南京農業大學
NANJING AGRICULTURAL UNIVERSITY

数字人文下的模型预训练

目录
CONTENT

- 预训练技术的基本知识
- 预训练方法与评价指标
- 预训练技术在数字人文中的应用背景
- 模型预训练程序

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

预训练技术的基本知识

1、预训练简介

◆早期的自然语言处理：人们通常采用**完全有监督**的方式构建数据集和模型，以期模型充分学习文本数据的语义特征，**人力与时间成本非常昂贵**，且训练出的模型**可移植性较差**。

◆预训练：其思想的本质是**从大规模无监督数据中学习语言共性特征**，再将这些特征转移到相应领域的任务上，这一方法的好处在于有效**降低了对标注数据的依赖**，预训练是解决上述问题的重要举措。

1、预训练简介

◆现在的自然语言处理领域借鉴了预训练思路，其基本手段是使用语言模型来实现语言特征的迁移。

- ●经过预训练的语言模型参数不再是随机初始化，而是更新后适用于相似数据集的。
- ●预训练的做法一般是将大量低成本收集的训练数据放在一起，经过某种预训练方法去学习其中的共性，然后将其中的共性迁移到特定任务的模型中，再使用相关特定领域的少量标注数据进行微调，使模型充分吸收领域数据的共性。

1、预训练简介

◆预训练方式：可以分为AR模型和AE模型两种。

- **AR模型**：基于链式法则计算句子概率；按照写作顺序从左到右计算下一个单词的条件概率；常用于GPT、Transformer等注重NLG（自然语言生成）的模型。
- **AE模型**：通过对已损坏句子的重构来更新模型参数；通过上下文来预测被遮盖的词，充分考虑了上下文信息；主要用于BERT模型的预训练，适合NLU（自然语言理解）任务。

2、预训练的适用范围和工具简介

◆适用范围：适用于精加工的**训练数据缺少、训练数据与预训练数据相似度高**的情况，数据集越小、数据相似程度越高，预训练的结果则更加显著。

◆工具简介：

对于神经网络架构模型（如LSTM,RNN等），通常只需要pytorch框架；

对于transformer结构的模型（如BERT,GPT），通常使用tensorflow或huggingface提供的transformers框架。

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

预训练方法与评价指标

0、语言模型预训练方法

- 在自然语言处理中，预训练被视为一个在特定语料上训练语言模型的过程。从传统机器学习和深度学习的角度上来看，语言模型是对各语言单位分布概率的建模。
- 对于一个给定的语言序 $S = w_1 w_2 \ldots w_{n-1} w_n$，语言模型就是预测该序列的出现的概率 $\mathrm{p}(\mathrm{S}) = \mathrm{P}(w_1, w_{2,\ldots} w_{n-1}, w_n)$ 。这一概率能够采用AE模型和AR模型两种计算方法。

1、n-gram模型

◆使用链式法则可以把语言模型计算过程表示为：

$$P(w_1,.....,w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1,w_2) * ..... * P(w_n|w_1,...,w_{n-1})$$

在此公式中，当前第n个词概率的计算，完全取决于前n-1个词。

该方法的好处在于较为简单，但计算后一个词的概率需完全依赖前面所有词的计算结果，使得在面对长序列时的计算复杂度激增。

解决上述问题可以引入马尔可夫假设进行序列建模：即认为当前字出现的概率仅取决于其前面一个或多个词出现的概率。在此基础上利用n元模型降低计算难度，及在估算条件概率时，忽略与当前词距离大于等于n的词的影响。

