基于大语言模型的桥梁设计规范问答系统

张乐业[1]，田翔翔[1]，张洪俊[2]

（1. 江苏财会职业学院，连云港 222061；

2. 万世先行数智交通科技有限公司，南京 210016）

摘要：本文构建了基于大语言模型的桥梁设计规范问答系统。尝试了三种实现方案：对 BERT 预训练模型进行全参数微调、对 BERT 预训练模型进行参数高效微调、从零开始自建语言模型。通过自建的问答任务数据集，基于 TensorFlow 及 Keras 深度学习平台框架，构建和训练模型，预测答案在用户给定的桥梁设计规范中的开始位置和结束位置。实验结果显示：BERT 预训练模型全参数微调方案在训练集、验证集和测试集上均达到了 100%的精度，系统能够从用户给定的桥梁设计规范中抽取出答案，以回答用户的各种提问；而 BERT 预训练模型参数高效微调方案和自建语言模型方案虽然在训练集上表现良好，但在测试集上的泛化能力有待提高。本文的研究为专业领域问答系统的发展提供了有益参考。

关键词：问答系统；大语言模型；桥梁设计规范；预训练模型；全参数微调；参数高效微调

中图分类号：TP391；U442.5　　**文献标志码：**A

Question answering system of bridge design specification based on large language model

Leye Zhang[1]，Xiangxiang Tian[1]，Hongjun Zhang[2]

（1. Jiangsu College of Finance & Accounting, Lianyungang, 222061, China;

2. Wanshi Antecedence Digital Intelligence Traffic Technology Co., Ltd, Nanjing, 210016, China）

Abstract: This paper constructs question answering system for bridge design specification based on large language model. Three implementation schemes are tried: full fine-tuning of the Bert pretrained model, parameter-efficient fine-tuning of the Bert pretrained model, and self-built language model from scratch. Through the self-built question and answer task dataset, based on the tensorflow and keras deep learning platform framework, the model is constructed and trained to predict the start position and end position of the answer in the bridge design specification given by the user. The experimental results show that full fine-tuning of the Bert pretrained model achieves 100% accuracy in the training-dataset, validation-dataset and test-dataset, and the system can extract the answers from the bridge design specification given by the user to answer various questions of the user; While parameter-efficient fine-tuning of the Bert pretrained model and self-built language model from scratch perform well in the training-dataset, their generalization ability in the test-dataset needs to be improved. The research of this paper provides a useful reference for the development of question answering system in professional field.

Keywords: question answering system; large language model; bridge design specification; pretrained model; full fine-tuning; parameter-efficient fine-tuning

0　引言

在工作中，专业人士经常需要在文献中查询信息。最常见的关键词搜索技术（Keyword Search），存在信息丢失、返回信息太多、信息无关等局限性[1]。传统的基于结构化数据或者基于常用提问集的问答系统（Question Answering System, QA），依赖于规则、模板或者有限的内置问题答案集合，难以应对复杂多变的用户需求，且开发维护成本高。近几年大语言模型（Large Language Model，LLM）的飞速发展，为基于自然语言处理（Natural Language Processing，NLP）的问答系统带来了新生，有望成为新一代搜索引擎。

ChatGPT 等生成式预训练模型，拥有语言理解和文本生成能力，能够回答用户的提问，但是生成结果会存在幻觉和偏差等现象，需要进一步微调才能用于专业领域。目前许多专业领域，已在积极开展基于大语言模型问答系统的研究。张金营基于大语言模型，构建电力知识库智能问答系统[2]；张春红研究了基于大语言模型的教育问答系统

作者简介：张乐业（2003—），男，2023 级人工智能技术应用学生

通信作者：田翔翔（1991—），男，讲师，主要研究方向：视觉传达设计，E-mail：3514956127@qq.com

[3]；雷天凤基于大语言模型，研究了竞品车型配置问答系统[4]；王婷基于大语言模型，构建果蔬农技知识问答系统[5]；丁志坤基于大语言模型，设计了 BIM 正向设计问答系统[6]；MM Lucas 提出新的提示词（prompt）方法，以提高基于大语言模型的医学问答系统推理能力[7]；Wonjin Yoon 研究了基于大语言模型的生物医学问答系统[8]。而大语言模型应用于桥梁设计领域问答系统的研究，尚未见报道。

本文自建问答任务数据集，采用对 BERT 预训练模型进行全参数微调（Full Fine-Tuning，FFT）、对 BERT 预训练模型进行参数高效微调（Parameter-Efficient Fine-Tuning，PEFT）、从零开始自建语言模型三种方案，实现桥梁设计规范问答系统。模型能够从用户事先给定的桥梁设计规范中抽取出答案，来回答用户各种提问（本文数据集与源代码开源地址 https://github.com/ zhangleye/Bridge-LLM-QA）。

