NLP Course Report
Chinese Word Segmentation and POS Tagging

Peking University
1801210840 Hui-Qiang Jiang
1701210963 Da-Wei Lee

2019 年 6 月 1 日

目录

1 实验目的

这次主要分为三大任务，中文分词 (CWS)、词性标注 (POS)、医学命名实体识别 (NER)。目的是为了学习在自然语言处理领域之中，对于中文文本资料的基本处理。这三类问题都属于序列标注问题，而序列标注问题是自然语言处理领域 (NLP) 的一个典型任务。其经常作为复杂自然语言处理问题的上游任务，常用于机器翻译 (Machine Translation)、问答 (Question Answer), 对话系统 (ChatBot), 情感分析 (Sentiment Analysis), 词义消歧 (Semantic Disambiguation) 等等。正确的分词、词性标准、命名实体识别对下游的模型训练效果起到较好的提升作用。

原始所提供的数据中，已经包含了答案，也就是说，对于分词来说已经分好词；而对于词性标注来说，是已经分好词且标注过词性的；对于医学命名实体识别来说，是分词并标注上命名实体信息的。

以原始数据来说，并无法直接进行训练，需要进行定的转换。一是需要转换成模型可训练的数据，二是需要转换回纯原始数据。前者以分词举例，由于我们之后使用的模型，是相当于对于每个字做标签分类，来区分是起始词 B、中间词 M、结束词 E 还是单词 S。而对于后者来说，由于以实际应用来说，我们肯定是丢入一般正常的句子，所以为了模拟必须先转换回去。

我们并未对原始数据做太多的调动，即便有很多不合理之处，但由于测试集是由训练集取出，其实相当于我们在一个「很像中文」的一个环境中，只是在这个世界中的中文，和我们现实所使用的中文有些许不同。例如我们保留"$$_" 之类的特殊符号或是被它所切开的英文，经过一定处理后与一般数据一同输入模型当中进行预测。我们倾向预测所提供数据中的答案，即便发现有很多完全一样的样例有著超过 5 种可能词性的这种情况，我们仍为这种问题提出相对应的解决方案。故并未使用任何外部训练数据来「污染」自己的结果。

2 实验原理

2.1 Word Segmentation

2.1.1 CRF

条件随机场 [?] 是一种无向概率图模型，常用在序列标注问题中。

条件随机场是在给定的随机变量 X （观测序列 $o_1,\cdots,o_i$）条件下，随机变量 Y （隐状态序列 $i_1,\cdots,i_i$ 的马尔科夫随机场。

条件随机场是一种无向概率图模型，在迭代过程中，通过寻找最大团块 (子图中任意两点间都有边相邻) 进行参数消除。无向图的联合概率值等于所有最大团块的势函数乘积除以归一化因子。这就意味着条件随机场是一个全局迭代的过程，其结果要等到最后才能知道，所造成的迭代效率也就偏低。

在实验过程中, 除了使用条件随机场加上一些词嵌入方法之外，还用了分词任务中常见的 BiLSTM 加上条件随机场 CRF 进行处理。

我们将序列标注问题转化为概率图模型，这就导致了概率图的输入值对结果起到了至关重要的影响。

条件随机场的条件概率和归一化因子如下:

$P(I|O) = \frac{1}{Z(O)} exp(\sum_{i,k} \lambda_k t_k(O_{i-1}, O_i, I, i) + \sum_{i,l} \mu_l I_l(O_i, I, i))$

$Z(O) = \sum_I exp(\sum_{i,k} \lambda_k t_k(O_{i-1}, O_i, I, i) + \sum_{i,l} \mu_l I_l(O_i, I, i))$

条件概率 $P(y|x)$ 表示了在给定的一条观测序列 $O = (o_1, \cdots, o_i)$ 条件下，CRF 求出隐状态序列 $I = (i_1, \cdots, i_i)$ 的概率值。

展开上式，得到:

$P(I|O) = \frac{1}{Z(O)} e^{\Sigma_i^T \Sigma_k^M \lambda_k f_k(O, I_{i-1}, I_i, i)} = \frac{1}{Z(O)} e^{[\Sigma_i^T \Sigma_j^J \lambda_j t_j(O, I_{i-1}, I_i, i) + \Sigma_i^T \Sigma_l^L \mu_l s_l(O, I_i, i)]}$

