

# 1 安装工具库和下载BERT中文权重

In [1]:

```
1 !pip install git+https://www.github.com/bojone/bert4keras.git
```

```
Collecting git+https://www.github.com/bojone/bert4keras.git
  Cloning https://www.github.com/bojone/bert4keras.git (https://www.github.com/bojone/bert4keras.git) to /tmp/pip-req-build-wso6kd0q
  Running command git clone -q https://www.github.com/bojone/bert4keras.git (https://www.github.com/bojone/bert4keras.git) /tmp/pip-req-build-wso6kd0q
Requirement already satisfied: keras in /usr/local/lib/python3.6/dist-packages (from bert4keras==0.8.1) (2.3.1)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6/dist-packages (from keras->bert4keras==0.8.1) (1.1.2)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras->bert4keras==0.8.1) (2.10.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (from keras->bert4keras==0.8.1) (3.13)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages (from keras->bert4keras==0.8.1) (1.18.5)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.6/dist-packages (from keras->bert4keras==0.8.1) (1.4.1)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from keras->bert4keras==0.8.1) (1.12.0)
Requirement already satisfied: keras-applications>=1.0.6 in /usr/local/lib/python3.6/dist-packages (from keras->bert4keras==0.8.1) (1.0.8)
Building wheels for collected packages: bert4keras
  Building wheel for bert4keras (setup.py) ... done
  Created wheel for bert4keras: filename=bert4keras-0.8.1-cp36-none-any.whl size=41579 sha256=def13bd0f4aff59b84173354e24c6959734fa75888e38fb39087ab4a2b1435a8
  Stored in directory: /tmp/pip-ephem-wheel-cache-lr2ci8ya/wheels/12/58/83/8ff5c864b80c860e6d9e9e0d90c04fafca05d01d21f9f6fcb
Successfully built bert4keras
Installing collected packages: bert4keras
Successfully installed bert4keras-0.8.1
```

In [2]:

```
1 !wget https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip
```

```
--2020-06-18 12:30:00-- https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip (https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip)
Resolving storage.googleapis.com (storage.googleapis.com)... 64.233.189.128, 2404:6800:4008:c04::80
Connecting to storage.googleapis.com (storage.googleapis.com)|64.233.189.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 381892918 (364M) [application/zip]
Saving to: 'chinese_L-12_H-768_A-12.zip'

chinese_L-12_H-768_ 100%[=====>] 364.20M  55.3MB/s   in 6.6s

2020-06-18 12:30:06 (55.3 MB/s) - 'chinese_L-12_H-768_A-12.zip' saved [381892918/381892918]
```

In [3]:

```
1 !ls
```

chinese\_L-12\_H-768\_A-12.zip sample\_data

In [4]:

```
1 !unzip chinese_L-12_H-768_A-12.zip
```

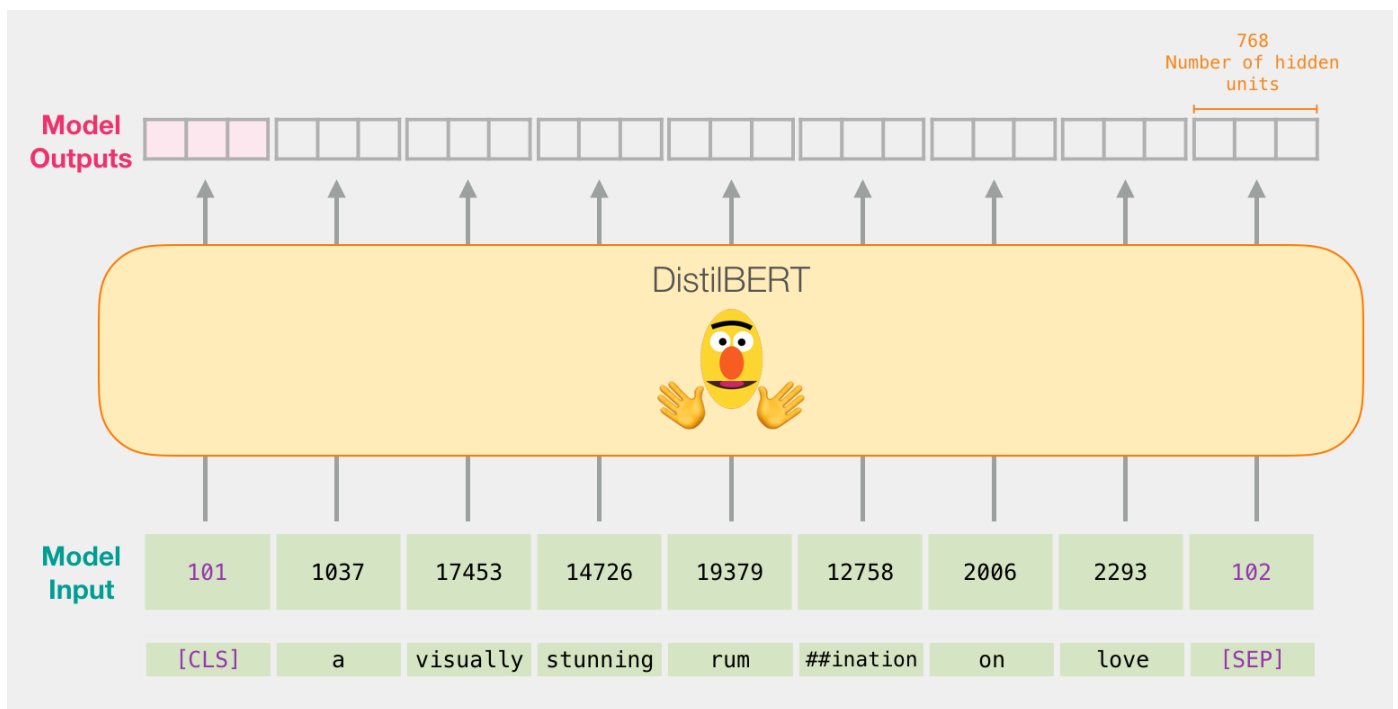
Archive: chinese\_L-12\_H-768\_A-12.zip  
creating: chinese\_L-12\_H-768\_A-12/  
inflating: chinese\_L-12\_H-768\_A-12/bert\_model.ckpt.meta  
inflating: chinese\_L-12\_H-768\_A-12/bert\_model.ckpt.data-00000-of-00001  
inflating: chinese\_L-12\_H-768\_A-12/vocab.txt  
inflating: chinese\_L-12\_H-768\_A-12/bert\_model.ckpt.index  
inflating: chinese\_L-12\_H-768\_A-12/bert\_config.json

In [6]:

```
1 !ls chinese_L-12_H-768_A-12
```

bert\_config.json bert\_model.ckpt.index vocab.txt  
bert\_model.ckpt.data-00000-of-00001 bert\_model.ckpt.meta

## 2 文本特征抽取



In [2]:

```
1 from bert4keras.backend import keras
2 from bert4keras.models import build_transformer_model
3 from bert4keras.tokenizers import Tokenizer
4 import numpy as np
5
6 config_path = './chinese_L-12_H-768_A-12/bert_config.json'
7 checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
8 dict_path = './chinese_L-12_H-768_A-12/vocab.txt'
9
10 tokenizer = Tokenizer(dict_path, do_lower_case=True) # 建立分词器
11 model = build_transformer_model(config_path, checkpoint_path) # 建立模型, 加载权重
```

Using TensorFlow backend.

