自然语言处理实验报告

1 实验问题描述

1.1 问题描述

本实验是一个利用百度 paddle 框架对新闻标题（中文）进行分类的简单项目。

1.2 实验步骤

1. 获取数据集。
2. 将数据处理成模型需要的形式。
3. 构建网络。
4. 训练模型。
5. 调试参数。
6. 利用模型进行预测。

2 数据处理介绍

2.1 数据集概述

实验中所用数据来源于百度飞桨网站，数据获取网址如下：

数据来源网址：常规赛-中文新闻文本标题分类数据集 - 飞桨 AI Studio (baidu.com)

该数据集分为训练集、验证集和测试集。其中训练集数据 752471 条、验证集数据 80000 条、测试集数据 83599 条，测试集数据没有 label。

训练集部分原始数据如图 1：

```
1  网民市民集体幻想中奖后如果你中了9000万怎么办      彩票
2  PVC期货有望5月挂牌  财经
3  午时三刻新作《幻神录–宿命情缘》  游戏
4  欧司朗LLFY网络提供一站式照明解决方案      家居
5  试探北京楼市向何方：排不完的队  涨不够的价  房产
6  个性测试：测你的生活无趣指数(图)      星座
7  浙江女排战前遇下马威天津球迷微博发出“威胁”  体育
8  澳央行行长重申未来需要加息  股票
9  海地总统选举第二轮投票将推迟举行      时政
10 苹果中国学生机开卖最高优惠1600元      科技
11 黄金分割位有支撑短期股指有望止跌企稳      股票
12 英国政府建议各部门每天发布2-10条微博客  时政
13 网游“吸金”新玩法      科技
14 短讯-阿森纳之王表态不回欧洲亨利在欧洲已赢得一切  体育
15 美男子因不满银行一怒推平豪宅      房产
16 刘孜喜得贵子不退出娱乐圈儿子名字叫NEMO  娱乐
```

图 1：原始数据

2.2 数据介绍

训练数据集包含新闻标题和新闻分类。实验中根据训练数据集整理出一份新闻分类的字典列表。如下所示：

[{0: '教育'}, {1: '社会'}, {2: '星座'}, {3: '房产'}, {4: '彩票'}, {5: '体育'}, {6: '游戏'}, {7: '财经'}, {8: '股票'}, {9: '时政'}, {10: '科技'}, {11: '娱乐'}, {12: '时尚'}, {13: '家居'}]

2.3 数据集定义与加载

本实验采用的是百度的飞桨平台进行编程实现，采用百度 paddle 框架和百度研发的 erine 预处理模型，因此需要将数据处理成 erine 模型的输入形式。

（1） Erine 模型简介

该模型是百度发布的一个预训练模型，采用 transformer 的 encoder 部分作为模型主干进行训练。使用 erine 进行预训练，可以帮助挖掘句子间的语义关系，更好的进行文本分类任务。

Erine 模型的数据输入格式如下：

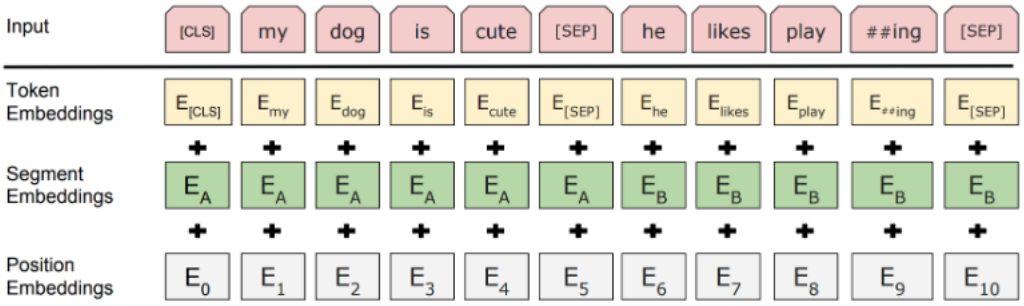

图 2：erine 模型的数据输入格式

Input 为输入的文本句子，token ids 是每个字的向量表示、segment ids 是用来表示每个字属于哪句话的向量，position_ids 表示位置信息，由模型自动生成。因此，使用该预训练模型需要输入的是 token_ids 和 segment_ids。

