高级机器学习报告 2——数据集预测

陈晟 [1)]

[1)](南京大学 计算机科学与技术系, 南京 中国 210033)

摘 要 高级机器学习第二次实验报告旨在让解决一个数据集预测的问题。该数据集是关于短期旅行租赁市场的业务，我们的任务是为房东开发建议服务，帮助他们设置租金等级，即通过模型算法预测含有不同特征的房屋租金等级(0-5).本次实验过程中，我首先对训练集数据进行了简单的分析，然后将数据进行合适的预处理后，将所有特征融合并且分词处理，选取 BERT 加 TextCNN 和 BERT 加 GRU 模型分别对其进行的训练，并且将训练集的一部分划出作为验证集，测试模型的泛化性能，然后通过多次尝试选取最适合的模型结构和训练轮次作为最终模型，并对测试集进行预测，保留结果。本次报告分为 5 个部分，首先对数据集进行分析介绍，然后介绍本次实验用到的 BERT，TextCNN 和 GRU 模型，并写明详细的实验流程和伪代码，并且说明实验结果，最终给出结论和分析。

关键词 数据集预测；BERT；卷积神经网络；循环神经网络；文本分类；特征融合；

Advanced Machine Learning Report 2 - Dataset Prediction

Sheng Chen[1)]

[1)](Department of computer science and technology, Nanjing University, Nanjing, China)

Abstract The Advanced Machine Learning 2nd Lab Report aims to solve the problem of prediction on a dataset. This dataset is about the business of the short-term travel rental market. Our task is to develop a recommendation service for landlords to help them set rent grades, that is, to predict the rent grades (0-5) of houses with different characteristics through the model algorithm. This experiment process , I first performed a simple analysis of the training set data, and then after appropriate preprocessing of the data, I selected the BERT plus TextCNN and BERT plus GRU models to train them respectively, and demarcated a part of the training set as a verification. Set, test the generalization performance of the model, and then select the most suitable model structure and training rounds as the final model through multiple attempts, make predictions on the test set, and retain the results. This report is divided into 5 parts. First, the data set is analyzed and introduced, and then the BERT, TextCNN and GRU models used in this experiment are introduced, and the detailed experimental process and pseudocode are written, and the experimental results are explained. Finally, the draw conclusions and analysis.

Key words Dataset prediction; BERT; Convolutional Neural Network; Recurrent Neural Network; Text Classification; Feature fusion;

1 问题理解与分析

本次实验的数据集是关于短期旅行租赁市场的业务，我们的任务是为房东开发建议服务，帮助他们设置租金，我们要通过收集这个数据集中的结构化表格数据和非结构化描述，并尝试构建一个分类器来预测每个房子的租金应该设置在什么级别（0 级-5 级）。

该数据集分为训练集和测试集，训练集具有标签（租金等级）而测试集没有标签，所以模型在训练集上训练后因为不知道模型训练后在测试集上的效果，需要将训练集拿出一部分来作为验证集来测试模型在未知数据集上的效果如何。训练集一共

陈晟，男，1999年生，硕士，主要研究领域为自然语言处理.E-mail:tiancai1073943777@qq.com.

有 15063 条数据样本记录，测试集一共有 5000 条数据记录（没有标签租金级别）。

训练集除了标签外共有 16 类特征，分别是 description, neighbourhood, latitude, longitude, type, accommodates, bathrooms, bedrooms, amentities, reviews, review_dating, review_scoresA, review_scores_B, review_scoresC, review_scoresD, insant_bookable. 中文含义分别是：对该房子的简单描述，其靠近哪，经纬度，房子类型，容纳人数，洗浴室个数，卧室个数，含有哪些配置，评论条数，评论率，评论分数 ABCD 以及是否能及时预定。出于自身的先验知识，我感觉比较重要的特征为 description，amentities，四个 review 分数。

刚拿到数据集我感觉到数据的特征很多，如果要要有一个较好的分类效果，可能要找到与标签 target 关系比较密切的特征来做分类。但是也可以反过来想，如果能把所有的特征都放进来一起训练一个模型，应该也能达到不错的效果。得益于深度学习 NLP 技术的发展，模型可以足够复杂到很好的捕捉到所有的特征信息，但是出于简单的考虑，我也许可以先尝试只将比较重要，基于自身的先验知识，我感觉比较重要的特征为 description，amentities，四个 review 分数，因为这几个特征中含有丰富的文字和数字，如果我是顾客，我也会先看这几个特征。所以我准备先将前两个都是文字型的特征融合起来输入到模型中，做文本分类任务。然后，我还会将 review 分数转换为字符串形式加入到之前的处理中联合训练。分析效果后，我最后再将所有的特征都融合转换成合适输出的字符串，转换成文本分类任务进行训练模型。