In [3]:

```
1 # 编码测试
2 token_ids, segment_ids = tokenizer.encode(u'北京最近的疫情情况得到了有效的控制')
3
4 print('\n ===== predicting =====\n')
5 print(model.predict([np.array([token_ids]), np.array([segment_ids])]))
```

===== predicting =====

```
[[[-0.3788255 -0.03549163 -0.23232993 ... 0.39397225 0.25821406
    0.7239427 ]
  [-0.73081243 0.4969071 0.5336969 ... -0.89273274 0.11071089
    0.10646826]
  [-0.15799534 0.20846073 0.32380506 ... 0.24519442 1.0479465
    0.18938088]
  ...
  [ 0.61453784 0.58808786 -0.41544795 ... 0.01150814 1.0123017
    0.62400264]
  [ 0.7205164 0.6138239 0.76781553 ... 0.26691562 -0.10644234
   -0.09748168]
  [-0.17477396 -0.6593223 0.0176075 ... 0.23238292 0.40088922
    0.41288805]]]
```

In [4]:

```
1
2 print('\n ===== reloading and predicting =====\n')
3 model.save('bert.model')
4 del model
5 model = keras.models.load_model('bert.model')
6 print(model.predict([np.array([token_ids]), np.array([segment_ids])]))
```

===== reloading and predicting =====

/usr/local/lib/python3.6/dist-packages/keras/engine/saving.py:341: UserWarning: No training configuration found in save file: the model was \*not\* compiled. Compile it manually.

warnings.warn('No training configuration found in save file: '

```
[[[-0.3788255 -0.03549163 -0.23232993 ...  0.39397225  0.25821406
   0.7239427 ]
 [-0.73081243  0.4969071  0.5336969 ... -0.89273274  0.11071089
   0.10646826]
 [-0.15799534  0.20846073  0.32380506 ...  0.24519442  1.0479465
   0.18938088]
 ...
 [ 0.61453784  0.58808786 -0.41544795 ...  0.01150814  1.0123017
   0.62400264]
 [ 0.7205164  0.6138239  0.76781553 ...  0.26691562 -0.10644234
  -0.09748168]
 [-0.17477396 -0.6593223  0.0176075 ...  0.23238292  0.40088922
   0.41288805]]]
```

In [5]:

```
1 !ls
```

bert.model chinese\_L-12\_H-768\_A-12 chinese\_L-12\_H-768\_A-12.zip sample\_data

### 3 文本分类

In [6]:

```
1 !wget https://github.com/bojone/bert4keras/raw/master/examples/datasets/sentiment.zip
```

```
--2020-06-18 12:43:38-- https://github.com/bojone/bert4keras/raw/master/examples/datasets/sentiment.zip (https://github.com/bojone/bert4keras/raw/master/examples/datasets/sentiment.zip)
```

```
Resolving github.com (github.com)... 140.82.112.4
```

```
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
```

```
HTTP request sent, awaiting response... 302 Found
```

```
Location: https://raw.githubusercontent.com/bojone/bert4keras/master/examples/datasets/sentiment.zip (https://raw.githubusercontent.com/bojone/bert4keras/master/examples/datasets/sentiment.zip) [following]
```

```
--2020-06-18 12:43:39-- https://raw.githubusercontent.com/bojone/bert4keras/master/examples/datasets/sentiment.zip (https://raw.githubusercontent.com/bojone/bert4keras/master/examples/datasets/sentiment.zip)
```

```
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133, 151.101.128.133, ...
```

```
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connected.
```

```
HTTP request sent, awaiting response... 200 OK
```

```
Length: 2936149 (2.8M) [application/zip]
```

```
Saving to: 'sentiment.zip'
```

```
sentiment.zip      100%[=====>]    2.80M  11.3MB/s   in 0.2s
```

```
2020-06-18 12:43:41 (11.3 MB/s) - 'sentiment.zip' saved [2936149/2936149]
```

In [7]:

```
1 !unzip sentiment.zip
```

```
Archive:  sentiment.zip
```

```
creating: sentiment/
```

```
inflating: sentiment/sentiment.test.data
```

```
inflating: sentiment/sentiment.train.data
```

```
inflating: sentiment/sentiment.valid.data
```

In [10]:

```
1 !ls
```

```
bert.model          chinese_L-12_H-768_A-12.zip  sentiment
chinese_L-12_H-768_A-12  sample_data                sentiment.zip
```

In [11]:

```
1 !head -10 sentiment/sentiment.train.data
```

贝贝好爱干净 每天出门都要洗澡 还喜欢喝蒙牛 不喜欢蹲地方 喜欢坐凳子上还喜欢和我坐在一起~ 1  
感觉好像是文科生看一本《高等数学》的教材一样，流水账一般，只是背景很好罢了，选择在这样一个竞争激烈的时代，写了那么一个催人奋进的故事，文笔不咋地。 0  
很安静，隔音设施不错. 服务员态度很好, 下次还会选这里 1  
1 感觉外观还可以，符合我的要求，体积虽不算小，但比它大的翻盖手机还是很多的。2 比一张IC卡比较要小，厚度也还可以。键盘很漂亮，按键键盘面是平的，屏幕一般不会碰到按键，设计的很好。3。。。一堆呢，不说了 1  
收到后，包装完好。笔记本封条完好。 性价比很高，DVD驱动盘包含VISTA所有必备的驱动，方便。 1  
《小狗钱钱》是我一个好朋友推荐并且送给我的，并且说就把它放在枕边 随时阅读，我抱着这样的想法一气呵成读完 觉得译者的语言翻译的很准确，没有多少翻译痕迹，（我通常不喜欢读翻译过来的书）并且语言形象生动，故事易于理解，令人身临其境，有不少借鉴意义，如果能够仔细的按照这个试试做做，（重在领会精神），会有不错的效果。正打算读第二遍。 推荐 1  
书的质量不好，翻一下就坏了，而且书的内容也不好，没有什么用，也不觉得好笑！根本就比不上“我是英语单词书”！！建议大家不要买！！ 0  
键盘很生硬，没有手感。 标配内存2G为两根1G的组成，很郁闷，当初没问清楚~~~（外面买很多都是单根2G，以后扩大内存不方便） 0  
蒙牛真果粒、美丽有新意。 1  
书本质量不错，但是感觉布局不是很合理，打开后感觉很乱，密密麻麻的，孩子也不喜欢它。 0

In [12]:

```
1 !wget https://storage.googleapis.com/albert_zh/albert_small_zh_google.zip
```

--2020-06-18 12:45:55-- https://storage.googleapis.com/albert\_zh/albert\_small\_zh\_google.zip (https://storage.googleapis.com/albert\_zh/albert\_small\_zh\_google.zip)  
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.204.128, 2404:6800:4008:c07::80  
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.204.128|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 53458815 (51M) [application/zip]  
Saving to: 'albert\_small\_zh\_google.zip'  
  
albert\_small\_zh\_goo 100%[=====>] 50.98M 38.0MB/s in 1.3s  
  
2020-06-18 12:45:57 (38.0 MB/s) - 'albert\_small\_zh\_google.zip' saved [53458815/53458815]

In [14]:

```
1 !ls
```

```
albert_small_zh_google.zip  chinese_L-12_H-768_A-12.zip  sentiment.zip  
bert.model                sample_data  
chinese_L-12_H-768_A-12   sentiment
```

In [15]:

```
1 !unzip albert_small_zh_google.zip
```

```
Archive:  albert_small_zh_google.zip
  inflating: albert_config_small_google.json
  inflating: albert_model.ckpt.data-00000-of-00001
  inflating: albert_model.ckpt.index
  inflating: albert_model.ckpt.meta
  inflating: checkpoint
  inflating: vocab.txt
```

In [16]:

```
1 !ls
```

```
albert_config_small_google.json      chinese_L-12_H-768_A-12
albert_model.ckpt.data-00000-of-00001 chinese_L-12_H-768_A-12.zip
albert_model.ckpt.index              sample_data
albert_model.ckpt.meta               sentiment
albert_small_zh_google.zip           sentiment.zip
bert.model                           vocab.txt
checkpoint
```