（2） 数据读取

本实验采用自定义数据集的形式，构造一个数据集类 MyDataSet 类，该类负责进行数据读取、数据获取和初步数据转换，实验中的训练集和验证集数据通过该类进行加载。

```
import paddle
import numpy as np
import paddlenlp
import pandas as pd
```

```
import re
from functools import partial
from paddlenlp.transformers import LinearDecayWithWarmup
from paddlenlp.data import Stack,Tuple,Pad
import warnings
warnings.filterwarnings('ignore')
import paddle.nn.functional as F
from paddlenlp.datasets import MapDataset

class MyDataSet(paddle.io.Dataset):
    def __init__(self,path):
        self.data_dict_list=self.__load_data(path)

    def __load_data(self,path):
        with open(path,'r',encoding='utf-8') as f:
            data=f.readlines()
            data_dict_list=[]
            for line in data:
                message=line.strip('\n')
                label=message[-2:]
                for i in range(len(Label_list)):
                    if Label_list[i]==label:
                        label=i
                text=message[:-3]
                temp={'text':text,'label':label}
                data_dict_list.append(temp)
        return data_dict_list

    def __getitem__(self, idx):
        return self.data_dict_list[idx]

    def __len__(self):
        return len(self.data_dict_list)
```

表 1：自定义数据集类

自定义数据集将原始数据每条处理成字典的形式，每条数据包含 text 和 label，如下：

{'text': '我国 300 名公民因将护照交给利比亚当局滞留', 'label': 9},

{'text': '英超-兰帕德 2 球飞翼传射切尔西 3-0 少赛 1 轮返榜首', 'label': 5},

{'text': '新《还珠》主创赴台见琼瑶小燕子卖乖送吻', 'label': 11},

{'text': '台检方认定 19 名失踪大陆游客及领队遇难', 'label': 9},

{'text': '解析京城城市综合体之现状', 'label': 3},

{'text': '详讯：美大城市房价创泡沫破裂以来新低', 'label': 8},

{'text': '30 所区外院校下月在新疆进行高考艺术类测试', 'label': 0},

（3） 数据读取器构建

实验中定义了 convert_example 函数将从自定义数据集中获取的每一条数据转换成 ernie 需要的 input_ids 和 token_type_ids。

```
# 该函数作用为将数据转换为 ernie 所需要的输入数据格式
def convert_example(example, tokenizer, max_seq_length=128, is_test=False):
    encoded_inputs=tokenizer(text=example['text'],max_seq_len=max_seq_length)
    input_ids = encoded_inputs['input_ids']
    token_type_ids = encoded_inputs['token_type_ids']

    if is_test:
        print('进入了 test:')
        return input_ids, token_type_ids
    else:
        label = np.array([example['label']], dtype='int64')
        return input_ids, token_type_ids, label
```

表 2：convert_example 函数定义

定义 create_dataloader 函数对从数据集获取的数据进行格式转换和自动批处理。

```
def create_dataloader(dataset, batch_size=1,
                      batchify_fn=None, trans_fn=None):
    # trans_fn 对应前边的 covert_example 函数，该函数处理每个样本为期望的格式
    if trans_fn:
        dataset = dataset.map(trans_fn)

    # 定义并初始化数据读取器
    return paddle.io.DataLoader(dataset, batch_size=batch_size,
                                shuffle=False, collate_fn=batchify_fn,
                                num_workers=1, drop_last=False,
return_list=True)
```

表 3：定义 dataloader 函数

定义训练数据读取器和验证数据读取器。

```
trans_func = partial(
    convert_example,
    tokenizer=tokenizer,
    max_seq_length=max_seq_len)
```

```
batchify_fn = lambda samples, fn=Tuple(
    Pad(axis=0, pad_val=tokenizer.pad_token_id),
    Pad(axis=0, pad_val=tokenizer.pad_token_type_id),
    Stack(dtype='int64')
): [data for data in fn(samples)]

train_dataset = MyDataSet(Train_path)
train_set = MapDataset(train_dataset)
train_data_loader = create_dataloader(
    train_set,
    batch_size=batch_size,
    batchify_fn=batchify_fn,
    trans_fn=trans_func
)
```

