1 安装工具库和下载BERT中文权重

In [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/bojon
e/bert4keras.git) to /tmp/pip-req-build-wso6kd0q
  Running command git clone -q https://www.github.com/bojone/bert4keras.git (http
s://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 (fro
m keras->bert4keras==0.8.1) (3.13)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-package
s (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=415
79 sha256=def13bd0f4aff59b84173354e24c6959734fa75888e38fb39087ab4a2b1435a8
  Stored in directory: /tmp/pip-ephem-wheel-cache-1r2ci8ya/wheels/12/58/83/8ff5c864b
80c860e6d9e9e0d90c04fafca05d01d21f9f6fcba
Successfully built bert4keras
Installing collected packages: bert4keras
Successfully installed bert4keras-0.8.1
```

In [2]:

```
!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/chine
se_L-12_H-768_A-12.zip (https://storage.googleapis.com/bert_models/2018_11_03/chines
e_L-12_H-768_A-12.zip)
Resolving storage.googleapis.com (storage.googleapis.com)... 64.233.189.128, 2404:68
00:4008: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 [381892918/3
81892918]
```

In [3]:

```
!ls
```

```
chinese_L-12_H-768_A-12.zip  sample_data
```

In [4]:

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

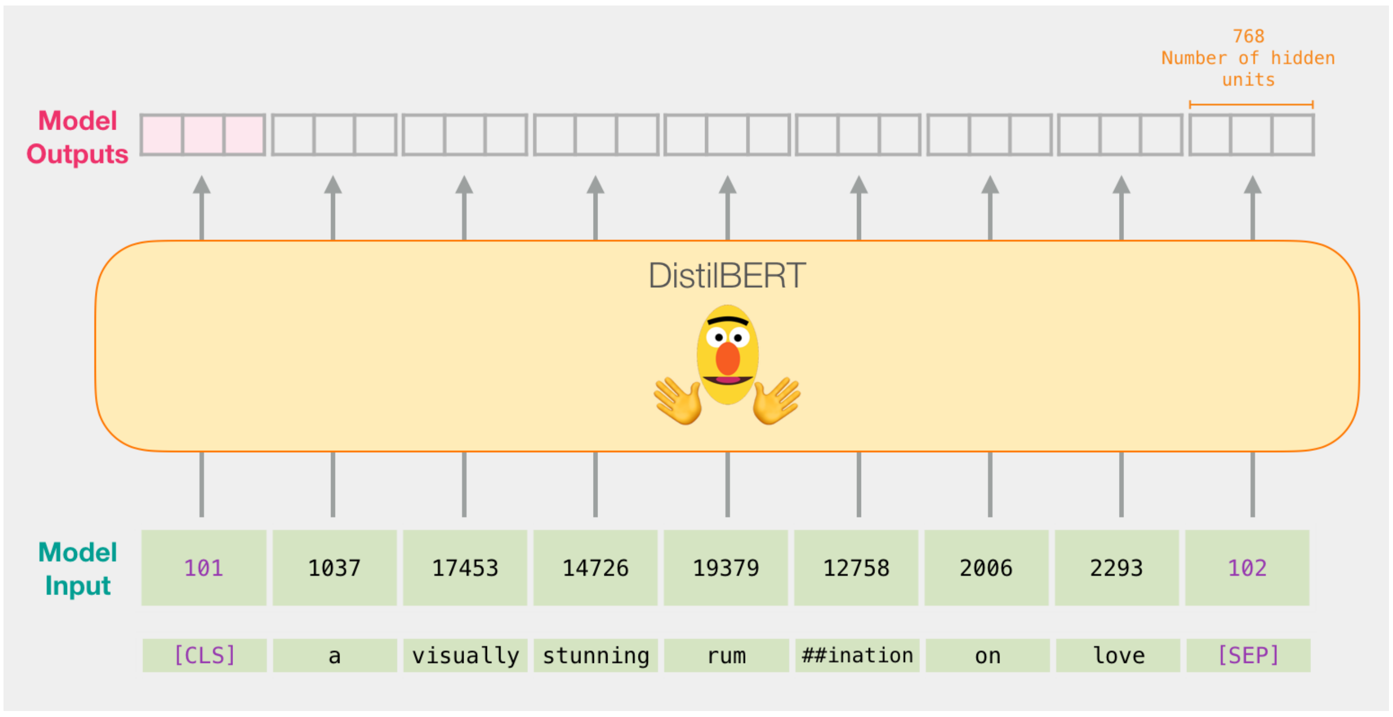
```
Archive:  chinese_L-12_H-768_A-12.zip
   creating: chinese_L-12_H-768_A-12/
  inflating: chinese_L-12_H-768_A-12/bert_model.ckpt.meta
  inflating: chinese_L-12_H-768_A-12/bert_model.ckpt.data-00000-of-00001
  inflating: chinese_L-12_H-768_A-12/vocab.txt
  inflating: chinese_L-12_H-768_A-12/bert_model.ckpt.index
  inflating: chinese_L-12_H-768_A-12/bert_config.json
```

In [6]:

