Simplify the Usage of Lexicon in Chinese NER

林荣盛

摘要

最近，许多工作都试图利用词库来提高中文命名实体识别（NER）的性能。作为一个代表，Lattice-LSTM[1]在几个公开的中文 NER 数据集上取得了新的基准结果。然而，Lattice-LSTM 有一个复杂的模型架构。这限制了它在许多需要实时 NER 的工业领域的应用。在这项工作中，我们提出了一种简单而有效的方法，将单词词库纳入字符嵌入中。这种方法避免了设计复杂的序列建模结构，对于任何神经网络 NER 模型来说，它只需要对字符表示层进行细微的调整就可以引入词库信息。对四个基准中文 NER 数据集的实验研究表明，我们的方法的推理速度比最先进的方法快 6.15 倍，同时性能也更好。实验结果还表明，所提出的方法可以很容易地与 BERT 等预训练的模型结合起来。

关键词：中文命名实体识别；LSTM 神经网络；自然语言处理

1 引言

命名实体识别（Named Entity Recognition，NER）是自然语言处理（Natural Language Processing，NLP）领域的子任务，通常解释为从一段非结构化文本中，将那些人类通过历史实践规律认识、熟知或定义的实体识别出来，同时也代表了具有根据现有实体的构成规律发掘广泛文本中新的命名实体的能力。实体是文本中意义丰富的语义单元，识别实体的过程分为两阶段，首先确定实体的边界范围，然后将这个实体分配到所属类型中去。

NER 的提取对象随着相关评测会议的进行不断丰富。最先开始的英文文本实体集中在三大基本类——person（人物）、organization（组织机构）、geographical location（地理位置）上，同时辅助于currency（货币）、time（时间日期）、percentage expression（百分数表达式）的识别，前者属于实体类（entity type），后者属于数字类（numeric type）。而 person 类下包含了名字、昵称、代称、外文译名等识别任务，location 类对城市、道路、区划等名词进行识别。随着 NER 研究的推进，提取实体范围更广，实体分类更加精细，不同语种、不同学科领域被包含进来。

促进中文 NER 发展的会议有 SigHAN、863 中文 IP 评测会议等。NER 在 SIGHAN Bakeoff-2010 之后，不再作为评测任务出现，后续如命名实体消歧、命名实体链接任务被加入信息抽取任务中，NER 最新进展被发表在 ACL、AAAI、COLING、EMNLP、NAACL 等 NLP 顶级会议中。

2 相关工作

本章将会介绍中文命名实体识别任务的研究现状以及一些当前研究人员重点关注的研究方法。

2.1 传统的中文 NER 任务方法

在单词自然分离的语言中（如英语），NER 任务被传统地表述为一个序列标记问题，并且使用基于神经网络的模型取得了最先进的结果。与英语的 NER 任务相比，中文的 NER 任务更加困难，因为中文的句子不是自然分割的。因此，中文 NER 任务的常见做法是首先使用现有的 CWS 系统对句子进行词的分割，然后应用词级序列标签模型对实体进行预测。然而，CWS 系统在对句子进行分词的时

候会引入错误，这将直接导致对实体边界的检测和实体类别的预测也出现错误。因此，改进的方法采用直接在字符层面上进行中文 NER 任务[2]。

2.2 基于 LatticeLSTM 的中文 NER 任务方法

现在的一些方法采用直接在字符层面上进行中文 NER，纯粹基于字符的 NER 方法的一个缺点是单词信息没有得到充分的利用。基于这种考虑，Zhang 和 Yang（2018）提出了 Lattice-LSTM，将词库信息纳入基于字符的 NER 模型中。此外，当一个字符与词库中的多个词相匹配时，作者建议保留所有与字符相匹配的词，而不是启发式地选择某一个词作为该字符的附加信息，即所有与该字符相匹配的词，让后续的 NER 模型来决定使用哪个词。为了实现这一想法，他们对 LSTM-CRF[3]模型的序列建模层进行了详细的修改。在对四个中文 NER 数据集的实验上已经验证了 Lattice-LSTM 的在中文 NER 任务上具有优秀的表现。然而，LatticeLSTM 的模型结构是相当复杂的。为了引入词库信息，Lattice-LSTM 在输入序列中的非相邻字符之间增加了几条额外的边，这大大降低了其训练和推理的速度。此外，LatticeLSTM 的结构很难转移到其他神经网络架构（如卷积神经网络和 Transformers），这些架构可能更适合于一些特定的任务。