表 4：定义数据转换的函数

综上，数据读取器的处理过程为：

1 实例化自定义数据集类，获取初步处理后的数据集合

2 将 1 中得到的数据集映射为 MapDadaset 类型，方便对数据进行编码

3 构造 data_loader。对 2 中数据集的每一个样本进行 convert_example 函数中定义的操作，之后调用 paddle.io.DataLoader 构造 dataloader，dataloader 自动对数据集进行分批操作，以便分批进行训练和调试。

3 模型构建与预测

3.1 模型建立

（1） 本实验采用 paddlenlp 预训练好的 ernie 模型后连接一个分类器来进行微调来对中文文本分类。定义 ErineForSequenceClassification 类，在该类中加载预训练好的 erine 模型、构造一个线性分类层和 dropout 层来防止过拟合，该类返回每个句子对应每个标签类别的向量表示。

Erine 模型有 erine-tiny\erine 等形式，实验中为了更快训练出结果，使用的是 ernie-tiny。

```
# 构建网络
class ErnieForSequenceClassification(paddle.nn.Layer):
    def __init__(self, num_class=14, dropout=None):
        super(ErnieForSequenceClassification, self).__init__()
        # 加载预训练好的 ernie
```

```
        self.ernie =
paddlenlp.transformers.ErnieModel.from_pretrained(MODEL_NAME)
        # self.dropout=paddle.nn.Dropout(dropout)
        self.dropout = paddle.nn.Dropout(dropout if dropout is not None else
self.ernie.config['hidden_dropout_prob'])
        self.classifier = paddle.nn.Linear(self.ernie.config['hidden_size'],
num_class)

    def forward(self, input_ids, token_type_ids=None):
        sequence_output, pooled_output = self.ernie(
            input_ids,
            token_type_ids)
        # print("在网络里面~")
        # print('pooled_output:{}'.format(pooled_output))
        pooled_output = self.dropout(pooled_output)
        logits = self.classifier(pooled_output)
        return logits
```

表 5：构建网络

3.2 模型训练

构造优化器

```
optimizer = paddle.optimizer.AdamW(
    learning_rate=lr_scheduler,
    parameters=model.parameters(),
    weight_decay=weight_decay,
    apply_decay_param_fun=lambda x: x in [
        p.name for n, p in model.named_parameters()
        if not any(nd in n for nd in ['bias', 'norm'])
    ])
```

表 6：构造优化器

定义训练函数和用于模型调试的验证函数

```
def evaluate(model, metric, data_loader):
    model.eval()
    # 每次使用测试集进行评估时，先重置掉之前的 metric 的累计数据，保证只是针对本次评估。
    metric.reset()
    losses = []
    # logits_list=[]
    # labels_list=[]
    acc_list = []
```

```
    total_acc = 0

    for step, batch in enumerate(data_loader):
        input_ids, segment_ids, labels = batch
        logits = model(input_ids, segment_ids)
        loss = F.cross_entropy(input=logits, label=labels)
        loss = paddle.mean(loss)
        losses.append(loss.numpy())
        correct = metric.compute(logits, labels)
        metric.update(correct)
        acc = metric.accumulate()
        acc_list.append(acc)
        total_acc += acc

    print('eval loss:%.5f,acc:%.5f' % (np.mean(losses), acc))
    print('acc_list={}'.format(acc_list))
    print('ave_acc={}'.format(np.mean(acc_list)))
    print('total_acc={},ave_acc={}'.format(total_acc, total_acc / len(acc_list)))
    metric.reset()

def train(model):
    global_step = 0
    for epoch in range(1, epochs + 1):
        print('epoch={}'.format(epoch))
        model.train()
        for step, batch in enumerate(train_data_loader, start=1):
            if (step == 100):
                print("step={}".format(step))
            input_ids, segment_ids, labels = batch
            logits = model(input_ids, segment_ids)
            loss = F.cross_entropy(input=logits, label=labels)
            probs = F.softmax(logits, axis=1)
            correct = metric.compute(probs, labels)
            # if step<5:
            #     print('probs.shape:{}'.format(probs.shape))
            #     pred_labels=np.argmax(probs,axis=1)
            #     true_labels=labels
            #     print('step:{},pred_labels:{},true_labels:{}'.
format(step,pred_labels,true_labels))
            #     # print('step:{},correct:{}'.format(step,correct))
            metric.update(correct)
            acc = metric.accumulate()
            global_step += 1
            loss.backward()
```

```
            optimizer.step()
            lr_scheduler.step()
            optimizer.clear_grad()


train(model)
```

