NER 模型测试报告

作者：纽约的自行车，小猪佩奇

日期：2020 年 10 月 20 日

1 前言

使用基于词汇增强的 LexiconAugmentedNER [1] 和适用于远程监督的 NegSamplingNER [2] 两种命名实体识别（named entity recognition, NER）算法在 7 个数据集上测试，与 baseline 模型 CRF [1]作比较，对比所选模型的性能。

2 算法简介

2.1 LexiconAugmentedNER

2.1.1 背景

在中文 NER 领域，输入有字级别和词级别两种。由于中文分词的歧义性，导致词级别的模型劣于字级别的模型。但是，基于字的 NER 没有利用词信息，而词信息对于实体边界通常起着重要作用。自从 Lattice-LSTM [3] 发表后，将词信息融入到字符表征中，成为中文 NER 的主流。但是 Lattice-LSTM 结构复杂，且无法并行计算。本文提出一种简化方法，在输入中即加入词信息，而不是在模型中。因此本文的方法适用于任何模型。

2.2 模型

模型创新点在于输入，作者称之为 SoftLexicon。输入仍是以字符为主，将句子中包含该字的所有词信息融入到该字的字符表征中。以句子 ‘‘李明住在中山西路学’’ 中的 ‘‘山’’ 字为例，如图 1。这句话中含有多个包含 ‘‘山’’ 的词，例如 ‘‘山西’’、‘‘中山西路’’、‘‘中山’’、‘‘山’’ 等。‘‘山’’ 字在这些词中出现的位置不同，有开头位置的，有中间位置的，有末尾位置的，还有单个词的。将这些词按照 ‘‘山’’ 字出现位置分类，分为 B、M、E、S 类。将 ‘‘山西’’ 分到 B 类，‘‘中山’’ 分到 E 类，若某类不存在词，则补充 “None’’。

有了每个字对应的词后，接下来就是考虑如何将词向量加入到字向量中。从模型结构图中可以很清楚的知道论文对词向量的处理。模型如图 2所示，模型主体采用 LSTM+CRF 结构，重点在于输入数据的处理。

