

# BERT 系列大模型下游任务微调与测试

杨洪易 ZY2203119  
yhongyi@buaa.edu.cn

## 摘要：

本文使用 BERT、ALBERT、RoBERTa 模型完成了命名实体识别、序列分类、问答任务，通过加载预训练模型并在具体任务数据集上微调，可以得到较好的性能，并通过提示工程的方法对下游任务进行了测试。

## 简介：

### 1.下游任务

#### ①命名实体识别

单句子标注任务也叫命名实体识别任务 (Named Entity Recognition)，简称 NER，常见的 NER 数据集有 CoNLL-2003 NER 等。该任务是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等，以及时间、数量、货币、比例数值等文字。

#### ②单句子分类任务

单句子分类任务属于序列级任务 (对于每个输入序列，只计算 Bert 模型中一个输出的损失)，使用 Bert 模型解决单句子分类任务需要对 Bert 模型做如下调整：

在 [SEP] 位置对应的输出后增加一个分类层 (全连接层+softmax 层)，用于输出最后的分类概率。

#### ③问答任务

问答任务属于 token 级任务 (对于每个输入序列，计算 Bert 模型部分输出的损失)，使用 BERT 模型解决问答任务需要对 Bert 模型做如下调整：

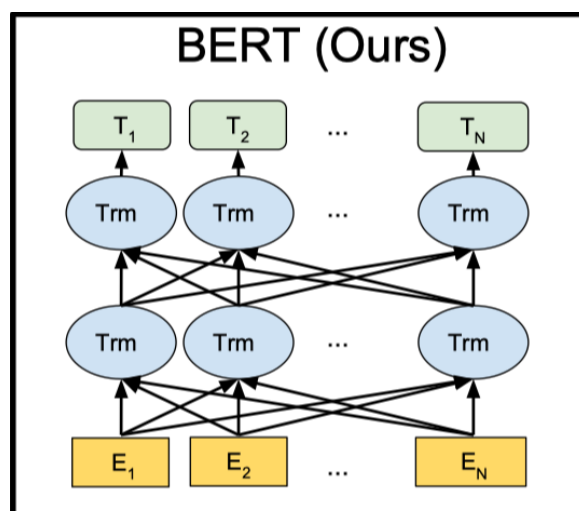
在 BERT 所有答案 token 对应的输出位置后增加一个分类层 (全连接层+softmax 层)，用于输出各个位置是答案开始和答案结束的概率；计算 loss 时，计算开始 loss 和结束 loss 的均值 loss 即可。

### 2.BERT 系列模型

#### ①BERT 模型

BERT 全称为 Bidirectional Encoder Representation from Transformers (来自 Transformers 的双向编码表示)，谷歌发表的论文 Pre-training of Deep Bidirectional Transformers for Language Understanding 中提出的一个面向自然语言处理任务的无监督预训练语言模型。是近年来自然语言处理领域公认的里程碑模型。

输入表示  $E = \text{Token Embeddings} + \text{Segment Embeddings} + \text{Position Embeddings}$



BERT 模型属于无监督的 Fine-tuning 方法，主要分为一下两个步骤：

- pre-training，在大量各种任务的无标签的数据上训练模型；
- fine-tuning，根据特定的 downstream 任务，给 BERT 模型添加输出层，使用预训练的参数对模型进行初始化，然后在该任务的有标签的数据集上对模型的参数进行微调。

