Homework 3 嵌套命名实体识别

人工智能学院 张含笑 181220067

1 任务概述

解决嵌套命名实体识别问题 (Nested NER) 之前，需要先解决普通NER任务，即将实体识别当做序列标注问题。

具体来说，对于一个长度为n的句子$S = \{s_1, s_2, ..., s_n\}$，其中$s_i$表示句子中的第$i$个token。序列标注给每个token打一个标签，来表示这个token所在的实体类别。一组相邻且类别相同的token构成的子序列 span(start, end, type) 就是我们想要的实体。这样，一个抽取实体的问题就被转化为一个给每个token进行分类的问题。

序列标注算法可用类似于上课讲过的词性标注算法求解，可用的方法有隐马尔科夫模型、条件随机场、LSTM网络等。在序列标注中，需要定义不同的Schema，来使得将序列标注的结果从token层面提高到实体层面，并且保持其唯一性。定义有效的Schema的意义在于消除从token标注复原出实体时的歧义。有两种最常见的Schema：

- BIO：即Beginning、Inside、Outside，对于一个实体的第一个token，标注为B-[type]，实体其他位置的token标注为I-[type]，不属于任何实体的token标注为O；这样，对于一个标签数 [公式] 的实体集，使用BIO标注将转变为 [公式] 个标签；
- BIOES：即Beginning、Inside、End、Outside、Single。其中End用来标识一个实体的结束，而Single用来标识仅包含一个token的实体。

解决一般实体识别的方法无法解决嵌套实体识别问题。因为一个token可以属于两个实体。对于Nested NER，也有一些基于序列标注的工作：

- 将分类任务的目标从单标签变成多标签：Schema不变，模型也不变，在最后分类的时候，从输出一个类到输出所有满足一个指定阈值Φ的所有类。
    - 完全不改变Schema，只是在输入训练集的时候，训练集中的label从原来的one-hot编码形式变成一个指定类别的均匀分布；在进行推理时，给定一个hard threshold，所有概率超过这个阈值的类别都会被预测出来，当做这个token的类。但这种方法模型学习的目标设置过难，阈值定义比较主观，很难泛化；
    - 修改Schema，将可能共同出现的所有类别两两组合，产生新的标签。这样做的好处是最后的分类任务仍然是一个单分类，因为所有可能的分类目标我们都在Schema中覆盖了。但这种方法指数级增加了标签，导致分布过于稀疏，很难学习；对于多层嵌套，需要定义非常多的复合标签；
- 修改模型的Decode过程，保证能够给一个token同时赋予多个类别。
    - 使用生成式的方法，如seq2seq中的Decoder来逐个生成每个token的标签。使用Decoder能够将输入的token数量与输出的类别数量解绑，允许给token打上超过一个的标签。
    - 如果一次分类无法解决实体嵌套的问题，那就对第一次的分类结果继续做分类，如是迭代，直到达到最大迭代次数或是不再有新的实体产生为止。

2 实验方法

2.1 词嵌入

观察数据得知，数据集内的词都是生物相关的，许多词作为合成词，由'-'连接两个单独的词合成新词。

对于这些合成词，如果将其作为单独的新词处理，则会丢失一部分语义——因为现实中我们理解合成词也会先将其分开理解再结合两个词的语义理解——因此在本实验中，将合成词拆开，作为单独的词进行词嵌入。