1、n-gram模型

◆一元语言模型，即unigram，认为当前词的出现不受周围词的影响，则可以语言序列S出现的概率表示如下：

$$p(S) = P(w_1) * P(w_2) * P(w_3) * ..... * P(w_n)$$

◆二元语言模型，即bigram，认为当前词的出现仅受它前面一个词的影响，则可以语言序列S出现的概率表示如下：

$$p(S) = P(w_1) * P(w_2|w_1) * P(w_3|w_2) * ..... * P(w_n|w_{n-1})$$

N元语言模型，以此类推。

1、n-gram模型

◆n-gram的优点：

（1）采用极大似然估计，参数易训练；

（2）完全包含了前 n-1 个词的全部信息；

（3）可解释性强，直观易理解。

◆n-gram的缺点：

（1）缺乏长距离依赖，只能建模到前 n-1 个词；

（2）随着 n 的增大，参数空间呈指数增长；

（3）数据稀疏，难免会出现未登录词的问题；

（4）单纯的基于统计频次，泛化能力差。

2、神经网络语言模型

◆神经网络语言模型直接通过一个神经网络结构评估n元条件概率。

对于一个给定的词序列S=w_1 w_2 …w_{n-1} w_n，其中w_i属于语料库中全部单词的集合。最终要训练的模型则能表示为：

$$f(w_t, w_{t-1}, w_{t-2}, \ldots w_{t-n+1}) = P(w_t = i | \text{context}) = P(w_t | w_1^{t-1})$$

求解时，模型需要满足以下两个约束条件：

$$f(w_t, w_{t-1}, w_{t-2}, \ldots w_{t-n+1}) > 0$$

$$\sum_i^{|V|} f(w_t, w_{t-1}, w_{t-2}, \ldots w_{t-n+1}) = 1$$

2、神经网络语言模型

◆神经网络语言模型的基本结构分为输入层，隐藏层，输出层三个部分。

◆在输入层首先将序列转化为one-hot编码，继而将这一元素映射到一个实向量中，这一实向量代表词表中第i个词的分布表示。

◆将输入层的向量通过前向传播完成对参数的更新，能够在一定程度上有效解决词袋模型带来的语义差异化和数据稀疏的问题。在相似的上下文语境中，神经网络语言模型可以预测出相似的目标词，这是传统的n元模型所难以实现的。

3、BERT模型

◆BERT的基本结构建立在双向Transformer编码器上，通过掩码语言模型（MLM）和下一句预测（NSP）两个无监督任务完成模型的预训练。

◆BERT模型属于自编码器（AR）语言模型，这种训练策略要求随机遮罩掉一些单词，在训练过程中根据上下文对被遮罩的单词进行预测，使预测概率最大化。

- 在MLM任务中，模型需要按比例随机遮蔽输入序列中的部分字符，根据上下文预测被遮蔽的单词，以完成词汇的深度双向表征的训练，能够同时利用上下文的语境的优势使其非常适合Transformer结构的网络。
- 在NSP任务中，BERT模型成对地读入句子，并判断给定的两个句子是否相邻，从而获得句子之间的关系。利用预训练获得参数可以建立BERT模型的微调过程，仅需模型的高层参数进行调整，即可适应不同的下游任务。

4、语言模型的评价指标

◆对于语言模型的优劣，除了利用下游任务的训练效果来评价之外，还可以针对模型内部的结构来评测，此时一般使用困惑度（PPL，perplexity）来衡量，困惑度的定义如下：

对于一个给定的序列 $S = w_1 w_2 \ldots w_{n-1} w_n$ ，w_n 表示序列中第n个词，则该序列的似然概率定义为：

$$p(S) = P(w_1 w_2 \ldots w_{n-1} w_n)$$

困惑度定义为：

$$PPL = P(w_1 w_2 \ldots w_{n-1} w_n)^{\frac{-1}{n}}$$

困惑度的大小反应了语言模型的好坏，一般情况下，困惑度越低，代表语言模型效果越好。

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

预训练技术在数字人文中的应用背景

1、基于外部词向量嵌入的古文处理

◆早期研究者通常将循环神经网络（RNN）或卷积神经网络（CNN）及其各种变体用于古文文本的处理，虽然深度神经网络的使用避免了复杂的特征工程，并在一定程度上取得超越统计学习模型的效果，但在标注数据集不足的情况仍然难以应对复杂语言环境下语料的处理。

◆该阶段先验知识的融合大多是通过引入外部词向量嵌入来解决的。这些训练好的外部词向量可以根据模型结构在嵌入层使用来初始化，或在隐藏层使用以对特定词汇进行增强。

1、基于外部词向量嵌入的古文处理

◆训练外部词向量的方式多种多样，其中word2vec以适中的复杂度、较快的训练速度和较佳的质量被大量用于深度学习模型之中。相关研究如王一钒等以word2vec中的CBOW方法训练古文词向量作为BiGRU-CRF的嵌入层初始参数，并应用于古文实体关系联合抽取。王莉军 将古文词向量作为BiLSTM-CRF的嵌入层，用以增强模型对中医古籍文本的分词性能。崔丹丹　利用“甲言”分词工具对《四库全书》进行分词，并将训练的词向量与Lattice-LSTM相结合，有效提升了古汉语命名实体识别的性能。