## ② ALBERT 模型

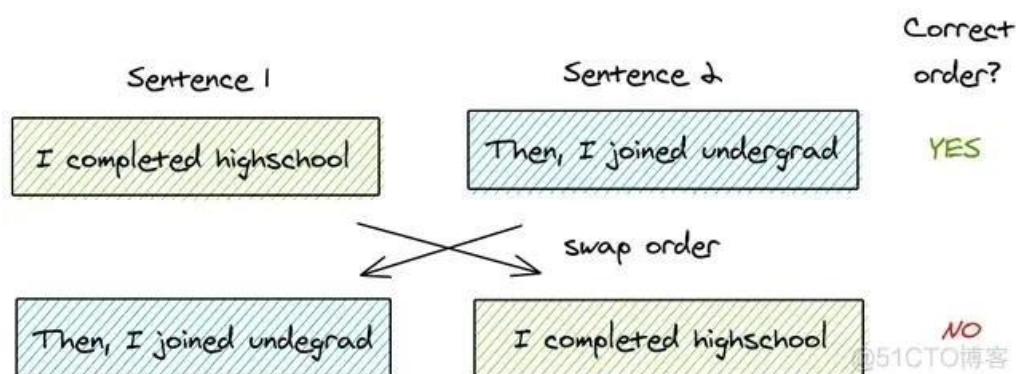
ALBERT 模型是在 BERT 模型的基础上进行改进的。它设计了参数减少的方法，用来降低内存消耗，同时加快 BERT 的训练速度。同时使用了 self-supervised loss（自监督损失函数）关注构建句子中的内在连贯性（coherence）。实验在 GLUE、RACE、SQuAD 数据集上取得了最好的效果。

ALBERT 使用了两种参数减少的方法：

- factorized embedding parameterization（词嵌入的因式分解）：把词嵌入矩阵进行分解，分解成更少的两个矩阵。
- cross-layer parameter sharing（交叉层的参数共享）：这种技术在深层的网络中更能减少参数。

ALBERT 利用了三种技术：

- factorized embedding parameterization（词嵌入的因式分解）
- cross-layer parameter sharing（交叉层的参数共享）
- sentence-order prediction (SOP, 句子顺序预测)



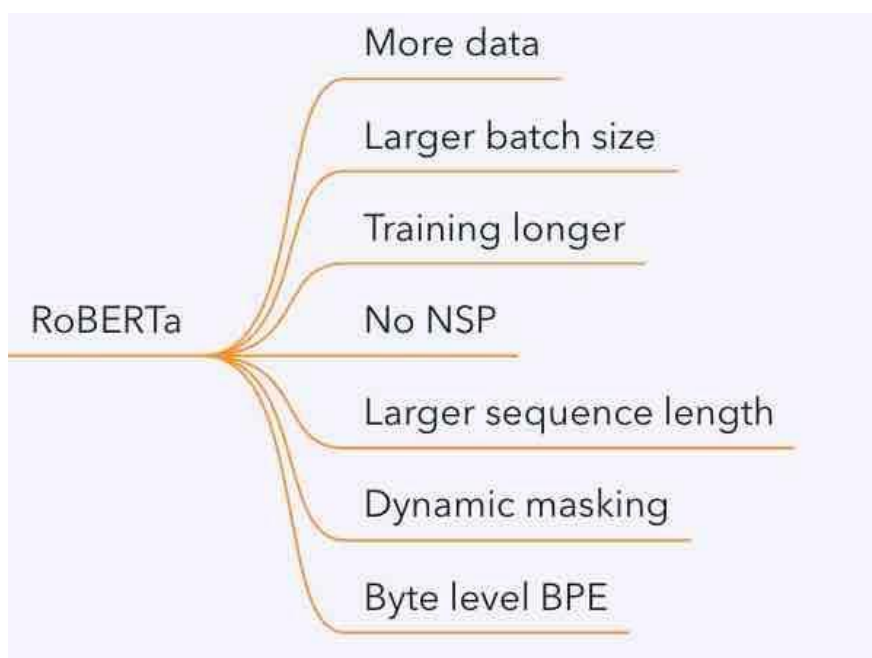
### ③ RoBERTa 模型

与 BERT 相比主要有以下几点改进：

- 更大的模型参数量（论文提供的训练时间来看，模型使用 1024 块 V100 GPU 训练了 1 天的时间）
- 更大 batch size。RoBERTa 在训练过程中使用了更大的 batch size。尝试过从 256 到 8000 不等的 batch size。
- 更多的训练数据（包括：CC-NEWS 等在内的 160GB 纯文本。而最初的 BERT 使用 16GB BookCorpus 数据集和英语维基百科进行训练）