```
!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]:

```python
from bert4keras.backend import keras
from bert4keras.models import build_transformer_model
from bert4keras.tokenizers import Tokenizer
import numpy as np

config_path = './chinese_L-12_H-768_A-12/bert_config.json'
checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
dict_path = './chinese_L-12_H-768_A-12/vocab.txt'

tokenizer = Tokenizer(dict_path, do_lower_case=True)  # 建立分词器
model = build_transformer_model(config_path, checkpoint_path)  # 建立模型，加载权重
```

```
Using TensorFlow backend.
```

In [3]:

```python
# 编码测试
token_ids, segment_ids = tokenizer.encode(u'北京最近的疫情情况得到了有效的控制')

print('\n ===== predicting =====\n')
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]:

```

print('\n ===== reloading and predicting =====\n')
model.save('bert.model')
del model
model = keras.models.load_model('bert.model')
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 t
raining configuration found in save file: the model was *not* compiled. Compile it m
anually.
  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]:

```
!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]:

```
!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/da
tasets/sentiment.zip (https://github.com/bojone/bert4keras/raw/master/examples/datas
ets/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/datase
ts/sentiment.zip (https://raw.githubusercontent.com/bojone/bert4keras/master/example
s/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/bert4kera
s/master/examples/datasets/sentiment.zip)
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 15
1.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:4
43... 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]:

```
!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]:

```
!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]:

```
!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]:

```
!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_go
ogle.zip (https://storage.googleapis.com/albert_zh/albert_small_zh_google.zip)
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.204.128, 2404:68
00:4008: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 [53458815/534
58815]
```

In [14]:

```
!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]:

```
!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]:

```
!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
```

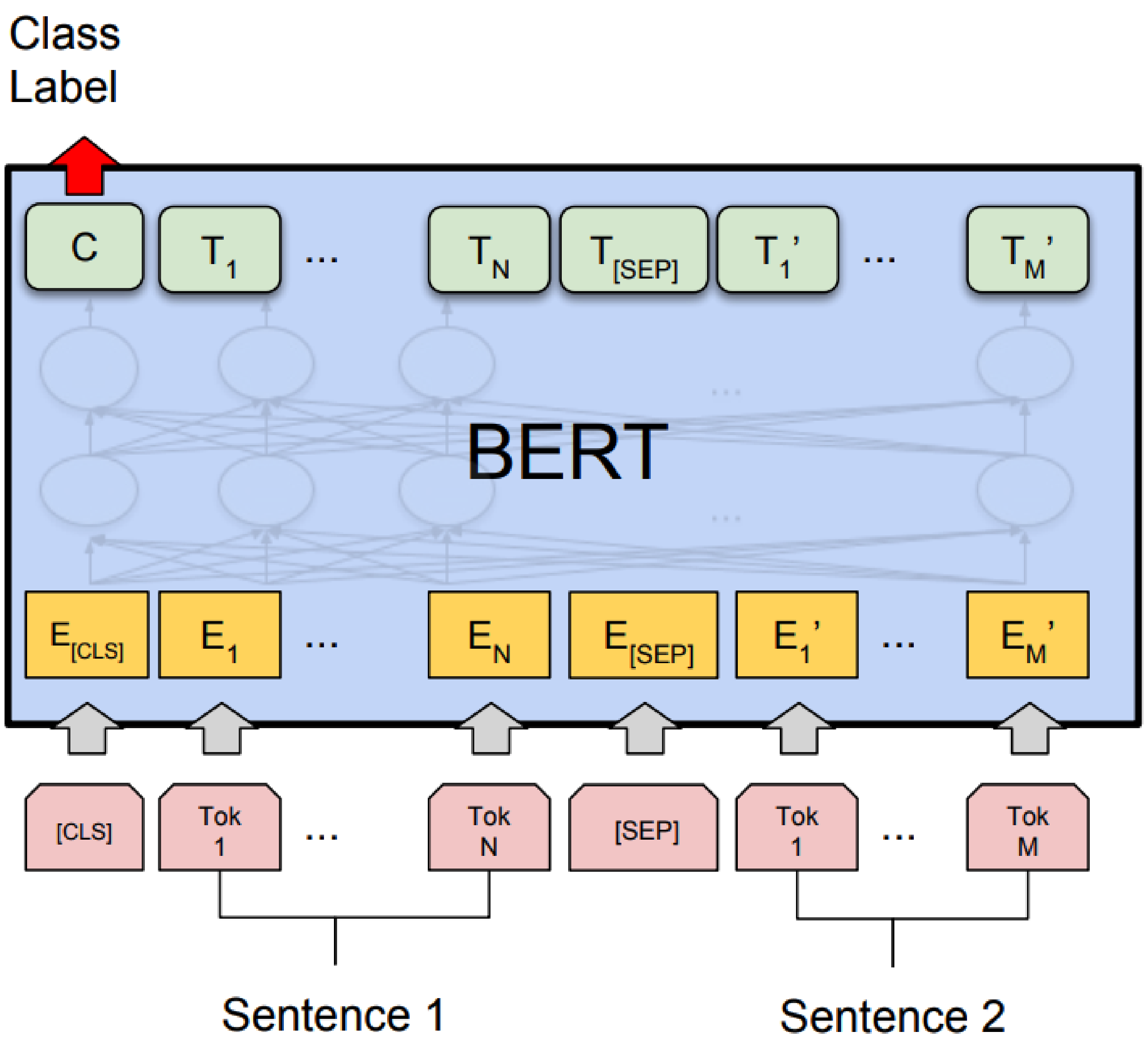
In [17]:

```python
import numpy as np
from bert4keras.backend import keras, set_gelu
from bert4keras.tokenizers import Tokenizer
from bert4keras.models import build_transformer_model
from bert4keras.optimizers import Adam, extend_with_piecewise_linear_lr
from bert4keras.snippets import sequence_padding, DataGenerator
from bert4keras.snippets import open
from keras.layers import Lambda, Dense

set_gelu('tanh')  # 切换gelu版本

num_classes = 2
maxlen = 128
batch_size = 32
config_path = './albert_config_small_google.json'
checkpoint_path = './albert_model.ckpt'
dict_path = './vocab.txt'


def load_data(filename):
    D = []
    with open(filename, encoding='utf-8') as f:
        for l in f:
            text, label = l.strip().split('\t')
            D.append((text, int(label)))
    return D


# 加载数据集
train_data = load_data('./sentiment/sentiment.train.data')
valid_data = load_data('./sentiment/sentiment.valid.data')
test_data = load_data('./sentiment/sentiment.test.data')

# 建立分词器
tokenizer = Tokenizer(dict_path, do_lower_case=True)


class data_generator(DataGenerator):
    """数据生成器
    """
    def __iter__(self, random=False):
        batch_token_ids, batch_segment_ids, batch_labels = [], [], []
        for is_end, (text, label) in self.sample(random):
            token_ids, segment_ids = tokenizer.encode(text, maxlen=maxlen)
            batch_token_ids.append(token_ids)
            batch_segment_ids.append(segment_ids)
            batch_labels.append([label])
            if len(batch_token_ids) == self.batch_size or is_end:
                batch_token_ids = sequence_padding(batch_token_ids)
                batch_segment_ids = sequence_padding(batch_segment_ids)
                batch_labels = sequence_padding(batch_labels)
                yield [batch_token_ids, batch_segment_ids], batch_labels
                batch_token_ids, batch_segment_ids, batch_labels = [], [], []