2、基于已有预训练模型微调的古文处理

◆以BERT为代表的大规模预训练语言模型的应用深刻地改变了古文智能处理地进程，“预训练+微调”处理模式被称为NLP第三范式。但由于BERT模型的预训练成本较大，多数数字人文研究者不具有足够的算力资源，对预训练模型的应用通常只保留微调这一过程。

◆部分研究如杜悦以BERT等模型对先秦典籍中构成历史的事件的实体进行实体识别。喻雪寒 等基于BERT,Roberta-CRF和GuwenBERT三种预训练模型从《左传》数据集中抽取战争事件。刘忠宝 基于BERT、BERT-Bilstm-crf模型标注《史记》中的重要历史事件，并以此为基础构建事理图谱。张琪 等基于BERT构建用于古文分词和词性标注的一体化模型，并将其应用于先秦典籍地词性标注中。

3、基于语言模型预训练的古文处理

◆另一些研究则从上游任务入手，通过对大规模语言模型的预训练实现古汉语语言特征的迁移。此类研究通常使用BERT类模型作为baseline,不同预训练语料的选取决定了这些模型不同的应用环境。

◆例如，日本京都大学学者KoichiYasuoka 基于guwenBERT继续训练，以繁体语料对词表进行了扩充，训练出能同时用于简体和繁体中文处理的roberta-classical-chinese-base-char模型，能较好地应对简繁混杂的情况。王东波 等基于繁体中文的文渊阁版本《四库全书》以BERT和Roberta模型为基础训练出适用于繁体古籍处理的古文预训练模型，更好地贴合文学、历史学等学科学者的语料处理需求。

3、基于语言模型预训练的古文处理

互联网已开源的古文预训练模型链接:

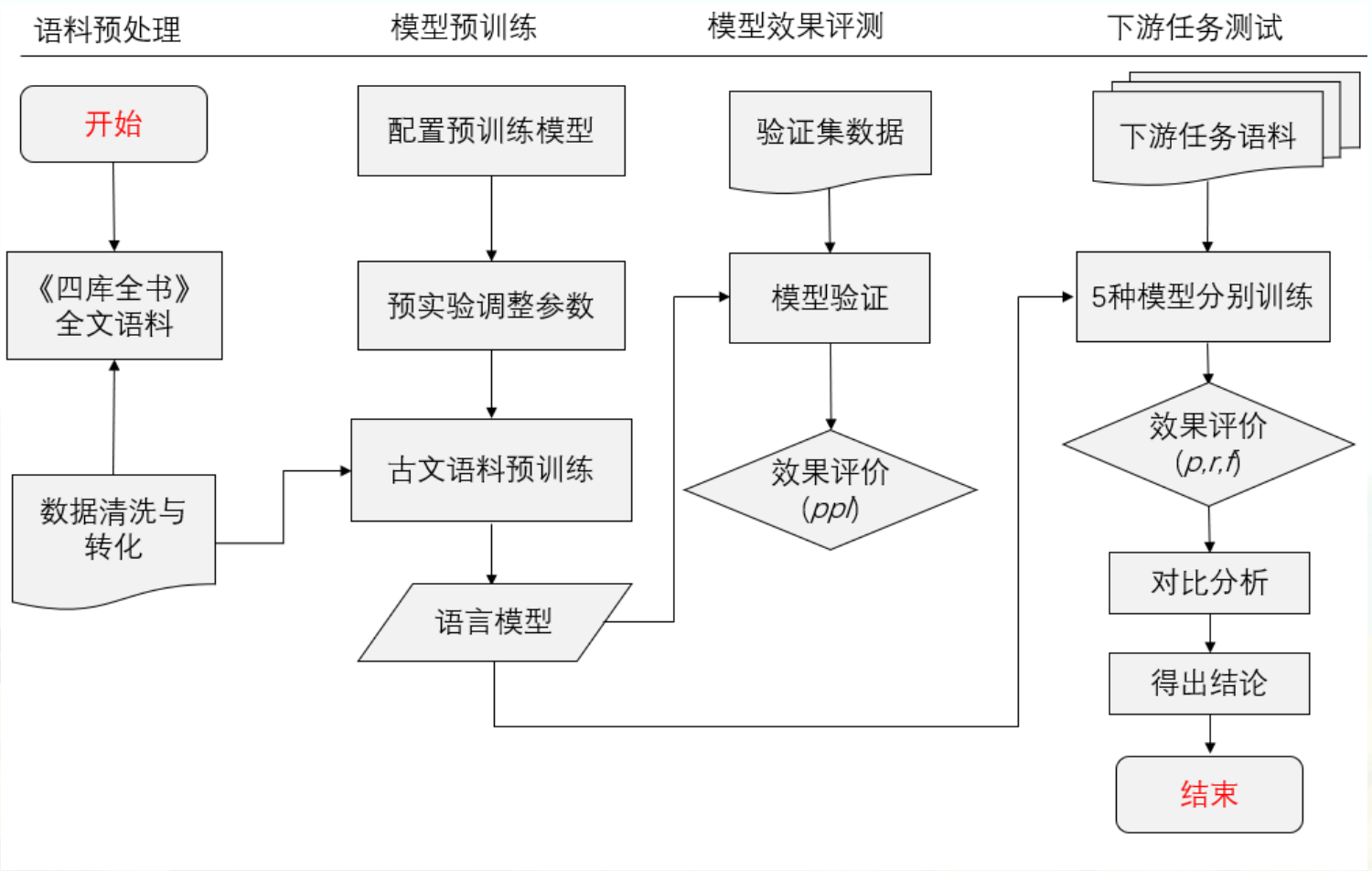
- GuwenBERT（适用于简体）：https://github.com/Ethan-yt/guwenbert
- roberta-classical-chinese-base-char（简繁均可）：https://huggingface.co/KoichiYasuoka/roberta-classical-chinese-base-char
- SikuBERT（适用于繁体）：https://github.com/hsc748NLP/SikuBERT-for-digital-humanities-and-classical-Chinese-information-processing

