2023.4.3 - AI小花 - 周周星分享

虎牙181469 · 3 天前

对于脱敏数据，有两种方式：
1）. 直接用开源的pretrain model，例如T5、Bart之类的，因为这些模型的词表都很大，数据里的脱敏id基本都被包含在词表里面；
2）. 则是通过自己创建词表。

在NLP中，模型大体是先通过将文本序列通过词表映射为id序列，然后将id序列通过embedding映射到向量空间中，得到该文本的初始化特征向量表示，然后再送入到我们的模型中进行编码，解码，得到对应的输出。

现在，针对开始说的第二种情况，我们改了词表，相应的encoder的embedding也得修改（resize）为词表的大小。对应decoder的embedding也一样。
改完之后，我们就可以把这个task adapt model保存下来了，形式差不多如下：

...
对应方法需要自己实现（问ChatGPT），或者找到开源代码里相应的地方

build_vocab() # 保存词表
tokenizer = BartTokenizer.from_pretrained('vocab_save_path')
tokenizer.save_pretrained('task_adapt_model')
model.encoder.resize_embedding(tokenizer.vocab_size)
model.decoder.resize_embedding(tokenizer.vocab_size)
model.save_pretrained('task_adapt_model')
...

在我得到了任务适应模型之后，就可以开始对模型进行任务数据的训练了。一般的，因为这个任务的脱敏数据和原始模型训练的数据是不一样的，所以我们需要让模型先适应这个数据。让模型去适应这个数据，则需要对模型进行预训练。

对于预训练，T5，Bart 之类的，都是和Bert差不多的，只不过Bert是一个 token 用一个MASK去替代，生成类的模型，是将一个 span（包含多个 token）用一个 MASK 去替代。在预训练的时候，一般不会用到答案，即本任务的 diag 字段，只需要用到 query 字段。所以，在做预训练任务的时候，例如：query

为 “12 561 61 314 694 315 ... ... 696”，我们先通过 tokenizer，把它变为对应的ids，得到“21 596 155 423 1226 496 ... ... 125”,然后我们通过代码选中一些 token 进行 mask（这里以mlm为例子），把 ids 序列中相应选中位置的地方置为 tokenizer.mask_token_id(假设为3)，得到“21 **3** 155 423 **3** 496 ... ... 125”，就可以了。对应于生成模型的 decoder ids，即 shift right 一下，“2 21 **3** 155 423 **3** 496 ... ... 125”（不明白的同学可以看看 Transformer，为啥要右移一位；一般模型内部会写好操作的，不需要自己生成）。然后对于 labels，即是原始的 ids，“21 596 155 423 1226 496 ... ... 125”，但是一般的，对于预训练任务，我们会把非 mask 的地方都置为-100，因为预训练的目的，只需要让模型去预测被 mask 的地方是什么词（在计算loss的时候，用的CE，可以ignore -100），因此，我们得到了 labels=“-100 596 -100 -100 1226 -100 ... ... -100”。到目前为止，我们的预训练就完成了。

在我们预训练之后，得到了相应的预训练权重，然后通过加载它，得到微调的模型，例如，
model=BartForConditionGeneration.from_pretrained('pretrain_save_path'),
然后开始我们的微调。
微调的话，与预训练不同的地方，就是，input_ids，就是原始的 query（没有mask），
decoder_input_ids 就是右移一位的 ans，labels 就是原始的 ans。把数据造完之后，就可以输入模型进行微调了。
特别地，在微调时，eval 的时候要用model.generate()，不然会导致 offline 特别高。
微调完，进行预测。

对于开始说到的第一种方式，我们就不需要 resize 什么的了，词表还是用的老词表，模型还是用的老模型，然后直接加载它进行数据的适应（预训练），微调即可。

==========

要拉开差距，
第一，是需要对模型做更好的预训练任务，比如上述的生成类型的预训练任务，把 span 用一个 mask 替代，让模型去预测该 mask 是哪个 span（可以理解为让模型拥有了生成能力）。又或者是，把 mlm 改成更负责的 mask 策略，如 ngram-mask，span-mask 等等，这些mask都是一个token对应一个 mask；