其中:

t_j 为第 i 状态的转移到 j 状态特征，对应权重 λ_j

s_i 为第 i 个的状态特征，对应权重 $_i$, 其用来表示是否满足 label 条件，用来打分评价该隐状态的权值。

然后对 Score 进行求和归一化得到其概率值。

2.1.2 BiLSTM CRF

如果我们用单模型，既在给定特征的基础上做一个最大概率分布计算，那么我们相当于假定我们的特征能很好的表示当前上下文信息，能表示需要分词的 Context。

但实际上并不是这样的，首先 One-Hot 可以直观的感觉，它只是学习到某些特定的词组合，并没有学习到词义层级的信息。

即使是用 TF-IDF，甚至采用预训练的词嵌入作为输入，也不能较好的表示上下文信息，容易丢失一些局部的特征，学习到的效果较为死板。而使用预训练获得的词向量表示容易放大原本的噪声，因为条件随机场是一种全局化最优隐层状态概率值，原有的噪音会被放大导致不太好的效果。所以一般单条件随机场模型在分词问题上效果一般。

通过双向 LSTM 获得上下文信息，从而使得概率图 CRF 的输入噪音较小。继而获得更好的学习效果，

而对比单纯使用 BiLSTM 的模型，CRF 层在 BiLSTM 表示 Context 信息的基础上做了一个选择最大概率分类的工作。通过无向概率图模型的，选择概率最大的最优 label 路径，可以明提升模型效果。

2.2 Part-of-speech Tagging

目前我们所使用的词性标注，基本上都是以「假设数据集正确」来做的，为了更好的符合数据集的结果，我们将数据集进行分析，并且建立一个知识图谱来做词与词性之间的映射

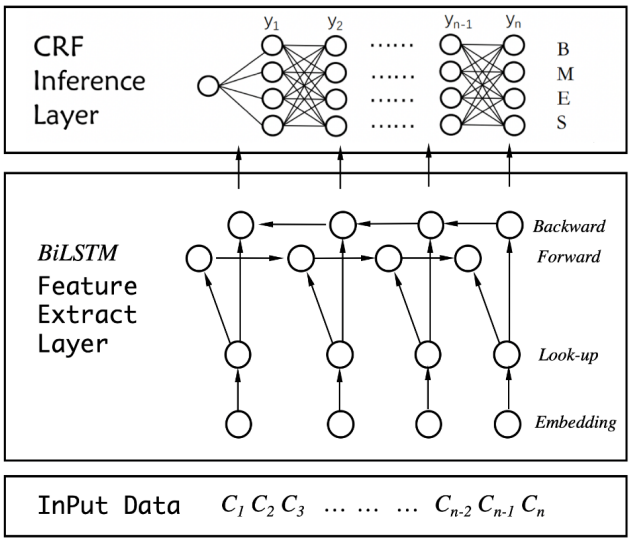

图 1: The overview of BiLSTM-CRF model structure

关系。

而这些映射关系由于是纯由训练数据做出来的，故在进行测试集或开发集的预测时，必然会出现没有见过的词与词性的对应。这时利用一些知识我们可以将一些容易判别的词性加以标注，并倾向于预测没出现过且无法规则判断的词为名词，毕竟专有名词经过我们统计是最有可能没有出现过的。

我们发现一些棘手的例子，在数据集中，有很多数据会有多种词性标注，故我们倾向于预测数据集中出现最多的那种词性。因为若用模型来预测很容易被误导而训练出很差的效果，所以牺牲一部分「错误的正确性」算是相对合理的。最原始模型基于所有见过的词进行出现频率上的统计，这在撇除分词效果（直接将 gold 分词结果读入当做模型输入），就可以直接达到 95% 左右的准确度。由于因为输入之分词与评价的集合完全相同，这相当于计算 precision 和 recall 的分母相同，故 precision、recall、f1-score 会呈现完全相同的结果。之后基于此模型在加上规则，这在数据很脏的时候特别有效，毕竟很多在词性标注的标准中，我们课堂使用的标准特别的「特殊」，所以若尝试使用外部标准，反而会使效果下降，而若使用机器学习，则会「教坏」它。随著规则的增多，提高了几个百分点，毕竟原先的效果已经很好了。