1　大语言模型和问答系统简介

1.1 大语言模型

1.大语言模型基于 Transformer 架构，使用大量文本数据训练，可以理解和生成自然语言文本[9]。它表现出惊人的通用性，可以处理多种自然语言任务，如文本分类、问答、对话等，被视为实现通用人工智能（Artificial General Intelligence，AGI）的可能途径。但是目前技术水平离真正的通用人工智能仍有不小的距离。

目前 AI 面对行业、业务场景多和需求呈现碎片化、多样化的特点，一般采用“预训练大模型+下游任务微调”的生产方式。经过预训练（自监督学习），大模型获得完成各种任务的通用能力；然后微调（监督学习），以适应特定任务。这与目前人类大学人才培养方案有点类似，前期先学习公共基础课程，后期再学习专业技能课程。

2.受硬件性能制约，本文采用规模较小的 BERT-base-chinese 预训练模型，模型配置为 L=12（Transformer 编码器串联数目）、A=12（Transformer 编码器通道数目）、H=768（词嵌入空间维数），网络参数总量为 1.1 亿个[10]。词汇表有 7 千多个汉字，满足本次研究要求。

BERT 模型在一个巨大的语料库上，针对两种特定的任务进行预训练。这两种任务是掩码语言和下句预测。 BERT 模型用于实际任务时，仅额外添加特定任务的输出层，然后微调即可，非常简单。

3.预训练大模型常见加载方式有：通过 PyTorch、TensorFlow 和 Hugging Face 的 Transformers 代码加载。

1.2 问答系统

问答系统是信息检索系统的一种高级形式，其研究兴起的主要原因是人们对快速、准确地获取信息的需求。问答系统是人工智能和自然语言处理领域中一个倍受关注并具有广泛发展前景的研究方向。目前代表性的技术路线有：基于结构化数据的问答系统、基于常用提问集的问答系统和基于自然语言处理的问答系统等[11]。

基于自然语言处理的问答系统，可以直接从语言模型中生成文本来回答（开放式的，可能没有标准答案），也可以从互联网或者阅读材料中查找答案（摘要式或者抽取式）。对于专业领域，当前技术水平下，抽取式较为适合，它从给定的文档资料中抽取文字片段来回答（如同考生在做语文或者英语的阅读理解题目，考生领悟给定的阅读材料，依据阅读材料具体细节来回答提问）。

2　软硬件环境和自建问答任务数据集

2.1 软硬件环境

操作系统为 Windows 11（用于 TensorFlow 2.10、Keras 2.10）和 WSL Ubuntu 22.04（用于 TensorFlow 2.16.1、Keras 3.4.0、Keras_nlp 0.12.1），CUDA Version 12.3。CPU 为 AMD Ryzen 5 2600，内存 64G DDR4，显卡为 RTX 4060 Ti 16G。未使用云计算。

2.2 自建问答任务数据集

目前国内现行的公路、市政桥梁设计规范有十几本，受成本制约，本文仅取《公路桥涵设计通用规范》（JTG D60-2015）[12]第 3 章的文字内容，制作问答任务数据集。

数据集采用 xlsx 文件格式，每行为一个样本，共有 226 个样本。表头为：context、question、answer、index_of_answer。其中 context 为给定的规范文字内容（阅读理解材料），question 为问题，answer 为标准答案（从 context 中抽取的文字片段），index_of_answer 为标准答案索引号（因为标准答案文字片段在 context 中会多次出现，但是仅有一处是符合要求的，所以使用该参数定位）。数据集具体内容如下：

