Report for NLP-Beginner task 4

基于LSTM+CRF的序列标注

叶栩冰

---

1 Task

用LSTM+CRF来训练序列标注模型：以Named Entity Recognition为例。

1. 参考
   1. 《神经网络与深度学习》 第6、11章
   2. https://arxiv.org/pdf/1603.01354.pdf
   3. https://arxiv.org/pdf/1603.01360.pdf
2. 数据集：CONLL 2003，https://www.clips.uantwerpen.be/conll2003/ner/
3. 实现要求：Pytorch
4. 知识点：
   1. 评价指标：precision、recall、F1
   2. 无向图模型、CRF
5. 时间：两周

2 Environment & Data

2.1 Environment

> Python ~= 3.6
>
> IDE : Jetbrains Pycharm
>
> torch ~= 1.10.0

2.2 Data

The CoNLL-2003 shared task data files contain four columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence. The first item on each line is a word, the second a part-of-speech (POS) tag, the third a syntactic chunk tag and the fourth the named entity tag. The chunk tags and the named entity tags have the format I-TYPE which means that the word is inside a phrase of type TYPE. Only if two phrases of the same type immediately follow each other, the first word of the second phrase will have tag B-TYPE to show that it starts a new phrase.

The data consists of three files per language: one training file and two test files testa and testb. The first test file will be used in the development phase for finding good parameters for the learning system. The second test file will be used for the final evaluation. There are data files available for English and German. The German files contain an extra column (the second) which holds the lemma of each word.

Example:

> **原数据**:
> EU NNP B-NP B-ORG
> rejects VBZ B-VP O
> German JJ B-NP B-MISC
> call NN I-NP O
> to TO B-VP O
> boycott VB I-VP O
> British JJ B-NP B-MISC
> lamb NN I-NP O
> . . O O
> **Input**: EU rejects German call to boycott British lamb.
> **Output**: B-ORG O B-MISC O O O B-MISC O O

> **原数据**:
> Peter NNP B-NP B-PER
> Blackburn NNP I-NP I-PER
> **Input**: Peter Blackburn
> **Output**: B-PER I-PER

Explain :

每行第一项是单词，第二项是POS标签，第三项是语法块标签，第四项是命名实体标签，也就是我们本次的任务。具体命名实体的每一个标签代表的意思，可以参考原数据集介绍。

数据集共有三个文件：train.txt, test.txt 和 dev.txt。由于文件数据的组织方式比较松散，因此需要预处理，并且需要去掉文件中所有的：-DOCSTART- -X- -X- O（代表某一个文档开始）

```
if elements[0] == '-DOCSTART-':
    continue
```

详见代码Utils.py/data_process.

3 Method & Model

3.1 Embedding

同task2，此外，本次作业除了给定的各种序列类别之外，还要另外多加3个类别，分别是: < pad >,< start >,< end >，分别代表padding（即补位，使句子达到同一个长度），句子开头和句子结尾，总共C类标签。

3.2 LSTM+CRF

3.2.1 序列标注

序列标注（Sequence Tagging）是NLP中最基础的任务，应用十分广泛，如分词、词性标注（POS tagging）、命名实体识别（Named Entity Recognition，NER）、关键词抽取、语义角色标注（Semantic Role Labeling）、槽位抽取（Slot Filling）等实质上都属于序列标注的范畴。

序列标注问题可以认为是分类问题的一个推广，或者是更复杂的结构预测（structure prediction）问题的简单形式。序列标注问题的输入是一个观测序列，输出是一个标记序列或状态序列。问题的目标在于学习一个模型，使它能够对观测序列给出标记序列作为预测。

首先给定一个训练集，

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_N)\}$$

这里，$x_i = (x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(n)})^T, i = 1, 2, \cdots, N$，是输入观测序列，$y_i = (y_i^{(1)}, y_i^{(2)}, \cdots, y_i^{(n)})^T$ 是相应的输出标记序列，n 是序列的长度，对不同样本可以有不同的值。学习系统基于训练集构建一个模型，表示为条件概率分布：

$$P(Y^{(1)}, Y^{(2)}, \cdots, Y^{(n)} | X^{(1)}, X^{(2)}, \cdots, X^{(n)})$$

标注系统按照学习得到的条件概率分布模型，对新的输入观测序列找到相应的输出标记序列。具体地，对一个观测序列 $x_{N+1} = (x_{N+1}^{(1)}, x_{N+1}^{(2)}, \cdots, x_{N+1}^{(n)})^T$ 找到使条件概率 $P((y_{N+1}^{(1)}, y_{N+1}^{(2)}, \cdots, y_{N+1}^{(n)})^T | (x_{N+1}^{(1)}, x_{N+1}^{(2)}, \cdots, x_{N+1}^{(n)})^T)$ 最大的标记序列 $y_{N+1} = (y_{N+1}^{(1)}, y_{N+1}^{(2)}, \cdots, y_{N+1}^{(n)})^T$。