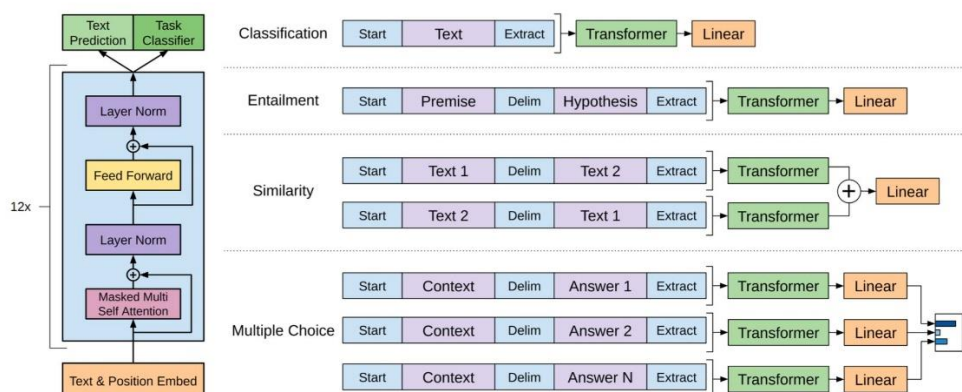
另外，RoBERTa 在训练方法上有以下改进：

- 去掉下一句预测(NSP)任务
- 动态掩码。BERT 依赖随机掩码和预测 token。RoBERTa 使用了动态掩码：每次向模型输入一个序列时都会生成新的掩码模式。
- 文本编码。考虑用更大的 byte 级别 BPE 词汇表来训练 BERT，这一词汇表包含 50K 的 subword 单元，且没有对输入作任何额外的预处理或分词。



### 3.参数微调

Fine-tuning 方式是指在已经训练好的语言模型的基础上，加入少量的 task-specific parameters, 例如对于分类问题在语言模型基础上加一层 softmax 网络，然后在新的语料上重新训练来进行 fine-tune。



实现:

## 1.环境






Anaconda3 python3.9.16

## 2.实验过程:

基本过程为加载预训练模型后针对数据集和任务进行微调，测试效果。

预训练模型加载:

使用 hugging face 预训练模型，找到对应模型类型下载保存到本地

 config.json	2023/6/4 11:07	JSON 源文件	1 KB
 pytorch_model.bin	2023/6/4 11:07	BIN 文件	399,657 KB
 special_tokens_map.json	2023/6/4 11:07	JSON 源文件	1 KB
 tokenizer_config.json	2023/6/4 11:07	JSON 源文件	1 KB
 vocab.txt	2023/6/4 11:06	文本文档	107 KB

### ①命名实体识别

NER 数据集保存在 bert\_ner/data 文件夹下，包含训练集、测试集、验证集三个 csv 文件，数据形式如图所示。

心	O
之	O
一	O
。	O
医	O
院	O
事	B-ORG
业	I-ORG
发	I-ORG
展	I-ORG
部	I-ORG
主	O
任	O
卡	B-PER
洛	I-PER
斯	I-PER
·	I-PER
马	I-PER
德	I-PER
里	I-PER
先	O
生	O
介	O
绍	O
说	O
，	O
这	O
是	O
全	O
墨	B-LOC
西	I-LOC
哥	I-LOC
最	O
大	O

读取并配置预训练的模型文件，以 albert 模型为例，读取方法参考 hugging 官网，利用 transformers 模块导入。from\_pretrained 后面可以使用本地模型文件也可以使用 hugging face 上的模型文件名。

```
tokenizer = BertTokenizer.from_pretrained("./albert_base_chinese")
config = AlbertConfig.from_pretrained("./albert_base_chinese", num_labels=num_labels)
model = AlbertNer.from_pretrained("./albert_base_chinese", config=config).to(device)
```

模型设计如下图，前向传播过程为 albert 模型提取特征后加入全连接层与 dropout，后配置隐藏层与交叉熵损失函数。

```
class AlbertNer(AlbertPreTrainedModel):
    def __init__(self, config):
        super(AlbertNer, self).__init__(config)
        self.num_labels = config.num_labels
        self.bert = AlbertModel(config) # 载入bert模型
        self.dropout = nn.Dropout(config.hidden_dropout_prob)
        # 简单的线性层
        self.classifier = nn.Linear(config.hidden_size, num_labels)
        # 初始化权重
        self.init_weights()

    def forward(self, input_ids, token_type_ids=None, attention_mask=None, labels=None, position_ids=None, head_mask=None):
        outputs = self.bert(input_ids, token_type_ids=token_type_ids, attention_mask=attention_mask,
                             position_ids=position_ids, head_mask=head_mask)
        sequence_output = outputs[0]

        sequence_output = self.dropout(sequence_output)
        logits = self.classifier(sequence_output)

        outputs = (logits,) + outputs[2:] # add hidden states and attention if they are here
        if labels is not None:
            loss_fct = nn.CrossEntropyLoss()
            loss = loss_fct(logits.view(-1, self.num_labels), labels.view(-1))
            outputs = (loss,) + outputs

        return outputs # (loss), scores, (hidden states), (attentions)
```