表 1 问答任务数据集

Tab.1 Dataset of question answering

	A	B	C	D
1	context	question	answer	index_of_answer
2	3.1.1 公路桥涵应根据公路功能和技术等级，考虑因地制宜、就地取材、便于施工和养护等因素进行总体设计，在设计使用年限内应满足规定的正常交通荷载通行的需要。 3.1.2公路桥涵线形设计应符合下列规定： 1中小桥涵线形设计应符合路线设计的总体要求。 2特大、大桥线形设计应综合考虑路线总体走向、桥区地质、地形、安全通行、通航、已有建筑设施、环境敏感区等因素。 3特大、大桥宜采用较高的平曲线指标，纵断面不宜设计成平坡或凹曲线。 3.1. 3公路桥涵结构应按承载能力极限状态和正常使用极限状态进行设计。	公路桥涵总体设计时应考虑哪些因素?	因地制宜、就地取材、便于施工和养护等	0
3	3.4.2桥面人行道、自行车道和拦护设施的布置应符合下列规定： 1高速公路上的桥梁不宜设人行道。一、二、三、四级公路上桥梁的桥上人行道和自行车道的设置，应根据需要而定，并应与前后路线布置协调。人行道、自行车道与行车道之间，应设护栏或路缘石等分隔设施。一个自行车道的宽度应为1.0m;当单独设置自行车道时，不宜小于两个自行车道的宽度。人行道的宽度宜为1.0m;大于1.0m时，按0.5m的级差增加。漫水桥和过水路面可不设人行道。 2通行拖拉机或畜力车为主的慢行道，其宽度应根据当地行驶拖拉机或畜力车车型及交通量而定；当沿桥梁一侧设置时，不应小于双向行驶要求的宽度。 3桥梁护栏设置应符合现行《公路交通安全设施设计规范》(JTG D81)的相关规定。 4路缘石高度可取用0.25~0.35m。当跨越急流、大河、深谷、重要道路、铁路、主要航道，或桥面常有积雪、结冰时，其路缘石高度宜取用较大值。	桥面人行道的宽度宜为多少?	1.0m	1
4	3. 4. 4涵洞宜设计为无压力式的。无压力式涵洞内顶点至洞内设计洪水频率标准水位的净高应符合表3.4.4的规定。 3.4.5立体交叉跨线桥桥下净空应符合下列规定： 1公路与公路立体交叉的跨线桥桥下净空及布孔除应符合本规范第3.4.1条桥涵净空的规定外，尚应满足桥下公路的视距和前方信息识别的要求，其结构形式应与周围环境相协调。 2铁路从公路上跨越通过时，其跨线桥桥下净空及布孔除应符合本规范第3.4.1条桥涵净空的规定外，尚应满足桥下公路的视距和前方信息识别的要求。 3农村道路与公路立体交叉的跨线桥桥下净空为： 1)当农村道路从公路上面跨越时，跨线桥桥下净空应符合本规范第3. 4. 1条建筑限界的规定； 2)当农村道路从公路下面穿过时，其净空可根据当地通行的车辆和交叉情况而定，人行通道的净高应大于或等于2.2m,净宽应大于或等于4.0m; 3)畜力车及拖拉机通道的净高应大于或等于2.7m,净宽应大于或等于4.0m;	跨线桥桥下畜力车及拖拉机通道的净宽应大于或等于多少?	4.0m	1

因为本文采用的 bert 预训练模型，最大输入长度是 512 个 token（含标点符号、括号、小数点等），用户可用 token 为 509 个（扣除[CLS]、[SEP]标记），故人为设定 context 不超过 474 个 token、question 不超过 35 个 token。

3 桥梁设计规范问答系统实现方案一： 对 BERT 预训练模型进行全参数微调

3.1 下载词汇表和创建分词器

从 Hugging Face 的 Transformers 库中下载预训练模型 BERT-base-chinese 的词汇表，然后创建分词器 tokenizer。一个汉字就是一个 token。

3.2 加载数据集和数据预处理

采用 pandas 模块的 read_excel()函数读取数据集。将数据集随机划分为训练集（200 个样本）、验证集（20 个样本）和测试集（6 个样本）。

然后将文本格式转换成 BERT 模型指定的整数格式向量。每个样本由样本输入 X 和标签 Y 组成。样本输入 X 由"input_ids"、"token_type_ids"、"attention_mask"三部分组成。标签 Y 由"start_token_idx"、"end_token_idx"两部分组成。

其中："input_ids"由样本的 context、question 拼接而成；"token_type_ids"为分段标记，context 取 0、question 取 1、尾部填充部分取 0；"attention_mask"尾部填充部分取 0，其它为 1；"start_token_idx"为样本 answer 在 context 中的起始位置索引号；"end_token_idx"为样本 answer 在 context 中的结束位置索引号。

3.3 模型构建

在问答任务中，给定一个上下文和一个问题，模型需要从上下文中找到与问题相关的答案。这需要在 BERT 模型的基础上添加了一个用于问答任务的头部模型（输出层）。该头部模型包含 2 个平行的 Dense 层(softmax 激活)，用于对 BERT 模型的输出进行答案的开始位置和结束位置的预测。

这里从 Hugging Face 的 Transformers 库中下载预训练模型 BERT-base-chinese。如果从 Transformers 库中下载的是基于 BERT 的问答系统模型，那么原模型已含有头部模型，此时直接微调即可。