虽然之前提到的 description，amentities，四个 review 分数较为重要但是其他特征如 neighbourhood, bedrooms 也许也是一些顾客的重要参考指标，因为也许顾客出于某些旅游或者人数要求也会根据这些特征进行选择，并且如经纬度这类特征对我们而言看似不一定有用，也看不出个啥，但是这对模型而言则不一定了，因为模型也许会学习到我们想象不到的知识从而对分类任务产生正向影响。所以我最后要采用的形式还是将所有特征处理后融合在一块形成字符串，将 target0-5 设置为标签，做 NLP 的文本分类任务处理。

2 算法模型介绍

这个部分主要介绍我所使用到的模型和算法，我使用了两种模型，第一种是 BERT 模型加 TextCNN 模型，第二种是 BERT 加 GRU 模型，我将分为三个部分 BERT，CNN 和 GRU 分别介绍。

2.1 BERT

BERT 模型的全称是 Bidirectional Encoder Representations from Transformer，基于 Transformer 的双向编码器表示，是一个预训练的语言表征模型，它强调了不再像以往一样采用传统的单向语言模型或者把两个单向语言模型进行浅层拼接的方法进行预训练，而是采用新的 masked language model（MLM），以致能生成深度的双向语言表征。BERT 模型的目标是利用大规模无标注语料训练、获得文本的包含丰富语义信息的 Representation，即：文本的语义表示，然后将文本的语义表示在特定 NLP 任务中作微调，最终应用于该 NLP 任务[1]。煮个栗子，BERT 模型训练文本语义表示的过程就好比我们在高中阶段学习语数英、物化生等各门基础学科，夯实基础知识；而模型在特定 NLP 任务中的参数微调就相当于我们在大学期间基于已有基础知识、针对所选专业作进一步强化，从而获得能够应用于实际场景的专业技能。

以往的预训练模型的结构会受到单向语言模型（从左到右或者从右到左）的限制，因而也限制了模型的表征能力，使其只能获取单方向的上下文信息。而 BERT 利用 MLM 进行预训练并且采用深层的双向 Transformer 组件（单向的 Transformer 一般被称为 Transformer decoder，其每一个 token（符号）只会 attend 到目前往左的 token。而双向的 Transformer 则被称为 Transformer encoder，其每一个 token 会 attend 到所有的 token。）来构建整个模型，因此最终生成能融合左右上下文信息的深层双向语言表征。[1]