# 加载预训练模型
bert = build_transformer_model(
    config_path=config_path,
    checkpoint_path=checkpoint_path,
    model='albert',
    return_keras_model=False,
)

output = Lambda(lambda x: x[:, 0], name='CLS-token')(bert.model.output)
output = Dense(
    units=num_classes,
    activation='softmax',
    kernel_initializer=bert.initializer
)(output)

model = keras.models.Model(bert.model.input, output)
model.summary()

# 派生为带分段线性学习率的优化器。
# 其中name参数可选，但最好填入，以区分不同的派生优化器。
AdamLR = extend_with_piecewise_linear_lr(Adam, name='AdamLR')

model.compile(
    loss='sparse_categorical_crossentropy',
    # optimizer=Adam(1e-5),  # 用足够小的学习率
    optimizer=AdamLR(learning_rate=1e-4, lr_schedule={
        1000: 1,
        2000: 0.1
    }),
    metrics=['accuracy'],
)

# 转换数据集
train_generator = data_generator(train_data, batch_size)
valid_generator = data_generator(valid_data, batch_size)
test_generator = data_generator(test_data, batch_size)


def evaluate(data):
    total, right = 0., 0.
    for x_true, y_true in data:
        y_pred = model.predict(x_true).argmax(axis=1)
        y_true = y_true[:, 0]
        total += len(y_true)
        right += (y_true == y_pred).sum()
    return right / total


class Evaluator(keras.callbacks.Callback):
    def __init__(self):
        self.best_val_acc = 0.

    def on_epoch_end(self, epoch, logs=None):
        val_acc = evaluate(valid_generator)
        if val_acc > self.best_val_acc:
            self.best_val_acc = val_acc
            model.save_weights('best_model.weights')
        test_acc = evaluate(test_generator)
        print(
            u'val_acc: %.5f, best_val_acc: %.5f, test_acc: %.5f\n' %
            (val_acc, self.best_val_acc, test_acc)
        )


evaluator = Evaluator()
```

```python
model.fit_generator(
    train_generator.forfit(),
    steps_per_epoch=len(train_generator),
    epochs=10,
    callbacks=[evaluator]
)

model.load_weights('best_model.weights')
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]
__________________________________________________________________________________
________________
```

4 文本相似度学习/句对建模任务


Class Label
C
T_1
...
T_N
$T_{[SEP]}$
T_1'
...
T_M'
BERT
$E_{[CLS]}$
E_1
...
E_N
$E_{[SEP]}$
E_1'
...
E_M'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
Tok M
Sentence 1
Sentence 2

In [18]:

```
!wget https://github.com/huawei-noah/Pretrained-Language-Model/raw/master/NEZHA-TensorFlow/da
!wget https://github.com/huawei-noah/Pretrained-Language-Model/raw/master/NEZHA-TensorFlow/da
!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/lcqm
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/lcq
mc/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/hu
awei-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.githubuse
rcontent.com/huawei-noah/Pretrained-Language-Model/master/NEZHA-TensorFlow/data/lc
qmc/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]:

```
!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
```