3.2.2 Neural Architectures for Named Entity Recognition

文章中，对于NER，提出了一种的神经结构，它不依赖术语资源或者特征，而仅仅依赖小规模的监督训练数据与未注释的语料库。由于一个名称通常由多个词条组成，因此对任意一个标记的词性标注决策进行联合推理是很重要的。在这里我们对比了两种模型，一种是具有序列条件随机层的双向LSTM（LSTM-CRF）。

3.2.2.1 LSTM

LSTM在此就不过多进行阐述了，在以前的作业中都有所提及，如下图。

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{c}_t &= (1 - \mathbf{i}_t) \odot \mathbf{c}_{t-1} + \\
&\quad \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}
$$

本方法中，对于一个给定包含n个词的句子$\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\right)$，每一个都被表示为为一个d维向量，LSTM计算了句子下文每t个词的一个输出值$\overrightarrow{\mathbf{h}_t}$。当然，也有特指上文的输出值$\overleftarrow{\mathbf{h}_t}$，这可以利用LSTM读句子的反向序列实现。我们将前者称为正向LSTM，后者称为反向LSTM。他们是两个截然不同的具有不同参数的网络。正向LSTM与反向LSTM统称为双向LSTM（Gravesand Schmidhuber, 2005）。使用该模型时的词表示是通过该单词的上下文得到的，即$\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$。这些表征有效包括了上下文中的单词表征，这对于许多词性标注应用都很有用。此部分得到的h作为CRF的输入。

3.2.2.2 CRF Tagging Models

一个相当简单但是极其有效的词性标记系统是使用$\mathbf{h}_t$作为特征为每一个输出y_t做出独立的词性标记决策（Ling et al.，2015b）。尽管这个模型成功解决了类似 POS tagging 这样简单的问题，但是当需要输出标签之间存在有很强的依赖关系时，它的独立分类决策仍会受到限制。而NER就是这样的一个任务，因为对带多标签的序列进行表征的语法施加了许多强约束（比如，I-PER不能遵循B-LOC；详见2.4），这样的话，模型一开始的独立性假设就不满足了。

因此，我们的模型不是独立地词性标记决策，而是使用条件随机场对它们进行联合建模（Lafferty et al., 2001）。对于这么一个输入句子：

$$
\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n),
$$

我们认为$\mathbf{P}$是双向LSTM输出的评分矩阵。$\mathbf{P}$的规模是$n \times k$，这里的 k 是不同标签的数量，而$P_{i,j}$对应句子中第 i 个词的第 j 各标签的评分。对于一个预测的序列$\mathbf{y} = (y_1, y_2, \ldots, y_n)$，我们定义它的评分为：

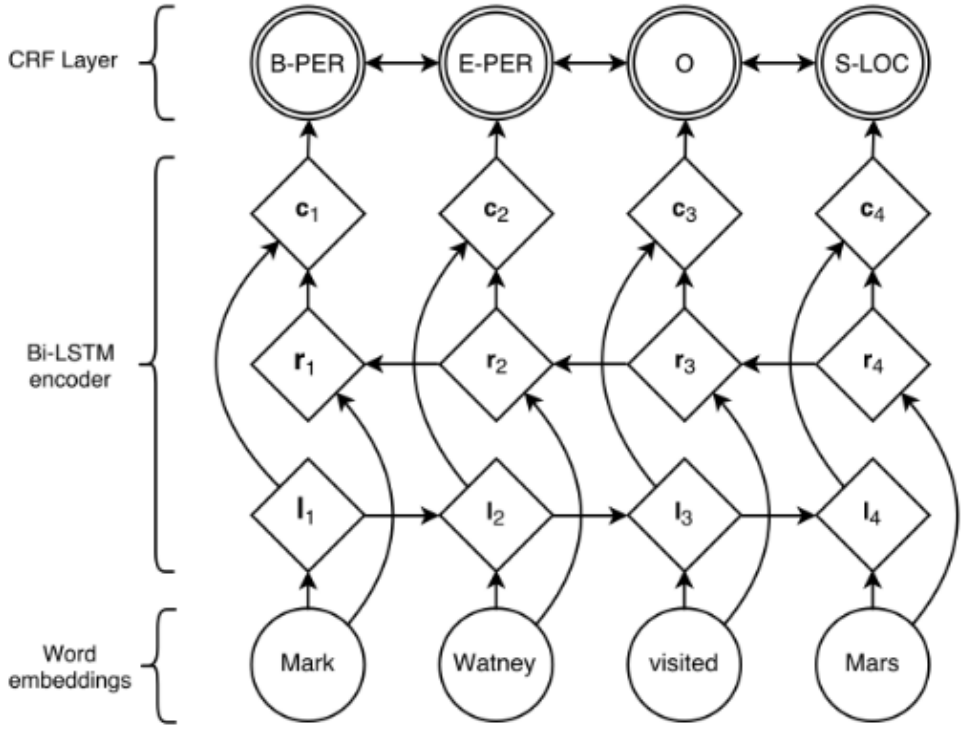
$$
s(\mathbf{X}, \mathbf{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}
$$

