自然语言处理课程实验报告

柳明
School of CEI/AI, Nanjing Normal University

1. 背景

中文分词（Chinese word segmentation, CWS ）任务常被视为一个序列标注任务，用序列标注任务的思路处理中文分词问题的关键在于：每个字在构造成词的时候，都有一个确定的位置，即对于词中的一个字来说，它对应的标签只能是词首字、词中字、词尾字或单字词。

不同于之前将分词问题转化为序列标注任务的做法，张岳等 [2]将中文分词问题转化为一种增量式地对序列进行多分类的任务，提出了一种以基于词的特征为输入的感知机算法用来处理中文分词问题。

2. 要点简述

本小节就阅读文章时和实验中遇到的一些要点做出记录和讨论。

基于词的特征. 文中 [2]也提到：之前常用的特征往往通过一个大小为 5 个字（character）的窗口在序列上滑动进行特征提取，这是一种基于字的特征。例如，下面是一个较为典型的特征模板：

$$C_n \ (n = -2, -1, 0, 1, 2) \tag{1}$$

$$C_n C_{n+1} \ (n = -2, -1, 0, 1) \tag{2}$$

$$C_{-1} C_1 \tag{3}$$

其中，C_n 表示处于第 n 个位置的字，当 $n = 0$ 时，表示当前字。以“我爱北京天安门”这句话为例，假设当前 C_n 为“北”字，那么使用上面这个特征模板抽取到的特征为：（1）{我，爱，北，京，天}；（2）{我爱，爱北，北京，京天}；（3）{爱京}。

可以看到特征模板 1 抽取出了一元特征（character unigram），特征模板 2 和 3 抽取出了二元特征（character bigram）。但这样的通过 5 个字符长度的窗口抽取的特征不能充分利用到窗口外的上下文信息，文中举了这样一个例子：“其中/国外/企业”和“中国/外企/业务”这两个句子中，以“外”为当前词的 5 字符长的窗口所截取的内容都是“中国外企业”，仅仅依靠这样一个局部窗口的提取的信息很难对序列做出正确的切分。

文章中设计的特征模板定义在 2 个连续的词上（word bigram），词级别的特征使得特征提取能覆盖到更广的上下文，模型可以充分利用更大范围的上下文信息做出决策。

全局特征向量和局部特征向量. 通过定义特征模板，每条已切分的句子都可被映射为一个全局特征向量。具体来说，将一句已切分的句子记为 $W_{1:|W|}=w_1w_2\cdots w_{|W|}$，全局特征向量通过如下方式计算。

$$\vec{\phi}\left(W_{1:|W|}\right)=\sum_{j=2}^{|W|}\vec{\phi}\left(w_{j-1},w_j\right) \tag{4}$$

即累加句中所有的局部特征 $\vec{\phi}(w_{j-1},w_j)$，通过对序列从左到右进行遍历，增量式地求出句子对应的全局特征向量。

柱搜索. 如前面所说，基于词的特征带来了范围更广的上下文信息，但相应地，特征维度会非常大并且特征向量是稀疏的。文中解码部分选择了用柱搜索（beam search）算法实现，为了对比，在此先简单分析一下如何将动态规划方法用于本问题 [3]。

首先将长为 n 个字符的输入序列记为 $C_{1:n}$，切分完的输出序列记为 $W_{1:|W|}$，解码的目标是根据感知机的权重向量 $\vec{\theta}$ 输出分数最高的切分 $\hat{W}$，即

$$\hat{W}=\arg\max_{w}\vec{\theta}\cdot\vec{\phi}\left(W\right) \tag{5}$$

记以词 $C_{b:e}$ 为最后一个词的序列为 $W\left(b,e\right)$，分数最高的以 $C_{b:e}$ 为最后一个词的输出序列为 $\hat{W}\left(b,e\right)$。假设 $\hat{W}\left(b,e\right)$ 中倒数第二个词为 $C_{b':b-1}$，那么，$\hat{W}\left(b,e\right)$ 中以字 C_{b-1} 为结尾的子序列也一定是所有的以 $C_{b':b-1}$ 为结尾词的切分序列中分数最高的，即 $\hat{W}\left(b',b-1\right)$，如图 1 所示。

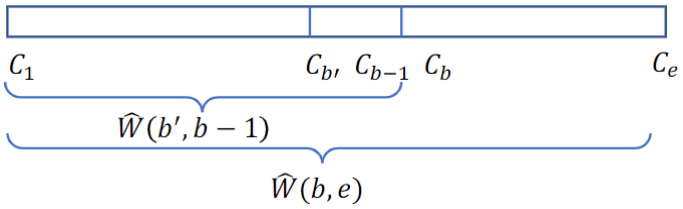

图 1. 子问题分解示意图。

注意到分数也同样可以增量式地计算，这样就可以写出计算分数的递推关系如式 6 所示。其中，$\hat{W}\left(b,e\right)$ 表示最后一个词为 $C_{b:e}=c_b,c_{b+1},\cdots,c_e$ 的最高分数的输出序列，起始字符的索引 $b\in[1,\cdots,n]$，结尾字符的索引 $e\in[b,\cdots,n]$。

$$\text{score}\left(\hat{W}(b,e)\right)=\max_{1\le b'\le b-1}\left(\text{score}\left(\hat{W}(b',b-1)\right)+\vec{\theta}\cdot\vec{\phi}_c(C_{1:n},b',b-1,e)\right) \tag{6}$$