3 本文方法

3.1 本文方法概述

在我们的工作中，我们提出了一种更简单的方法来实现 Lattice-LSTM 的思想，即把每个字符的所有匹配词纳入到基于字符的 NER 模型中。我们设计模型的首要原则是实现快速的推理速度。为此，我们建议将词库信息编码在字符嵌入中，并设计了一个编码方案，以确保我们设计的编码方案尽可能多地保留了词汇表的匹配结果。与 Lattice-LSTM 相比，我们的方法避免了对复杂模型结构的需求，更容易实现，并且可以通过调整字符表示层快速适应任何适当的神经 NER 模型。此外，经过消融研究证明，我们的方法在纳入更完整、更明显的词库信息以及引入更有效的词汇加权策略方面具有优势。这项工作的贡献可以概括为以下几点:

(1) 我们提出了一种简单而有效的方法，将词库纳入中文 NER 的字符表示中。

(2) 所提出的方法可以转移到不同的序列标签架构中，并且可以很容易地与 BERT（Devlin 等人，2018）[Runge]等预训练的模型相结合。

我们在四个公开的中文 NER 数据集上进行了实验。实验结果表明，当用单层 Bi-LSTM 实现序列建模层时，我们的方法在推理速度和序列标注性能方面都比最先进的方法有相当大的改进。模型的结构如图 1所示:

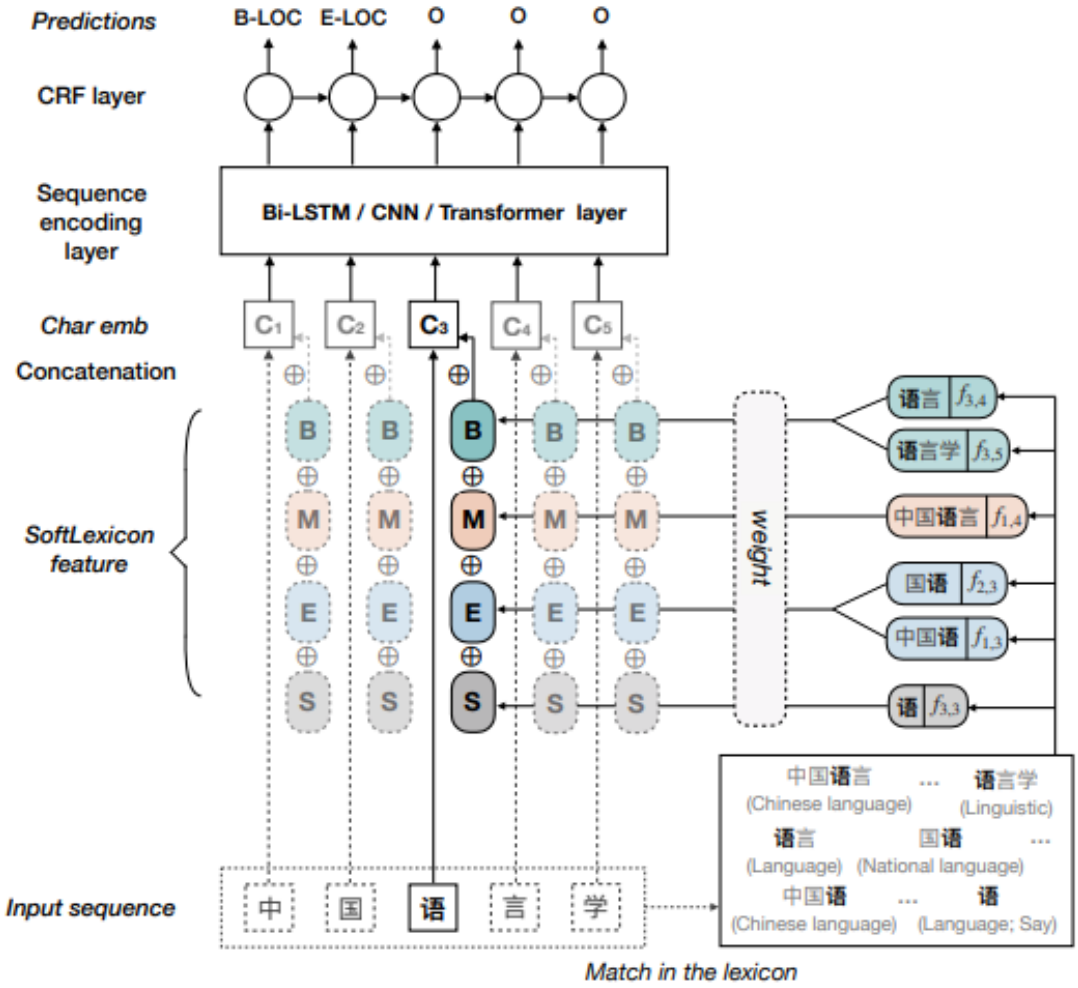

图 1: 模型总体结构

3.2 SoftLexicon 模块

纯粹基于字符的 NER 模型的问题是，它不能利用单词信息。为了解决这个问题，我们提出了 SoftLexicon 方法，如下所述，将单词信息引入到字符表示中。在下文中。对于任何输入序列 $s = \{c_1, c_2, ..., c_n\}$, $w_{i,j}$ 表示其子序列 $\{c_i, c_{i+1}, ..., c_j\}$。我们将此方法分为了三步：

(1) 首先，为了保留分割信息，将每个字符 c_i 的所有匹配词都归类进四个词组”BMES”，这四个词组由四个分割标签标记。对于输入序列 $= \{c_1, c_2, ..., c_n\}$ 中的每个字符 c_i，这四个数组的构建方法如下所示:

$$B(c_i) = \{w_{i,k}, \forall w_{i,k} \in L, i < k \leqslant n\},$$

$$M(c_i) = \{w_{j,k}, \forall w_{j,k} \in L, 1 \leqslant j < i < k \leqslant n\},$$

$$E(c_i) = \{w_{j,i}, \forall w_{j,i} \in L, 1 \leqslant j < i\},$$

$$S(c_i) = \{c_i, c_i \in L\}$$

这里，L 表示我们在这项工作中使用的词库。此外，如果一个词组是空的，则在空的词组中加入一个特殊的词”NONE”。通过这种方式，我们不仅可以引入词的嵌入，而且没有信息损失，因为匹配结果可以准确地从字的四个词组中恢复。

(2) 将词组里的信息进行压缩。在获得每个字符的”BMES ” 词集后，每个词集被压缩为一个固定维度的向量。为实现这个压缩，我们探索了两种方法来实现这种压缩。第一种实现方式是直观的均值

法。即使用如下公式对词集进行计算:

$$v^s(S) = \frac{1}{|S|}\sum_{w\in S} e^w(w)$$

这里，S 表示一个词集，e^w 表示词嵌入查询表。然而，实证研究的结果显示，这种算法的表现并不理想。因此，引入了一种加权算法来进一步利用词的信息。为了保持计算效率，我们没有选择像注意力那样的动态加权算法。相反，我们倾向于使用每个词的频率作为其权重的指示。由于一个词的频率是一个可以离线获得的固定值，这可以大大加快每个词的权重的计算。具体来说，让 $z(w)$ 表示一个词库单词 w 在统计数据中出现的频率，词集 S 的加权表示方法如下:

$$v^s(S) = \frac{4}{Z}\sum_{w\in S} z(2)e^w(w)$$

where:

$$Z = \sum_{w\in B\cup M\cup E\cup S} z(w)$$

在这里，对四个词组中的所有词进行了权重归一化，以进行整体的比较。在这项工作中，统计数据集是由训练集数据和开发集数据组合而成的。当然，如果任务中存在未标记的数据任务中，未标记的数据集也可以作为统计数据集。