In [17]:

```
1 import numpy as np
2 from bert4keras.backend import keras, set_gelu
3 from bert4keras.tokenizers import Tokenizer
4 from bert4keras.models import build_transformer_model
5 from bert4keras.optimizers import Adam, extend_with_pieewise_linear_lr
6 from bert4keras.snippets import sequence_padding, DataGenerator
7 from bert4keras.snippets import open
8 from keras.layers import Lambda, Dense
9
10 set_gelu('tanh') # 切换gelu版本
11
12 num_classes = 2
13 maxlen = 128
14 batch_size = 32
15 config_path = './albert_config_small_google.json'
16 checkpoint_path = './albert_model.ckpt'
17 dict_path = './vocab.txt'
18
19
20 def load_data(filename):
21     D = []
22     with open(filename, encoding='utf-8') as f:
23         for l in f:
24             text, label = l.strip().split('\t')
25             D.append((text, int(label)))
26     return D
27
28
29 # 加载数据集
30 train_data = load_data('./sentiment/sentiment.train.data')
31 valid_data = load_data('./sentiment/sentiment.valid.data')
32 test_data = load_data('./sentiment/sentiment.test.data')
33
34 # 建立分词器
35 tokenizer = Tokenizer(dict_path, do_lower_case=True)
36
37
38 class data_generator(DataGenerator):
39     """数据生成器"""
40
41     def __iter__(self, random=False):
42         batch_token_ids, batch_segment_ids, batch_labels = [], [], []
43         for is_end, (text, label) in self.sample(random):
44             token_ids, segment_ids = tokenizer.encode(text, maxlen=maxlen)
45             batch_token_ids.append(token_ids)
46             batch_segment_ids.append(segment_ids)
47             batch_labels.append([label])
48             if len(batch_token_ids) == self.batch_size or is_end:
49                 batch_token_ids = sequence_padding(batch_token_ids)
50                 batch_segment_ids = sequence_padding(batch_segment_ids)
51                 batch_labels = sequence_padding(batch_labels)
52                 yield [batch_token_ids, batch_segment_ids], batch_labels
53                 batch_token_ids, batch_segment_ids, batch_labels = [], [], []
54
55
56 # 加载预训练模型
57 bert = build_transformer_model(
58     config_path=config_path,
59     checkpoint_path=checkpoint_path,
```



```

60     model='albert',
61     return_keras_model=False,
62 )
63
64 output = Lambda(lambda x: x[:, 0], name='CLS-token')(bert.model.output)
65 output = Dense(
66     units=num_classes,
67     activation='softmax',
68     kernel_initializer=bert.initializer
69 )(output)
70
71 model = keras.models.Model(bert.model.input, output)
72 model.summary()
73
74 # 派生为带分段线性学习率的优化器。
75 # 其中name参数可选，但最好填入，以区分不同的派生优化器。
76 AdamLR = extend_with_piecewise_linear_lr(Adam, name='AdamLR')
77
78 model.compile(
79     loss='sparse_categorical_crossentropy',
80     # optimizer=Adam(1e-5), # 用足够小的学习率
81     optimizer=AdamLR(learning_rate=1e-4, lr_schedule={
82         1000: 1,
83         2000: 0.1
84     }),
85     metrics=['accuracy'],
86 )
87
88 # 转换数据集
89 train_generator = data_generator(train_data, batch_size)
90 valid_generator = data_generator(valid_data, batch_size)
91 test_generator = data_generator(test_data, batch_size)
92
93
94 def evaluate(data):
95     total, right = 0., 0.
96     for x_true, y_true in data:
97         y_pred = model.predict(x_true).argmax(axis=1)
98         y_true = y_true[:, 0]
99         total += len(y_true)
100         right += (y_true == y_pred).sum()
101     return right / total
102
103
104 class Evaluator(keras.callbacks.Callback):
105     def __init__(self):
106         self.best_val_acc = 0.
107
108     def on_epoch_end(self, epoch, logs=None):
109         val_acc = evaluate(valid_generator)
110         if val_acc > self.best_val_acc:
111             self.best_val_acc = val_acc
112             model.save_weights('best_model.weights')
113         test_acc = evaluate(test_generator)
114         print(
115             u'val_acc: %.5f, best_val_acc: %.5f, test_acc: %.5f\n' %
116             (val_acc, self.best_val_acc, test_acc)
117         )
118
119
120 evaluator = Evaluator()

```

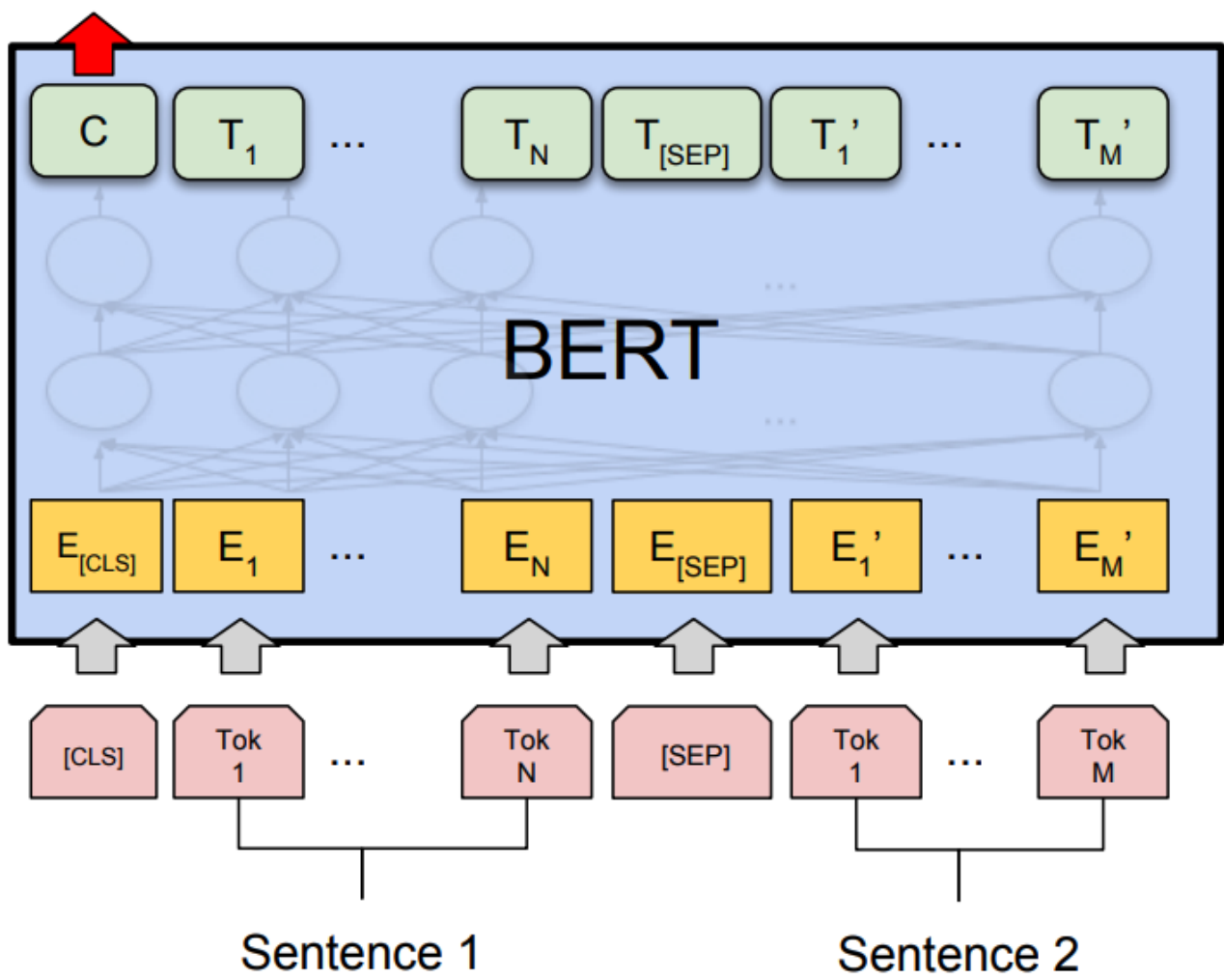
```
121 model.fit_generator(  
122     train_generator.forfit(),  
123     steps_per_epoch=len(train_generator),  
124     epochs=10,  
125     callbacks=[evaluator]  
126 )  
127  
128 model.load_weights('best_model.weights')  
129 print(u'final test acc: %05f\n' % (evaluate(test_generator)))
```

Model: "model\_3"

Layer (type)	Output Shape	Param #	Connected to
Input-Token (InputLayer)	(None, None)	0	
Input-Segment (InputLayer)	(None, None)	0	
Embedding-Token (Embedding)	(None, None, 128)	2704384	Input-Token[0][0]
Embedding-Segment (Embedding)	(None, None, 128)	256	Input-Segment[0][0]
Embedding-Token (Embedding)	(None, None, 128)	0	Embedding-Token[0][0]