基于 Python3.10 编程语言、TensorFlow2.10 及 Keras2.10 深度学习平台框架，构建模型，代码见下图：

```python
def create_model():
    encoder = TFBertModel.from_pretrained("bert-base-chinese")
    input_ids = layers.Input(shape=(max_len,), dtype=tf.int32)
    token_type_ids = layers.Input(shape=(max_len,), dtype=tf.int32)
    attention_mask = layers.Input(shape=(max_len,), dtype=tf.int32)
    embedding = encoder(input_ids, token_type_ids=token_type_ids, attention_mask=attention_mask)[0]
    start_logits = layers.Dense(1, name="start_logit", use_bias=False)(embedding)
    start_logits = layers.Flatten()(start_logits)
    end_logits = layers.Dense(1, name="end_logit", use_bias=False)(embedding)
    end_logits = layers.Flatten()(end_logits)
    start_probs = layers.Activation(keras.activations.softmax)(start_logits)
    end_probs = layers.Activation(keras.activations.softmax)(end_logits)
    model = keras.Model( inputs=[input_ids, token_type_ids, attention_mask], outputs=[start_probs, end_probs] )
    loss = keras.losses.SparseCategoricalCrossentropy(from_logits=False)
    optimizer = keras.optimizers.Adam(lr=learning_rate)
    model.compile(optimizer=optimizer, loss=[loss, loss], metrics=['accuracy'])
    return model
```

图 1 方案一模型代码

Fig. 1 Code for model of scheme one

模型摘要见下表：

表 2 方案一模型摘要

Tab.2 Model summary of scheme one

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 512)]	0	[]
input_3 (InputLayer)	[(None, 512)]	0	[]
input_2 (InputLayer)	[(None, 512)]	0	[]
tf_bert_model (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 512,768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	102267648	['input_1[0][0]', 'input_3[0][0]', 'input_2[0][0]']
start_logit (Dense)	(None, 512, 1)	768	['tf_bert_model[0][0]']
end_logit (Dense)	(None, 512, 1)	768	['tf_bert_model[0][0]']
flatten (Flatten)	(None, 512)	0	['start_logit[0][0]']
flatten_1 (Flatten)	(None, 512)	0	['end_logit[0][0]']
activation (Activation)	(None, 512)	0	['flatten[0][0]']
activation_1 (Activation)	(None, 512)	0	['flatten_1[0][0]']
Total params: 102,269,184			
Trainable params: 102,269,184			
Non-trainable params: 0			

损失函数采用稀疏多分类交叉熵（SparseCategoricalCrossentropy）。

3.4 微调和测试

模型接收上下文和问题作为输入，模型输出是答案开始位置和结束位置的预测。通过最小化预测结果与标签 Y 的差异，来优化 BERT 基础模型、头部模型的权重参数。采用“Adam” 优化器更新神经网络参数，监控指标为精度（accuracy）。使用回调函数保存每轮模型权重。

微调过程的损失、精度变化曲线见下图（前 28 轮）：

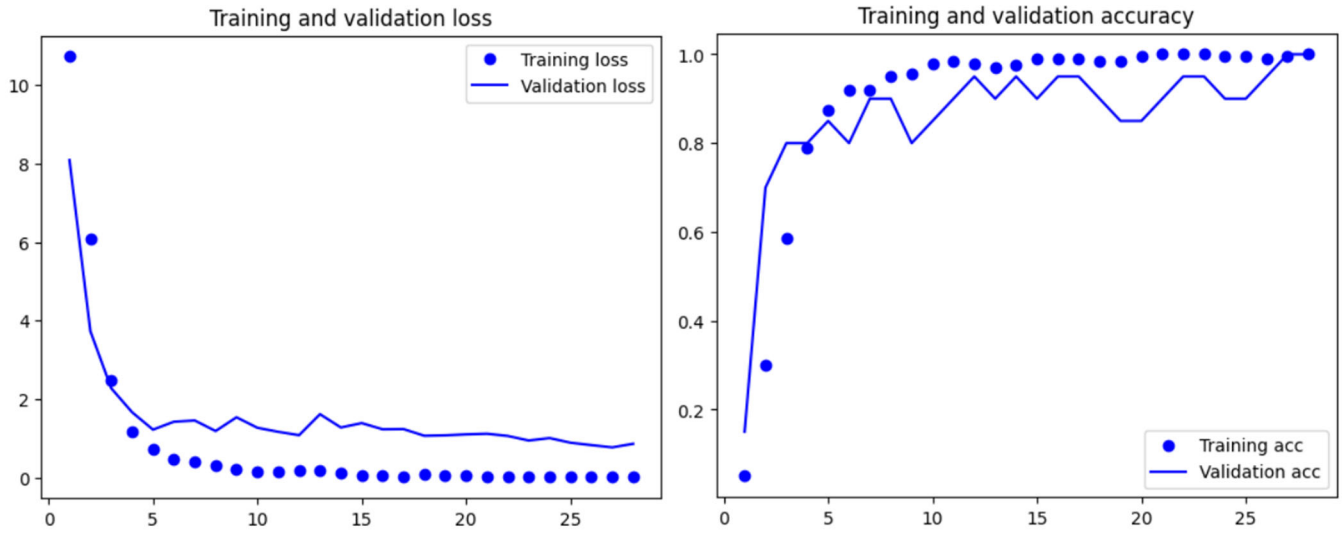

图 2 方案一微调损失和精度曲线

Fig. 2 Fine tuning loss and accuracy curves of scheme one

经过 28 轮的微调，模型在训练集、验证集、测试集上的精度均为 100%。可见本方案的性能非常好。

经过测试，微调之前将 BERT 基础模型冻结，仅微调头部模型，性能不佳。

4 桥梁设计规范问答系统实现方案二： 对 BERT 预训练模型进行参数高效微调

与全参数微调不同，参数高效微调只训练模型的一小部分参数。这里采用 Lora 微调。Lora 的思想很简单，即在原始预训练模型旁边增加一个旁路，执行降维再升维的操作。