(3) 与字符嵌入相结合。最后一步是将四个词组的嵌入合并为一个固定维度的特征。并将其添加到每个字符的表示中。为了保留尽可能多的信息。我们选择将四个词组的嵌入串联起来。每个字符的最终嵌入由以下方法获得：

$$e^s(B, M, E, S) = [v^s(B); V^s(M); v^s(E); v^s(S)],$$

$$x^c \leftarrow [x^c; e^s(B, M, E, S)]$$

v^s 表示加权函数。

3.3 损失函数定义

给定一组人工标注的训练数据 $\{(s_i, y_i)\}|_{i=1}^N$，我们使用句子级别的带有 L_2 正则化的对数似然损失函数用来训练模型。

$$L = \sum_{i=1}^{N} log(P(y_i|s_i)) + \frac{\lambda}{2}||\Theta||^2$$

4 复现细节

4.1 与已有开源代码对比

与原作者已经开源的代码相比，我们重新使用 pytorch 实现了一遍作者提出的模型，实现的模型结构基本上与作者的模型结构一样。我们还稍微修改了一下作者的源代码，使得作者的代码能够顺利的在自己的机器上跑起来，以进行训练。并在作者开源的代码上新加入了一个小功能，即实现了通过手动输入一段文本，然后用已经训练好的模型对这段文本进行实体识别任务，并将文本以及预测的结果自动编排好存放进预先定义好的路径中去的功能。该功能弥补了原作者的代码不能直接对输入的文本进行预测的缺陷，使得项目更加的完善，实用性也更加高，能更好的帮助到需要进行命名实体识别

任务的人。

4.2 实验环境搭建

我们使用 pytorch0.4.1 对模型进行实现，并且我们的项目可以在 CPU 上进行训练，也可以在 GPU 上进行训练，但推荐在 GPU 上训练，在 GPU 上的训练速度比在 CPU 上训练更快，而且结果更加准确。所以，请尽量在 GPU 上进行训练。我们的代码实现了可以根据参数来选择模型的具体结构的功能，不同的模型具有不同的训练速度和训练结果，可以根据自己设置的参数来选择。编码层具体可以选择三种模型结构，即 LSTM、CNN、Transformer 三种结构。实现代码如下图 2所示:

```
if self.model_type == 'lstm':
    self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers=num_layer, batch_first=True, bidirectional=biflag
    self.drop = nn.Dropout(dropout)
if self.model_type == 'cnn':
    self.cnn = CNNmodel(input_dim, hidden_dim, num_layer, dropout, gpu)
## attention model
if self.model_type == 'transformer':
    self.attention_model = AttentionModel(d_input=input_dim, d_model=hidden_dim, d_ff=2*hidden_dim, head=4,
    for p in self.attention_model.parameters():
        if p.dim() > 1:
            nn.init.xavier_uniform_(p)
```

图 2: 模型选择

4.3 界面分析与使用说明

我们提供了两种方式运行本项目。第一种是直接在 main 文件里点击运行即可运行文件，但需要在运行文件前手动修改需要设置的参数。手动设置好参数后，只需要点击运行按钮即可进行训练。第二种是使用脚本文件运行我们的项目，你可以在脚本文件里添加需要设置的参数，设置好参数后，点击运行后，可以直接运行本项目。参数设置实例如下图 3所示:

```
python main.py --test data/demo.test.char --modelname demo --savedset data/data.dset
```