## 4 文本相似度学习/句对建模任务

Class  
Label



In [18]:

```
1 !wget https://github.com/huawei-noah/Pretrained-Language-Model/raw/master/NEZHA-TensorFlow/da
2 !wget https://github.com/huawei-noah/Pretrained-Language-Model/raw/master/NEZHA-TensorFlow/da
3 !wget https://github.com/huawei-noah/Pretrained-Language-Model/raw/master/NEZHA-TensorFlow/da

--2020-06-18 13:00:22-- https://github.com/huawei-noah/Pretrained-Language-Model/
raw/master/NEZHA-TensorFlow/data/lcqmc/dev.tsv (https://github.com/huawei-noah/Pre
trained-Language-Model/raw/master/NEZHA-TensorFlow/data/lcqmc/dev.tsv)
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/huawei-noah/Pretrained-Language-Model/
master/NEZHA-TensorFlow/data/lcqmc/dev.tsv (https://raw.githubusercontent.com/huaw
ei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc/dev.tsv) [fol
lowing]
--2020-06-18 13:00:23-- https://raw.githubusercontent.com/huawei-noah/Pretrained-
Language-Model/master/NEZHA-TensorFlow/data/lcqmc/dev.tsv (https://raw.githubuserc
ontent.com/huawei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc
c/dev.tsv)
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133,
151.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7892 (7.7K) [text/plain]
Saving to: 'dev.tsv'

dev.tsv          100%[=====>]    7.71K  --.-KB/s    in 0s

2020-06-18 13:00:23 (59.9 MB/s) - 'dev.tsv' saved [7892/7892]

--2020-06-18 13:00:25-- https://github.com/huawei-noah/Pretrained-Language-Model/
raw/master/NEZHA-TensorFlow/data/lcqmc/test.tsv (https://github.com/huawei-noah/Pr
etrained-Language-Model/raw/master/NEZHA-TensorFlow/data/lcqmc/test.tsv)
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/huawei-noah/Pretrained-Language-Model/
master/NEZHA-TensorFlow/data/lcqmc/test.tsv (https://raw.githubusercontent.com/hua
wei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc/test.tsv) [f
ollowing]
--2020-06-18 13:00:26-- https://raw.githubusercontent.com/huawei-noah/Pretrained-
Language-Model/master/NEZHA-TensorFlow/data/lcqmc/test.tsv (https://raw.githubuser
content.com/huawei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc
/test.tsv)
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133,
151.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7892 (7.7K) [text/plain]
Saving to: 'test.tsv'

test.tsv         100%[=====>]    7.71K  --.-KB/s    in 0s

2020-06-18 13:00:26 (68.4 MB/s) - 'test.tsv' saved [7892/7892]

--2020-06-18 13:00:28-- https://github.com/huawei-noah/Pretrained-Language-Model/
raw/master/NEZHA-TensorFlow/data/lcqmc/train.tsv (https://github.com/huawei-noah/P
retrained-Language-Model/raw/master/NEZHA-TensorFlow/data/lcqmc/train.tsv)
```

```
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/huawei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc/train.tsv (https://raw.githubusercontent.com/huawei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc/train.tsv)
[following]
--2020-06-18 13:00:29-- https://raw.githubusercontent.com/huawei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc/train.tsv (https://raw.githubusercontent.com/huawei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lcqmc/train.tsv)
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34117 (33K) [text/plain]
Saving to: 'train.tsv'

train.tsv          100%[=====>]  33.32K  --.-KB/s    in 0.01s

2020-06-18 13:00:30 (2.39 MB/s) - 'train.tsv' saved [34117/34117]
```

In [19]:

```
1 !head -5 train.tsv
```

```
喜欢打篮球的男生喜欢什么样的女生      爱打篮球的男生喜欢什么样的女生  1
我手机丢了，我想换个手机      我想买个新手机，求推荐  1
大家觉得她好看吗      大家觉得跑男好看吗？  0
求秋色之空漫画全集      求秋色之空全集漫画  1
晚上睡觉带着耳机听音乐有什么害处吗？  孕妇可以戴耳机听音乐吗？ 0
```

In [20]:

```
1 import numpy as np
2 from bert4keras.backend import keras, set_gelu, K
3 from bert4keras.tokenizers import Tokenizer
4 from bert4keras.models import build_transformer_model
5 from bert4keras.optimizers import Adam
6 from bert4keras.snippets import sequence_padding, DataGenerator
7 from bert4keras.snippets import open
8 from keras.layers import Dropout, Dense
9
10 set_gelu('tanh') # 切换gelu版本
11
12 maxlen = 128
13 batch_size = 64
14 config_path = './chinese_L-12_H-768_A-12/bert_config.json'
15 checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
16 dict_path = './chinese_L-12_H-768_A-12/vocab.txt'
17
18
19 def load_data(filename):
20     D = []
21     with open(filename, encoding='utf-8') as f:
22         for l in f:
23             text1, text2, label = l.strip().split('\t')
24             D.append((text1, text2, int(label)))
25     return D
26
27
28 # 加载数据集
29 train_data = load_data('train.tsv')
30 valid_data = load_data('dev.tsv')
31 test_data = load_data('test.tsv')
32
33 # 建立分词器
34 tokenizer = Tokenizer(dict_path, do_lower_case=True)
35
36
37 class data_generator(DataGenerator):
38     """数据生成器"""
39
40     def __iter__(self, random=False):
41         batch_token_ids, batch_segment_ids, batch_labels = [], [], []
42         for is_end, (text1, text2, label) in self.sample(random):
43             token_ids, segment_ids = tokenizer.encode(
44                 text1, text2, maxlen=maxlen
45             )
46             batch_token_ids.append(token_ids)
47             batch_segment_ids.append(segment_ids)
48             batch_labels.append([label])
49             if len(batch_token_ids) == self.batch_size or is_end:
50                 batch_token_ids = sequence_padding(batch_token_ids)
51                 batch_segment_ids = sequence_padding(batch_segment_ids)
52                 batch_labels = sequence_padding(batch_labels)
53                 yield [batch_token_ids, batch_segment_ids], batch_labels
54                 batch_token_ids, batch_segment_ids, batch_labels = [], [], []
55
56
57 # 加载预训练模型
58 bert = build_transformer_model(
59     config_path=config_path,
```

```

60     checkpoint_path=checkpoint_path,
61     with_pool=True,
62     return_keras_model=False,
63 )
64
65 output = Dropout(rate=0.1)(bert.model.output)
66 output = Dense(
67     units=2, activation='softmax', kernel_initializer=bert.initializer
68 )(output)
69
70 model = keras.models.Model(bert.model.input, output)
71 model.summary()
72
73 model.compile(
74     loss='sparse_categorical_crossentropy',
75     optimizer=Adam(2e-5), # 用足够小的学习率
76     # optimizer=PiecewiseLinearLearningRate(Adam(5e-5), {10000: 1, 30000: 0.1}),
77     metrics=['accuracy'],
78 )
79
80 # 转换数据集
81 train_generator = data_generator(train_data, batch_size)
82 valid_generator = data_generator(valid_data, batch_size)
83 test_generator = data_generator(test_data, batch_size)
84
85
86 def evaluate(data):
87     total, right = 0., 0.
88     for x_true, y_true in data:
89         y_pred = model.predict(x_true).argmax(axis=1)
90         y_true = y_true[:, 0]
91         total += len(y_true)
92         right += (y_true == y_pred).sum()
93     return right / total
94
95
96 class Evaluator(keras.callbacks.Callback):
97     def __init__(self):
98         self.best_val_acc = 0.
99
100     def on_epoch_end(self, epoch, logs=None):
101         val_acc = evaluate(valid_generator)
102         if val_acc > self.best_val_acc:
103             self.best_val_acc = val_acc
104             model.save_weights('best_model.weights')
105         test_acc = evaluate(test_generator)
106         print(
107             u'val_acc: %.5f, best_val_acc: %.5f, test_acc: %.5f\n' %
108             (val_acc, self.best_val_acc, test_acc)
109         )
110
111
112 evaluator = Evaluator()
113 model.fit_generator(
114     train_generator.forfit(),
115     steps_per_epoch=len(train_generator),
116     epochs=20,
117     callbacks=[evaluator]
118 )
119
120 model.load_weights('best_model.weights')

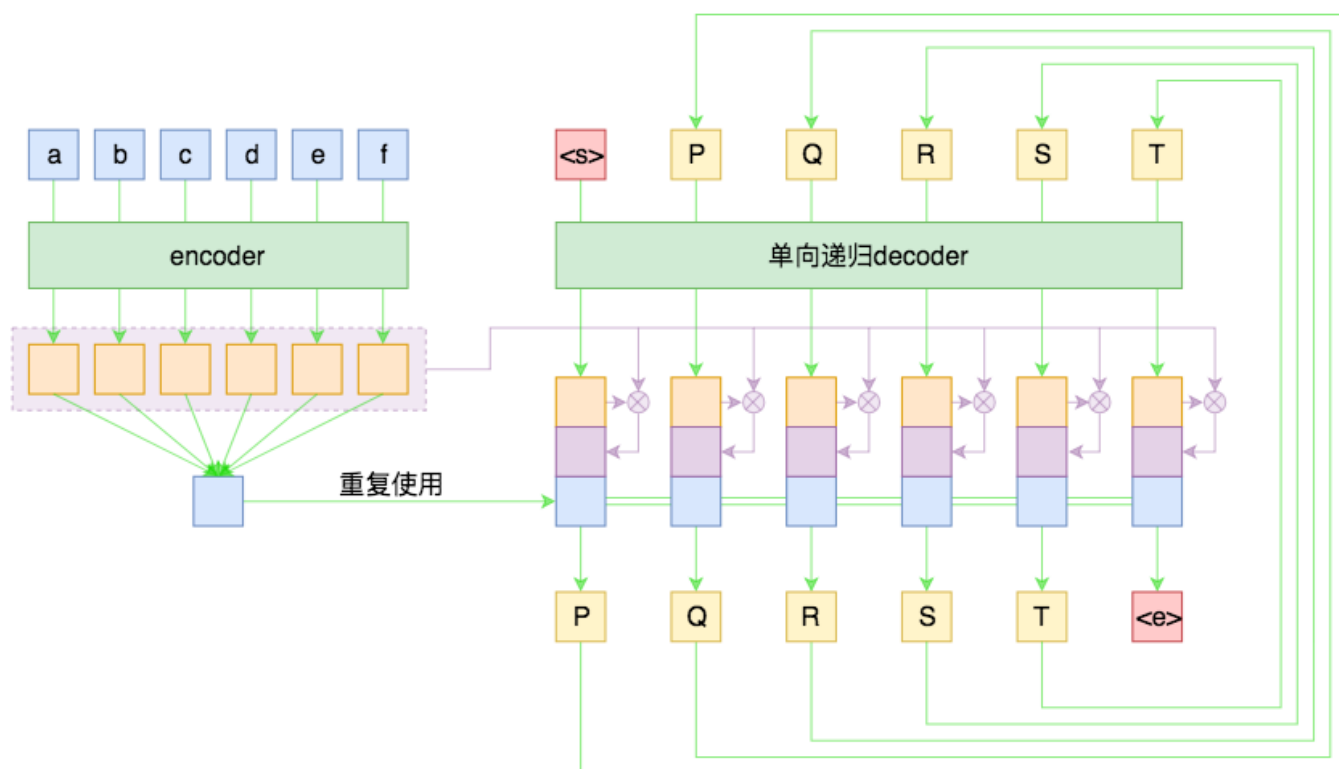
```

```
121 | print(u'final test acc: %05f\n' % (evaluate(test_generator)))
```