表 7：定义训练函数与验证函数

3.3 模型保存与加载

模型训练时间较长，因此为了便于后续测试和验证，一般需要在训练结束后将模型保存下来，这样之后直接加载模型保存的文件就可以进行测试。

```
# ----------------------------------   保存模型用于调试

# 模型保存的名称
model_name = "ernie_for_news_classification"

paddle.save(model.state_dict(), "{}.pdparams".format(model_name))
paddle.save(optimizer.state_dict(), "{}.optparams".format(model_name))
tokenizer.save_pretrained('./tokenizer')
```

表 8：保存模型用于调试

```
# ----------------------------------   保存模型用于预测（推理）
from paddle.static import InputSpec
# 1.切换 eval()模式
model.eval()
# 2. 构造 InputSpec 信息
input_ids = InputSpec([32, 26],'int64')
segment_ids=InputSpec([32,26],'int64')
# 3.调用 paddle.jit.save 接口转为静态图模型
path = "model_for_predict/linear"
paddle.jit.save(
    layer=model,
    path=path,
    input_spec=[input_ids,segment_ids])
```

表 9：保存模型用于预测

```
# ----------------------------------   加载模型用于预测
import paddle
path = "model_for_predict/linear"
loaded_model = paddle.jit.load(path)
```

表 10：加载模型用于预测

3.4 模型验证与预测

（1） 模型验证

加载验证数据集来验证模型效果，验证数据集需要与训练数据集进行同样的处理操作。如图：

```
valid_dataset = MyDataSet(Valid_path)
valid_set = MapDataset(valid_dataset)
# test_set = MapDataset(test_dataset)
valid_data_loader = create_dataloader(
    valid_set,
    batch_size=batch_size,
    batchify_fn=batchify_fn,
    trans_fn=trans_func
)

evaluate(model, metric,valid_data_loader)
```

表 11：加载验证数据集进行验证

模型在验证数据集上的分类效果还可以，总准确率约为 0.958，如图。

```
eval loss:0.13060,acc:0.95774
acc_list=[0.875, 0.921875, 0.9375, 0.9296875, 0.9375, 0.9427083333333
142857, 0.94140625, 0.9444444444444444, 0.946875, 0.9488636363636364,
66, 0.9495192307692307, 0.9508928571428571, 0.9520833333333333, 0.951
470588235, 0.9513888888888888, 0.9506578947368421, 0.9515625, 0.95238
3125, 0.9497282608695652, 0.9505208333333334, 0.9525, 0.953125, 0.954
953125, 0.9536637931034483, 0.9520833333333333, 0.9526209677419355, 0
45454545454546, 0.9558823529411765, 0.9553571428571429, 0.95399305555
64864865, 0.9547697368421053, 0.9543269230769231, 0.95546875, 0.95579
53571428571429, 0.9556686046511628, 0.9566761363636364, 0.95763888888
47826086, 0.9574468085106383, 0.9576822916666666, 0.9559948979591837,
205882352942, 0.9585336538461539, 0.9587264150943396, 0.9589120370370
909091, 0.9598214285714286, 0.9599780701754386, 0.9595
ave_acc=0.9576342451458822
total_acc=2394.0856128647083,ave_acc=0.9576342451458834
```