[1]aawantNER

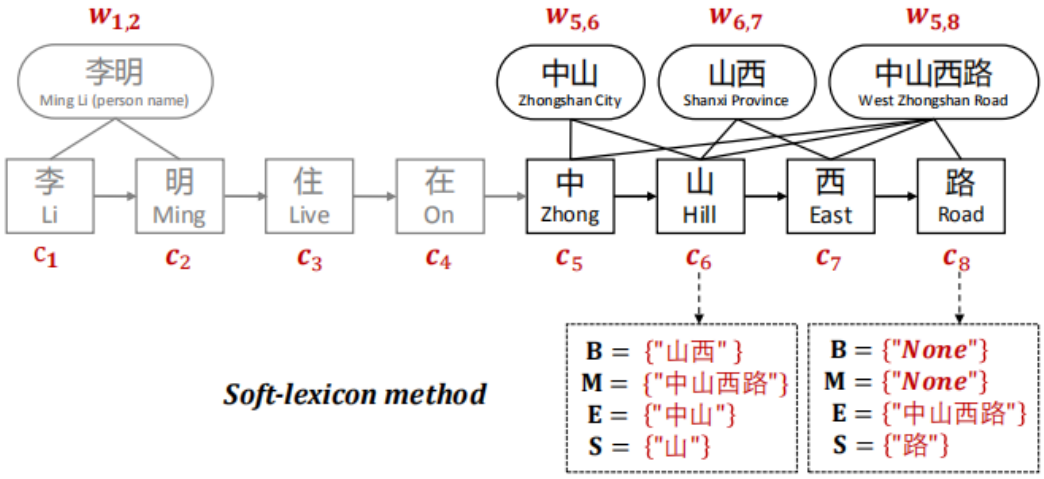

图 1: SoftLexicon 方法

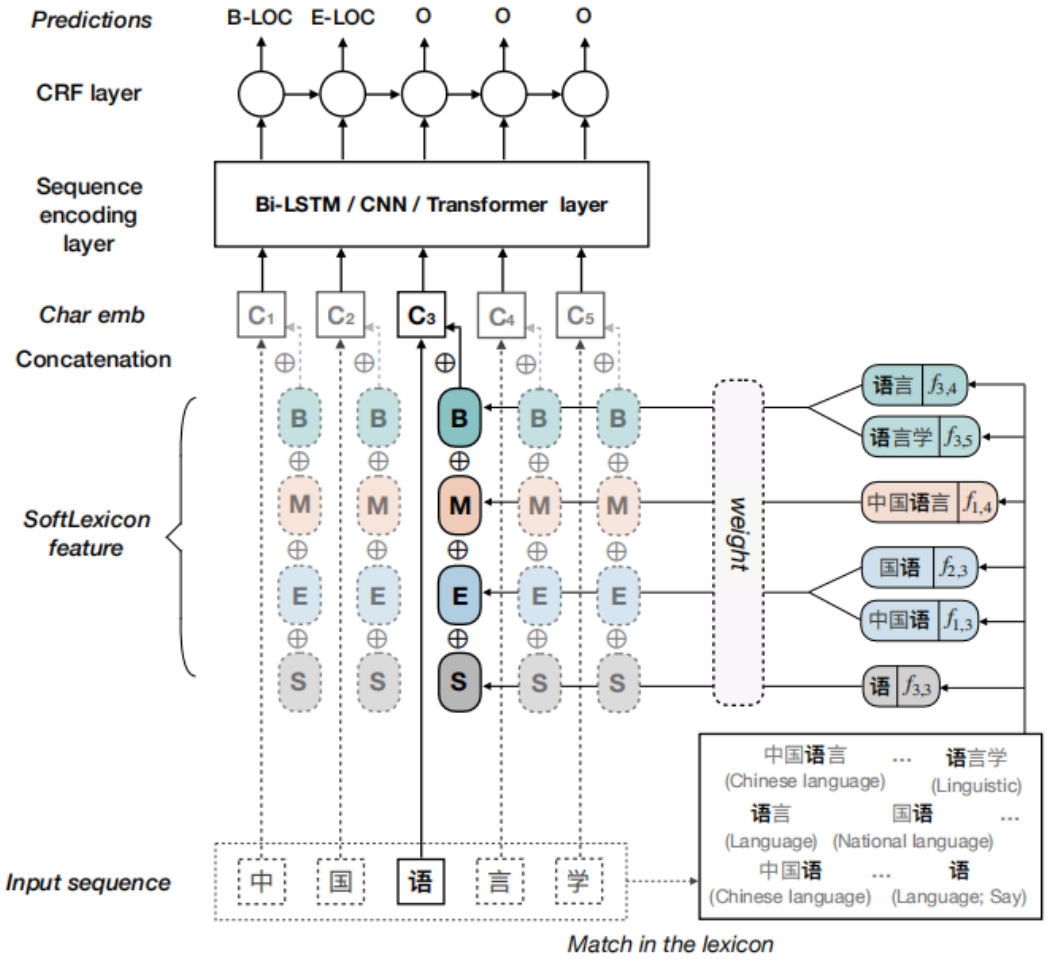

图 2: LexiconAugmentedNER 模型结构

首先将每个类别的词向量加权，得到一个类别词向量，如式 1：

$$v^s(S) = \frac{4}{Z} \sum_{w \in S} z(w) e^w(w) \tag{1}$$

S 表示词集合，w 表示词，$z(w)$ 表示词权重，由词在语料库中的频数代替。$e^w(w)$ 表示词向量。Z 表示归一化，$z = \sum_{w \in B \cup M \cup E \cup S} z(w)$。

将四个类别的词向量拼接，得到一个固定维度的词向量 $e^s(B,M,E,S)$。然后将该词向量与字向量 x^c 拼接，得到最终的输入向量 x^c。

$$e^s(B,M,E,S) = [v^s(B); v^s(M); v^s(E); v^s(S)] \tag{2}$$

$$x^c \leftarrow [x^c; e^s(B,M,E,S)] \tag{3}$$

2.3 NegSamplingNER

2.3.1 背景

远程监督标注数据会存在部分实体未标注问题。使用存在大量未标注实体的数据集训练模型会存在两个问题：一是导致训练样本减少；二是未标注实体会误导模型。第一个问题可使用预训练模型缓解，本文提出使用负样本和 span-level 分类方法解决第二个问题。

2.3.2 模型

句子中存在部分未标注实体，为了使其不影响模型训练，将这些未标注实体删除即可，但是我们无法预知句子中那些字符是未标注实体。一个简单的解决方法是随机删除标注为 “O’’ 的字符，留下的字符作为样本训练模型，但这样破坏了句子结构，训练效果不好。既然无法对输入作改变，那么就只能改变输出了。随机 mask 部分位置的输出，目的是删除未标注实体。虽然 mask 输出没有破坏输入时的句子结构，但这种方法仍难以准确找到未标注实体，mask 效果并不好。