当隐藏了 Transformer 的详细结构，将其当作一个黑箱后，多个 Transfomer 的组合可以用下图表示：

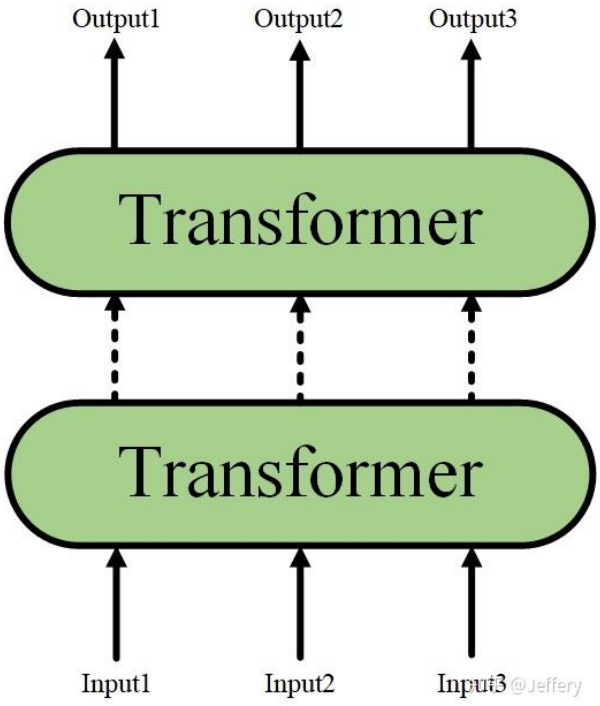

图 1 Transfomer 堆叠图

最终，经过多层 Transformer 结构的堆叠后，形成 BERT 的主体结构：

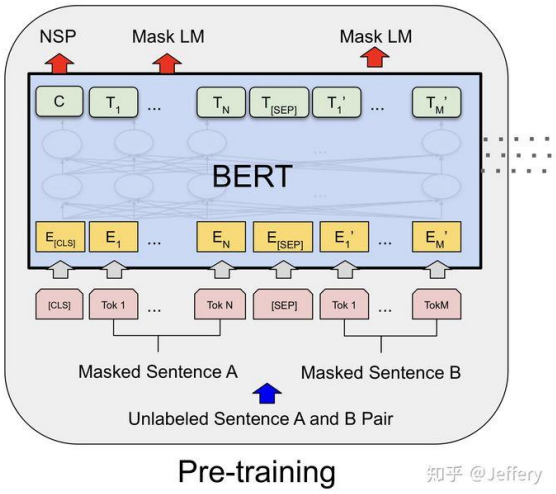

图 2 BERT 主体结构

BERT 的输入为每一个 token 对应的表征（图中的粉红色块就是 token，黄色块就是 token 对应的表征），并且单词字典是采用 WordPiece 算法来进行构建的。为了完成具体的分类任务，除了单词的 token 之外，作者还在输入的每一个序列开头都插入特定的分类 token（[CLS]），该分类 token 对应的最后一个 Transformer 层输出被用来起到聚集整个序列表征信息的作用。由于 BERT 是一个预训练模型，其必须要适应各种各样的自然语言任务，因此模型所输入的序列必须有能力包含一句话（文本情感分类，序列标注任务）或者两句话以上（文本摘要，自然语言推断，问答任务），因此 BERT 加入了[CLS]符号，其 embedding 代表整个句子的信息以及[SEP]符号用于分割句子。

介绍完 BERT 的输入，实际上 BERT 的输出也就呼之欲出了，因为 Transformer 的特点就是有多少个输入就有多少个对应的输出。如图 2 所示：C 为分类 token（[CLS]）对应最后一个 Transformer 的输出，T_i 则代表其他 token 对应最后一个 Transformer 的输出。对于一些 token 级别的任务（如，序列标注和问答任务），就把 T_i 输入到额外的输出层中进行预测。对于一些句子级别的任务（如，自然语言推断和情感分类任务），就把 C 输入到额外的输出层中，这里也就解释了为什么要在每一个 token 序列前都要插入特定的分类 token。

虽然 NLP 领域没有像 ImageNet 这样质量高的人工标注数据，但是可以利用大规模文本数据的自监督性质来构建预训练任务。因此 BERT 构建了两个预训练任务，分别是 Masked Language Model 和 Next Sentence Prediction。并通过这种自监督的方式来让模型学习得文本自身的特征，以更好的为下一步特定任务的训练提供良好的基础。

2.2 TextCNN

在说 TextCNN 前，先介绍一下传统的 CNN。卷积神经网络 CNN 与普通神经网络的区别在于，卷积神经网络包含了一个由卷积层和子采样层（池化层）构成的特征抽取器[2]。在卷积神经网络的卷积层中，一个神经元只与部分邻层神经元连接。在 CNN 的一个卷积层中，通常包含若干个特征图(feature Map)，每个特征图由一些矩形排列的的神经元组成，同一特征图的神经元共享权值，这里共享的权值就是卷积核。卷积核一般以随机小数矩阵的形式初始化，在网络的训练过程中卷积核将学习得到合理的权值。共享权值（卷积核）带来的直接好处是减少网络各层之间的连接，同时又降低了过拟合的风险。子采样也叫做池化（pooling），通常有均值子采样（mean pooling）和最大值子采样（max pooling）两种形式。子采样可以看作一种特殊的卷积过程。卷积和子采样大大简化了模型复杂度，减少了模型的参数。[2]

卷积神经网络主要由这几类层构成：输入层、卷积层，ReLU 层、池化（Pooling）层和全连接层（全连接层和常规神经网络中的一样）。通过将这

些层叠加起来，就可以构建一个完整的卷积神经网络。在实际应用中往往将卷积层与 ReLU 层共同称之为卷积层，所以卷积层经过卷积操作也是要经过激活函数的。具体说来，卷积层和全连接层（CONV/FC）对输入执行变换操作的时候，不仅会用到激活函数，还会用到很多参数，即神经元的权值 w 和偏差 b；而 ReLU 层和池化层则是进行一个固定不变的函数操作。卷积层和全连接层中的参数会随着梯度下降被训练，这样卷积神经网络计算出的分类评分就能和训练集中的每个图像的标签吻合了。