NER 处理部分较为庞大，主要为对文本进行处理，随后抽取特征与标签，这里展示了对文本的部分 token 化过程，句子开始设置[CLS]标志、句尾添加[SEP]标志，字转换为 id，实体类型转换为标签后，做分类训练。

```
def convert_examples_to_features(examples, label_list, max_seq_length, tokenizer):
    """Loads a data file into a list of `InputBatch`s."""
    # 表示从1开始对label进行index化
    label_map = {label: i for i, label in enumerate(label_list, 1)}

    features = []
    # 每个example实例
    for (ex_index, example) in enumerate(examples):
        textlist = example.text_a.split(' ')
        labellist = example.label.split(' ')
        tokens = []
        labels = []
        # 每个token
        for i, word in enumerate(textlist):
            # 分词，如果是中文，就是分字，但是对于一些不在BERT的vocab.txt中得字符会被进行WordPice处理（例如中文的引号）
            token = tokenizer.tokenize(word)
            tokens.extend(token)
            label_1 = labellist[i]
            labels.append(label_1)
```

随后在预训练模型上对三个模型进行了训练微调，训练过程为基本的加载数据输入模型后反向传播，使用 Adam 优化器优化。设置了 10 个训练周期，花费时间较长，微调过程使得预训练模型能够更好的适应下游任务需求，能够在对应数据集上达到较高的准确率，未经微调的预训练模型准确率较低。

## ② 单句子分类任务

数据集选用 kaggle 的 News Headlines Dataset For Sarcasm Detection 数据集，最终希望通过模型分别标题是否具有讽刺意味，其格式如下图所示，带有是否 sarcastic 的标签与对应的文本和文章链接。

```
{
  "is_sarcastic": 1,
  "headline": "thirtysomething scientists unveil doomsday clock of hair loss",
  "article_link": "https://www.theonion.com/thirtysomething-scientists-unveil-doomsday-clock-of-hair-loss-1819586205"
},
{
  "is_sarcastic": 0,
  "headline": "dem rep. totally nails why congress is falling short on gender, racial equality",
  "article_link": "https://www.huffingtonpost.com/entry/dem-rep-totally-nails-why-congress-is-falling-short-on-gender-racial-equality-us-57455f7e4b059bb1170b207"
},
{
  "is_sarcastic": 0,
  "headline": "eat your veggies: 9 deliciously different recipes",
  "article_link": "https://www.huffingtonpost.com/entry/eat-your-veggies-9-deliciously-different-recipes-us-59230747e4b07617aedc0e1a"
},
{
  "is_sarcastic": 1,
  "headline": "inclement weather prevents liar from getting to work",
  "article_link": "https://local.theonion.com/inclement-weather-prevents-liar-from-getting-to-work-1819576031"
},
{
  "is_sarcastic": 1,
  "headline": "mother comes pretty close to using word 'streaming' correctly",
  "article_link": "https://www.theonion.com/mother-comes-pretty-close-to-using-word-streaming-correctly-1819575546"
},
{
  "is_sarcastic": 0,
  "headline": "my white inheritance",
  "article_link": "https://www.huffingtonpost.com/entry/my-white-inheritance-us-59230747e4b07617aedc0e1a"
},
{
  "is_sarcastic": 0,
  "headline": "5 ways to file your taxes with less stress",
  "article_link": "https://www.huffingtonpost.com/entry/5-ways-to-file-your-taxes-with-less-stress-us-59230747e4b07617aedc0e1a"
},
{
  "is_sarcastic": 1,
  "headline": "richard branson's global warming donation nearly as much as cost of failed balloon trips",
  "article_link": "https://www.theonion.com/richard-bransons-global-warming-donation-nearly-as-much-as-cost-of-failed-balloon-trips-1819568749"
}
```