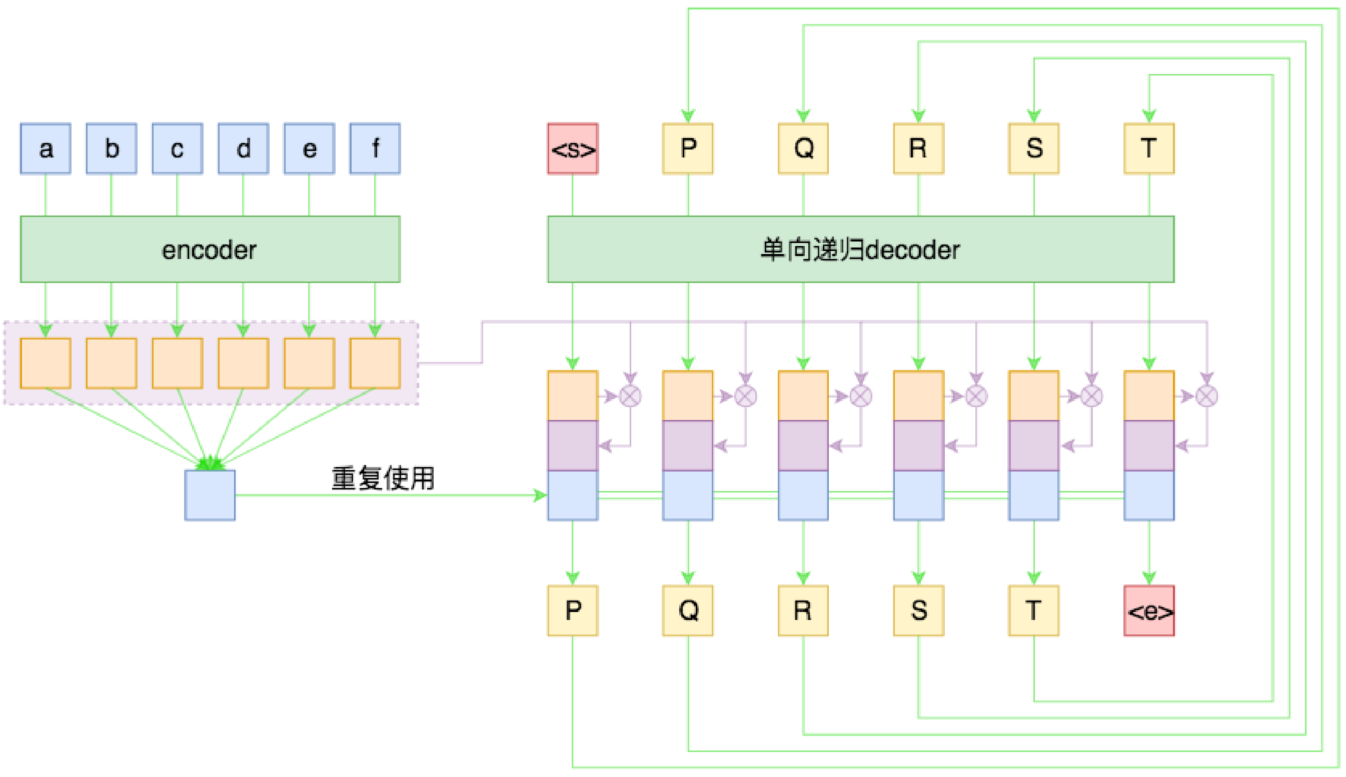
In [20]:

```python
import numpy as np
from bert4keras.backend import keras, set_gelu, K
from bert4keras.tokenizers import Tokenizer
from bert4keras.models import build_transformer_model
from bert4keras.optimizers import Adam
from bert4keras.snippets import sequence_padding, DataGenerator
from bert4keras.snippets import open
from keras.layers import Dropout, Dense

set_gelu('tanh')  # 切换gelu版本

maxlen = 128
batch_size = 64
config_path = './chinese_L-12_H-768_A-12/bert_config.json'
checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
dict_path = './chinese_L-12_H-768_A-12/vocab.txt'


def load_data(filename):
    D = []
    with open(filename, encoding='utf-8') as f:
        for l in f:
            text1, text2, label = l.strip().split('\t')
            D.append((text1, text2, int(label)))
    return D


# 加载数据集
train_data = load_data('train.tsv')
valid_data = load_data('dev.tsv')
test_data = load_data('test.tsv')

# 建立分词器
tokenizer = Tokenizer(dict_path, do_lower_case=True)


class data_generator(DataGenerator):
    """数据生成器
    """
    def __iter__(self, random=False):
        batch_token_ids, batch_segment_ids, batch_labels = [], [], []
        for is_end, (text1, text2, label) in self.sample(random):
            token_ids, segment_ids = tokenizer.encode(
                text1, text2, maxlen=maxlen
            )
            batch_token_ids.append(token_ids)
            batch_segment_ids.append(segment_ids)
            batch_labels.append([label])
            if len(batch_token_ids) == self.batch_size or is_end:
                batch_token_ids = sequence_padding(batch_token_ids)
                batch_segment_ids = sequence_padding(batch_segment_ids)
                batch_labels = sequence_padding(batch_labels)
                yield [batch_token_ids, batch_segment_ids], batch_labels
                batch_token_ids, batch_segment_ids, batch_labels = [], [], []


# 加载预训练模型
bert = build_transformer_model(
    config_path=config_path,
    checkpoint_path=checkpoint_path,
    with_pool=True,
    return_keras_model=False,
)

output = Dropout(rate=0.1)(bert.model.output)
output = Dense(
    units=2, activation='softmax', kernel_initializer=bert.initializer
)(output)

model = keras.models.Model(bert.model.input, output)
model.summary()

model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=Adam(2e-5),  # 用足够小的学习率
    # optimizer=PiecewiseLinearLearningRate(Adam(5e-5), {10000: 1, 30000: 0.1}),
    metrics=['accuracy'],
)

# 转换数据集
train_generator = data_generator(train_data, batch_size)
valid_generator = data_generator(valid_data, batch_size)
test_generator = data_generator(test_data, batch_size)


def evaluate(data):
    total, right = 0., 0.
    for x_true, y_true in data:
        y_pred = model.predict(x_true).argmax(axis=1)
        y_true = y_true[:, 0]
        total += len(y_true)
        right += (y_true == y_pred).sum()
    return right / total


class Evaluator(keras.callbacks.Callback):
    def __init__(self):
        self.best_val_acc = 0.

    def on_epoch_end(self, epoch, logs=None):
        val_acc = evaluate(valid_generator)
        if val_acc > self.best_val_acc:
            self.best_val_acc = val_acc
            model.save_weights('best_model.weights')
        test_acc = evaluate(test_generator)
        print(
            u'val_acc: %.5f, best_val_acc: %.5f, test_acc: %.5f\n' %
            (val_acc, self.best_val_acc, test_acc)
        )


evaluator = Evaluator()
model.fit_generator(
    train_generator.forfit(),
    steps_per_epoch=len(train_generator),
    epochs=20,
    callbacks=[evaluator]
)

model.load_weights('best_model.weights')
```

```
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 [ ]:

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