3 实验内容

最一开始当然是从观察数据开始，并且先将数据处理成可以训练的模式。

我们主要处理三阶段的可训练数据。第一个就是处理成分词用的训练数据，将数据转换为针对每一个字的四分类问题，在段落 1也提过，我们所预测的集合为 {B, M, E, S}。字的

部份我们就先不做处理，保留之后转成各种不同的 embedding 表示。第二阶段数据则是给词性标注的数据，毕竟我们是两个任务并行进行的，但词性标注必须依赖先有分词的结果，故我们将词性标注的 gold 数据处理后形成模拟的分词结果（不直接使用分词的 gold 数据处理，主要是因为分词的数据连 <sub> <sup> 之类的 tag 都被搞坏了，根本没办法用）。第三阶段数据是在最终的文本数据提供之前，先利用现有数据还原成原始文章的形式，这是用于测试从无到有的模型输出效果，之后待最后的目标文档提供后只要直接替换这边就可以快速的套用并输出结果。

再来，则是模型的搭建与训练和测试，这部份在前面的段落 2中有详细的叙述。

最后是模型的评价，原先使用了所提供的评价脚本，却发现里面有很多的坑，特别是因为数据中的"//w" 和"</sup>" 这类数据会被误判成词性，还有空白行也会一同被纳入评测，这导致最后评分出来的表现有很大的误差，故为了保险起见还是针对了这部份进行我们自己版本的改正。

Detail of Word Segmentation

在分词过程中，测试了 CRF 单模型，Embed 使用了 One-Hot, TF-IDF, FastText.

因为在测试过程中发现用预训练词向量效果不好，就没有测试基于 Bert 或者 ELMo 的实验。

在跑单 CRF 过程中，因为输入的是一个句子数 * 句子长度 * 特征数的一个三维矩阵。当利用最大句子长度对齐时，在特征数较多的情况下（即 One-Hot）会导致 Memory Error。例如训练集 6106 句，最大长度 1060，One-Hot 2868 维度。用 int16 存储，需要 68GB 内存，实际测试中用了 int32 内存直接打到 120GB+。故在处理这个情况下，使用了 reshape.

另外，在分词这个任务，除了最基本的 MASK 操作之外，实际上带有一个隐藏的条件，即句尾一定分词。虽然为了学习到上下文信息，不易直接将句尾字符舍去，但在评测过程中，需要手动更改。

因为使用单模型，效果较为一般，为了获得更多上下文信息，在 encoder 层按常规接了一个 BiLSTM。

实验下来，效果比较稳定。像单模型 CRF，训练集和开发集偏差有将近 25%, 而 BiLSTM-CRF 模型偏差在 5% 以内。

4 实验结果

Word Segmentation

在训练集、开发集与测试集数据中测试结果。（表 1）

Model	Embed	Train Set			Dev Set			Test Set			Epoch
		P	R	F1	P	R	F1	P	R	F1	
CRF	One-Hot	98.86	99.19	99.03	76.36	75.06	75.70	78.54	75.68	77.08	130
CRF	TF-IDF	33.87	29.69	31.64	31.14	37.42	33.99	31.69	37.42	34.32	90
CRF	FastText	30.24	33.21	31.65	34.59	37.29	35.89	36.13	37.99	37.03	50
BiLSTM-CRF	One-Hot	94.46	95.19	94.83	90.02	91.08	90.55	91.01	91.32	91.16	8-75

表 1: Best Evaluation in DevSet about Word segmentation (in %)

