司法阅读理解

实验报告

姓名:王雨静

班级 ： 1期NLP训练营

日期:2021年4月24日

一、案例简介

任务说明

裁判文书中包含了丰富的案件信息，比如时间、地点、人物关系等等，通过机器智能化地阅读理解裁判文书，可以更快速、便捷地辅助法官、律师以及普通大众获取所需信息。本次任务覆盖多种法律文书类型，包括民事、刑事、行政，问题类型为多步推理，即对于给定问题，只通过单句文本很难得出正确回答，模型需要结合多句话通过推理得出答案。

数据说明

本任务数据集包括约5100个问答对，其中民事、刑事、行政各约1700个问答对，均为需要多步推理的问题类型。为了进行评测，按照9:1的划分，数据集分为了训练集和测试集。注意 该数据仅用于本课程的学习，请勿进行传播。

发放的文件为train.json和dev.json，为字典列表，字典包含字段为：

- _id：案例的唯一标识符。
- context：案例内容，抽取自裁判文书的事实描述部分。数据格式与HotpotQA数据格式一致，不过只包含一个篇章，篇章包括标题（第一句话）和切割后的句子列表。
- question：针对案例提出的问题，每个案例只标注一个问题。
- answer：问题的回答，包括片段、YES/NO、据答几种类型，对于拒答类，答案应该是"unknown"。
- supporting_facts：回答问题的依据，是个列表，每个元素包括标题（第一句话）和句子编号（从0开始）。

同学们需根据案例描述和问题，给出答案及答案依据，最终会综合两部分的效果并作为评判依据，评价方法与HotpotQA一致。

我们提供基础的模型代码在baseline目录下

评分要求

分数由两部分组成。首先，使用已有代码在训练数据上进行训练，并且完成开发集评测，这部分占50%，评分依据为模型的开发集性能和报告，报告主要包括对于模型基本原

理的介绍，需要同学阅读代码进行学习。 第二部分，进行进一步的探索和尝试，我们将在下一小节介绍可能的尝试，并在报告中汇报尝试的方法以及结果，这部分占50%。同学需要提交代码和报告，在报告中对于两部分的实验都进行介绍。

探索和尝试

- 使用2019年的阅读理解数据集（CJRC作为辅助数据集，帮助模型提高阅读理解能力
- 使用别的预训练语言模型完成该实验，例如THUNLP提供的司法BERT
- 对于新的模型架构进行探索，例如加入图神经网络（GNN）来加强模型的推理能力

参考资料

- CAIL2020——阅读理解

二、模型原理介绍

2.1 processing

原始数据结构：

- _id：案例的标识符，int
- context：案例内容，篇章包括标题（第一句话）和切割后的句子列表。
    - [[标题，[句子，句子.....]]]
- question：问题，str
- answer：回答，包括片段、YES/NO、据答几种类型，对于拒答类，答案应该是"unknown"。str
- supporting_facts：回答问题的依据，是个列表，每个元素包括标题（第一句话）和句子编号（从0开始）。
    - [[标题，句子编号], [标题, 句子编号].....]

2.1.1 第一步：从train.json到train_example.pkl.gz

（具体注释请看processing.ipynb）
将依据从列表转化成tuple，[标题，句子编号] --> (标题，句子编号)。
对案例内容中的句子一句几句处理，把句子切割成一个一个单字（token），并记录每一句话在总token中的位置。如果回答不是yes/no/unknown，记录答案在出现的起始和结束位置。
转换后的格式：

- qas_id: 案例标识符，int。
- qas_type: '' 空的str。
- doc_tokens: context中的字，token的列表。
- question_text: 问题，str。
- sent_num: 句子数，int。
- sent_names: [(标题，句子编号），（标题，句子编号）...]
- sup_fact_id: 证据的句子编号, [句子编号，句子编号....]
- para_start_end_position: 段落在tokens中的位置[(start, end)]
- sent_start_end_position: 每一个句子在tokens中的位置[(start, end)]

- entity_start_end_position: 空白列表[]
- orig_answer_text: 回答，str。
- start_position: 如果回答是片段，在tokens中的起始位置。[start, start,...]
- end_position: 如果回答是片段，在tokens中的结束位置。[end, end,...]

2.1.2 第二步：从train_example.pkl.gz到train_feature.pkl.gz

根据回答统计answer type，‘yes’是1，’no’是2，‘unknown’是3，其他（片段）是0。
用bert模型来对query分词，并加入起始token（’[CLS]’）和结束token（’[SEP]’），然后把query和context的token合到一起，并更新para、sent、answer在all_doc_tokens中的起始和结束位置。舍去大于max_seq_length的token，将query_tokens和all_doc_tokens使用bert tokenizer转化为id。生成遮罩mask。使用0填补id、mask使得他们等于最长度。更新sentence_spans（每一句话在tokens中的位置）。
转化后的格式：

- qas_id: 案例识别码，int。
- doc_tokens：query和context的token列表。
- doc_input_ids: doc_tokens对应的id列表。
- doc_input_mask: 有token的位置是1，后面用0填补。
- doc_segment_ids: query的位置是0，context token的位置是1，后面0填补。
- query_tokens: query的分词列表。
- query_input_ids: query_toekns对应的id列表。
- query_input_mask: 有token的位置是1，后面用0填补。
- query_segment_ids: query的位置是1，后面0填补。
- sent_spans: 句子在all_tokens中的位置。[(start, end).....]
- sup_fact_ids: 证据的句子编号。
- ans_type: 回答类型，int。
- token_to_origin_map: token到context原文的mapping。
- start_position: 回答在all_tokens中开始的位置。可能为空。
- end_position: 回答在all_tokens中结束的位置。可能为空。

