示该模型在复杂语境下的优势。此外，本文通过实验验证了该方法在特定任务中的性能，并对其局限性与优化方向进行讨论。研究表明，LSTM 通过门控机制有效捕获长距离依赖关系，而 CRF 通过条件概率建模实现全局优化，从而提升序列标注的准确性。本文的研究为进一步改进 LSTM+CRF 模型提供了参考，并为相关任务提供了有效的解决方案。

关键词— LSTM，CRF，NER，序列标注，自然语言处理，神经网络

1. 引言

在自然语言处理（NLP）领域，序列标注任务是基础且重要的研究方向，广泛应用于命名实体识别（NER）、词性标注、分词等任务。传统方法如隐马尔可夫模型（HMM）、条件随机场（CRF）等在特定场景下取得了较好效果，但由于其依赖于人工特征工程，难以捕获复杂的上下文信息。随着深度学习的兴起，循环神经网络（RNN）及其变体长短时记忆（LSTM）网络在序列建模任务中展现出强大的能力。

然而，LSTM 仅能建模局部时间依赖关系，在序列标注任务中，独立预测每个标签可能导致全局最优性受损。为此，LSTM 与 CRF 的结合成为近年来研究的热点。LSTM 通过门控机制捕获长期依赖信息，而 CRF 则能够利用标签间的依赖关系进行全局优化，从而提高序列标注的准确率。

本文旨在深入研究 LSTM+CRF 结构的工作原理、优势及其在 NLP 任务中的应用。我们将首先介绍 LSTM 和 CRF 的基本理论，并分析其结合的数学原理。随后，通过实验验证 LSTM+CRF 在序列标注任务中的性能，并探讨其优化策略和应用前景。

2. 研究方法

2.1 LSTM+CRF 模型结构

该模型分为四层：第一层为词向量嵌入层，对单词进行表示；第二层为双向 LSTM 层，负责特征提取；第三层为全连接层，用于特征转换；第四层为 CRF 层，实现序列标注的全局优化。模型架构如下：

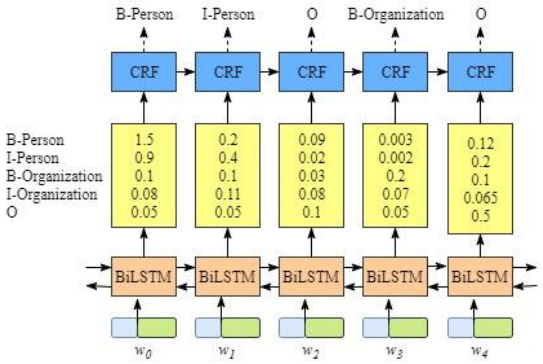


图 1 Bilstm-CRF 的模型结构

2.1.1 Embedding 层

在该模型中，无法直接使用字符串数据作为输入，而需要将其转换为数值列表（即 tensor 张量）才能进行计算。

本实验采用 PyTorch 库中封装的 nn.Embedding 函数进行词向量嵌入，其主要作用如下：

1. 整数索引到词向量映射：nn.Embedding 可将整数索引映射为对应的词嵌入向量，使文本数据能够以数值形式输入神经网络，如文本分类、情感分析等任务。

2. 学习词嵌入：词嵌入向量是可训练参数，在模型训练过程中，nn.Embedding 可根据数据特点学习最优的词表示，以提升模型性能。

3. 支持变长输入序列：nn.Embedding 可处理不同长度的文本，将每个单词映射为对应的词向量，并将其拼接或平均，作为模型的输入。

因此，我们首先需要构建一个词典，为文本数据中的每个中文字符分配唯一的整数索引。然后，将文本序列转换为对应的整数索引序列，并输入模型，以便在训练过程中学习合适的词向量表示。

2.1.2 BiLTSM 层

在本次实验中，BiLSTM 的作用是学习每个单词对应的命名实体概率。

在介绍双向 LSTM 之前，首先需要了解 LSTM 的概念。

LSTM（长短期记忆网络）是一种循环神经网络（RNN）架构，专用于处理时间序列数据和文本数据等具有时间依赖性的任务。该模型由 Hochreiter 和 Schmidhuber 于 1997 年提出，旨在解决传统 RNN 存在的梯度消失和梯度爆炸问题。

LSTM 的设计目标是捕捉长期依赖关系，并高效处理长序列数据。在序列建模过程中，LSTM 通过门控机制控制信息的流动和遗忘，从而有效保留长期记忆。这一门控机制包括遗忘门、输入门和输出门，分别决定哪些信息需要遗忘、存入记忆以及输出，以优化序列学习效果。

LSTM 网络的核心是 单元（cell），其中包含 遗忘门（Forget Gate）、输入门（Input Gate） 和 输出门（Output Gate），以及 单元状态（Cell State）。这些门的作用是控制信息的传递和更新，它们通过 Sigmoid 函数和乘法运算来决定数据的去留。

- **遗忘门** 负责判断需要丢弃的历史信息。
- **输入门** 决定当前时刻要加入的新信息。
- **输出门** 影响最终的输出结果，决定当前状态如何传递给下一步。