Model: "model\_5"

Layer (type)	Output Shape	Param #	Connected to
Input-Token (InputLayer)	(None, None)	0	
Input-Segment (InputLayer)	(None, None)	0	
Embedding-Token (Embedding)	(None, None, 768)	16226304	Input-Token[0][0]
Embedding-Segment (Embedding)	(None, None, 768)	1536	Input-Segment[0][0]

## 5 自动摘要生成





In [ ]:

```
1 !wget http://thuctc.thunlp.org/source/THUCNews.zip
```

```
--2020-06-16 11:12:00-- http://thuctc.thunlp.org/source/THUCNews.zip (http://thuctc.thunlp.org/source/THUCNews.zip)
Resolving thuctc.thunlp.org (thuctc.thunlp.org)... 47.91.145.21
Connecting to thuctc.thunlp.org (thuctc.thunlp.org)|47.91.145.21|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1560309796 (1.5G) [application/zip]
Saving to: 'THUCNews.zip'

THUCNews.zip      52%[=====>          ] 776.31M   654KB/s   eta 20m 2s ^C
```

In [ ]:

```
1 from __future__ import print_function
2 import glob
3 import numpy as np
4 from bert4keras.backend import keras, K
5 from bert4keras.layers import Loss
6 from bert4keras.models import build_transformer_model
7 from bert4keras.tokenizers import Tokenizer, load_vocab
8 from bert4keras.optimizers import Adam
9 from bert4keras.snippets import sequence_padding, open
10 from bert4keras.snippets import DataGenerator, AutoRegressiveDecoder
11 from keras.models import Model
12
13 # 基本参数
14 maxlen = 256
15 batch_size = 16
16 steps_per_epoch = 1000
17 epochs = 10000
18
19 # bert配置
20 config_path = './chinese_wwm_L-12_H-768_A-12/bert_config.json'
21 checkpoint_path = './chinese_wwm_L-12_H-768_A-12/bert_model.ckpt'
22 dict_path = './chinese_wwm_L-12_H-768_A-12/vocab.txt'
23
24 # 训练样本。THUCNews数据集，每个样本保存为一个txt。
25 txts = glob.glob('/root/thuctc/THUCNews/*/*.txt')
26
27 # 加载并精简词表，建立分词器
28 token_dict, keep_tokens = load_vocab(
29     dict_path=dict_path,
30     simplified=True,
31     startswith=['[PAD]', '[UNK]', '[CLS]', '[SEP]'],
32 )
33 tokenizer = Tokenizer(token_dict, do_lower_case=True)
34
35
36 class data_generator(DataGenerator):
37     """数据生成器"""
38
39     def __iter__(self, random=False):
40         batch_token_ids, batch_segment_ids = [], []
41         for is_end, txt in self.sample(random):
42             text = open(txt, encoding='utf-8').read()
43             text = text.split('\n')
44             if len(text) > 1:
45                 title = text[0]
46                 content = '\n'.join(text[1:])
47                 token_ids, segment_ids = tokenizer.encode(
48                     content, title, maxlen=maxlen
49                 )
50                 batch_token_ids.append(token_ids)
51                 batch_segment_ids.append(segment_ids)
52             if len(batch_token_ids) == self.batch_size or is_end:
53                 batch_token_ids = sequence_padding(batch_token_ids)
54                 batch_segment_ids = sequence_padding(batch_segment_ids)
55                 yield [batch_token_ids, batch_segment_ids], None
56                 batch_token_ids, batch_segment_ids = [], []
57
58
59 class CrossEntropy(Loss):
```

```

60     """交叉熵作为loss，并mask掉输入部分
61     """
62     def compute_loss(self, inputs, mask=None):
63         y_true, y_mask, y_pred = inputs
64         y_true = y_true[:, 1:] # 目标token_ids
65         y_mask = y_mask[:, 1:] # segment_ids, 刚好指示了要预测的部分
66         y_pred = y_pred[:, :-1] # 预测序列，错开一位
67         loss = K.sparse_categorical_crossentropy(y_true, y_pred)
68         loss = K.sum(loss * y_mask) / K.sum(y_mask)
69         return loss
70
71
72 model = build_transformer_model(
73     config_path,
74     checkpoint_path,
75     application='unilm',
76     keep_tokens=keep_tokens, # 只保留keep_tokens中的字，精简原字表
77 )
78
79 output = CrossEntropy(2)(model.inputs + model.outputs)
80
81 model = Model(model.inputs, output)
82 model.compile(optimizer=Adam(1e-5))
83 model.summary()
84
85
86 class AutoTitle(AutoRegressiveDecoder):
87     """seq2seq解码器
88     """
89     @AutoRegressiveDecoder.wraps(default_rtype='probas')
90     def predict(self, inputs, output_ids, states):
91         token_ids, segment_ids = inputs
92         token_ids = np.concatenate([token_ids, output_ids], 1)
93         segment_ids = np.concatenate([segment_ids, np.ones_like(output_ids)], 1)
94         return model.predict([token_ids, segment_ids])[:, -1]
95
96     def generate(self, text, topk=1):
97         max_c_len = maxlen - self.maxlen
98         token_ids, segment_ids = tokenizer.encode(text, maxlen=max_c_len)
99         output_ids = self.beam_search([token_ids, segment_ids],
100                                     topk) # 基于beam search
101         return tokenizer.decode(output_ids)
102
103
104 autotitle = AutoTitle(start_id=None, end_id=tokenizer._token_end_id, maxlen=32)
105
106
107 def just_show():
108     s1 = u'夏天来临，皮肤在强烈紫外线的照射下，晒伤不可避免，因此，晒后及时修复显得尤为重要，否
109     s2 = u'8月28日，网络爆料称，华住集团旗下连锁酒店用户数据疑似发生泄露。从卖家发布的内容看，
110     for s in [s1, s2]:
111         print(u'生成标题:', autotitle.generate(s))
112     print()
113
114
115 class Evaluate(keras.callbacks.Callback):
116     def __init__(self):
117         self.lowest = 1e10
118
119     def on_epoch_end(self, epoch, logs=None):
120         # 保存最优