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

模型预训练程序

0、语言模型预训练实验流程

◆全套的语言模型预训练构建实验包含上游任务训练和下游任务的效果测试两个过程，以我们的SikuBERT预训练模型为例，下图展示了实验的全部流程。



1、环境配置

- 目前模型预训练的两大主流框架是pytorch和tensorflow。
- Pytorch框架的优点在于轻量级、资源占用量低和维护性强，是深受学术界欢迎的框架，但仍有代码不直观，不易读的缺点。
- Tensorflow框架封装了大量的接口，虽然这些接口调用简单，但代码的冗余度却更高，且用户需要花费大量时间去记忆不同的接口，因此多适用于工业界。

2、程序内容

◆语言模型的预训练可以参考transformers官方的示例和开源的github项目。transformers官方示例网址为：

- https://github.com/huggingface/transformers/tree/v3.4.0/examples/language-modeling
- 基于该项目改进的使用于中文模型预训练的代码见以下链接:
- https://github.com/zhusleep/pytorch_chinese_lm_pretrain

2、程序内容

●可调用参数及参数功能说明：

参数名	参数作用
model_name_or_path	指示需要预训练的模型名称和路径，以及是否从头训练
model_type	预训练模型的种类，可选则transformers支持的所有模型，目前已经可以使用二十余种
config_name	如果config_name的位置与model_name_or_path不同，则将config_name参数赋值，否则不赋值
tokenizer_name	关于分词器的路径，用于文本切分
train_data_file	训练数据路径
eval_data_file	验证数据路径
line_by_line	是否将数据集中不同的文本行作为不同的序列处理
mlm	即掩码语言模型，预训练操作的核心，训练BERT类模型必须选取此参数
mlm_probability	在掩码语言模型中，一个序列被遮罩的字符数和全部字符数的比率，通常为15%
plm_probability	排列语言模型的掩码标记的跨度与周围的上下文长度的比率
max_span_length	掩码标记的最大跨度，作用在于限定连续遮罩的最大词长，值为5说明一个句子最多连续遮罩5个字符
block_size	输入模型最大的句子长度，当为默认值-1时代表以整个句子做为输入
overwrite_cache	覆盖缓存的训练和测试集文件，可用于程序调试