4.1 下载词汇表和创建分词器

同方案一。

4.2 加载数据集和数据预处理

同方案一。

4.3 模型构建

这里从 KerasNLP 库中下载预训练模型 BERT-base-chinese，同时调用 enable_lora()方法冻结 BERT 基础模型的所有权重，建立 Lora 旁路。头部模型同方案一。

基于 Python3.11 编程语言、TensorFlow2.16.1 及 Keras3.4 深度学习平台框架，构建模型，代码见下图：

```
def create_model():
    encoder = keras_nlp.models.Backbone.from_preset("bert_base_zh", load_weights=True)
    encoder.enable_lora(rank=16) #rank=4、8、16
    input_ids = layers.Input(shape=(max_len,), dtype=tf.int32)
    token_type_ids = layers.Input(shape=(max_len,), dtype=tf.int32)
    attention_mask = layers.Input(shape=(max_len,), dtype=tf.int32)
    embedding = encoder( {"token_ids":input_ids, "segment_ids":token_type_ids, "padding_mask":attention_mask} )
    start_logits = layers.Dense(1, name="start_logit", use_bias=False)(embedding['sequence_output'])
    start_logits = layers.Flatten()(start_logits)
    end_logits = layers.Dense(1, name="end_logit", use_bias=False)(embedding['sequence_output'])
    end_logits = layers.Flatten()(end_logits)
    start_probs = layers.Activation(keras.activations.softmax)(start_logits)
    end_probs = layers.Activation(keras.activations.softmax)(end_logits)
    model = keras.Model( inputs=[input_ids, token_type_ids, attention_mask], outputs=[start_probs, end_probs] )
    loss = keras.losses.SparseCategoricalCrossentropy(from_logits=False)
    optimizer = keras.optimizers.Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss=[loss, loss],metrics=['accuracy','accuracy'] )
    return model
```

图 3 方案二模型代码

Fig. 3 Code for model of scheme two

模型摘要见下表：

表 3 方案二模型摘要

Tab.3 Model summary of scheme two

Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 512)	0	-
input_layer_1 (InputLayer)	(None, 512)	0	-
input_layer (InputLayer)	(None, 512)	0	-
bert_backbone (BertBackbone)	[(None, 768), (None, 512, 768)]	105,831,1…	input_layer_2[0]… input_layer_1[0]… input_layer[0][0]
start_logit (Dense)	(None, 512, 1)	768	bert_backbone[0]…
end_logit (Dense)	(None, 512, 1)	768	bert_backbone[0]…
flatten (Flatten)	(None, 512)	0	start_logit[0][0]
flatten_1 (Flatten)	(None, 512)	0	end_logit[0][0]
activation (Activation)	(None, 512)	0	flatten[0][0]
activation_1 (Activation)	(None, 512)	0	flatten_1[0][0]

Total params: 105,832,704 (403.72 MB)
Trainable params: 3,583,488 (13.67 MB)
Non-trainable params: 102,249,216 (390.05 MB)