文本处理过程较为简单，基本过程为读取对应的标签与文本后对样本进行打乱，读取部分在 DataModules.py 下的 SequenceDataset 类里，下图展示了部分读取过程，按照数据集中字典的方式获取文本与标签，并随后创建对应的训练集测试集迭代器。

```
class SequenceDataset(Dataset):
    def __init__(self, dataset_file_path, tokenizer, regex_transformations={}):
        # Read JSON file and assign to headlines variable (list of strings)
        df = pd.read_json(dataset_file_path, lines=True)
        df = df.drop(['article_link'], axis=1)
        self.headlines = df.values
        # Regex Transformations can be used for data cleansing.
        # e.g. replace
        # '\n' -> ' ',
        # 'wasn't' -> 'was not'
        self.regex_transformations = regex_transformations
        self.tokenizer = tokenizer

    def __len__(self):
        return len(self.headlines)

    def __getitem__(self, index):
        is_sarcastic, headline = self.headlines[index]
        for regex, value_to_replace_with in self.regex_transformations.items():
            headline = re.sub(regex, value_to_replace_with, headline)
```

类 BERT 模型的 embedding 都有三层，包括 input\_ids, segment\_ids, attention\_mask

```
for step, batch in enumerate(train_iterator):
    model.train(True)
    # Here each element of batch list refers to one of [input_ids, segment_ids, attention_mask, labels]
    inputs = {
        'input_ids': batch[0].to(DEVICE),
        'token_type_ids': batch[1].to(DEVICE),
        'attention_mask': batch[2].to(DEVICE)
    }
```

随后在预训练模型上对三个模型进行了训练微调，训练过程为基本的加载数据输入模型后反向传播，使用 Adam 优化器优化，设置了 2 个训练周期。

### ③ 问答任务

使用了 Taipei\_QA\_new.txt 繁体中文数据集，主要内容为帮助民众解答生活相关问题寻找对应的负责部门或单位，数据集形式如下，第一列为答案第二列为相关问题。。

臺北市政府文化局	2018臺北藝術節FAQ
臺北市政府文化局	臺北市信義區Neo19大樓後方人行道後場使用作業通告(107/06/03-05)
臺北市政府文化局	2018台北電影節FAQ
臺北市政府文化局	臺灣新文化運動紀念館位置與聯絡方式
臺北市政府文化局	臺灣新文化運動紀念館開館及參觀時間?
臺北市政府文化局	2018臺北兒童藝術節FAQ
臺北市政府文化局	藝文補助之申請時間及計畫執行時間
臺北市政府文化局	新芳春茶行地點、開放時間、聯絡方式
臺北市政府文化局	如何前往松山文創園區
臺北市政府文化局	被指定古蹟之建築物要符合何種條件，才能辦理容積移轉?
臺北市政府文化局	所有權人接獲古蹟公告後，如不服指定程序該如何處理?
臺北市政府文化局	臺北市古蹟歷史建築紀念建築聚落建築群考古遺址史蹟及文化景觀審議會如何組成?
臺北市政府文化局	如何申請「古蹟」指定、「歷史建築」登錄? 「古蹟」指定、「歷史建築」登錄的程序為何?
臺北市政府文化局	錢穆故居營業時間? 是否須收取門票? 聯絡電話? 交通資訊?
臺北市政府文化局	林語堂故居營業時間? 是否須收取門票? 聯絡電話? 交通資訊?

同样加载预训练模型，使用了 BertConfig, BertForSequenceClassification, BertTokenizer 模块（其余两个模型同理），更接近序列分类，但在特征处理与提取方面有所差异，由于答案相对固定，使用了转 label 形式，即多分类问题。下图截取部分特征转换程序。

```
def convert_data_to_feature(tokenizer, train_data_path):
    with open(train_data_path, 'r', encoding='utf-8') as f:
        data = f.read()
        qa_pairs = data.split("\n")

    questions = []
    answers = []
    for qa_pair in qa_pairs:
        qa_pair = qa_pair.split()
        try:
            a, q = qa_pair
            questions.append(q)
            answers.append(a)
        except:
            continue

    assert len(answers) == len(questions)

    ans_dic = DataDic(answers)
    question_dic = DataDic(questions)

    q_tokens = []
    max_seq_len = 0
    for q in question_dic.data:
        bert_ids = to_bert_ids(tokenizer, q)
        if len(bert_ids) > max_seq_len:
            max_seq_len = len(bert_ids)
        q_tokens.append(bert_ids)
```