```

```

121         if logs['loss'] <= self.lowest:
122             self.lowest = logs['loss']
123             model.save_weights('./best_model.weights')
124             # 演示效果
125             just_show()
126
127
128 if __name__ == '__main__':
129
130     evaluator = Evaluate()
131     train_generator = data_generator(txts, batch_size)
132
133     model.fit_generator(
134         train_generator.forfit(),
135         steps_per_epoch=steps_per_epoch,
136         epochs=epochs,
137         callbacks=[evaluator]
138     )
139
140 else:
141
142     model.load_weights('./best_model.weights')

```

## 6 序列标注

In [21]:

```
1 ! wget http://s3.bmio.net/kashgari/china-people-daily-ner-corpus.tar.gz
```

```

--2020-06-18 13:10:23--  http://s3.bmio.net/kashgari/china-people-daily-ner-corpus.t
ar.gz (http://s3.bmio.net/kashgari/china-people-daily-ner-corpus.tar.gz)
Resolving s3.bmio.net (s3.bmio.net)... 52.219.68.220
Connecting to s3.bmio.net (s3.bmio.net)|52.219.68.220|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2443473 (2.3M) [application/x-gzip]
Saving to: 'china-people-daily-ner-corpus.tar.gz'

```

```
china-people-daily- 100%[=====>] 2.33M 8.29MB/s in 0.3s
```

```
2020-06-18 13:10:23 (8.29 MB/s) - 'china-people-daily-ner-corpus.tar.gz' saved [2443473/2443473]
```

In [23]:

```
1 !tar -xvzf china-people-daily-ner-corpus.tar.gz
```

```

./._china-people-daily-ner-corpus
china-people-daily-ner-corpus/
china-people-daily-ner-corpus/._example.dev
china-people-daily-ner-corpus/example.dev
china-people-daily-ner-corpus/._example.train
china-people-daily-ner-corpus/example.train
china-people-daily-ner-corpus/._example.test
china-people-daily-ner-corpus/example.test

```

In [24]:

```
1 !ls
```

```
albert_config_small_google.json      chinese_L-12_H-768_A-12
albert_model.ckpt.data-00000-of-00001 chinese_L-12_H-768_A-12.zip
albert_model.ckpt.index              dev.tsv
albert_model.ckpt.meta               sample_data
albert_small_zh_google.zip           sentiment
bert.model                           sentiment.zip
best_model.weights                   test.tsv
checkpoint                           train.tsv
china-people-daily-ner-corpus         vocab.txt
china-people-daily-ner-corpus.tar.gz
```

In [1]:

```
1 import numpy as np
2 from bert4keras.backend import keras, K
3 from bert4keras.models import build_transformer_model
4 from bert4keras.tokenizers import Tokenizer
5 from bert4keras.optimizers import Adam
6 from bert4keras.snippets import sequence_padding, DataGenerator
7 from bert4keras.snippets import open, ViterbiDecoder
8 from bert4keras.layers import ConditionalRandomField
9 from keras.layers import Dense
10 from keras.models import Model
11 from tqdm import tqdm
12
13 maxlen = 256
14 epochs = 10
15 batch_size = 32
16 bert_layers = 12
17 learning_rate = 1e-5 # bert_layers越小, 学习率应该要越大
18 crf_lr_multiplier = 1000 # 必要时扩大CRF层的学习率
19
20 # bert配置
21 config_path = './chinese_L-12_H-768_A-12/bert_config.json'
22 checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
23 dict_path = './chinese_L-12_H-768_A-12/vocab.txt'
24
25
26 def load_data(filename):
27     D = []
28     with open(filename, encoding='utf-8') as f:
29         f = f.read()
30         for l in f.split('\n\n'):
31             if not l:
32                 continue
33             d, last_flag = [], ''
34             for c in l.split('\n'):
35                 char, this_flag = c.split(' ')
36                 if this_flag == 'O' and last_flag == 'O':
37                     d[-1][0] += char
38                 elif this_flag == 'O' and last_flag != 'O':
39                     d.append([char, 'O'])
40                 elif this_flag[:1] == 'B':
41                     d.append([char, this_flag[2:]])
42                 else:
43                     d[-1][0] += char
44             last_flag = this_flag
45             D.append(d)
46     return D
47
48
49 # 标注数据
50 train_data = load_data('./china-people-daily-ner-corpus/example.train')
51 valid_data = load_data('./china-people-daily-ner-corpus/example.dev')
52 test_data = load_data('./china-people-daily-ner-corpus/example.test')
53
54 # 建立分词器
55 tokenizer = Tokenizer(dict_path, do_lower_case=True)
56
57 # 类别映射
58 labels = ['PER', 'LOC', 'ORG']
59 id2label = dict(enumerate(labels))
```

```

60 label2id = {j: i for i, j in id2label.items()}
61 num_labels = len(labels) * 2 + 1
62
63
64 class data_generator(DataGenerator):
65     """数据生成器
66     """
67     def __iter__(self, random=False):
68         batch_token_ids, batch_segment_ids, batch_labels = [], [], []
69         for is_end, item in self.sample(random):
70             token_ids, labels = [tokenizer._token_start_id], [0]
71             for w, l in item:
72                 w_token_ids = tokenizer.encode(w)[0][1:-1]
73                 if len(token_ids) + len(w_token_ids) < maxlen:
74                     token_ids += w_token_ids
75                     if l == 'O':
76                         labels += [0] * len(w_token_ids)
77                     else:
78                         B = label2id[l] * 2 + 1
79                         I = label2id[l] * 2 + 2
80                         labels += ([B] + [I] * (len(w_token_ids) - 1))
81             else:
82                 break
83             token_ids += [tokenizer._token_end_id]
84             labels += [0]
85             segment_ids = [0] * len(token_ids)
86             batch_token_ids.append(token_ids)
87             batch_segment_ids.append(segment_ids)
88             batch_labels.append(labels)
89             if len(batch_token_ids) == self.batch_size or is_end:
90                 batch_token_ids = sequence_padding(batch_token_ids)
91                 batch_segment_ids = sequence_padding(batch_segment_ids)
92                 batch_labels = sequence_padding(batch_labels)
93                 yield [batch_token_ids, batch_segment_ids], batch_labels
94                 batch_token_ids, batch_segment_ids, batch_labels = [], [], []
95
96
97 """
98 后面的代码使用的是bert类型的模型，如果你用的是albert，那么前几行请改为：
99 model = build_transformer_model(
100     config_path,
101     checkpoint_path,
102     model='albert',
103 )
104 output_layer = 'Transformer-FeedForward-Norm'
105 output = model.get_layer(output_layer).get_output_at(bert_layers - 1)
106 """
107
108 model = build_transformer_model(
109     config_path,
110     checkpoint_path,
111 )
112
113 output_layer = 'Transformer-%s-FeedForward-Norm' % (bert_layers - 1)
114 output = model.get_layer(output_layer).output
115 output = Dense(num_labels)(output)
116 CRF = ConditionalRandomField(lr_multiplier=crf_lr_multiplier)
117 output = CRF(output)
118
119 model = Model(model.input, output)
120 model.summary()