2.2 modeling

联合使用bert和grah fusion net建模。
将doc_ids, sengment_ids, doc_mask输入bert模型，提取语义，得到context_encoding_layer。
然后将context_encoding_layer，和其他初始的batch信息输入fusion net，来建模寻找答案。
根据context_encoding_layer，用线性层判断答案的开始和结束位置的概率。
将句子的token和context_encoding_layer相乘，取最大值，输入线性层，寻找证据的位置。
将context_encoding_layer在dim = 1上的最大值输入线性层来找回答的类型。
用开始和结位置的概率来预测答案开始和结束的位置。使用了上三角矩阵和下三角矩阵来限制答案的最长长度15。减去1e30 * query mapping来限制模型不能预测query的内容。
输出start_logits, end_logits, type_logits, sp_logits, start_position, end_position。

criterion: Cross Entropy Loss (reduction = ‘mean’）
binary_criterion: Binary Cross Entropy With Logits Loss（reduction = ‘mean’）
sp_loss_fct: Binary Cross Entropy With Logits Loss (reduction = ‘none’)

各个loss的意义：
loss1: 开始和结束位置的Loss，用Cross Entropy Loss来计算。
loss2: 回答类型，用Cross Entropy Loss来计算。乘以type_lamda（该loss的权重）。
loss3: 每句话是否为证据，用Binary Cross Entropy With LogitsLoss来计算。乘以sp_lamda（该loss的权重），除以句子总数。
loss: loss1 + loss2 + loss3。

三、实验结果

```
'em': 0.4425,                    （0.5944）
'f1': 0.5263,                    （0.6822）
'prec': 0.5380,
'recall': 0.5332,
'sp_em': 0.2163,                 （0.3571）
'sp_f1': 0.5801,                 （0.5401）
'sp_prec': 0.6678,
'sp_recall': 0.5556,
'joint_em': 0.1310,              （0.2704）
'joint_f1': 0.3407,              （0.3701）
'joint_prec': 0.3975,
'joint_recall': 0.3317
```

实验结果不如readme中给出的（上面括号中的）。

这可能是因为做实验时为了能够load下模型和数据集，将batch size设为了1，导致模型没法很好得考虑全局的情况，容易在个别案例上过拟合。

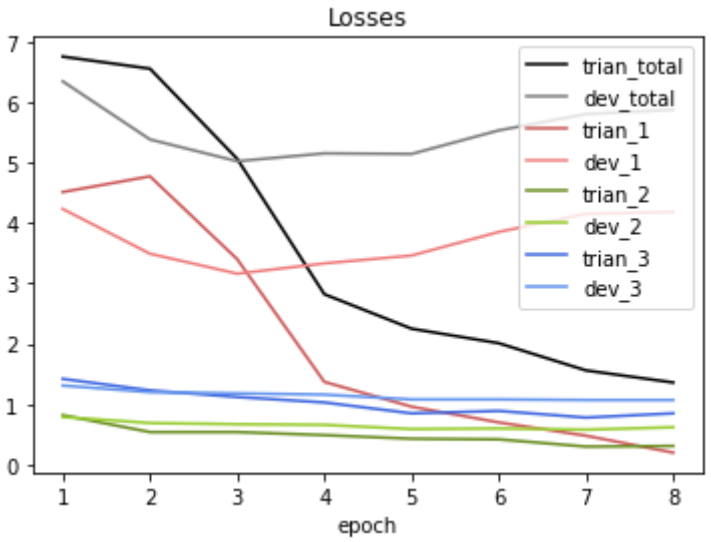

Fig1. losses

从Fig1中可以看到，在epoch 3以后，训练集上的loss还在不断下降，但是开发集上的loss却呈现出了上升趋势，这说明模型在训练集上过拟合了，这也导致模型的泛化能力较差，最后的效果不是很好。

四、改进思路

4.1 分词方法

这个实验对context用的是字级别的分词，这可能会丢失很多词级别的信息。
可以考虑使用bert的词表进行分词。如果有司法方面的词表，那就更好了，这样能够捕捉更多词级别的语义。

4.2 实体识别

可以考虑加入实体识别，利用已有的知识图谱来辅助逻辑上的处理和推断。对于一个案件，可以考虑识别出人物、地点、时间等信息，辅助案情的判断。

4.3 使用其他bert模型

可以考虑使用在大量相关材料（如司法案件、法律法条等文本）上预训练过的bert模型。这样的模型能够对司法相关词汇更好地建模，且词汇的语义会更偏向于司法方面，而非生活用语。
（给出的司法bert，下载后遇到了无法直接load的问题，尚未解决）
（目前使用的是bert-base-uncased，是在英文语料上训练的，不意外效果不好）

4.4使用更大的数据集进行训练

使用更大量的合理的数据集能有助于模型学习到更多有用的知识，帮助改善模型。
(时间有限，尚未写合适的函数来读取相关数据）

4.5 调整参数

建议使用略大一些的batch size，如16， 32等。同时可以调整一下其他参数，如学习率。在开发集上进行参数调优，找到最合适的一组。

4.6 模型改进

这个模型中预测基本只使用了单个的线性层，可以考虑使用更复杂的结构，如多个线性层 + 激活层，或者加入卷积等。或者可以参考图神经网络，加入更合适的结构。

五、反思小结

在这次实验中，学习和实践了结合使用上层的预训练模型（提取语义）和下层的预测模型（推理回答）来进行司法阅读理解。在有限的硬件设施下，使用gradient accumulation的方法，加大了batch size（原本batch size = 1），提高模型的效果。希望之后能有机会尝试改进模型，取得更好的效果。