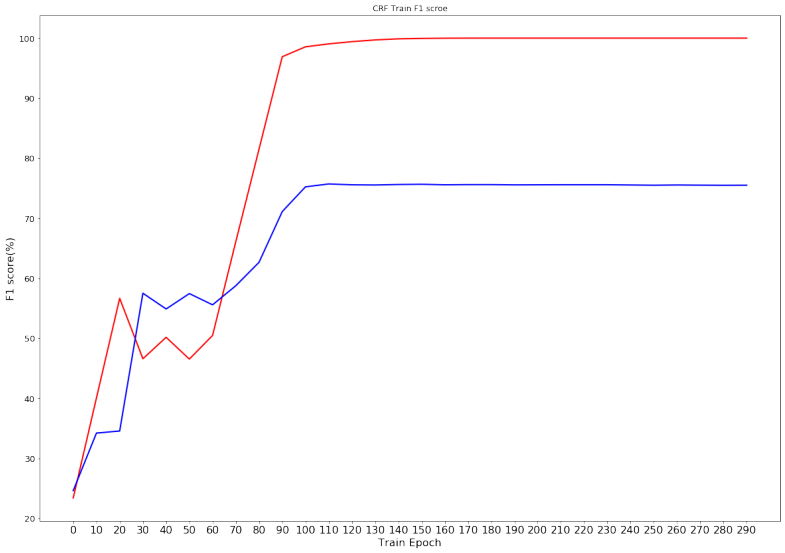

图 2: Comparison between train epoch and the f1 score in TrainSet & TestSet By CRF

Part-of-speech Tagging

在这个测试之中，排除了分词错误的可能性，利用从 testset1 中的 test_pos1.txt 中提取出分词结果（不使用 test_cws1.txt 是因为里面的 tag 全是乱的完全无法使用）。（表 2）

由于后面为使整体一致，补上于训练集和开发集上的评价结果，但中间状态由于并不是一个定版，故有些测试数据无法得到。

关于这些数据可以观察到几点，首先 precision、recall、f1-score 值在此项测试中会相等的原因在词性标注的实验原理中 2.2已经叙述过了。再来是训练集并不因规则而有所提升，这也很容易想，因为由于我们是从训练集建立的关联关系，故训练集并不会存在 OOV 的问题，故提升上只会在开发集和测试集中体现。

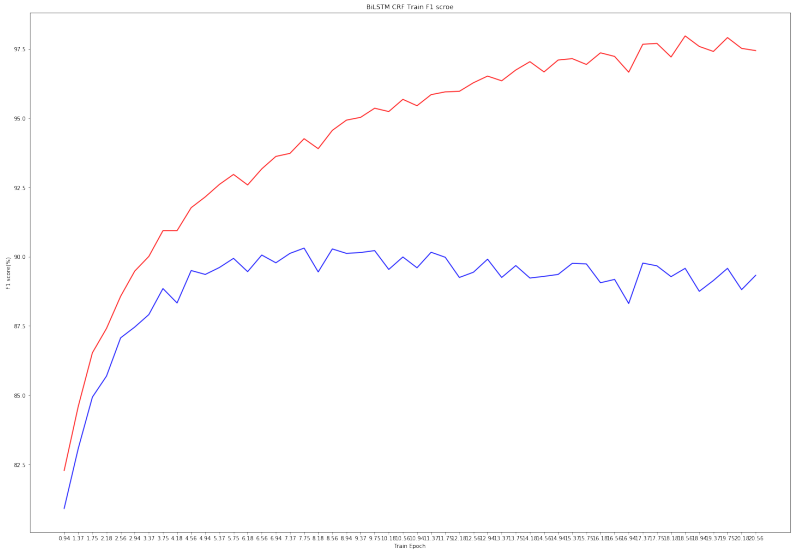

图 3: Comparison between train epoch and the f1 score in TrainSet & TestSet By BiLSTM-CRF

Method	F1-Score on Train Set	F1-Score on Dev Set	F1-Score on Test Set
Pure Mapping	97.98	95.57	95.39
Add Rules	-	-	95.83
Even More Rules	97.98	96.06	95.86

表 2: Part-of-speech tagging on Gold CWS (in %) (precision = recall = f1-score)

5 误差分析

一开始花了挺多时间在检查 F1 score 一致性问题上，一致性主要要注意 MASK，还有句尾一定分词这个隐藏的条件，另外给的评测脚本和自己写的 F1 score 还是有些偏差，但无伤大雅。

到最后开始调模型的时候发现单模型 CRF TF-IDF 效果竟然比 One-Hot 还差。

TF-IDF 的思路是先做一个窗口获得附近 centre word 附近 n-gram 词汇信息，做相应的词频转换。然后丢到 sklearn 的 pipeline 中做 TF-IDF 的转换。

因为 TF-IDF 测试集需要和训练集对齐，在这个处理上把未出现的词都塞到最后一个 word 的 dict 中，赋予所有词频为 0.