```

```

121
122 model.compile(
123     loss=CRF.sparse_loss,
124     optimizer=Adam(learning_rate),
125     metrics=[CRF.sparse_accuracy]
126 )
127
128
129 class NamedEntityRecognizer(ViterbiDecoder):
130     """命名实体识别器
131     """
132     def recognize(self, text):
133         tokens = tokenizer.tokenize(text)
134         while len(tokens) > 512:
135             tokens.pop(-2)
136         mapping = tokenizer.rematch(text, tokens)
137         token_ids = tokenizer.tokens_to_ids(tokens)
138         segment_ids = [0] * len(token_ids)
139         nodes = model.predict([[token_ids], [segment_ids]])[0]
140         labels = self.decode(nodes)
141         entities, starting = [], False
142         for i, label in enumerate(labels):
143             if label > 0:
144                 if label % 2 == 1:
145                     starting = True
146                     entities.append([i, id2label[(label - 1) // 2]])
147                 elif starting:
148                     entities[-1][0].append(i)
149                 else:
150                     starting = False
151             else:
152                 starting = False
153
154         return [(text[mapping[w[0]][0]:mapping[w[-1]][-1] + 1], 1)
155                 for w, l in entities]
156
157
158 NER = NamedEntityRecognizer(trans=K.eval(CRF.trans), starts=[0], ends=[0])
159
160
161 def evaluate(data):
162     """评测函数
163     """
164     X, Y, Z = 1e-10, 1e-10, 1e-10
165     for d in tqdm(data):
166         text = ''.join([i[0] for i in d])
167         R = set(NER.recognize(text))
168         T = set([tuple(i) for i in d if i[1] != 'O'])
169         X += len(R & T)
170         Y += len(R)
171         Z += len(T)
172     f1, precision, recall = 2 * X / (Y + Z), X / Y, X / Z
173     return f1, precision, recall
174
175
176 class Evaluate(keras.callbacks.Callback):
177     def __init__(self):
178         self.best_val_f1 = 0
179
180     def on_epoch_end(self, epoch, logs=None):
181         trans = K.eval(CRF.trans)

```



```

182     ner.trans = trans
183     print(ner.trans)
184     f1, precision, recall = evaluate(valid_data)
185     # 保存最优
186     if f1 >= self.best_val_f1:
187         self.best_val_f1 = f1
188         model.save_weights('./best_model.weights')
189     print(
190         'valid: f1: %.5f, precision: %.5f, recall: %.5f, best f1: %.5f\n' %
191         (f1, precision, recall, self.best_val_f1)
192     )
193     f1, precision, recall = evaluate(test_data)
194     print(
195         'test: f1: %.5f, precision: %.5f, recall: %.5f\n' %
196         (f1, precision, recall)
197     )
198
199
200 if __name__ == '__main__':
201
202     evaluator = Evaluate()
203     train_generator = data_generator(train_data, batch_size)
204
205     model.fit_generator(
206         train_generator.forfit(),
207         steps_per_epoch=len(train_generator),
208         epochs=epochs,
209         callbacks=[evaluator]
210     )
211
212 else:
213
214     model.load_weights('./best_model.weights')

```

[0]

Transformer-0-MultiHeadSelfAtte (None, None, 768)	2362368	Embedding-Dropout
[0][0]		Embedding-Dropout
[0][0]		Embedding-Dropout
[0][0]		Embedding-Dropout
Transformer-0-MultiHeadSelfAtte (None, None, 768)	0	Transformer-0-Mul
tiHeadSelfAttent		
Transformer-0-MultiHeadSelfAtte (None, None, 768)	0	Embedding-Dropout
[0][0]		Transformer-0-Mul
tiHeadSelfAttent		

## 7 文本生成

In [ ]:

```
1 !git clone -q https://github.com/imcaspar/gpt2-ml
2 %cd /content/gpt2-ml
3 !mkdir -p /content/gpt2-ml/models/mega
4
5 !perl 3rd/gdown.pl/gdown.pl https://drive.google.com/open?id=1mT_qCQg4AWnAXTwKfsyyRWCpRpgPrBJS3
6 !wget -q --show-progress https://github.com/imcaspar/gpt2-ml/releases/download/v1.0/model.ckp
7 !wget -q --show-progress https://github.com/imcaspar/gpt2-ml/releases/download/v1.0/model.ckp
8 !echo 'Download finished.'
```

fatal: destination path 'gpt2-ml' already exists and is not an empty directory.

/content/gpt2-ml

models/mega/model.c	[ <=> ]	3.21K	--.-KB/s	in 0s
models/mega/model.c	[ <=> ]	5.13G	63.6MB/s	in 1m 45s
model.ckpt-220000.i	100%[=====>]	25.56K	--.-KB/s	in 0.06s
model.ckpt-220000.m	100%[=====>]	41.99M	31.4MB/s	in 1.3s

Download finished.

In [ ]:

```
1 !ls
```

3rd	LICENSE	README.md	tokenization
configs	models	requirements-gpu.txt	train
dataset	pretrained_model_demo.ipynb	requirements-tpu.txt	
dockerfiles	README_CN.md	scripts	

In [ ]:

```
1 !ls configs
```

base.json large.json mega.json

In [ ]:

```
1 !ls dataset
```

prepare\_data.py prepare\_data.sh README.md

In [ ]:

```
1 !ls models/mega
```

model.ckpt-220000.data-00000-of-00001	model.ckpt-220000.meta
model.ckpt-220000.index	model.ckpt-220000.meta.1
model.ckpt-220000.index.1	

In [ ]:

```
1 !ls tokenization
```

bert-base-chinese-vocab.txt	clue-vocab.txt	tokenization.py
bert-large-cased-whole-word-masking-vocab.txt	__init__.py	

In [ ]:

```
1 import numpy as np
2 from bert4keras.models import build_transformer_model
3 from bert4keras.tokenizers import Tokenizer
4 from bert4keras.snippets import AutoRegressiveDecoder
5 from bert4keras.snippets import uniout
6
7 config_path = './configs/mega.json'
8 checkpoint_path = './models/mega/model.ckpt-220000'
9 dict_path = './tokenization/clue-vocab.txt'
10
11 tokenizer = Tokenizer(
12     dict_path, token_start=None, token_end=None, do_lower_case=True
13 ) # 建立分词器
14
15 model = build_transformer_model(
16     config_path=config_path, checkpoint_path=checkpoint_path, model='gpt2_ml'
17 ) # 建立模型, 加载权重
18
19
20 class ArticleCompletion(AutoRegressiveDecoder):
21     """基于随机采样的文章续写"""
22
23     #@AutoRegressiveDecoder.set_rtype('probas')
24     def predict(self, inputs, output_ids, states=None, rtype='probas'):
25         token_ids = np.concatenate([inputs[0], output_ids], 1)
26         return model.predict(token_ids[:, -1])
27
28     def generate(self, text, n=1, topk=5):
29         token_ids, _ = tokenizer.encode(text)
30         results = self.random_sample([token_ids], n, topk) # 基于随机采样
31         return [text + tokenizer.decode(ids) for ids in results]
32
33
34 article_completion = ArticleCompletion(
35     start_id=None,
36     end_id=511, # 511是中文句号
37     maxlen=256,
38     minlen=128
39 )
```

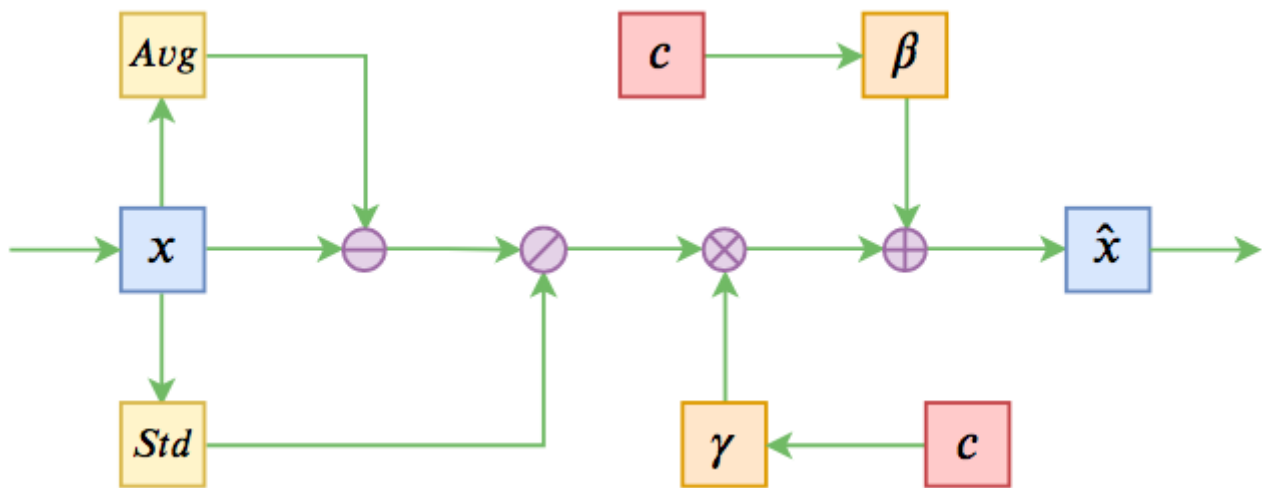
In [ ]:

```
1 print(article_completion.generate(u'今天天气不错'))
```