通过这种设计，LSTM 能够在处理序列数据时保留重要的信息，同时避免传统 RNN 容易出现的梯度消失问题。

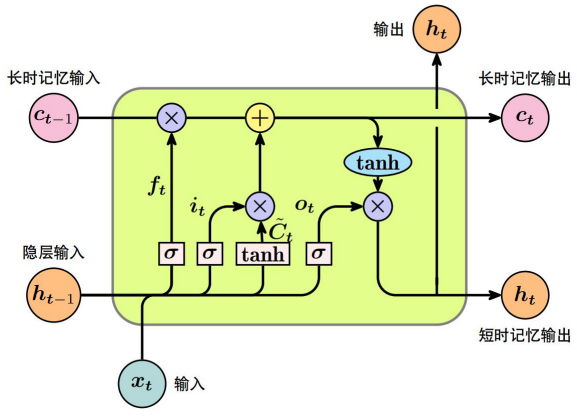


图 2 LSTM 元胞结构图

因此，LSTM 可以保留过去的信息，当前单词能够与前面的内容建立关系。然而，单向的 LSTM 只能学习到过去的信息，而双向 LSTM 就可以学习到两边的信息，这对于处理需要理解上下文信息的自然语言数据再合适不过了。因此，本实验中使用 BiLSTM，能够考虑到上下文信息并学习到每一个单词对应的实体标签概率。

在本实验中，我们采用 BiLSTM，使模型能够结合前后语境，更准确地学习每个单词对应的实体标签概率。

2.1.3 全连接层

在 BiLSTM 中，每个单词的输出是一个维度为 Hidden_size 的向量。为了将其转换为具体的标签概率，需要通过一个全连接层，将 Hidden_size 映射到 tag_size。本实验中，标签总数为 9，包括数据集中的 7 个标签，以及额外的起始符和终止符。

通过全连接层的映射，BiLSTM 的输出能够对应到 9 个命名实体标签的概率，从而为后续的序列标注任务提供基础。

2.1.4 CRF 层

全连接层的输出为每个单词对应各个标签的概率。虽然可以直接选择概率最高的标签作为预测结果，但由于单词处于句子上下文之中，单独取最大概率并不一定能得到最优的序列标注结果。因为句子中的单词具有上下文依赖性，整体的最优标注序列并不等同于逐个单词最大概率的简单组合。

为了解决这个问题，CRF（条件随机场）用于在全局范围内优化标注结果，通过考虑序列中不同实体之间的关系、上下文信息以及标签间的转移概率，提高命名实体识别的准确性和鲁棒性。

CRF 在命名实体识别中的作用如下：

1. 建模序列依赖关系：CRF 能够捕捉实体间的联系，例如人名通常包含姓氏或称谓，地名可能存在空间分布规律等。通过建模这种依赖关系，CRF 能更精准地识别实体边界。

2. 整体优化标注结果：相较于隐马尔可夫模型等局部模型，CRF 进行全局优化，通过最大化整个序列的标注概率来得到最优的标注序列，从而减少局部错误，提高标注精度。

3. 利用上下文信息：CRF 不仅关注当前单词的标注，还结合整个序列的上下文信息，增强模型对复杂语义和依赖关系的理解，适用于上下文影响较强的实体识别任务。

4. 缓解标注偏置问题：在实际任务中，模型可能倾向于选择较常见的标签，导致忽略一些罕见实体。CRF 通过联合建模减少这一偏置，提高模型的泛化能力和鲁棒性。

综合来看，通过 CRF，模型能够获取整句话最优的标注序列，即最大化整个句子的标注似然概率，从而提升命名实体识别的整体表现。

3. 实验与结果分析

3.1 数据处理

本次实验采用了一个相对小众的数据集：Youku，该数据集与优酷视频平台相关，主要包含娱乐领域的中文命名实体数据。数据集分为训练集和测试集，其中训练集包含约 10,000+ 条中文语句，测试集约 2,000+ 条中文语句。数据集格式如下：

```
变 B-TELEVISION
形 I-TELEVISION
金 I-TELEVISION
刚 I-TELEVISION
4 B-MISC
： O
绝 B-TELEVISION
迹 I-TELEVISION
重 I-TELEVISION
生 I-TELEVISION
```

数据集内容由两列组成，第一列表示中文语句，第二列表示命名实体标签，标签内容包括'O', 'B-MISC', 'I-TELEVISION', 'B-TELEVISION', 'I-MISC', 'B-PER', 'I-PER'，其中‘B’前缀表示命名实体开头，‘-’之后的后缀表示该实体的类别，例如‘TELEVISION’即表示电视节目相关的实体。‘I’表示命名实体的内部，‘O’表示命名实体的外部。每一句话通过一个空行分割。