图 3: 参数设置实例

4.4 创新点

我们在作者开源的代码上新加入了一个小功能，即实现了通过手动输入一段文本，然后用已经训练好的模型对这段文本进行实体识别任务，并将文本以及预测的结果自动编排好存放进预先定义好的路径中去的功能。该功能弥补了原作者的代码不能直接对输入的文本进行预测的缺陷，使得项目更加的完善，实用性也更加高，能更好的帮助到需要进行命名实体识别任务的人。

5 实验结果分析

我们对复现的模型进行了多次的训练，并对训练的结果进行了详细的分析。

5.1 计算效率研究

下表显示了 SoftLexicon 方法在用双 LSTM 层实现序列建模层时的推理速度。该速度是根据使用 GPU（NVIDIA TITAN X）的模型每秒处理的平均句子数来评估的。从表中，我们可以看到，当解码时，在相同的批处理量（=1）下，我们提出的方法比 Lattice-LSTM 和 LR-CNN 的效率高得多，比 Lattice-LSTM 快 6.15 倍。带有 bichar 的 SoftLexicon(LSTM) 的推理速度接近于没有 bichar 的推理速度，因为

我们只是将一个额外的特征连接到字符嵌入表示中。BERT-Tagger 和 SoftLexicon(LSTM)+BERT 模型的推理速度是有限的，因为 BERT 结构的层次很深。然而，在所有的数据集上，SoftLexicon（LSTM）+BERT 模型的速度仍然比 LatticeLSTM 和 LR-CNN 的速度快。速度对比图 4所示:

Models	OntoNotes	MSRA	Weibo	Resume
Lattice-LSTM	1×	1×	1×	1×
LR-CNN (Gui et al., 2019)	2.23×	1.57×	2.41×	1.44×
BERT-tagger	2.56×	2.55×	4.45×	3.12×
BERT + LSTM + CRF	2.77×	2.32×	2.84×	2.38×
SoftLexicon (LSTM)	6.15×	5.78×	6.10×	6.13×
SoftLexicon (LSTM) + bichar	6.08×	5.95×	5.91×	6.45×
SoftLexicon (LSTM) + BERT	2.74×	2.33×	2.85×	2.32×

图 4: 速度对比

5.2 训练结果对比

我们用自己重新搭建的模型在 Resume 数据集上进行训练，并与原作者实现的模型的训练结果进行了对比。对比结果显示，我们实现的模型在训练结果上效果略低于作者实现的模型，即在准确率、召回率和

我们实现的模型训练结果如图 5所示:

Models	P	R	F1
SoftLexicon(LSTM)	0.9361	0.9361	0.9361
SoftLexicon(LSTM)+bichar	0.9365	0.9411	0.9388
SoftLexicon(LSTM)+BERT	0.9470	0.9539	0.9504

图 5: 我们实现的模型训练结果

从训练结果可以看出，我们的模型的效果只是略微的降低，所以复现的结果大体上还是比较成功的，基本的实现了作者提出的模型。对于效果为什么会略微降低的问题，可能存在两个因素导致这种结果。首先，可能是由于我们复现的模型在一些细微的参数上没有与作者的模型保持一致，导致了效果的略微降低。为解决这个问题，在未来我们会继续参数调优的工作，尽量将模型的效果调至最优的状态。其次，可能是由于我们搭建的模型并没有完美的复制作者的模型，使得模型的性能并没有作者的好，导致了模型效果略微降低。对此，我们将继续研究作者的模型，使得将来我们可以完美的复制作者的模型，是模型的效果达到最优。

5.3 创新点结果展示

我们添加的新功能可以手动输入一段文本，然后利用训练好的模型对文本中的实体进行识别。经过测试，我们添加的新功能完美的达到了预期的效果，能够利用训练好的模型进行预测文本中的实体，并能够将结果进行保存到指定的路径。当然，我们的模型是在 Resume 数据集上训练的，所以我们的模型只能对类似简历的文本进行预测，预测实体类型包括机构、人名、地点、头衔等，这些类型以外的实体类型暂时还不支持。当然，也可以使用自己的数据集对模型进行训练，用训练好的模型对你的目标实体类型进行预测即可。这里，我们从百度百科上复制了一段类似简历的文本，并对其中存在的

