Lab3 BERT

和泳毅 PB19010450

一、实验目标

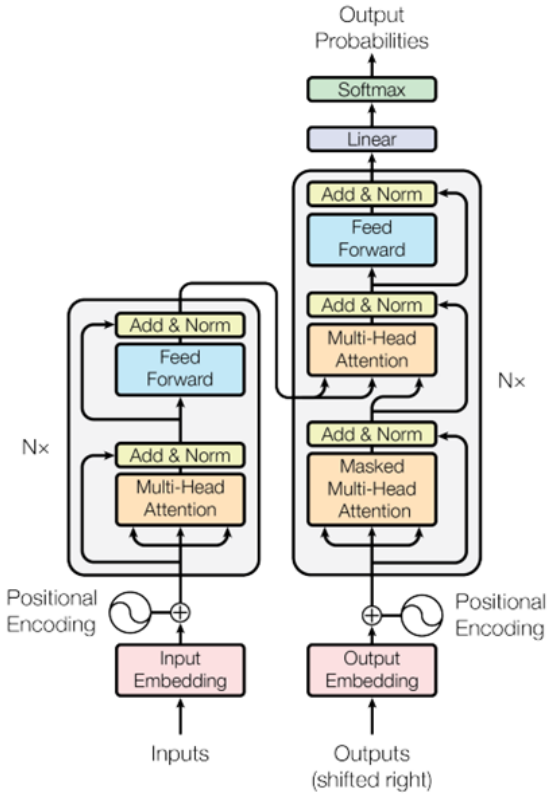
1. 利用实验二的数据编写BERT的语言模型，并基于训练好的词向量，利用少量的训练数据，微调BERT模型用于文本分类；
2. 调整训练集大小、MAXLEN、学习率等参数对比结果；
3. 与RNN、LSTM、GRU的结果对比。

二、数据集介绍

Tiny Imagenet是斯坦福大学提供的图像分类数据集，其中包含200个类别，每个类别包含500张训练图像，50张验证图像及50张测试图像。

三、实验原理

1.Transformer



模型大致分为Encoder和Decoder两个部分，分别对应上图中的左右两部分。

其中Encoder由N个相同的层堆叠在一起，每一层又有两个子层。第一个子层是一个Multi-Head Attention(多头的自注意机制)，第二个子层是一个简单的Feed Forward(全连接前馈网络)。两个子层都添加了一个residual和layer normalization的操作。

模型的Decoder同样是堆叠了N个相同的层，不过和编码器中每层的结构稍有不同。对于解码器的每一层，除了编码器中的两个子层Multi-Head Attention和Feed Forward，解码器还包含一个子层Masked Multi-Head Attention，如图中所示每个子层同样也用了residual以及layer normalization。

模型的输入由Input Embedding和Positional Encoding(位置编码)两部分组合而成，模型的输出由Decoder的输出简单的经过softmax得到。

1.2 模型输入

输入部分包含两个模块，Embedding和Positional Encoding。

Embedding层

Embedding层的作用是将某种格式的输入数据，例如文本，转变为模型可以处理的向量表示，来描述原始数据所包含的信息。Embedding层输出的可以理解为当前时间的特征，如果是文本任务，这里就可以是Word Embedding，如果是其他任务，就可以是任何合理方法所提取的特征。

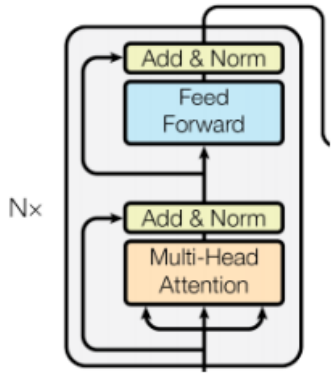
Positional Encodding

Positional Encodding的作用是为模型提供当前时间步的前后出现顺序的信息。因为Transformer不像RNN那样的循环结构有前后不同时间步输入间天然的先后顺序，所有的时间步是同时输入，并行推理的，因此在时间步的特征中融合进位置编码的信息是合理的。

1.3 Encoder

编码器作用是用于对输入进行特征提取，为解码环节提供有效的语义信息每个编码器层由两个子层连接结构组成：

第一个子层包括一个多头自注意力层和规范化层以及一个残差连接；第二个子层包括一个前馈全连接层和规范化层以及一个残差连接。如下图所示：



注意力计算：它需要三个指定的输入Q（query），K（key），V（value），然后通过下面公式得到注意力的计算结果：

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

EncoderLayer中另一个核心的子层是 Feed Forward Layer，在进行了Attention操作之后，encoder和decoder中的每一层都包含了一个全连接前向网络，对每个position的向量分别进行相同的操作，包括两个线性变换和一个ReLU激活输出：