随后在预训练模型上对三个模型进行了训练微调, 训练过程为基本的加载数据输入模型后反向传播, 使用 AdamW 优化器优化, 设置了 30 个训练周期进行微调。

## 结果:

### ①命名实体识别

BERT、ALBERT、RoBERTa 三种模型分别训练结果如下:

6/02/2023 20:18:44 - INFO - __main__ -					
	precision	recall	f1-score	support	
LOC	0.9505	0.9736	0.9619	36517	
ORG	0.9158	0.9294	0.9225	20574	
PER	0.9790	0.9741	0.9766	17618	
micro avg	0.9475	0.9616	0.9545	74709	
macro avg	0.9484	0.9590	0.9537	74709	
weighted avg	0.9477	0.9616	0.9545	74709	

6/03/2023 22:32:35 - INFO - __main__ -					
	precision	recall	f1-score	support	
LOC	0.9402	0.9583	0.9492	36517	
ORG	0.8944	0.9261	0.9100	20574	
PER	0.9426	0.9664	0.9544	17618	
micro avg	0.9280	0.9513	0.9395	74709	
macro avg	0.9257	0.9503	0.9378	74709	
weighted avg	0.9282	0.9513	0.9396	74709	

6/04/2023 11:29:07 - INFO - __main__ -					
	precision	recall	f1-score	support	
LOC	0.9606	0.9664	0.9635	36517	
ORG	0.9262	0.9139	0.9200	20574	
PER	0.9819	0.9780	0.9799	17618	
micro avg	0.9562	0.9547	0.9555	74709	
macro avg	0.9562	0.9528	0.9545	74709	
weighted avg	0.9561	0.9547	0.9554	74709	

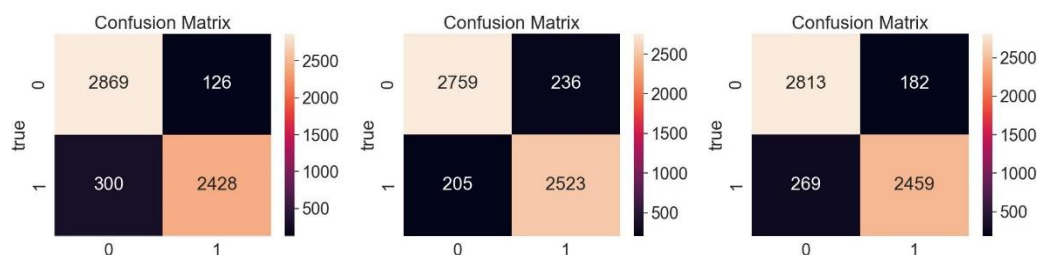
综合来看在 NER 任务上 BERT、RoBERTa 表现出的性能较好, f1 得分也较高, 可以看出在一些指标上 RoBERTa 相比 BERT 有所提升, ALBERT 相对较差。

### ②单句子分类任务

BERT、ALBERT、RoBERTa 三种模型分别训练结果如下:

Training Accuracy 93.7718 - Validation Accuracy 92.1195
Training Accuracy 93.9160 - Validation Accuracy 92.2943
Training Accuracy 93.0687 - Validation Accuracy 92.5564

accuracy: 0.925563515638651 recall: 0.8900293255131965 precision 0.9506656225528582 F1-score 0.9193487315410829	accuracy: 0.9229425126681811 recall: 0.9248533724340176 precision 0.9144617615077927 F1-score 0.9196282121377802	accuracy: 0.925563515638651 recall: 0.8900293255131965 precision 0.9506656225528582 F1-score 0.9193487315410829
--	---	--