实体进行了预测，预测结果的一部分如图 6所示:

```
一,O 直,O 在,O 中,B-ORG 国,M-ORG 中,M-ORG 医,M-ORG 研,M-ORG 究,M-ORG 院,E-ORG （,O 2,O 0,O 0,O 5,O 年,O 更,O 名,O 为,O
中,B-ORG 国,M-ORG 中,M-ORG 医,M-ORG 科,M-OR 学,M-ORG 院,E-ORG ）,O 工,O 作,O ，,O 期,O 间,O 晋,O 升,O 为,O 硕,B-TITLE 士,M-TITLE
生,M-TITLE 导,M-TITLE 师,E-TITLE 、,O 博,B-TITLE 士,M-TITLE 生,M-TITLE 导,M-TITLE 师,E-TITLE 。,O 现,O 为,O 中,B-ORG 国,M-ORG
中,M-ORG 医,M-ORG 科,M-ORG 学,M-ORG 院,E-ORG 首,B-TITLE 席,M-TITLE 科,M-TITLE 学,M-TITLE 家,E-TITLE ，,O 终,B-TITLE 身,M-TITLE
研,M-TITLE 究,M-TITLE 员,E-TITLE 兼,O 首,B-TITLE 席,M-TITLE 研,M-TITLE 究,M-TITLE 员,E-TITLE ，,O
```

图 6: 创新点结果展示

可以看到，文本中存在的各种类型的实体都被准确预测出来了。所以，这个功能达到了很好的效果。

6 总结与展望

在复现这篇论文的过程中，遇到了很多的问题，比如代码会有 bug、环境没有配置好导致项目运行不成功等等一系列的问题。最终，依靠自己上网查阅相关资料，咨询同学以及请教老师，成功的解决了这些问题，使项目成功运行了起来。通过这个课程，我的动手能力和分析问题的能力都有不同程度的提高。同时也扩展了我的视野，丰富了我的知识，也增强了我解决实际问题的能力，受益匪浅，对我今后的科研生活有着巨大的帮助。这些问题也提醒着我，在今后的研究生学习中，要更加仔细和谨慎，避免一些不必要的失误。在今后的学习中需要更加刻苦，更加努力。

在这项工作中，我们解决了在中文 NER 中利用词库的计算效率问题。为了获得一个高性能的中文 NER 系统的推理速度，我们提出了一种新型的的方法，将词库信息纳入字符嵌入。在四个基准的中文 NER 数据集上的实验研究表明我们的方法可以实现更快的推理速度和更好的性能，是目前表现最好的方法。但我们的工作还是有些不足之处，例如词库的里的词语正确率还不是很高，词语正确率对模型的最终准确率会有着不小的影响。所以，在未来，我们可以改进词库信息，使词库里的词语更加的准确，从而使得模型的准确率进一步提高。

参考文献

[1] MA R, PENG M, ZHANG Q, et al. Simplify the Usage of Lexicon in Chinese NER[C/OL]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020: 5951-5960. https://aclanthology.org/2020.acl-main.528. DOI: 10.18653/v1/2020.acl-main.528.

[2] SUI D, CHEN Y, LIU K, et al. Leverage Lexical Knowledge for Chinese Named Entity Recognition via Collaborative Graph Network[C/OL]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 3830-3840. https://aclanthology.org/D19-1396. DOI: 10.18653/v1/D19-1396.

[3] CHIU J P, NICHOLS E. Named Entity Recognition with Bidirectional LSTM-CNNs[J/OL]. Transactions of the Association for Computational Linguistics, 2016, 4: 357-370. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00104/1567392/tacl_a_00104.pdf. https://doi.org/10.1162/tacl%5C

_a%5C_00104. DOI: 10.1162/tacl_a_00104.