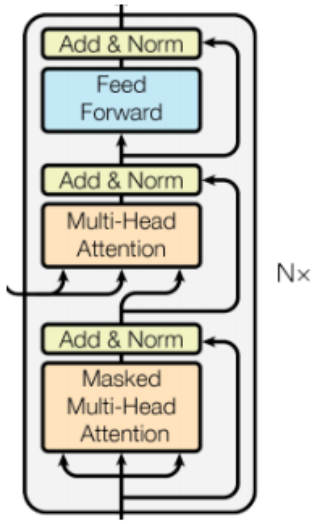
$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Attention模块每个时间步的输出都整合了所有时间步的信息，而Feed Forward Layer每个时间步只是对自己的特征的一个进一步整合，与其他时间步无关。

1.4 Decoder

解码器的作用：根据编码器的结果以及上一次预测的结果，输出序列的下一个结果。

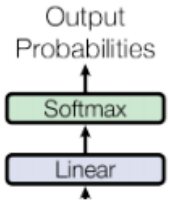
每个解码器层由三个子层连接结构组成，第一个子层连接结构包括一个多头自注意力子层和规范化层以及一个残差连接，第二个子层连接结构包括一个多头注意力子层和规范化层以及一个残差连接，第三个子层连接结构包括一个前馈全连接子层和规范化层以及一个残差连接。



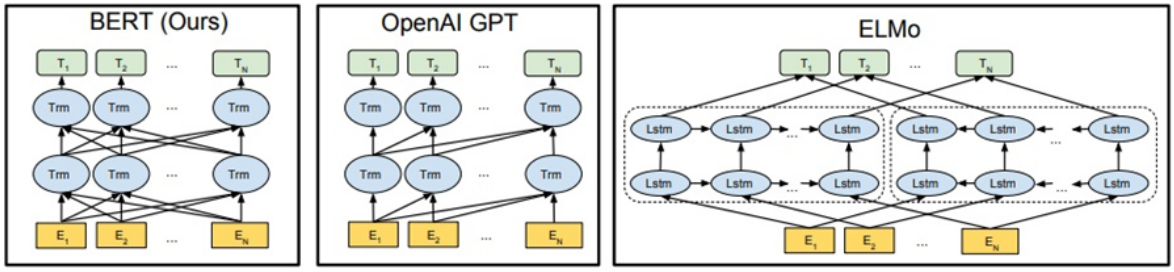
解码器层中的各个子模块，都与编码器中的实现相同。

1.5 模型输出

每个时间步都过一个 线性层 + softmax层：



2、BERT



BERT是双向的Transformer block连接。

2.1 Masked LM

第一步预训练的目标就是做语言模型，如果使用预训练模型处理其他任务，那人们想要的肯定不止某个词左边的信息，而是左右两边的信息。而考虑到这点的模型ELMo只是将left-to-right和right-to-left分别训练拼接起来。直觉上来讲我们其实想要一个deeply bidirectional的模型，但是普通的LM又无法做到，因为在训练时可能会“穿越”。所以作者用了一个加mask的做法。

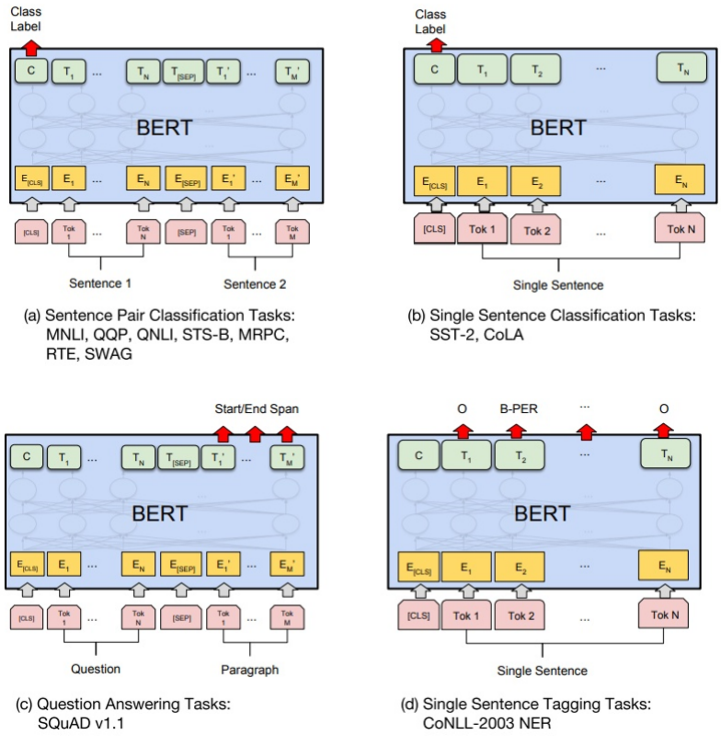
Mask如何做也是有技巧的，如果一直用标记[MASK]代替（在实际预测时是碰不到这个标记的）会影响模型，所以随机mask的时候10%的单词会被替代成其他单词，10%的单词不替换，剩下80%才被替换为[MASK]。要注意的是Masked LM预训练阶段模型是不知道真正被mask的是哪个词，所以模型每个词都要关注。