那么根据式 6，使用动态规划求解的过程中就需要维护两张表：一张表对所有的 $b\in[1,\cdots,n]$，$e\in[b,\cdots,n]$，存储 $\text{score}\left(\hat{W}(b,e)\right)$ 到表中的对应位置上；另一张表 bp 用来存储解码过程中满足 $\max_{b'}$ 的 b'。最后输出最高分数的序列，即

$$\hat{W}=\arg\max_{b\in[1,\cdots,n]}\text{score}\left(\hat{W}(b,n)\right) \tag{7}$$

根据以上分析，可以在算法 1 中给出序列切分问题的动态规划解码算法流程。

由于求解过程中对 b、e 和 b' 进行了枚举，容易看出算法的时间复杂度为 $O\left(n^3\right)$。结合之前的叙述，采用词级别的特征的优点是（1）提供了范围更广的上下文；（2）基于输出结构做特征提取，直接对序列切分问题

Algorithm 1: A dynamic programming decoding algorithm.

Input : Sequence $C_{1:n} = c_1, c_2, \cdots, c_n$, perceptron parameter $\vec{\theta}$

Output: Segmented sequence $W_{1:|W|} = w_1 w_2 \cdots w_{|W|}$

```
// initialization;
for e ∈ [1, ··· , n] do
    for b ∈ [1, ··· , e] do
        table [b, e] ← −∞;
        bp [b, e] ← −1;
    end
    table [1, e] ← θ⃗ · φ⃗_c (C_{1:n}, 0, 0, e);
end
// dynamic programming;
for e ∈ [2, ··· , n] do
    for b ∈ [2, ··· , e] do
        for b′ ∈ [1, ··· , b − 1] do
            if table [b′, b − 1] + θ⃗ · φ⃗_c (C_{1:n}, b′, b − 1, e) > table [b, e] then
                table [b, e] ← table [b′, b − 1] + θ⃗ · φ⃗_c (C_{1:n}, b′, b − 1, e);
                bp [b, e] ← b′;
            end
        end
    end
end
max_score ← max_{b′∈[1,··· ,n]} table [b′, n];
W_{1:|W|} ← backtrace with bp;
```

进行建模。但是也带来了两个问题：（1）特征维度非常大，特征向量也很稀疏；（2）解码效率低，使用 word bigram 的特征的动态规划解码算法时间复杂度为 $O(n^3)$，如果使用 word trigram 的特征，最优子问题的分解同样类似于之前所述，但时间复杂度也会相应地增加到 $O(n^4)$。

所以在文中选择了基于柱搜索的解码方法，使用柱搜索能够使得模型可以采用不满足马尔科夫假设的特征，解码器从左至右增量式地处理输入序列，可以比动态规划方法更快地构建输出序列。

但是也应该注意到，动态规划方法是一种精确搜索（exact search）方法，算法停机时能够保证搜索到的是满足条件的最优解，而柱搜索是一种非精确搜索方法，解码器不能保证最终一定可以搜索到模型打分最高的候选输出，选择柱搜索也是在解的最优性和解码效率之间做一种权衡。

Lazy update. 文中使用了平均感知机 [1]作为处理序列切分问题的模型，在这里先回顾一下平均感知机算法，见算法 2。

容易看出，平均感知机训练算法在训练过程中维护一个和向量 $\vec{\sigma}$，将每轮迭代的感知机权重向量 $\vec{\theta}$ 加到和向量 $\vec{\sigma}$ 上（算法 2 第 8 行），满足停机条件之后再利用和向量 $\vec{\sigma}$ 求出感知机的平均权重向量。但是也应当注

Algorithm 2: Average perceptron training algorithm.

Input : $D = \{(x_i, y_i)\}\big|_{i=1}^{N}$

1 $\vec{\theta} \leftarrow \vec{0}$; $\vec{\sigma} \leftarrow \vec{0}$; $t \leftarrow 0$; *// initialization*
2 **repeat**
3 **for** $i \in [1, \cdots, N]$ **do**
4 $z_i \leftarrow \arg\max_{\hat{z}} \vec{\theta} \cdot \vec{\phi}(x_i, \hat{z})$; *// decoding*
5 **if** $z_i \neq y_i$ **then**
6 $\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(x_i, y_i) - \vec{\phi}(x_i, z_i)$
7 **end**
8 $\vec{\sigma} \leftarrow \vec{\sigma} + \vec{\theta}$;
9 **end**
10 $t \leftarrow t + 1$;
11 **until** $t = T$;
12 $\vec{\sigma} \leftarrow \vec{\sigma}/NT$;

