

# 作业一报告

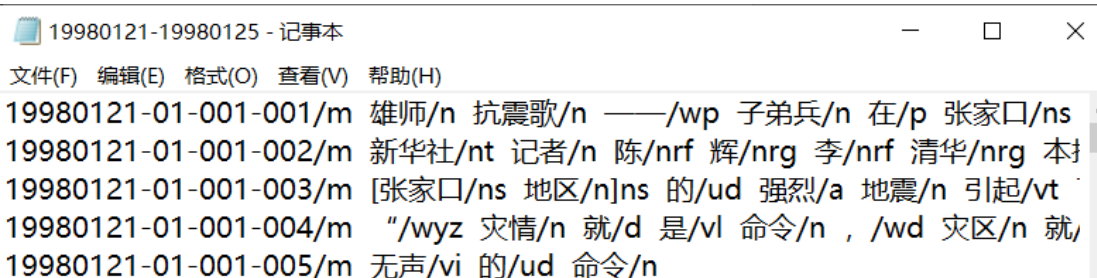
## 1 概述

本次作业对 1998 年 1 月人民日报做了命名实体识别，识别对象为机构名，使用了 one-hot 特征并采用逻辑回归的方式建立模型，最终得到的 F1-measure 收敛至。在实验中 one-hot 特征采取 500-1500 和 5000-15000 维的形式，梯度下降采用小批量梯度下降，并比较了根据两种常用词典进行训练得到结果的不同。

## 2.实验流程

### 2.1 分割原始数据

将原文件重命名为“renming.txt”,根据每行开头日期将其划分为三个 txt 文件，“19980101-19980120.txt”、“19980121-19980125.txt”、“19980126-19980131.txt”，分别作为训练集、验证集、测试集的初始文本文件。该步骤运行 data 文件夹下 split\_collection.py 即可完成，运行结果符合预期，以下是“19980121-19980125.txt”文件部分截图：



### 2.2 获取常用词词典

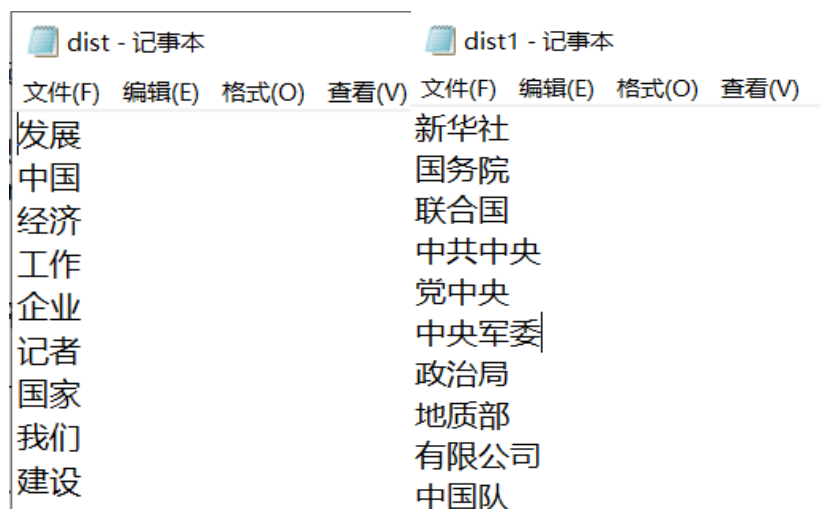
扫描训练集文件，初步得到训练集样本数约为 73 万，接下来采用两种方式获得常用词典 dist 和 dist1。

dist 词典大小基于 one-hot 向量大小设置，one-hot 向量大小可以通过调整所有 py 文件中 size 的大小来改变。

dist 词典基于停用词表，停用词表包括标点符号、部分功能词、部分词汇词，结合网上资料和训练集本身特征手动选取，现保存于 data 文件夹内。扫描训练集文本文件，统计不出现于停用词表中的各词出现的频率，选取频率最高的 size 个词写入词典文件“dist.txt”中。

dist1 词典基于高频机构名，即在确保高频机构名完全选取形成常用词典。扫描训练集文本文件，统计机构名出现的频率，选取频率最高的 size 个词写入词典文件“dist1.txt”中。

dist 和 dist1 可分别运行 get\_count.py 和 get\_count1.py 获得，以下分别为“dist.txt”和“dist1.txt”文件部分截图：