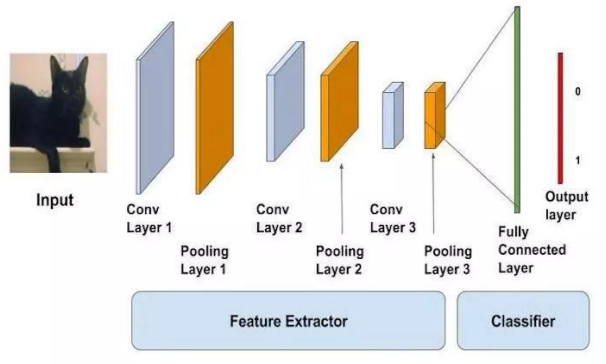

图 3 传统 CNN 处理流程

Yoon Kim[3] 在论文 (2014 EMNLP) Convolutional Neural Networks for Sentence Classification 提出 TextCNN。将卷积神经网络 CNN 应用到文本分类任务，利用多个不同 size 的 kernel 来提取句子中的关键信息（类似于多窗口大小的 n-gram），从而能够更好地捕捉局部相关性。

每一个单词的 embedding 固定，所以 kernel size 的宽度不变，只能改变高度。kernel 的通道可以理解为用不同的词向量表示。输入句子的长度不一样，但卷积核的个数一样。每个卷积核抽取单词的个数不一样，高度低的形成 feature maps 长度就较长，高度高的形成 feature maps 的长度就较短，但 feature maps 的长度不影响后面的输入，因为通过 Max-over-time pooling 层后，每一个 feature maps 都只取一个最大值，所以最终形成的向量与卷积核的个数一致。

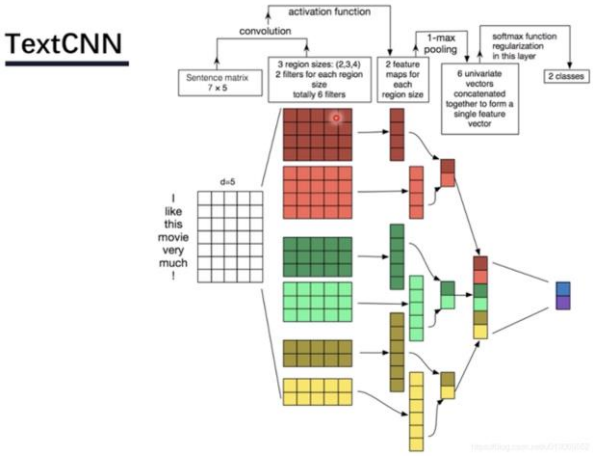

图 4 TextCNN 处理文本流程

2.3 GRU

GRU 输入输出的结构与普通的 RNN 相似，其中的内部思想与 LSTM 相似。与 LSTM 相比，GRU 内部少了一个门控，参数比 LSTM 少，但是却也能够达到与 LSTM 相当的功能。GRU 是 LSTM 的一个变种，也是为了解决长期记忆和反向传播中的梯度等问题而提出来的。GRU 和 LSTM 在很多情况下实际表现上相差无几，但是 GRU 计算更简单，更易于实现，其结构如下图所示：

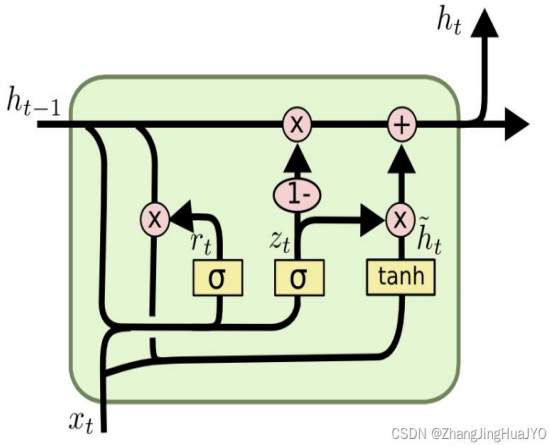

图 5 GRU 内部结构

GRU 只有两个门。GRU 将 LSTM 中的输入门和遗忘门合二为一，称为更新门（update gate），上图中的 z_t，控制前边记忆信息能够继续保留到当前时刻的数据量，或者说决定有多少前一时间步的信息和当前时间步的信息要被继续传递到未来；GRU 的另一个门称为重置门（reset gate），上图中的 r_t，控制要遗忘多少过去的信息。[4]概括来说，LSTM 和 CRU 都是通过各种门函数来将重要特征保留下来，这样就保证了在 long-term 传播的时候也不会