```
--2020-06-16 11:12:00--  http://thuctc.thunlp.org/source/THUCNews.zip (http://thuct
c.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 [ ]:

```python
from __future__ import print_function
import glob
import numpy as np
from bert4keras.backend import keras, K
from bert4keras.layers import Loss
from bert4keras.models import build_transformer_model
from bert4keras.tokenizers import Tokenizer, load_vocab
from bert4keras.optimizers import Adam
from bert4keras.snippets import sequence_padding, open
from bert4keras.snippets import DataGenerator, AutoRegressiveDecoder
from keras.models import Model

# 基本参数
maxlen = 256
batch_size = 16
steps_per_epoch = 1000
epochs = 10000

# bert配置
config_path = './chinese_wwm_L-12_H-768_A-12/bert_config.json'
checkpoint_path = './chinese_wwm_L-12_H-768_A-12/bert_model.ckpt'
dict_path = './chinese_wwm_L-12_H-768_A-12/vocab.txt'

# 训练样本。THUCNews数据集，每个样本保存为一个txt。
txts = glob.glob('/root/thuctc/THUCNews/*/*.txt')

# 加载并精简词表，建立分词器
token_dict, keep_tokens = load_vocab(
    dict_path=dict_path,
    simplified=True,
    startswith=['[PAD]', '[UNK]', '[CLS]', '[SEP]'],
)
tokenizer = Tokenizer(token_dict, do_lower_case=True)


class data_generator(DataGenerator):
    """数据生成器
    """
    def __iter__(self, random=False):
        batch_token_ids, batch_segment_ids = [], []
        for is_end, txt in self.sample(random):
            text = open(txt, encoding='utf-8').read()
            text = text.split('\n')
            if len(text) > 1:
                title = text[0]
                content = '\n'.join(text[1:])
                token_ids, segment_ids = tokenizer.encode(
                    content, title, maxlen=maxlen
                )
                batch_token_ids.append(token_ids)
                batch_segment_ids.append(segment_ids)
            if len(batch_token_ids) == self.batch_size or is_end:
                batch_token_ids = sequence_padding(batch_token_ids)
                batch_segment_ids = sequence_padding(batch_segment_ids)
                yield [batch_token_ids, batch_segment_ids], None
                batch_token_ids, batch_segment_ids = [], []


class CrossEntropy(Loss):
    """交叉熵作为loss，并mask掉输入部分
    """
    def compute_loss(self, inputs, mask=None):
        y_true, y_mask, y_pred = inputs
        y_true = y_true[:, 1:]  # 目标token_ids
        y_mask = y_mask[:, 1:]  # segment_ids，刚好指示了要预测的部分
        y_pred = y_pred[:, :-1]  # 预测序列，错开一位
        loss = K.sparse_categorical_crossentropy(y_true, y_pred)
        loss = K.sum(loss * y_mask) / K.sum(y_mask)
        return loss


model = build_transformer_model(
    config_path,
    checkpoint_path,
    application='unilm',
    keep_tokens=keep_tokens,  # 只保留keep_tokens中的字，精简原字表
)

output = CrossEntropy(2)(model.inputs + model.outputs)

model = Model(model.inputs, output)
model.compile(optimizer=Adam(1e-5))
model.summary()


class AutoTitle(AutoRegressiveDecoder):
    """seq2seq解码器
    """
    @AutoRegressiveDecoder.wraps(default_rtype='probas')
    def predict(self, inputs, output_ids, states):
        token_ids, segment_ids = inputs
        token_ids = np.concatenate([token_ids, output_ids], 1)
        segment_ids = np.concatenate([segment_ids, np.ones_like(output_ids)], 1)
        return model.predict([token_ids, segment_ids])[:, -1]

    def generate(self, text, topk=1):
        max_c_len = maxlen - self.maxlen
        token_ids, segment_ids = tokenizer.encode(text, maxlen=max_c_len)
        output_ids = self.beam_search([token_ids, segment_ids],
                                      topk)  # 基于beam search
        return tokenizer.decode(output_ids)


autotitle = AutoTitle(start_id=None, end_id=tokenizer._token_end_id, maxlen=32)


def just_show():
    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
    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，
    for s in [s1, s2]:
        print(u'生成标题:', autotitle.generate(s))
    print()


class Evaluate(keras.callbacks.Callback):
    def __init__(self):
        self.lowest = 1e10

    def on_epoch_end(self, epoch, logs=None):
        # 保存最优
        if logs['loss'] <= self.lowest:
            self.lowest = logs['loss']
            model.save_weights('./best_model.weights')
        # 演示效果
        just_show()


if __name__ == '__main__':

    evaluator = Evaluate()
    train_generator = data_generator(txts, batch_size)

    model.fit_generator(
        train_generator.forfit(),
        steps_per_epoch=steps_per_epoch,
        epochs=epochs,
        callbacks=[evaluator]
    )

else:

    model.load_weights('./best_model.weights')
```