2.2 Next Sentence Prediction

因为涉及到QA和NLI之类的任务，增加了第二个预训练任务，目的是让模型理解两个句子之间的联系。训练的输入是句子A和B，B有一半的几率是A的下一句，输入这两个句子，模型预测B是不是A的下一句。

2.3 Fine-tunning

分类：对于sequence-level的分类任务，BERT直接取第一个[CLS]token的final hidden state，加一层权重后softmax预测概率。其他预测任务需要进行一些调整，如图：



四、核心代码

1. 数据读入

```
(ds_train, ds_test), ds_info = tfds.load('imdb_reviews',
                                         split = (tfds.Split.TRAIN, tfds.Split.TEST),
                                         as_supervised=True,
                                         with_info=True)
```

2. Tokenization

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
```

3. 参数设置

```
max_length = 300
batch_size = 16
learning_rate = 2e-5
number_of_epochs = 3
```

4. 数据集处理

```
def convert_example_to_feature(review):
    return tokenizer.encode_plus(review,
                add_special_tokens = True, # add [CLS], [SEP]
                max_length = max_length, # max length of the text that can go to BERT
                pad_to_max_length = True, # add [PAD] tokens
                return_attention_mask = True, # add attention mask to not focus on pad tokens
              )

def map_example_to_dict(input_ids, attention_masks, token_type_ids, label):
  return {
      "input_ids": input_ids,
      "token_type_ids": token_type_ids,
      "attention_mask": attention_masks,
  }, label

def encode_examples(ds, limit=-1):
  input_ids_list = []
  token_type_ids_list = []
  attention_mask_list = []
  label_list = []

  if (limit > 0):
      ds = ds.take(limit)

  for review, label in tfds.as_numpy(ds):

    bert_input = convert_example_to_feature(review.decode())
    input_ids_list.append(bert_input['input_ids'])
```

```
    token_type_ids_list.append(bert_input['token_type_ids'])
    attention_mask_list.append(bert_input['attention_mask'])
    label_list.append([label])

  return tf.data.Dataset.from_tensor_slices((input_ids_list, attention_mask_list, token_type_ids_list, label_list)).map(map_example_to_dict)

train_num = 5000
ds_train_encoded = encode_examples(ds_train,train_num).shuffle(10000).batch(batch_size)
ds_val_encoded = encode_examples(ds_train.skip(train_num)).shuffle(10000).batch(batch_size)

ds_test_encoded = encode_examples(ds_test).batch(batch_size)

```

5. 模型设置

```
model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased')

optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, epsilon=1e-08)

loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')

model.compile(optimizer=optimizer, loss=loss, metrics=[metric])

```

五、实验结果

统一设置参数：

```
max_length = 300
batch_size = 16
learning_rate = 2e-5
number_of_epochs = 3
train_num = 5000
```

训练样本数量

训练样本数	训练集acc	验证集acc
2500	0.9683	0.9024
5000	0.9710	0.9054
10000	0.9755	0.9103

增大训练样本数量对正确率的影响甚微，反而回极大地增加训练时长，相比之下选取合适数量的小样本来训练性价比更高。

学习率

学习率	训练集acc	验证集acc
2e-5	0.9710	0.9054
1e-5	0.9698	0.9051
1e-4	0.9651	0.8902

学习率对结果的影响不是很大。

MAXLEN

MAXLEN	训练集acc	验证集acc
300	0.9710	0.9054
200	0.9622	0.8912
100	0.9518	0.8518

由于文本长度不小，所以MAXLEN取得过小对结果影响很大，而MAXLEN取得过大也会增加训练时间，所以MAXLEN的取值也要合适。

与RNN、LATM、GRU对比

模型	训练集acc	验证集acc
RNN	0.7952	0.7822
LSTM	0.9093	0.8742
GRU	0.9352	0.8814
BERT	0.9755	0.9103

可见BERT的效果要比其他模型更好，且比复杂网络训练的时间更少。

测试

选取参数：

```
max_length = 300
batch_size = 16
learning_rate = 2e-5
number_of_epochs = 3
train_num = 10000
```

正确率为0.9134。

六、实验总结

通过本次实验可以看到，BERT这种预训练模型在处理文本分类上有着很大的优势，可以使用少量样本进行训练，只需要进行微调即可获得非常好的结果，并且其训练时间要比LSTM等复杂网络少。