在该任务中 ALBERT 模型取得了最高的 F1 分数。

### ③ 问答任务

以提示工程的方式对提问进行修饰，做出如下三句提问

```
q_inputs = ['為何路邊停車格有編號的要收費，無編號的不用收費', '債權人可否向稅捐稽徵處申請查調債務人之財產、所得資料，應去往何處', '想做大腸癌篩檢，不知如何辦理']
```

训练三个模型，BERT、ALBERT、RoBERTa 分别花费 30、60、120 周期达到测试集准确率 80%，这可能由于 ALBERT、RoBERTa 使用了 tiny 版模型。下图展示了 BERT 训练结果。

```
epoch:30 batch: 42 test_loss:0.9968 test_acc:80.2083
epoch:30 batch: 43 test_loss:1.0085 test_acc:80.2326
epoch:30 batch: 44 test_loss:1.0121 test_acc:80.1136
epoch:30 batch: 45 test_loss:0.9932 test_acc:80.4167
epoch:30 batch: 46 test_loss:1.0003 test_acc:80.4348
epoch:30 batch: 47 test_loss:0.9984 test_acc:80.5851
epoch:30 batch: 48 test_loss:0.9982 test_acc:80.4688
epoch:30 batch: 49 test_loss:1.0029 test_acc:80.3571
epoch:30 batch: 50 test_loss:0.9849 test_acc:80.6167
```

三个模型的问答结果分别如下所示：

為何路邊停車格有編號的要收費，無編號的不用收費	臺北市停車管理工程處
債權人可否向稅捐稽徵處申請查調債務人之財產、所得資料，應去往何處	臺北市稅捐稽徵處稅務管理科
想做大腸癌篩檢，不知如何辦理	臺北市立聯合醫院
為何路邊停車格有編號的要收費，無編號的不用收費	臺北市政府環境保護局資源循環管理科
債權人可否向稅捐稽徵處申請查調債務人之財產、所得資料，應去往何處	臺北市政府地政局土地登記科
想做大腸癌篩檢，不知如何辦理	臺北市政府產業發展局農業發展科



為何路邊停車格有編號的要收費，無編號的不用收費  
臺北市停車管理工程處

債權人可否向稅捐稽徵處申請查調債務人之財產、所得資料，應去往何處  
臺北市稅捐稽徵處稅務管理科

想做大腸癌篩檢，不知如何辦理  
臺北市立聯合醫院

可以看出 BERT、RoBERTa 得出了正确的答案，而 ALBERT 模型得到的答案明显有误。

## 结论：

在所选的三种下游任务中，在命名实体识别与问答任务上 BERT 与 RoBERTa 效果较好，同时在句子序列分类任务上 ALBERT 性能更优。

分析原因参考 ALBERT 论文，ALBERT 模型相比 BERT 做了轻量化，可以产生轻量级模型文件，但参考原论文结果 ALBERT 只是在模型结构更加复杂时，效果好于 BERT，对于 ALBERT 随着模型的深度增加，模型的性能提升。出现实验中现象可能受到数据集特性、中文特性、数据集特性影响，表现有所浮动。同时，参考 RoBERTa 论文，经过增大训练时长，更大的批规模 (BATCH SIZE)，更大的数据集，增长预训练的序列长度等等方式，模型的效果有显著提升，由于设备限制，实验中训练并不充分，故可能并未发挥 RoBERTa 模型全部性能。借此机会学习了一下 BERT 模型，希望对科研有所启发。能力有限，多有参考。

## 参考：

[1]BERT 论文：《BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding》

[2]ALBERT 论文：《ALBERT: A lite BERT for self-supervised learning of language representations》

[3]RoBERTa 论文：《RoBERTa: A Robustly Optimized BERT Pretraining Approach》

[4] [BERT 和 ALBERT\\_albert 和 bert 的区别](#)

[5] [p208p2002/bert-downstream-task-notes: BERT 各類下游任務實作細節 \(github.com\)](#)

[6] [moon-hotel/BertWithPretrained: An implementation of the BERT model and its related downstream tasks based on the PyTorch framework \(github.com\)](#)

[7] [649453932/Bert-Chinese-Text-Classification-Pytorch: 使用 Bert, ERNIE, 进行中文文本分类 \(github.com\)](#)

[8] [RoBERTa 模型原理总结 - 知乎 \(zhihu.com\)](#)