这里的$\mathbf{A}$是一个转换分数的矩阵，使得$A_{i,j}$表示为从标签 i 到标签 j 转换的分数。y_0和y_n是一个句子的起始标签与尾标签，我们将它们添加到一个可能标签的集合里。因此$\mathbf{A}$是一个规模为$k+2$的方阵。对于所有可能的标签序列，一个 softmax 产生序列$\mathbf{y}$的概率是：

$$
p(\mathbf{y}|\mathbf{X}) = \frac{e^{s(\mathbf{X}, \mathbf{y})}}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y}_\mathbf{X}} e^{s(\mathbf{X}, \tilde{\mathbf{y}})}}.
$$

在训练过程中，我们将预测正确的标签序列的对数概率最大化即可，从而获得最大评分从而预测出输出序列。

3.2.2.3 Parameterization and Training



与每个词条的每个词性标注决策相关的评分是由双向LSTM计算的输出（每个词的向量）和二元语法的转移评分一起计算出来的。结构示意图如上图。圆圈代表观测变量，菱形代表其父母的确定性函数，双圆圈代表随机变量。

这些表示可以在 ci 中被联系起来并被线性投影到一个尺寸等于独立标签数量的层上。我们不使用 softmax 作为该层的输出，而是使用之前说过的CRF以考虑到相邻标签的相关性，从而产生每一个词的最终预测 y_i 。另外，我们观察到在 ci 和CRF层之间添加一个隐含层可以稍稍改进我们的结果。所以本论文所有模型的实验也都加入了隐含层。给出观察的词，对于一个已注释语料库中的NER标签观测序列，这些参数通过训练将其对应的等式最大化。

3.3 Model

```
super(Named_Entity_Recognition, self).__init__()
        self.len_feature = len_feature
        self.len_words = len_words
        self.len_hidden = len_hidden
        self.dropout = nn.Dropout(drop_out)
        if weight is None:
            x = nn.init.xavier_normal_(torch.Tensor(len_words, len_feature))
            self.embedding = nn.Embedding(num_embeddings=len_words,
embedding_dim=len_feature, _weight=x).cuda()
        else:
            self.embedding = nn.Embedding(num_embeddings=len_words,
embedding_dim=len_feature, _weight=weight).cuda()
        self.lstm = nn.LSTM(input_size=len_feature, hidden_size=len_hidden,
batch_first=True, bidirectional=True).cuda()
        self.fc = nn.Linear(2 * len_hidden, type_num).cuda()
        self.crf = CRF(type_num, pad_id, start_id, end_id).cuda()
```

4 Train & Result

4.1 Train

由于数据量较大，因此借助colab完成。本作业在训练前对样本进行了按句子长短排序的处理，为防止长句子的存在使短句子需要padding过长的现象发生。

具体训练参数如下：

- sample：train.txt test.txt
- batch_size = 128
- lr = 0.001
- epochs = 50
- l_h, l_f：50

4.2 Result

Model	Precision	F1-score	Recall
LSTM + CRF + Glove - train	0.909292	0.902417	0.908333
LSTM + CRF + Random - train	0.942022	0.948812	0.944470
LSTM + CRF + Glove - test	0.724831	0.741003	0.732566
LSTM + CRF + Random - test	0.586279	0.600217	0.593728

4.3 Analysis

1. Glove & Random

   由于前三次作业均做了两种方法embedding的对比试验，后续作业不在对此进行讨论。但是值得一提的是，本作业中的两种对比实验效果是最为明显的（准确率差异高达20%），这是我在之前作业中未探索到的。并且在训练集测试集上二者的效果相反，train中Random的表象更好。作业二、三中虽然glove对模型的结果有提升，但是微乎其微。

2. 对CRF+LSTM的一些想法。

   - 本方法最大的特点是，在训练中学习字粒度特征，而不是手工建立单词前缀和后缀信息的特征工程。 学习字粒度的embedding有利于学习特定任务或者特定领域中的表示（representation）。
   - 字符查找表是随机初始化的，包含了每个字符的embedding。通过一个双向lstm将word中的每个character相关联起来，正向lstm得到character embedding的正向序列，反向lstm得到反向序列。
   - 从双向lstm得到正向和反向的character embedding序列，拼接起来得到一个word embedding，这个embedding是字粒度（character-level）。
   - 最后将上问题中字粒度的word embedding与词粒度的embedding拼接起来得到最后的embedding。词粒度的embedding是通过查找word lookup-tabel得到。