损失函数采用稀疏多分类交叉熵（SparseCategoricalCrossentropy）。

4.4 微调和测试

采用“Adam” 优化器更新 Lora 模型、头部模型的神经网络参数，监控指标为精度（accuracy）。使用回调函数保存每轮模型权重。

微调过程的损失、精度变化曲线见下图（前 25 轮）：

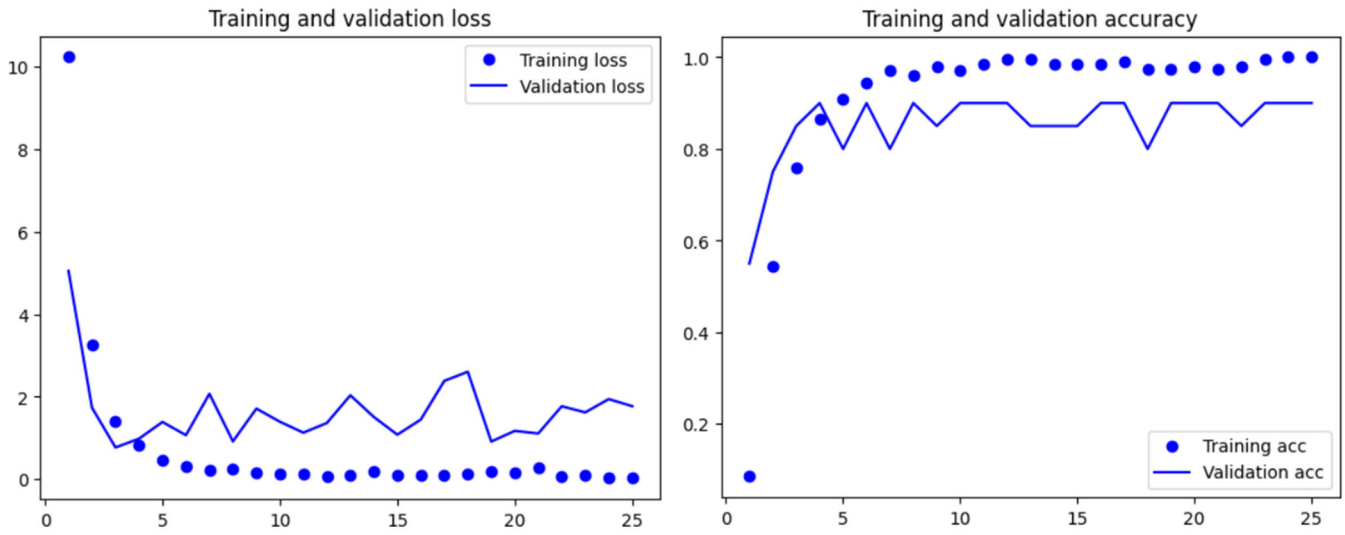

图 4 方案二微调损失和精度曲线

Fig. 4 Fine tuning loss and accuracy curves of scheme two

经过 25 轮的微调，模型在训练集、验证集、测试集上的精度分别为 100%、90%、67%。可见本方案的泛化性能不佳，这与数据集规模太小、微调方式有关。

5 桥梁设计规范问答系统实现方案三：从零开始自建语言模型

大模型如果不进行预训练，而是直接在最终任务数据集上训练，会缺乏泛化性能。原因是监督学习的数据集制作成本高昂，通常样本数量非常少，故无法满足大模型的学习要求。这时，预训练就派上了用场，它先在一个庞大而低廉的数据集上进行自监督学习，然后将学到的通用知识转移到最终任务上。

这里从零开始自建 Transformer 语言模型，然后采用一个中等规模的无标签中文新闻数据集，进行预训练，最后将其在问答任务数据集上微调。

5.1 下载预训练数据集

在 https://huggingface.co/datasets/YeungNLP/firefly-pretrain-dataset 下载 CNewSum_v2.jsonl（新闻数据集，275596 个样本，用于预训练）、prose.jsonl（散文数据集，658 个样本，用于验证）。

5.2 下载词汇表和创建分词器

同方案一。

5.3 预训练任务

采用掩码语言任务进行预训练，即样本随机掩盖 15%的单词，训练模型来预测被掩盖的单词。这相当于完形填空任务。损失函数采用稀疏多分类交叉熵（SparseCategoricalCrossentropy）。

5.4 加载预训练数据集和数据预处理

采用 pandas 模块的 read_json ()函数读取预训练数据集。然后将文本格式转换成整数格式向量。每个样本由样本输入 X 和标签 Y 组成。样本输入 X 由"token_ids"、"mask_positions"二部分组成。标签 Y 由"mask_ids"、"mask_weights"两部分组成。

其中："token_ids"为样本经过掩码后的整数序列；"mask_positions"为被掩码的位置；"mask_ids"为掩码位置上的原始 token id；"mask_weights"为标记，尾部填充部分取 0，其它为 1。

5.5 Transformer 语言模型构建

基于 Python3.11 编程语言、TensorFlow2.16.1 及 Keras3.4 深度学习平台框架，构建模型。预训练样本（一维向量），首先经过词嵌入空间 Embedding 层（整合了位置嵌入），转变为序列（二维矩阵）。然后经过 3 个 Transformer 编码器，实现上下文感知。具体代码见下图：

```
inputs = keras.Input(shape=(SEQ_LENGTH,), dtype="int32")
embedding_layer = keras_nlp.layers.TokenAndPositionEmbedding(
    vocabulary_size=tokenizer.vocabulary_size(),
    sequence_length=SEQ_LENGTH,
    embedding_dim=MODEL_DIM, )
outputs = embedding_layer(inputs)
outputs = keras.layers.LayerNormalization(epsilon=NORM_EPSILON)(outputs)
outputs = keras.layers.Dropout(rate=DROPOUT)(outputs)
for i in range(NUM_LAYERS):
    outputs = keras_nlp.layers.TransformerEncoder(
        intermediate_dim=INTERMEDIATE_DIM,
        num_heads=NUM_HEADS,
        dropout=DROPOUT,
        layer_norm_epsilon=NORM_EPSILON,
    )(outputs)
encoder_model = keras.Model(inputs, outputs)
```