2、程序内容

●导入相关模块：

```
import logging   # 展示程序运行的过程
import math
import os
from dataclasses import dataclass, field
from typing import Optional
from transformers import BertTokenizer, BertForMaskedLM, BertConfig, BertLMHeadModel
#导入分词器、MLM、配置文件等接口
from transformers import ( CONFIG_MAPPING, MODEL_WITH_LM_HEAD_MAPPING,AutoConfig,
AutoModelWithLMHead,AutoTokenizer, DataCollatorForLanguageModeling, HfArgumentParser,
LineByLineTextDataset, PreTrainedTokenizer, TextDataset,
Trainer, TrainingArguments, set_seed)
# AutoConfig,AutoModelWithLMHead,AutoTokenizer三个方法可以自动指定模型的参数
# TextDataset方法是用于文本处理的方法，导入的目的在于统一数据集的输入格式
# Trainer类是整个程序的核心，所有与训练有关的参数都可以赋值给这个类，随后再通过train（）
方法实现训练
```

2、程序内容

●数据集读取：

```
def get_dataset ( args: DataTrainingArguments, tokenizer: PreTrainedTokenizer, evaluate=False) :
file_path = args.eval_data_file if evaluate else args.train_data_file
    if args.line_by_line:
        return LineByLineTextDataset ( tokenizer=tokenizer, file_path=file_path, block_size=args.block_size)
#从训练与测试集中一行一行的读入数据，并基于BERT自带的tokenizer对读入的序列进行分词
    else:
        return TextDataset (
            tokenizer=tokenizer, file_path=file_path, block_size=args.block_size, overwrite_cache=args.overwrite_cache)
```

2、程序内容

●加载模型基本参数，参见本仓库。

●整合参数至训练器：

```
# 初始化训练器
  trainer = Trainer（model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    prediction_loss_only=True,）
  #Trainer类中包含与训练相关的各种参数，以上的参数是类中参数的一部分，还有一些参数仅使用默认值，例如分布式计算功能。
```

2、程序内容

●训练：

```
    # 训练部分
#do_train操作的实现逻辑
    if training_args.do_train:
model_path = (model_args.model_name_or_path
            if  model_args.model_name_or_path  is  not  None  and  os.path.  isdir
 (model_args.model_name_or_path)
            else None)  #调用训练方法实现训练
trainer.train (model_path=model_path)
trainer.save_model ()
if trainer.is_world_master () :#存储训练后的模型
tokenizer.save_pretrained (training_args.output_dir)
```

2、程序内容

- 验证部分，以困惑度作为评价指标判断训练出语言模型的好坏，参见本仓库。
- 程序运行的命令，将运行指令写入sh文件中，运行命令sh xxx. sh，参见本仓库。
- 如果要从头训练，只需修改部分参数。从头训练需要的语料数量较大，且计算资源开销更大，建议从头训练前应估算训练成本，参见本仓库。

2、预训练模型训练和使用的注意事项

◆在模型的预训练和下游任务的使用中的常见错误

●预训练与下游任务使用的pytorch版本不一致。

使用较高版本的pytorch（1.6版本及以上）训练出的预训练模型，如果在较低版本的环境中加载时会出现如下报错。

```
pytorch_model.bin is a zip archive (did you mean to use torch.jit.load() ?)
```

这是因为pytorch1.6及之后,更换了保存模型文件的方式。默认使用zip文件格式来保存权重文件，导致这些权重文件无法直接被1.5及以下的pytorch加载。解决此问题需要使用如下代码重新加载模型并保存。

2、预训练模型训练和使用的注意事项

●强行将pytorch训练出的模型和tensorflow训练出的模型互相转换导致模型参数转化异常。

目前，主要的开源代码是由tensorflow和pytorch两种框架分别编写的，实现具体任务的代码可能只有pytorch版或tensorflow版，对于基于tensorflow的代码来说，必然不能直接使用pytorch版本的模型用于训练。pytorch版本的BERT模型转tensorflow模型可以参huggingface官方提供的代码进行转换，官方链接如下：

https://github.com/huggingface/transformers/blob/master/src/transformers/models/bert/convert_bert_pytorch_checkpoint_to_original_tf.py