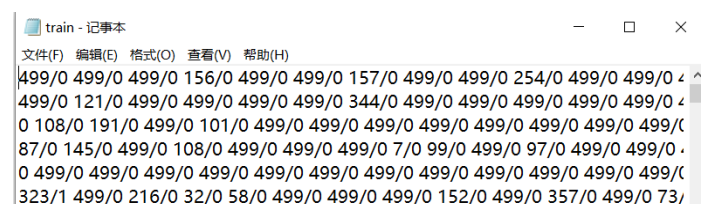


## 2.3 预处理训练集、验证集、测试集

根据获得的常用词典转化训练集、验证集、测试集，减少此后训练时的内存消耗。

分别转化三个集合，将每个词转化为 word1/word2 的形式。读取集合文件，对每一个词判断其本身是否在常用词典中出现，若出现则 word1 标记为其在常用词典中的序号，否则标记为窗口大小-1，同时判断词性，若为 nt，则标记 word2 为 1，否则为 0。需要特判[]内的词性情况，具体操作方法是创建临时列表，记录[]内所有词后一并修改，并根据[]词性判断词性。

由于存在两个常用词典，所以共可以获得六个转化后的文件，分别为“train.txt”、“verify.txt”、“test.txt”、“train1.txt”、“verify1.txt”、“test1.txt”，本步骤运行 predeal.py 即可完成，以下为 size 大小为 500 情况下“train.txt”文件部分截图：



```
499/0 499/0 499/0 156/0 499/0 499/0 157/0 499/0 499/0 254/0 499/0 499/0 499/0 499/0 499/0 121/0 499/0 499/0 499/0 499/0 344/0 499/0 499/0 499/0 499/0 499/0 108/0 191/0 499/0 101/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 87/0 145/0 499/0 108/0 499/0 499/0 499/0 7/0 99/0 499/0 97/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 499/0 323/1 499/0 216/0 32/0 58/0 499/0 499/0 499/0 152/0 499/0 357/0 499/0 73/
```

## 2.4 训练并计算 F1-measure

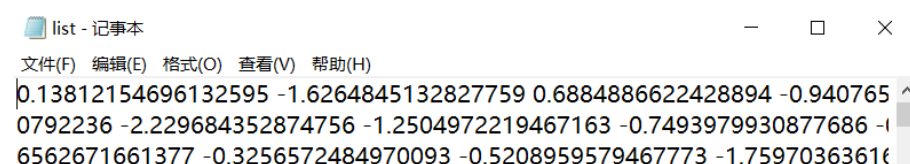
根据预处理过后的训练集，对 theta 进行梯度下降，这里使用小批量梯度下降的方式，每次迭代使用 batch 个样本对 theta 进行更新（本次作业 batch 设为 128），为确保效率，每对训练集所有样本遍历一次计算一次验证集上的 F1-measure，并将 F1-measure 和对应 theta 写入历史文件。

theta 初值设为根据正态分布取随机值的 15000 位向量，将预处理后的训练集的所有样本读入并将词序号和词性标签分别转化为 x、y 序列。

接下来开始训练，设定 epochs 参数，一个 epoch 表明训练集中每个样本都参与训练一次，每经过一次 epoch 计算一次 F1-measure 并记录到历史文件，更新临时 theta 文件中的 theta 值（用于中断训练后再次进行时读取）。每次 epoch 包含多次迭代操作，每次迭代按顺序取 batch 个样本对 theta 进行更新，若取到训练集最后不足 batch 个样本则取小于 batch 个样本进行更新。对样本序号为 i 的样本进行更新时，需要建立 15000 位临时向量 x\_data，根据 i-1 号样本、i 号样本、i+1 号样本对应的 x 值，更新 x\_data，此操作可以看做是三个 5000 维 one-hot 向量的拼接，然后根据梯度下降公式更新梯度值 grad。当 batch 内样本均被计算过后利用 grad 更新 theta，重置 grad 并计算下一个 batch，直到所有样本均参与过一次训练。