图 5 方案三模型代码

Fig.5 Code for model of scheme three

模型摘要见下表：

表 4 方案三模型摘要

Tab.4 Model summary of scheme three

Layer (type)	Output Shape	Param #
input_layer_5 (InputLayer)	(None, 512)	0
token_and_position_embedding_4 (TokenAndPositionEmbedding)	(None, 512, 512)	11,079,680
layer_normalization_4 (LayerNormalization)	(None, 512, 512)	1,024
dropout_16 (Dropout)	(None, 512, 512)	0
transformer_encoder_12 (TransformerEncoder)	(None, 512, 512)	2,078,172
transformer_encoder_13 (TransformerEncoder)	(None, 512, 512)	2,078,172
transformer_encoder_14 (TransformerEncoder)	(None, 512, 512)	2,078,172

Total params: 17,315,220 (66.05 MB)
Trainable params: 17,315,220 (66.05 MB)
Non-trainable params: 0 (0.00 B)

5.6 预训练

采用“Adam” 优化器更新神经网络参数，监控指标为精度（accuracy）。

预训练过程的损失、精度变化曲线见下图：

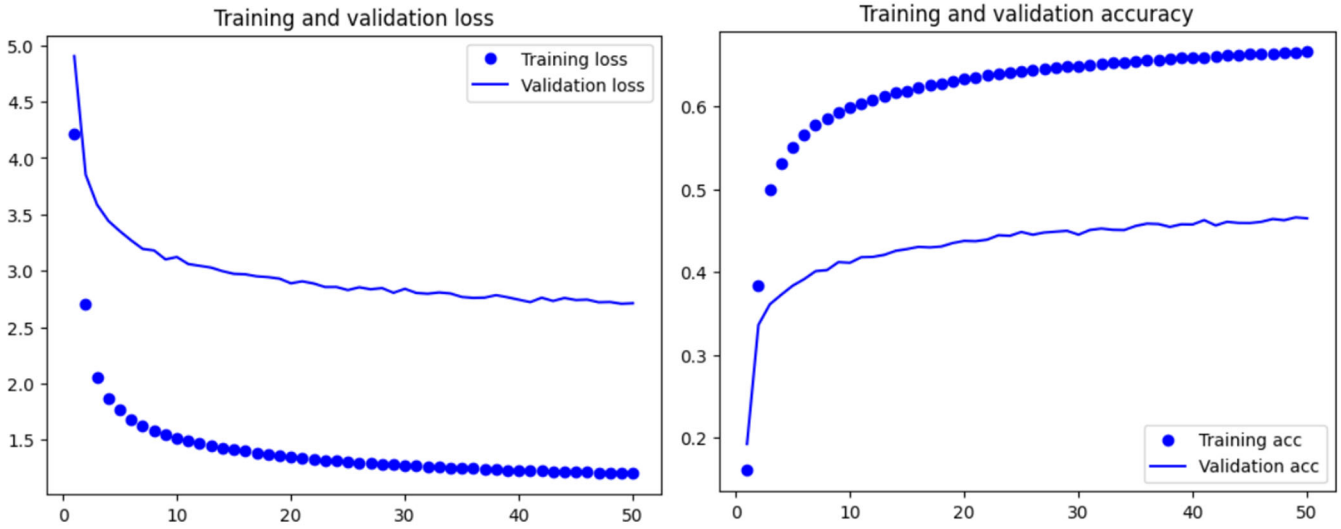

图 6 方案三预训练损失和精度曲线

Fig.6 Pre-training loss and accuracy curves of scheme three

5.7 问答任务微调

采用全参数微调，具体方法雷同方案一。

微调过程的损失、精度变化曲线见下图：

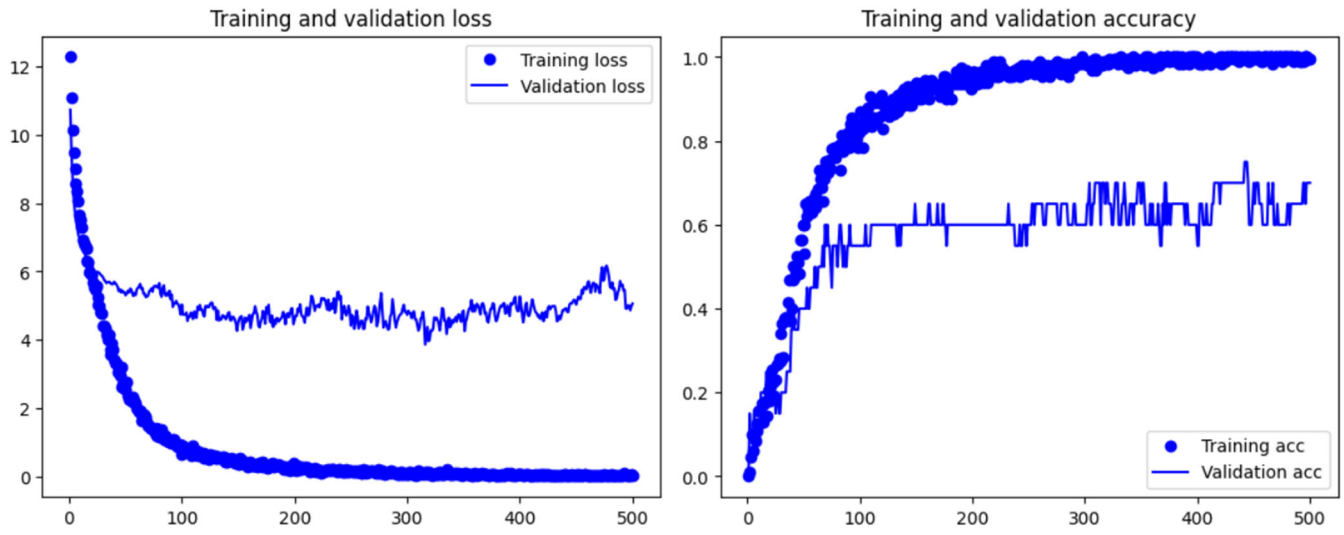

图 7 方案三微调损失和精度曲线

Fig.7 Fine tuning loss and accuracy curves of scheme three

经过 500 轮的微调，模型在训练集、验证集、测试集上的精度分别为 100%、70%、67%。可见本方案的泛化性能不佳，这与数据集规模太小、预训练模型质量有关。

6 结语

本文自建问答任务数据集，采用三种不同的方案，实现了基于大语言模型的桥梁设计规范问答系统，展示了大语言模型在专业领域问答系统中的应用潜力。