图 3：验证数据集验证结果

（2） 模型预测

测试数据集没有标签数据，需要另外定义函数来进行数据集获取和数据编码处理。

```
class TestDataSet(paddle.io.Dataset):
    def __init__(self,path):
        self.data_dict_list=self.__load_data(path)

    def __load_data(self,path):
        with open(path,'r',encoding='utf-8') as f:
            data=f.readlines()
            data_dict_list=[]
            for line in data:
                message=line.strip('\n')
                text=message
```

```
                temp={'text':text}
                data_dict_list.append(temp)
        return data_dict_list

    def __getitem__(self, idx):
        return self.data_dict_list[idx]

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

def convert_example02(example, tokenizer, max_seq_length=128):
    encoded_inputs = tokenizer(text=example['text'], max_seq_len=max_seq_length)
    input_ids = encoded_inputs['input_ids']
    token_type_ids = encoded_inputs['token_type_ids']
    return input_ids, token_type_ids


def create_dataloader02(dataset, batch_size=1,
                        batchify_fn=None, trans_fn=None):
    # trans_fn 对应前边的 covert_example 函数，使用该函数处理每个样本为期望的格式
    if trans_fn:
        dataset = dataset.map(trans_fn)
    print('进入了 create_dataloader02')

    # 定义并初始化数据读取器
    return paddle.io.DataLoader(dataset, batch_size=batch_size,
                                shuffle=False, collate_fn=batchify_fn,
                                num_workers=1, drop_last=False, return_list=True)


batchify_fn02 = lambda samples, fn=Tuple(
    Pad(axis=0, pad_val=tokenizer.pad_token_id),
    Pad(axis=0, pad_val=tokenizer.pad_token_type_id),
): [data for data in fn(samples)]


trans_func02 = partial(
    convert_example02,
    tokenizer=tokenizer,
    max_seq_length=max_seq_len)
```

表 12：定义对测试数据进行读取的类和函数

```
Test_path='./data/data103654/test.txt'
from paddlenlp.datasets import MapDataset
```

```
batch_size=32

# 加载测试数据集
test_dataset=TestDataSet(Test_path)
test_set = MapDataset(test_dataset)
test_data_loader = create_dataloader02(
    test_set,
    batch_size=batch_size,
    batchify_fn=batchify_fn02,
    trans_fn=trans_func02
)
```

表 13：加载测试数据集

```
def predict(model,data_loader):
    model.eval()
    results = []

    for step, batch in enumerate(data_loader):
        input_ids, segment_ids = batch
        logits = model(input_ids, segment_ids)
        probs = F.softmax(logits, axis=1)
        idx = paddle.argmax(probs, axis=1).numpy()
        idx = idx.tolist()
        results.extend(idx)
 return results
```

表 14：定义预测函数

```
results = predict(loaded_model,test_data_loader)
print(results)
```

表 15：预测测试数据集中前面 1000 条新闻的标签

测试数据集前 1000 条新闻为：

```
]: [{'text': '北京君太百货璀璨秋色 满100省353020元'},
    {'text': '教育部：小学高年级将开始学习性知识'},
    {'text': '专业级单反相机 佳能7D单机售价9280元'},
    {'text': '星展银行起诉内地客户 银行强硬客户无奈'},
    {'text': '脱离中国的实际 强压人民币大幅升值只能是梦想'},
    {'text': '内城土地稀缺 对开发商提出更高要求(组图)'},
    {'text': '亚欧首脑会议举行第二次全体会议(组图)'},
    {'text': '荷兰主帅球迷面前表决心耍酷 罗本不在范佩西成一哥'},
    {'text': '搭配18-105VR镜头 尼康D90带票7109元'},
    {'text': '百盛购物中心美罗城店 同一专柜花120得200'}]
```

图 4：测试数据集前 1000 条新闻部分数据

```
83612,进入了test:
83613,进入了test:
83614,进入了test:
[10, 9, 10, 8, 8, 3, 9, 5, 10, 3]
```

图 5：显示前 10 条新闻预测结果 id

```
1 pred_label
```

```
['科技', '时政', '科技', '股票', '股票', '房产', '时政', '体育', '科技', '房产']
```

图 6：id 转为 label