每次 epoch 后计算验证集 F1-measure 值。读入验证集文件，根据 theta 对每个样本进行计算，如第 i 号样本根据 i-1、i、i+1 号样本形成 15000 位 one-hot 向量拼接，然后即可预测 i 号样本词性，设预测值为 y\_predict，样本自身有标签值为 y\_true，若 y\_predict=y\_true=1，则 TP 加 1，若 y\_predict=1、y\_true=0，FP 值加 1，若 y\_predict=0、y\_true=1，FN 值加 1，可得查准率为  $P = TP / (TP + FP)$ ，查全率为  $R = TP / (TP + FN)$ ，即可得 F1-measure 为  $2PR / (P + R)$ 。

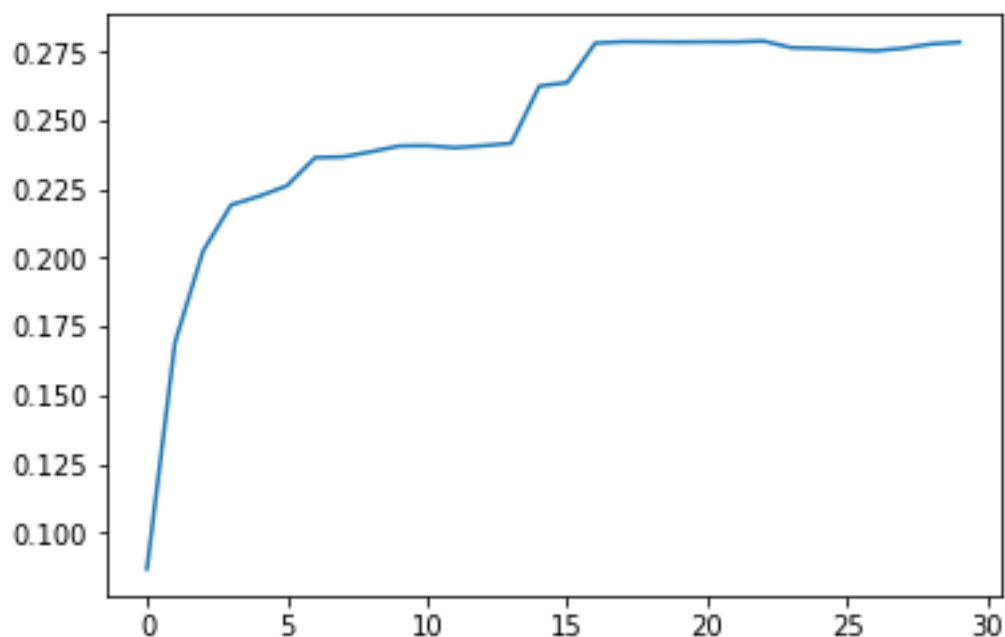
运行 main.py，修改参数即可分别对基于两个词典获得的特征进行训练得到不同结果，生成 F1-measure 历史文件“list.txt”、“list1.txt”，list 文件每行保存一个 F1-measure 和对应的 theta 值，“list.txt”文件部分截图如下所示：



```
0.13812154696132595 -1.6264845132827759 0.6884886622428894 -0.940765 0792236 -2.229684352874756 -1.2504972219467163 -0.7493979930877686 -1 6562671661377 -0.3256572484970093 -0.5208959579467773 -1.75970363616
```