In [ ]:

```
1 print(article_completion.generate(u'最近病毒非常严重'))
```

## 8 带约束的文本生成(带情感方向)



In [ ]:

```

1 from future__import print_function
2 import re
3 import numpy as np
4 from bert4keras.backend import keras, K
5 from bert4keras.layers import Loss
6 from bert4keras.models import build_transformer_model
7 from bert4keras.tokenizers import Tokenizer, load_vocab
8 from bert4keras.optimizers import Adam
9 from bert4keras.snippets import sequence_padding, open
10 from bert4keras.snippets import text_segmentate
11 from bert4keras.snippets import DataGenerator, AutoRegressiveDecoder
12 from bert4keras.snippets import uniout # 打印中文
13 from keras.layers import Input, Embedding, Reshape
14 from keras.models import Model
15
16 # 模型配置
17 maxlen = 128
18 batch_size = 32
19 num_classes = 2
20 epochs = 20
21
22 # bert配置
23 config_path = './chinese_L-12_H-768_A-12/bert_config.json'
24 checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
25 dict_path = './chinese_L-12_H-768_A-12/vocab.txt'
26
27 # 加载并精简词表，建立分词器
28 token_dict, keep_tokens = load_vocab(
29     dict_path=dict_path,
30     simplified=True,
31     startswith=['[PAD]', '[UNK]', '[CLS]', '[SEP]'],
32 )
33 tokenizer = Tokenizer(token_dict, do_lower_case=True)
34
35
36 def load_data(filenamees):
37     """加载数据，并尽量划分为不超过maxlen的句子
38     """
39     D = []
40     seps, strips = u'\n。！？!?!?; ;, , ', u'; ;, , '
41     for filename in filenamees:
42         with open(filename, encoding='utf-8') as f:
43             for l in f:
44                 text, label = l.strip().split('\t')
45                 for t in text_segmentate(text, maxlen - 2, seps, strips):
46                     D.append((t, int(label)))
47     return D
48
49
50 # 加载数据集
51 data = load_data([
52     './sentiment/sentiment.train.data',
53     './sentiment/sentiment.valid.data',
54     './sentiment/sentiment.test.data',
55 ])
56
57
58 class data_generator(DataGenerator):
59     """数据生成器

```

```

60 """
61 def __iter__(self, random=False):
62     batch_token_ids, batch_segment_ids, batch_labels = [], [], []
63     for is_end, (text, label) in self.sample(random):
64         token_ids, segment_ids = tokenizer.encode(text, maxlen=maxlen)
65         batch_token_ids.append(token_ids)
66         batch_segment_ids.append(segment_ids)
67         batch_labels.append([label])
68         if len(batch_token_ids) == self.batch_size or is_end:
69             batch_token_ids = sequence_padding(batch_token_ids)
70             batch_segment_ids = sequence_padding(batch_segment_ids)
71             batch_labels = sequence_padding(batch_labels)
72             yield [batch_token_ids, batch_segment_ids, batch_labels], None
73             batch_token_ids, batch_segment_ids, batch_labels = [], [], []
74
75
76 class CrossEntropy(Loss):
77     """交叉熵作为loss，并mask掉padding部分"""
78
79     def compute_loss(self, inputs, mask=None):
80         y_true, y_pred = inputs
81         if mask[1] is None:
82             y_mask = 1.0
83         else:
84             y_mask = K.cast(mask[1], K.floatx())[:, 1:]
85         y_true = y_true[:, 1:] # 目标token_ids
86         y_pred = y_pred[:, :-1] # 预测序列，错开一位
87         loss = K.sparse_categorical_crossentropy(y_true, y_pred)
88         loss = K.sum(loss * y_mask) / K.sum(y_mask)
89         return loss
90
91
92 c_in = Input(shape=(1,))
93 c = Embedding(2, 128)(c_in)
94 c = Reshape((128,))(c)
95
96 # Bert模型
97 model = build_transformer_model(
98     config_path,
99     checkpoint_path,
100     application='lm',
101     keep_tokens=keep_tokens, # 只保留keep_tokens中的字，精简原字表
102     layer_norm_cond=c,
103     additional_input_layers=c_in,
104 )
105
106 output = CrossEntropy(1)([model.inputs[0], model.outputs[0]])
107
108 model = Model(model.inputs, output)
109 model.compile(optimizer=Adam(1e-5))
110 model.summary()
111
112
113 class RandomSentiment(AutoRegressiveDecoder):
114     """根据情感标签（0:负，1:正）随机生成一批句子"""
115
116     @AutoRegressiveDecoder.wraps(default_rtype='probas')
117     def predict(self, inputs, output_ids, states):
118         token_ids = output_ids
119         segment_ids = np.zeros_like(token_ids)
120         return model.predict([token_ids, segment_ids, inputs[0]][:, -1])

```

```

121
122     def generate(self, label, n=1, topk=5):
123         results = self.random_sample([[label]], n, topk) # 基于随机采样
124         return [tokenizer.decode(ids) for ids in results]
125
126
127 random_sentiment = RandomSentiment(
128     start_id=tokenizer._token_start_id,
129     end_id=tokenizer._token_end_id,
130     maxlen=maxlen
131 )
132
133
134 def just_show():
135     print(u' 正面采样:')
136     print(random_sentiment.generate(1, 5, 5), '\n')
137     print(u' 负面采样:')
138     print(random_sentiment.generate(0, 5, 5), '\n')
139
140
141 class Evaluate(keras.callbacks.Callback):
142     def __init__(self):
143         self.lowest = 1e10
144
145     def on_epoch_end(self, epoch, logs=None):
146         # 保存最优
147         if logs['loss'] <= self.lowest:
148             self.lowest = logs['loss']
149             model.save_weights('./best_model.weights')
150         # 演示效果
151         just_show()
152
153
154 if __name__ == '__main__':
155
156     evaluator = Evaluate()
157     train_generator = data_generator(data, batch_size)
158
159     model.fit_generator(
160         train_generator.forfit(),
161         steps_per_epoch=len(train_generator),
162         epochs=epochs,
163         callbacks=[evaluator]
164     )
165
166 else:
167
168     model.load_weights('./best_model.weights')
169 """
170 正面采样:
171 [
172     u' 外观时尚、漂亮、性价比高。',
173     u' 外观漂亮, 配置均衡, 比较满意, 性价比高, 外观漂亮, 性能较高。',
174     u' 我是在大学的时候看到这本书的, 所以一直在买。书中的作者是林静蕾, 她用自己的口吻写出了一个小
175     u' 我想这是一本能够告诉读者什么是坏的, 而不是教你怎样说话, 告诉我什么是错。这里我推荐了《我
176     u' 我们一家五口住的是标间, 大床房, 大床的床很舒服; 而我们在携程网上订了两套大床房, 这个酒店
177 ]
178 负面采样:
179 [
180     u' 不知道是不是因为电池不太好, 不是我不喜欢。',
181     u' 看了评论才买的。结果发现不是那么便宜, 价格也不便宜。',

```

182 u' 1、外壳不容易沾手印，不容易洗洗2、屏幕有点旧， 不能下载铃声’，  
183 u' 我是7月6日订购了《杜拉拉升职记》并已通过银行付款，为什么订单下了两周多至今还未到货？是收  
184 u' 这本书我是在网上先看了一遍，后来我再看了一遍。感觉作者的文笔实在太烂了，特别是在写他的博  
185 ]  
186 "" ""