本文换了一种思路，放弃常用的 CRF 解码方式，使用 span-level 分类方法解码。使用 CRF 解码，只能 mask 字符，但是使用 span 分类方法解码，可以 mask 整个 span。对句子的每个 span 分类，判断其是否是实体，并进一步判断实体类型。输入句子 $x = [x_1, x_2, ..., x_n]$，标签为 $y = [y_1, y_2, ..., y_m]$，其中 $y_k = (i_k, j_k, l_k)$，i_k 表示实体在句子的起始位置，j_k 表示实体在句子的结束位置，l_k 表示实体类型。标签集合 y 中全为正样本，负样本由对句子采样得到。首先将句子全排列切分，长度为 n 的句子可切成 $(n+1)n/2$ 份，随机取 [0.35n] 个 span 作为负样本。由于切分的 span 足够多，采样的 span 很少，所以负样本中很大概率不包括未标注实体，因此很好的将未标注实体删除了。由于标签中不存在未标注实体，因此不影响模型训练。这种 span 分类加上负采样的方法很好的解决了远程监督中未标注实体问题。

模型由编码器（BERT 或者 LSTM）和解码器（MLP）组成，如图 3。

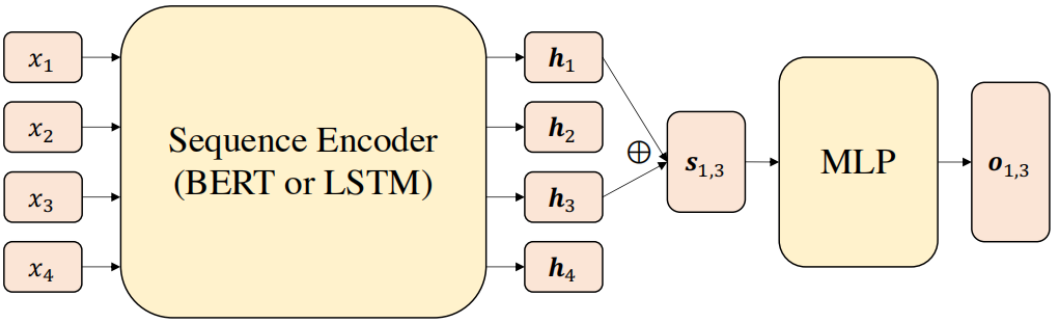

图 3: NegSamplingNER 模型结构

编码器的输入输出和常规方法相同。采样过程就是随机挑选一个起始位置和一个结束位置，作为一个负样本，将起始位置和结束位置的向量拼接得到负样本表征。正样本的表征也是将起始位置和结束位置的向量拼接得到。拼接方式如式 4：

$$s_{i,j} = h_i \oplus h_j \oplus (h_i - h_j) \oplus (h_i \bullet h_j) \tag{4}$$

将正负样本向量表征输入 MLP 分类，得到每个样本的类型，负样本的类型为 “O’’。MLP 计算过程如式 5：

$$o_{i,j} = Softmax\left(Utanh\left(Vs_{i,j}\right)\right) \tag{5}$$

模型损失函数如下：

$$L = \sum_{(i,j,l)\in y} -log(o_{i,j}[l]) + \sum_{i',j',l'\in \hat{y}} -log(o_{i',j'}[l']) \tag{6}$$

第一项是正样本集合 y 的损失，第二项是负样本集合 $\hat{y}$ 的损失。

3 实验

3.1 数据集介绍

测试数据集共 7 个，均为中文数据。其中 ds_30W 是远程监督生成的数据，其余六个为开源数据。数据集统计结果如表 1所示。

Datasets	Train	Dev	Test	Class
1998 人民日报	13312	1902	3804	6
MSRA	32453	4730	9179	6
literature	35815	5117	10234	9
weibo	1323	189	378	7
cluener	8463	1234	2394	10
BosonBLP	1400	205	395	6
ds_30W	255766	36538	73076	6

表 1: 数据集统计情况。Train、Dev、Test 栏表示句子数量，Class 栏表示实体类型数量。

1998 人民日报[2]和 MSRA[3]是论文中常用的公开数据集，数据量适中。weibo[4]和 BosonNLP[5]的数据量少，适用于测试少样本模型。literature[6]用于实体识别和关系抽取任务。cluener[7]是 CLUE[8]公布的细粒度 NER 数据集。远程监督生成的数据存在漏标和错标现象，本次从 ds_30W 数据中随

[2]https://github.com/InsaneLife/ChineseNLPCorpus/tree/master/NER/renMinRiBao

[3]https://github.com/InsaneLife/ChineseNLPCorpus/tree/master/NER/MSRA

[4]https://github.com/hltcoe/golden-horse

[5]http://static.bosonnlp.com/dev/resource

[6]https://github.com/lancopku/Chinese-Literature-NER-RE-Dataset

[7]https://github.com/corpus-dataset/cluener2020

[8]https://github.com/CLUEbenchmark/CLUE

机抽取 200 条样本，人工检查漏标和错标情况，检查统计如表 2所示。漏标率 6.28%，错标率 5.6%，总体看来 ds_30W 的数据标注良好。

entity type	miss	error	total
person	6	7	108
number	6	13	118
LOC	5	5	188
ORG	7	1	38
SI	8	3	85
time	12	10	120
total	44	39	657