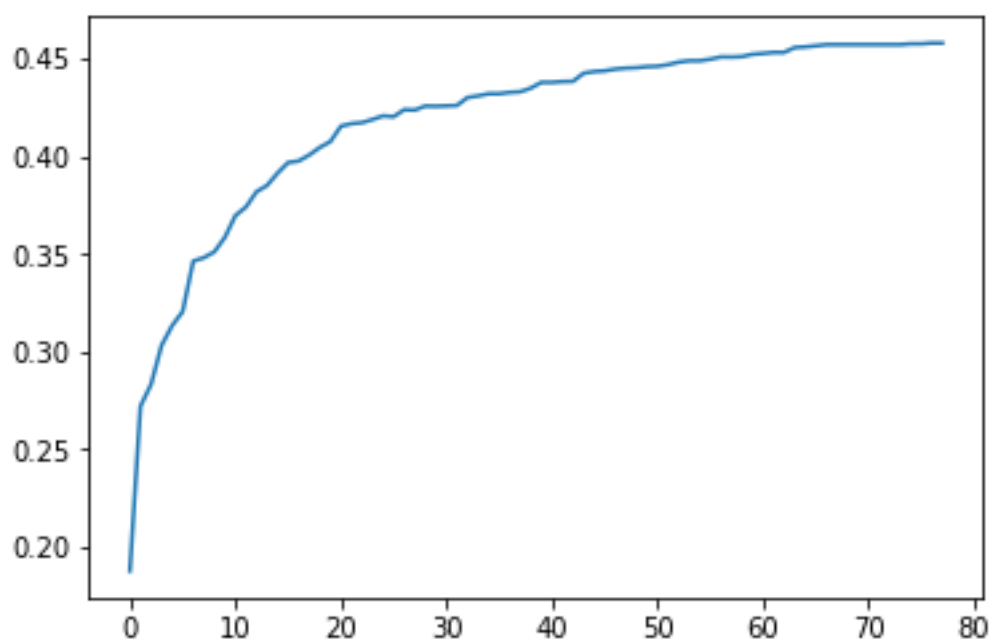
### 3.结果及分析

**3.1** 最初 size 取 500 位, 使用基于停用词表的词典 dist 进行训练, 得到结果较差, F1-measure 最终收敛于 0.277-0.278, F1-measure 迭代图如下:



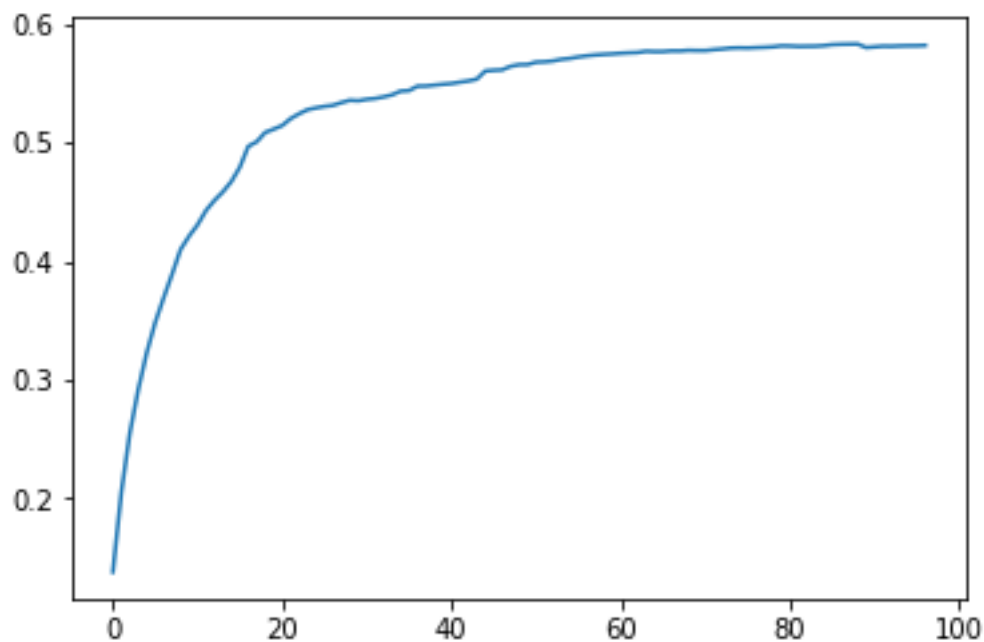
用最终 theta 值在测试集上计算得 F1-measure=0.2793

**3.2** 在 3.1 中观察到查全率较低的情况, 推测是因为 size 较小且取到的 nt 词极少, 所以针对性创建基于高频机构名词典 dist1, size 仍取 500 位的情况下进行训练, 得到结果上升显著, 在验证集上 F1-measure 最终收敛于 0.457-0.458, 查准率和查全率均有显著上升, F1-measure 迭代图如下:



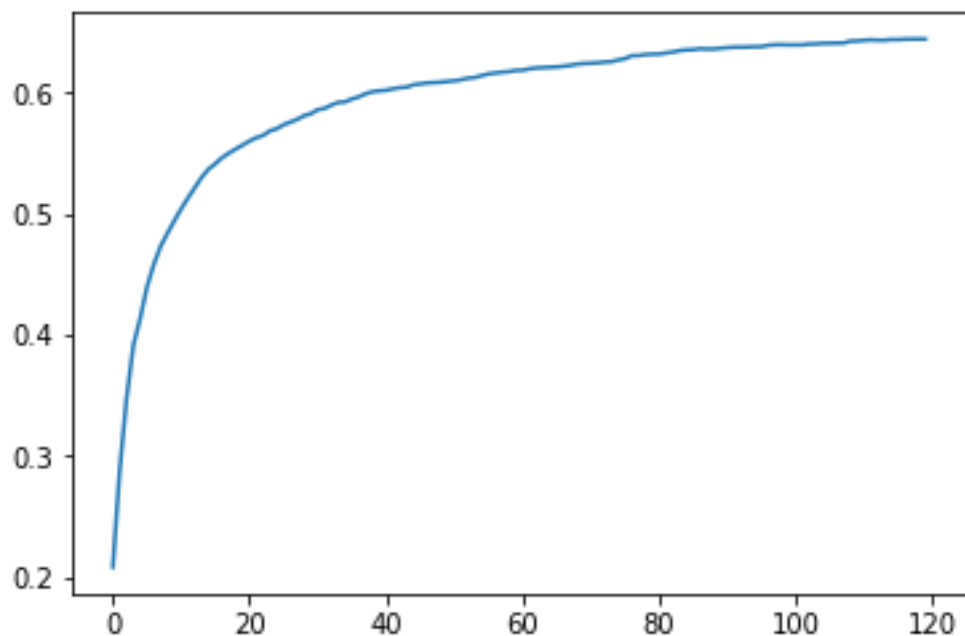
用最终 theta 值在测试集上计算得 F1-measure=0.4558

**3.3** 尝试扩大 size 大小至 5000 位, 使用基于停用词表的词典 dist 进行训练, 得到结果较 500 位情况有显著提升, 推测主要原因是相较 3.1 取入了大量的 nt 词, 在验证集上 F1-measure 最终收敛于 0.581-0.582, 查准率查全率均有显著上升, F1-measure 迭代图如下:



用最终 theta 值在测试集上计算得 F1-measure= 0.5601

**3.4** 在 size 取 5000 位的情况下, 使用基于高频机构名词典 dist1 进行训练, 结果相较 3.3 略有上升, 但并不明显, 在验证集上 F1-measure 最终收敛于 0.643-0.644, F1-measure 迭代图如下:



用最终 theta 值在测试集上计算得 F1-measure= 0.6165

推测若继续扩大 size, 基于两个词典的训练结果将逐渐趋于一致。

#### 4.总结

第一次做知识工程的任务遇到了很多困难,所幸都一一克服了,其中碰到的内存问题给我带来了最大的麻烦,pytorch 框架下 torch 具有的历史值会累计在内存中,如果不及时处理,内存很快就会占满,系统也因此多次崩溃。

本次任务的模型虽不算复杂,但我认为仍具有很大的优化空间,主要集中在特征提取上,可以考虑继续扩大 size 大小,同时结合前人经验优化停用词表的设计,或许可以结合两个词典的经验设计出更合适的常用词典。