我们选择对数据以列表的格式输入，每一个列表元素表示一句话的信息，为一个二元组，格式为（句子，实体标签），例如，上述信息经过数据处理之后就是：

[(['变', '形', '金', '刚', '4', '：', '绝', '迹', '重', '生'], ['B-TELEVISION', 'I-TELEVISION', 'I-TELEVISION', 'I-TELEVISION', 'B-MISC', 'O', 'B-TELEVISION', 'I-TELEVISION', 'I-TELEVISION', 'I-TELEVISION'])]

后续会将相应内容转换为张量，同时关联到 GPU 设备上。

3.2 实验流程

本次实验使用课堂发布的基线 baseline，做了如下调整：

1. 更改 tag_to_ix 字典。

因为实验使用了 Youku 数据集，标签不是简单的 BIO，更改后字典为：

```
tag_to_ix = {"B-TELEVISION": 0, "I-TELEVISION": 1, "B-MISC": 2, "I-MISC":3,"B-PER":4,"I-PER":5,"O":6,START_TAG: 7, STOP_TAG: 8}
```

2. 将所有张量和模型映射到 GPU 设备。

由于本实验涉及大量张量运算，在 CPU 上训练速度较慢，因此使用 .cuda() 函数将所有张量和模型映射到 GPU 加速训练。

3. 添加损失函数结果打印与可视化。

为便于分析实验结果，增加了损失值的打印功能，并绘制损失曲线。

4. 引入测试集评估模型泛化能力。

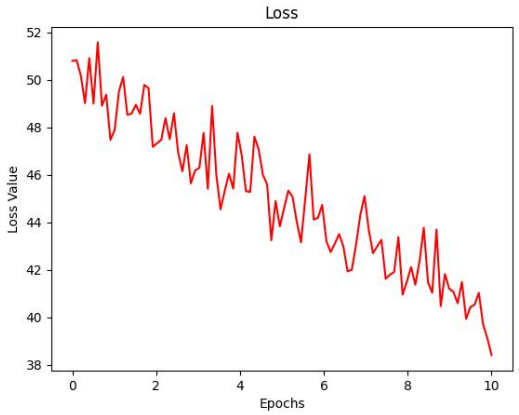
为了更全面地评估模型性能，额外添加了测试集，并在测试集上计算模型的准确率，以检验其泛化能力。

3.2.1 实验流程如下：

1. 对数据按照上述操作进行预处理。
2. 使用训练集训练模型，保存模型参数，绘制损失函数变化图。
3. 加载模型参数，在测试集上计算准确率。
4. 自己编写若干句话，调用模型观察结果。

3.2.2 实验结果：

首先，我绘制出训练时损失函数的变化图：



观察到损失函数波动下降，验证了模型收敛。

使用测试集计算代码的准确率（测试函数如下）：

```
with torch.no_grad():
    print("---------begin test----------")
    total=0
    corret=0
    cnt=0
    for sentence, tags in test_data:
        cnt+=1
        sentence_in = prepare_sequence(sentence, word_to_ix)
        sentence_in = sentence_in.cuda()
        targets = torch.tensor([tag_to_ix[t] for t in tags], dtype=torch.long)
        targets = targets.cuda()
        loss = model.neg_log_likelihood(sentence_in, targets)
        _,pre = model(sentence_in)
        pre=torch.tensor(pre)
        pre=pre.tolist()
        targets =targets.tolist()
        total+=len(pre)
        corret+=sum(1 for a, b in zip(pre, targets) if a == b)
        if(cnt%100==0):
            print(loss.item())
            print(corret/total)
```

得到的最终准确率为：0.7340697674418605

本次实验的准确率计算方式为 预测正确的标签数量占总样本数量的比值。实验结果较为理想，但仍有较大的优化空间。可能的原因包括：训练轮数不足（由于数据量较大，单轮训练耗时较长，因此未进行过多轮训练），以及 模型参数仍有优化空间，可通过进一步调整超参数提升性能。

4. 结论

在本次实验中，虽然我们基于初始的 baseline 代码开始，但该代码仅对简单数据进行了模拟，无法直接应用于真实数据。因此，我对代码进行了较大的改动。

由于源代码默认以 CPU 为基础进行训练，而面对 Youku 这个大规模数据集时，CPU 的训练速度显得非常缓慢，首个问题就是如何将所有张量迁移到 GPU。在多次遇到“模型绑定了不同设备”的错误后，我最终成功将所有张量映射到 CUDA 设备上，以加速训练过程。

在评估模型时，我最初计算了模型的汉明损失、精确率和召回率，但由于该任务与传统的分类问题有所不同，得到的结果并不理想。最终，我改为计算模型的 总体预测准确率，以便更好地评估模型的表现。

总体而言，这次实验加深了我对 BiLSTM 和 CRF 的理解，学习了文本特征标注方法。在处理类似文本分类任务时，结合命名实体标注可以为数据提供更多信息，从而提升模型的效果。

5. 参考文献

[1] CreateMoMo, "CRF Layer on the Top of BiLSTM - 1," CreateMoMo Blog, September 12, 2017.

[2] lavender 喵, "BiLSTM+CRF 模型详解," 知乎专栏, 2022 年 10 月 14 日.