意到，当向量维度非常高时，每轮迭代中计算和向量 $\vec{\sigma}$ 的计算代价非常高，可以举一个较为直观的例子：实验中，在 5 折交叉验证的第三种数据划分上训练 3 轮之后，平均感知机的权重向量维度为 $3,819,516$，其中非零特征有 $2,249,786$ 个。

所以在训练过程具体实现方面，作者们采用了称为 lazy update 的优化方法或者说技巧，出发点很自然：训练过程中处理每条样本后对权重向量 $\vec{\theta}$ 进行更新，需要更新的维度只是向量全部维度中的一小部分。特别地，为平均感知机权重向量的每一维定义一个更新信息向量 $\vec{\tau} = (n, t)$ ，存放该维度最近一次更新时的句子索引 n 和训练轮次 t。每条句子处理完之后，在权重向量里，只对和当前句子中的特征相对应的维度进行更新。

这种优化方法 “lazy” 之处就在于，对原训练算法 2 的第 8 行的处理进行了分解，不采用将整个向量累加的方式，而只将当前句子中需要更新的特征维度加到和向量上。记在第 t 个轮次中，处理第 n 个样本之前的各个向量的第 s 维度记为：$\theta_s^{n-1,t}$、$\sigma_s^{n-1,t}$ 和 $\tau_s^{n-1,t} = (n_{\tau,s}, t_{\tau,s})$，假设解码器对这句话的解码结果为 $z_{n,t}$ 且 $z_{n,t} \neq y_n$，那么就要根据 $\vec{\phi}(y_n)$ 和 $\vec{\phi}(z_{n,t})$ 的维度对权重向量中对应的维度进行更新，如公式 8 到公式 11 所示。

$$\sigma_s^{n,t} = \sigma_s^{n-1,t} + \theta_s^{n-1,t} \times ((t - t_{\tau,s}) \times N + (n - n_{\tau,s})) \tag{8}$$

$$\theta_s^{n,t} = \theta_s^{n-1,t} + \vec{\phi}(y_n) - \vec{\phi}(z_{n,t}) \tag{9}$$

$$\sigma_s^{n,t} = \sigma_s^{n,t} + \vec{\phi}(y_n) - \vec{\phi}(z_{n,t}) \tag{10}$$

$$\tau_s^{n,t} = (n, t) \tag{11}$$

对和向量的第 s 维 $\sigma_s^{n,t}$，因为采用了 lazy update 的方法，所以从该维度最近的一次更新到当前更新中，这个维度一直没有进行累加，所以更新之前应当将这部分补偿回来，根据更新信息向量 $\tau_s^{n-1,t} = (n_{\tau,s}, t_{\tau,s})$ 中的信息，这一维特征跳过累加求和的次数为 $(t - t_{\tau,s}) \times N + (n - n_{\tau,s})$，即进行如公式 8 所示更新。

3. 实验设置与实验结果

实验设置. 数据集为人民日报一、二月份语料（1998 年 1 月、1998 年 2 月），共 38329 行。评估方式采用 5 折交叉验证，通过调用 scikit-learn 库实现数据集的划分，其中固定 KFold 方法的 random_state 参数（即随机数种子）为 10，便于之后重现同样的数据划分结果，表 1 里列出了大致的数据划分情况。

	训练数据索引	训练数据大小	测试数据大小
split_1	[1, 2, 3, ... , 38326, 38327, 38328]	30663	7666
split_2	[0, 1, 2, ... ,38326, 38327, 38328]	30663	7666
split_3	[0, 2, 3, ... ,38325, 38326, 38328]	30663	7666
split_4	[0, 1, 2, ... ,38324, 38326, 38327]	30663	7666
split_5	[0, 1, 3, ... ,38325, 38327, 38328]	30664	7665

表 1. 实验数据划分。

超参数方面，根据文中 [2]的对比实验结果将 beam_size 设置为 16。由于训练速度较慢[1]，所以仅将训练迭代轮数设置为 3。

实验结果. 5 折交叉验证的评估结果如表 2 所示。

	P	R	F1
split_1	94.9	95.0	95.0
split_2	95.0	94.9	95.0
split_3	**95.2**	**95.1**	**95.2**
split_4	**95.2**	94.9	95.1
split_5	95.1	94.9	95.0
AVG	95.1	95.0	95.1

表 2. 评估结果。其中评估指标如下：P - precision(%), R - recall(%), F - F1-measure.

参考文献

[1] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)*, pages 1–8, 2002. 3

[2] Y. Zhang and S. Clark. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 840–847, 2007. 1, 5

[3] Y. Zhang and Z. Teng. *Natural language processing: a machine learning perspective*. Cambridge University Press, 2021. 2

[1]训练速度因机器的不同也会有所差异。按实验经验来说，5 个数据划分全部训练完并保存平均感知机的权重，再读取保存的权重之后对测试数据进行序列切分，最后输出评估结果，这整个过程大约需要 13 小时。