BERT 预训练模型全参数微调方案在训练集、验证集和测试集上均达到了 100%的精度，性能优益，验证了大语言模型在问答系统上的适用性。尽管 BERT 预训练模型参数高效微调和自建语言模型方案在测试集上遇到了泛化性能的挑战，但未来可以通过扩大数据集规模、优化模型结构等，来提高模型的泛化能力。

本研究不仅为桥梁设计领域提供了一个高效的问答工具，也为其他专业领域问答系统的构建提供了有价值的参考和方法论。

参考文献

[1]吴友政,赵军,段湘煜,等.问答式检索技术及评测研究综述[J].中文信息学报,2005,(03):1-13.

[2]张金营,王天堃,么长英,等.基于大语言模型的电力知识库智能问答系统构建与评价[J/OL].计算机科学,1-10[2024-08-12].http://kns.cnki.net/kcms/detail/50.1075.TP.20240528.0931.002.html.

[3]张春红,杜龙飞,朱新宁,等.基于大语言模型的教育问答系统研究[J].北京邮电大学学报(社会科学版),2023,25(06):79-88.DOI:10.19722/j.cnki.1008-7729.2023.0094.

[4]雷天凤,张永,龚春忠,等.基于大语言模型的竞品车型配置问答系统设计与应用研究[J].汽车科技,2024,(03):73-80.

[5]王婷,王娜,崔运鹏,等.基于人工智能大模型技术的果蔬农技知识智能问答系统[J].智慧农业(中英文),2023,5(04):105-116.

[6]丁志坤,李金泽,刘明辉.基于大语言模型的 BIM 正向设计问答系统研究[J].土木工程与管理学报,2024,41(01):1-7+12.DOI:10.13579/j.cnki.2095-0985.2024.20240046.

[7]Lucas M M , Justin Y , Pomeroy J K ,et al.Reasoning with large language models for medical question answering[J].Journal of the American Medical Informatics Association, 2024.DOI:10.1093/jamia/ocae131.

[8]Yoon W , Lee J , Kim D ,et al.Pre-trained Language Model for Biomedical Question Answering[C]//2020.DOI:10.1007/978-3-030-43887-6_64.

[9]弗朗索瓦·肖莱.Python 深度学习[M].第二版.北京:人民邮电出版社,2023.

[10]苏达哈尔桑·拉维昌迪兰.BERT 基础教程 Transformer 大模型实战[M] .北京:人民邮电出版社,2023.

[11]文森,钱力,胡懋地,等.基于大语言模型的问答技术研究进展综述[J].数据分析与知识发现,2024,8(06):16-29.

[12]中华人民共和国交通运输部.公路桥涵设计通用规范：JTG D 60-2015[S].北京:人民交通出版社,2015 年.