表 2: ds_30W 数据集统计情况。miss 表示漏标数量，error 表示错标数量，total 表示实体总数。

3.2 实验设置

本次测试的 baseline 模型为 aawantNER，使用公司现有代码测试。LexiconAugmentedNER[9]和 NegSamplingNER[10]的代码均来源于论文作者公开的源代码。LexiconAugmentedNER 使用 BiLSTM+CRF 结构，NegSamplingNER 使用 Bert+MLP 结构。LexiconAugmentedNER 在 7 个数据集上测试，由于数据集大小不一，训练模型时针对不同数据集设置不同超参数。NegSamplingNER 在远程监督生成的数据集上测试，该数据划分的句子长度太长，导致内存溢出。本次测试将句子切分，限制长度不超过 100，并且删除不包含实体的句子。由于 ds_30W 数据集太大，在服务器 GPU 环境下跑完一个 epoch 大约耗时 4 小时。为了方便调参，在随后的训练中随机选用 10% 的训练数据，因此测试结果是选用 10% 训练数据得到的结果。

3.3 测试结果

测试结果如表 3所示。在 7 个数据集上 LexiconAugmentedNER 模型均优于 aawantNER 模型。LexiconAugmentedNER 的论文中作者同样也在 MSRA 和 weibo 数据上测试，论文中的测试结果和本文的测试结果相差较大。其中本文在 MSRA 上的测试结果优于论文中的结果 2 个百分点，论文中的结果为 0.941。但是在 weibo 上的测试结果远低于论文中的结果，论文中的 F1 值为 0.598，两者相差约 13 个百分点。在这两个数据集上，本文测试时使用的超参数和论文作者给出的一致。测试结果相差较大的原因可能是数据划分不同，论文中的训练集、验证集和测试集使用官方划分方式，而本文对数据重新划分。在远程监督数据集上 NegSamplingNER 模型同样优于 aawantNER 模型，但是提升效果有限，这是因为 ds_30W 数据标注良好，未标注率仅有 6.28%，而 NegSamplingNER 模型在未标注率超过 40% 的数据上才能取得明显优势。LexiconAugmentedNER 模型在 ds_30W 数据上效果好于 NegSamplingNER 的原因可能是前者在全部的 ds_30W 数据上训练，而后者只使用了 10% 训练数据。同样使用 10% 数据训练的情况下，后者的 F1 值高于前

[9]https://github.com/ricklitong/LexiconAugmentedNER

[10]https://github.com/tencent-ailab/NegSampling-NER

者 1.3 个百分点。LexiconAugmentedNER 模型使用 BiLSTM 作为编码器，在大量数据上训练速度较快，便于调参，且内存占用少，因此使用全部 ds_30W 数据训练。而 NegSamplingNER 模型使用 bert 作为编码器，训练速度慢，只能被迫使用 10% 的 ds_30W 数据数据训练。

Datasets	aawantNER			LexiconAugmentedNER			NegSamplingNER		
	P	R	F1	P	R	F1	P	R	F1
1998 人民日报	0.947	0.916	0.931	0.951	0.959	**0.955**	-	-	-
MSRA	0.96	0.933	0.946	0.963	0.963	**0.963**	-	-	-
literature	0.856	0.706	0.823	0.846	0.851	**0.848**	-	-	-
weibo	0.441	0.154	0.357	0.585	0.381	**0.462**	-	-	-
cluener	0.743	0.652	0.693	0.754	0.742	**0.748**	-	-	-
BosonNLP	0.726	0.584	0.644	0.687	0.658	**0.672**	-	-	-
ds_30W	0.917	0.9	0.901	0.933	0.935	**0.934**	0.914	0.936	0.925

表 3: 测试结果对比。LexiconAugmentedNER 在 7 个开源数据集上测试，NegSamplingNER 在 ds_30W 数据集上测试。最好结果用黑色加粗字体标出。

参考文献

[1] Ruotian, Ma and Minlong, Peng and Qi, Zhang and Zhongyu, Wei and Xuanjing, Huang. Simplify the usage of lexicon in chinese ner[J]. ACL, 2020.

[2] Yangming, Li and Lemao, Liu and Shuming, Shi. Empirical analysis of unlabeled entity problem in named entity recognition[J]. ICLR, 2021.

[3] Yue, Zhang and Jie, Yang. Chinese NER Using Lattice LSTM[J]. ACL, 2018.