学习率，模型中不同的层用不同的学习率，对抗训练（fgm、pgd等等），awp对抗训练，swa（模型权重平均），伪标签迭代（单模型预测测试集，然后带标签的测试集加入到训练集一起重新微调一个模型；或者用多模型预测得到伪标签），一致性损失，标签平滑（哪些id应该赋予更高的权重）；

第三，则是改模型的结构（最难），一般在 NLP 比赛中没什么人能够做到通过改模型结构取胜的。

==========

一些初期的实验记录：

online :

无预训练，加载原始的模型：2.78左右

加预训练：2.88左右

offline :

精调（调model.generate()的参数）： 线下 2.7x -> 2.9

- `input_ids`：用于生成文本的输入序列，可以是一个整数张量或一个列表。
- `max_length`：生成序列的最大长度。
- `min_length`：生成序列的最小长度。
- `do_sample`：是否从模型的输出中随机采样。
- `early_stopping`：是否启用提前停止。如果为 `True`，则一旦模型生成了一个完整的文本序列，就会停止生成。
- `num_beams`：beam search的数量。beam search是一种搜索算法，用于在生成序列时优化模型的预测。
- `temperature`：从模型的输出中抽取样本时使用的温度。较高的温度将导致更多的随机性和多样性，但降低了质量。
- `top_k`：在选择要在下一步中生成的单词时，只考虑概率排名前 `k` 个的单词。
- `top_p`：在选择要在下一步中生成的单词时，只考虑累计概率超过指定概率的最小集合。对于动态调节数量保持优秀性能及生成生动多样性的策略(nucleus sampling)效果显著。
- `repetition_penalty`：一个惩罚系数，用于在生成序列时惩罚重复的单词。
- `length_penalty`：一个长度惩罚系数，用于在生成序列时惩罚过长或过短的文本。
- `no_repeat_ngram_size`：一个整数，用于预防n-gram在生成文本中的重复。

已赞 分享

评论

框框 · 3 天前 · #1楼 回复

男枪哥yyds

抹香鲸7kph · 3 天前 · #2楼

该评论已被用户删除

抹香鲸7kph · 3 天前 · #3楼 回复

男枪哥yyds

玖月初识 · 3 天前 · #4楼 回复

男枪哥yyds

aeeeeeep · 3 天前 · #5楼

回复

男枪哥yyds

独角鲸gqlu · 5 小时前 · #6楼

回复

男枪哥yyds

布式鲸gy4g · 2 小时前 · #7楼

回复

小白有几个问题想问下大佬：词表是一个列表还是字典呢，如果是字典的话键和值都是id嘛，这里有点疑惑。然后使用自己的词表的时候，需要把EOS，CLS等特殊符号也要加进去吗？最后，因为我们用的自己的词表，为什么还要用BartTokenizer.from_pretrained方法呀，可以使用tokenizer.add_tokens嘛？希望得到大佬回复，感谢！

虎牙181469 · 1 小时前 · #8楼

回复

回复 #7 布式鲸gy4g

词表是字典，键值都是id，你可以把这一段脱敏后的文本当做新的语言来对待，你现在就是需要重新去训练一个你自己的生成模型；

要把EOS和CLS加进去；

都可以，主要看你自己做的词典怎么做的，比如Bart的分词器，用的是BPE生成的vocab，得到的东西就用BartTokenizer.from_pretraind()，也可以直接做一个Bert的词表Sub-word，SentencePiece，直接用BertTokenizer加载也一样；

直接在原来的基础上加token，embedding也变大了，重新做词表的目的就是为了让embedding变小，多余的删了（除数字id外的token）。

bigbear · 33 分钟前 · #9楼

回复

大佬，为什么大家都叫你“男枪哥”呀

虎牙181469 · 7 分钟前 · #10楼

回复

回复 #9 bigbear

关注下虎牙181469

发布评论

B I H 插入 帮助

发布