丢失。此外 GRU 相对于 LSTM 少了一个门函数，因此在参数的数量上也是要少于 LSTM 的，所以整体上 GRU 的训练速度要快于 LSTM 的。不过对于两个网络的好坏还是得看具体的应用场景。

3 算法模型实现及伪代码

3.1 算法模型实现

我使用 python3.7 编写本次报告的代码。首先用 python 来进行模型的建立和算法的实现需要导入一些必要的包利于计算，并且避免代码的冗余我会使用 pytorch1.11 作为框架进行代码的编写。

导入包之后，利用 pandans 库带的函数打开训练集数据，并对其进行预处理。在这里我针对数据特征本身的信息量做了简单处理，考虑到 description 中含有较多的信息，所以我将没有 description 特征的数据行删除。其次就是对其他特征空行的处理。对于数值型特征，如 review_score 我将缺失值都设置为 0，因为没有人打分只能假设分数较低不太受欢迎。其次，我将其他字符型数据的缺失值利用 BERT 词表中的[UNK]表示未知。在对缺失值进行处理后，我将这 16 个特征中的字符型特征使用 pytorch_pretrained_bert 中自带的分词操作进行分词，并将数值特征转换成字符串形式（这里对 latitude 和 longitude 的操作是取其小数点后两位，因为隔行的这两个特征值的小数点前几乎都相同），最后将 16 个特征进行拼接处理，组成一个较长的含有所有特征信息的分好词的结构。最后由 BERT 输入要求在头部和尾部加上[CLS]和[SEP]符号。

然后是对标签，即训练集 target 的处理，这个处理就很简单，直接将 target 值放入一个和上面对每个样本分词一一对应的一个列表中，缺失值统一当 0 处理。

由于 BERT 最长读入的每个样本的分词数量是 512，而少部分样本的分词数量超过了这个值，所以经过多次实验和折中，我选取 300 作为最终最长的分数数量，如果没有达到 300 我将 BERT 的填充符号[PAD]加入到分词尾部。由于用到了 BERT 预训练模型，就要将上面得到的分词信息转化为 tokens 即用 long 型数字代表分词形式输入到 BERT 才能得到每个分词的输出向量，所以韩式利用 pytorch_pretrained_bert 库中的操作进行 token 化，然后输入到 BERT 模型中得到每一行样本的向量表示。

如果只用 BERT 得到的最终每句话的表示输入到全连接层后得到分类结果效果不是很好，所以在 BERT 添加上部分介绍的 TextCNN，此时每一行样本的表示会是比较长的形式，将 BERT 得到的每一行样本的向量表示然后交给 TextCNN 处理，将句子中的所有向量浓缩为设定的一个特定维度的向量，再输入到线性层，输出为标签类别个数 6，向量的每个值都代表各个标签的一个预测概率值，我们会取最大的那个作为预测的标签。

训练的损失函数为预测值的 softmax 处理后和真实标签求交叉熵，优化器为 AdamW，其参数为默认值，在 1080 显卡上训练。在调试参数的过程中，为了避免模型过于复杂，就将 BERT 参数的梯度关闭了，只对 TextCNN 或 GRU 的参数进行梯度下降。

3.2 算法伪代码

算法伪代码

train_tokens = [], train_labels = [], valid_tokens = [], valid_labels = [], test_tokens=[]

初始化的 BERT+TextCNN/GRU 模型 $M(\theta)$，θ 表示模型参数

读取训练数据 *train_datas*

train_datas 预处理并更新

#分词得到 BERT 的输入

for k ← 1, 2, ..., N do

train_datas['features'][k]='[CLS]'+ train_datas[k] + '[SEP]'

x = tokenize(train_datas[k])

if len(x)>300

x = x[0:300]