所以一开始怀疑是不是训练集和开发集对齐发生错误，毕竟 33% 左右的正确率基本上意味着 random。不过因为时间关系，后面就没去 check 了。

同样替换 embedding 为常见的预训练词向量，比如说 FastText，GloVe，ELMo，BERT。

本来想把实验做做完，但做 FastText 下来效果很差，就直接改 BiLSTM-CRF 了。

总的来说，在实验过程中 TF-IDF 训练效果明显低于正常值，且其训练集基本上达到最优效果，很有可能是测试集合训练集特征为对应上。

6 思考题

1. NER **部分存在同一个词被多个** NER **嵌套的情况，但是序列标注的** NER **模型对于一个词往往只能有一个** NER **标注，如何解决该问题？** e.g:[[肺动脉]bod 狭窄]sym

通常有这种多重嵌套的状况，一般来说都会是不同的属性，因此我们的命名实体模型，可以做一些特化，特别针对某类型的医学命名实体进行识别。在识别时，分别使用多个特化过的模型，就可以将所有 NER 的标签都找出来并标上。

2. **分词任务与** pos/ner **任务其实是紧密相关的，如果在分词阶段出现错误，该部分误差就会传递下去，如何解决该问题?**

这基本上是无解，我们无法阻止分词的误差传递到词性标注，但我们可以让词性标注不那么依赖分词，这也是为什么我们使用基于知识图谱的词性标注模型，这就可以避免因为分词效果不彰，导致词性标注的训练效果也很糟糕的这样的结果。

3. **训练数据量十分有限，在不使用人工标注的情况下，如何扩充数据量，并进一步优化模型效果?**

以分词和词性标注来说，可以将所有分词与词性标注结果做成一个辞典，并且构造一些句法逻辑，在利用这些句法逻辑来生成句子，并利用这些生成的句子来训练。

但以医学命名实体识别就相对困难，毕竟我们不可能自己自创医学命名实体，但若上网爬数据，相当于做了人工标注便不符合规范。可能的作法大概是例如将一些句子中相同类型的位置，比如替换器官名称等等，来达到扩充数据量的效果。

7 实验分工及感想

一开始也没有太特别的分工，主要先阅读一些相关的论文以及逐行研究目前流行之分词工具的底层源代码来收集灵感，接著一起决定要使用什么样的模型、使用哪些 library，并且拟定大方向架构。后来因为我们这学期修了太多扎实的课，包含数门信科的自然语言处理课程，导致时间被压缩的很紧迫，故决定并行动工，由姜慧强实现分词、李大为来做预处理与实现词性标注。相关工作细节皆公开透明开源于 Github[1] 上。

因为实验数据中噪音比较大，另外在校对 F1score 过程中浪费了一些时间。由于数据基本上已经脏到不能当做一般数据来使用，也不可能找其他数据集来训练，因为所基于的标准

[1] https://github.com/pku-nlp-forfun/CWS_POS_NER

完全不同，也没有很多特有的东西，比如”$$_” 之类的东西。所以一旦想要追求模型表现，那就不可能做出能通用于其他语料的版本。那我们最后决定主要倾向针对本次任务进行优化，并且不使用任何外部数据。但由于数据量对于模型训练还是偏少，故过度的追求模型表现只是纯粹的自我打击，也让我们学会了，如果是任务结果导向，能跳脱机器学习的迷思也许才是其中的关键，仔细去看结巴分词的实现，里面也有很多正则表达式，毕竟并不是所有任务、数据都适合机器学习。

在实验过程中，加强了对 CRF、概率图模型的认识，提高了动手时间能力，强化了对 Trick 的使用，对个人实践能力有较高的帮助。

另外在实践过程中遇到一些挫折，当然这些挫折原因多种多样，我们还是想尽可能用更科学的方式，更严谨的态度来完成这次实验。