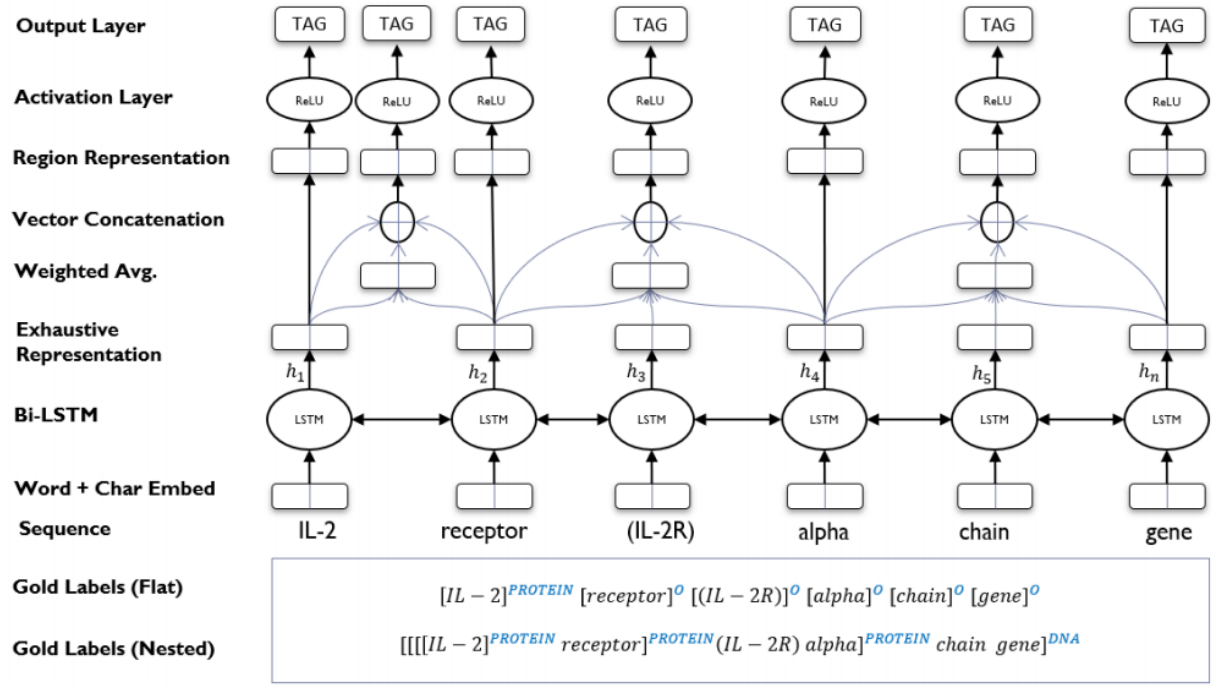
2.2 Nested NER实现

我最终选择的是 [Sohrab and Miwa, 2018] 中提出的方法——抛弃序列标注的方法，将句子S中所有的子序列全部枚举出来，即得到一个子序列集合：$s_1, s_1s_2, s_1s_2s_3, \ldots, s_2, s_2s_3, \ldots, s_n$。训练一个分类器$C$，负责将子序列映射到给定的标签集合中。

这种方法对于长句子时会遇到很多问题：时空复杂度极高、分类器训练十分困难、负样本极多等，一下方法可以减弱这些问题：

- 模型训练困难：给每个类型单独训练一个分类器；
- 时空复杂度：假设最长的实体长度为n，全枚举子序列时只枚举长度小于n的所有情况；
- 负样本很多：在执行分类之前，先训练一个或多个通用的筛选器，或通过人工规则首先筛掉一批负样本；在训练过程中对负样本进行采样，而非使用全部。

模型基本结构如下：



设置一个超参数——子序列的最大长度，用于控制可能产生的子序列的数量，模型对所有可能的子序列进行分类。当句子的单词数量为l，最大实体长度为m，实体类型数量为n时，模型需要对这个句子进行$O(lmn)$次分类。

先将单词的表示输入LSTM模型，得到输出后拼接得到子序列的表示，利用子序列的表示进行分类。对子序列进行表示时并分类时，将子序列的边界单独取出，和非边界的子序列内部的均值进行拼接，作为该子序列的表示。例如子序列$s_i - s_j$，设h_i是单词s_i经过LSTM的输出，子序列表示为：

$$R(i,j) = \left[h_i; \frac{1}{j-i+1} \sum_{k=i}^{j} h_k; h_j \right]$$

这些输出经过ReLU激活函数和线性全连接层确定是否将对应子序列归为某一类。

3 代码实现

运行方式：命令行输入python train.py即可运行。

代码实现参考了 [Sohrab and Miwa, 2018] 的仓库https://github.com/csJd/deep_exhaustive_model. 分为如下几个部分：

- 调整数据格式：txt文件里是句子对应(start,end G#type)，先将句子按照长度分成 l+#('-') 行，每行一个单词和四列标签，如果这个单词属于某个type的子序列，则标签为B/I-type. B指自序里的开头，I指子序列的其他部分；不属于任何type则记为O。如果这个单词只属于一个type，则只有第一列标签不为O；如果属于大于一个type，则一次向之后的列填充B/I-type. 这部分代码在dataset.py中。
- 模型设置：CharLSTM对单个单词进行编码；ExhaustiveModel包含了CharLSTM、词向量拼接、ReLU、Linear部分，句子输入经过这几个部分最终得到分类结果。这部分代码在model.py中。
- 模型训练：根据调整格式后的数据生成词典(json文件)，得到不同的词对应的序号。这里初始词向量的表示采用PubMed-shuffle-win-30.bin，可以在 https://drive.google.com/open?id=0BzMCqpcgEJgiUWs0ZnU0NlFTam8 下载。评价标准是交叉熵，用Adam优化器进行优化。对训练数据训练n_epoch轮，每轮训练结束后用dev集进行评估，用当前最优的模型替代之前的模型保存下来。最后对test集进行预测。这部分代码在train.py中。
- 模型评估：包含评估和预测。计算模型在dev上的f1值，并对test集进行预测。这部分代码在eval.py中。

4 实验结果

实验中，我对几个模块分别做了消融实验，计算它们对实验结果的影响。对最大子序列长度等超参数做了调节，比较实验结果。

提交后最高的f1值为0.68.

参考文献

[Gupta et al., 2017] Gupta, N., Singh, S., and Roth, D. (2017). Entity linking via joint encoding of types, descriptions, and context. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2681–2690.

[Lample et al., 2016] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California. Association for Computational Linguistics.

[Li et al., 2019] Li, X., Feng, J., Meng, Y., Han, Q., Wu, F., and Li, J. (2019). A unified mrc framework for named entity recognition. arXiv preprint arXiv:1910.11476.

[Ma and Hovy, 2016] Ma, X. and Hovy, E. (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In Proceedings of the 54th Annual Meeting of the Association for Computational

Linguistics (Volume 1: Long Papers).

[Miwa and Bansal, 2016] Miwa, M. and Bansal, M. (2016). End-to-end relation extraction using lstms on sequences and tree structures. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1105–1116.

[Sohrab and Miwa, 2018] Sohrab, M. G. and Miwa, M. (2018). Deep exhaustive model for nested named entity recognition. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2843–2849, Brussels, Belgium. Association for Computational Linguistics.

[Straková et al., 2019] Straková, J., Straka, M., and Hajic, J. (2019). Neural architectures for nested NER through linearization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 326–5331, Florence, Italy. Association for Computational Linguistics.

[Zheng et al., 2019] Zheng, C., Cai, Y., Xu, J., Leung, H.-f., and Xu, G. (2019). A boundary-aware neural model for nested named entity recognition. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 357–366, Hong Kong, China. Association for Computational Linguistics.