#padding 处理

while len(x)<300

x = x + '0'

y = train_datas['target'][k]

train_tokens ← x ∪tokens

train_labels ← y ∪labels

#对验证集和测试集处理相同

end

将 tokens 和对应 labels 装入 batch 中得到 *Dataload*

定义交叉熵损失 *HCE(.)*，优化器 *O(.)*
#输入模型
for epoch ← 1, 2, ... do
for b ← 1, 2, ... do
logits ← M(Dataload[b]; θ)
loss ← HCE(logits, Dataload[b])
θ ← O(θ)
end
end
得到训练好的模型 *M(θ')*
prob ← M(valid_tokens;θ')
labels_predict ← argmax(prob)
acc ← sum(labels_predict == valid_labels) / len(valid_labels)
选取效果最好的训练模型 *M(θ'')*

prob ← M(test_tokens;θ')
labels_predict ← argmax(prob)
储存结果

4 结果描述和分析

实验过程中，对训练数据集做如下划分：选取前 12724 个数据训练模型，剩余的数据作为验证集，对于 BERT-TextCNN，选择卷积核大小分别为 2,3,4，卷积个数为 256 个，dropout 设置为 0.5，并且尾部设置两个线性层，当训练 5 个 epoch 时，采用 AdamW 优化器，参数默认，交叉熵损失函数，batch_size 设置为 128，训练集精度能达到 56.72%，验证集精度达到 56.36%，没有发生过拟合。如果采用 Adam 优化器，参数默认，则训练集精度 55.98%，验证集精度 55.78%。

虽然损失一直在下降，训练集精度也一直在上升，但是如果继续训练，就会发生过拟合，即训练集精度上升，但是验证集精度下降，这可能是因为过度学习到了训练集本身的一些诸如顺序之类的特征导致的，所以我最终选择了只训练 5 个 epoch，总训练时间接近 15 分钟。

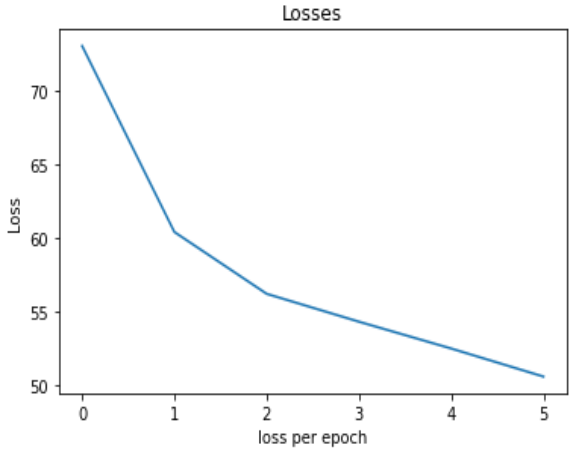

图 6 Bert_TextCNN 训练损失

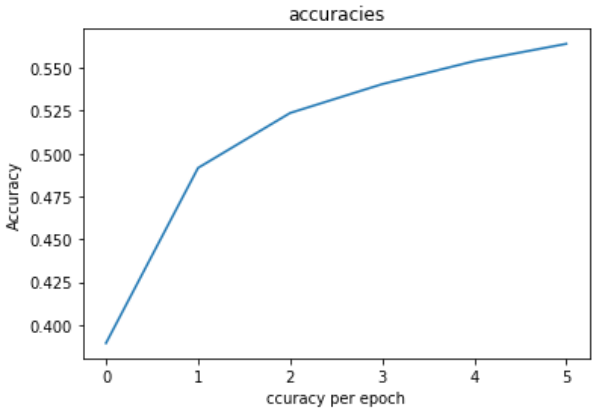

图 7 Bert_TextCNN 训练精度

其次是 Bert+GRU 的结果，Bert+GRU 结果最终和 Bert+TextCNN 差不多，这也侧面印证了用 NLP 方法处理，也许当前的极限便是 60%（我最高达到过 60%），是否能达到该极限和模型的初始化，或者是随机数相关。Bert+GRU 模型，除了模型设置之外其他设置和 Bert+TextCNN 一致，BertGRU 设置为双向 RNN，并且隐藏层大小为 256（双层为 512），共两层隐藏层。由于 Bert+GRU 训练速度要比 Bert+TextCNN 慢，所以我训练了 7 个 epoch，总训练时间接近 20 分钟。训练 7 个 epoch，训练集精度可达 54.49%，验证集精度为 54.29%，再训练就会过拟合了，结果如下，由于训练 5 个 epoch 效果交叉，所以

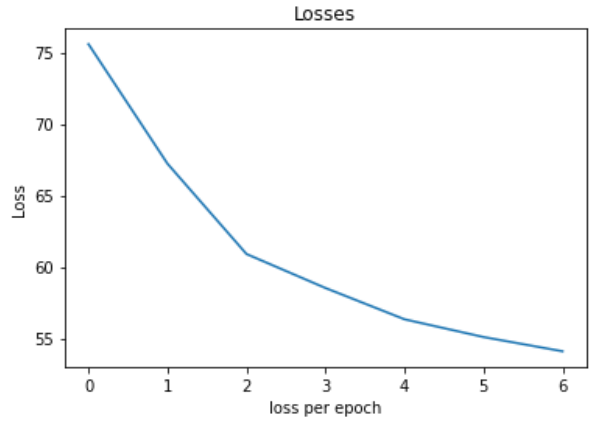

图 8 Bert_GRU 训练损失

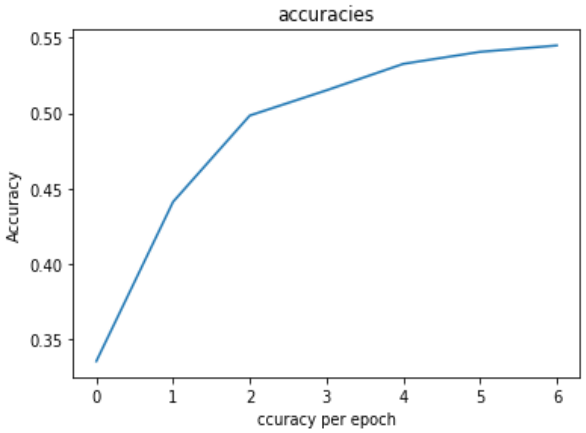

图 9 Bert_GRU 训练精度

最后，我最终选择训练 5 个 epoch 的 Bert+CNN 模型作为最终的模型预测测试集，这里我将所有训练数据完整的训练模型，没用划分验证集后的训练集，以增加泛化性，然后保存结果为 txt 文件。

5 结论和讨论

本次实验对了一个现实生活中的短期旅行租房数据集进行了模型建立和训练，意在解决实际问题。本数据集的 16 个特征都各自含有特别的信息，在实验的过程中只有不断地尝试才能选取最佳合适的特征来进行模型训练。由于科研导致时间紧迫，所以我直接选取了所有特征放入模型训练。不过我也做出了一些尝试，比如先开始我只用 description 特征喂入模型当成 BERT+TextCNN/GRU 的文本分类处理，但是发现训练集中划分出来的验证集精度在训练 5 个 epoch 之后精度就不能再提升了，只能接近到 50%.之后，我又融入了 amenities 特征，发现没有明显的提升，最后我融入了所有特征，并且第一次先将四个评论分数的和作为一个 token，这样发现效果提升了几个百分点，然后我将四个评论分数分别拆分作为 4 个独立的 tokens，再次训练，发现这才能让验证集精度达到 60%.所以尝试过后，我认为比较重要的特征可能就在于四个评论分数以及 description 当中。

之后我尝试增大训练 epoch 数量，发现随着 epoch 的增加，模型能够学习到的训练集知识越来越多，可以达到 90%以上，但是验证集上的精度反而会下降到 56%，这说明多次训练会导致过拟合问题，学习到训练集本身的如顺序之类的知识，从而模型泛化性能较差，所以 epoch 数量也要控制在 10 以内。

最后，我对实验的其他可行性进行了思考和讨论，如果想要进一步提高测试集/验证集精度，也许可以对 16 个特征进行良好的其他预处理，并且对重要的特征设置权重，以达到其和 target 标签强相关或者弱相关的关系，以此来更好的学习到有益的知识，并且不至于过拟合，来达到更好的效果。总之，通过本次实验，我受益颇多。

参 考 文 献

[1] Devlin J , Chang M W , Lee K , et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. 2018.

[2] Szegedy C , Liu W , Jia Y , et al. Going Deeper with Convolutions[J]. IEEE Computer Society, 2014.

[3] Kim Y . Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.

[4] Cho K , Merrienboer B V , Gulcehre C , et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation[J]. Computer Science, 2014.