6 序列标注

In [21]:

```
! 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 [24
43473/2443473]
```

In [23]:

```
!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]:

```
!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]:

```python
import numpy as np
from bert4keras.backend import keras, K
from bert4keras.models import build_transformer_model
from bert4keras.tokenizers import Tokenizer
from bert4keras.optimizers import Adam
from bert4keras.snippets import sequence_padding, DataGenerator
from bert4keras.snippets import open, ViterbiDecoder
from bert4keras.layers import ConditionalRandomField
from keras.layers import Dense
from keras.models import Model
from tqdm import tqdm

maxlen = 256
epochs = 10
batch_size = 32
bert_layers = 12
learing_rate = 1e-5  # bert_layers越小，学习率应该要越大
crf_lr_multiplier = 1000  # 必要时扩大CRF层的学习率

# bert配置
config_path = './chinese_L-12_H-768_A-12/bert_config.json'
checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
dict_path = './chinese_L-12_H-768_A-12/vocab.txt'


def load_data(filename):
    D = []
    with open(filename, encoding='utf-8') as f:
        f = f.read()
        for l in f.split('\n\n'):
            if not l:
                continue
            d, last_flag = [], ''
            for c in l.split('\n'):
                char, this_flag = c.split(' ')
                if this_flag == 'O' and last_flag == 'O':
                    d[-1][0] += char
                elif this_flag == 'O' and last_flag != 'O':
                    d.append([char, 'O'])
                elif this_flag[:1] == 'B':
                    d.append([char, this_flag[2:]])
                else:
                    d[-1][0] += char
                last_flag = this_flag
            D.append(d)
    return D


# 标注数据
train_data = load_data('./china-people-daily-ner-corpus/example.train')
valid_data = load_data('./china-people-daily-ner-corpus/example.dev')
test_data = load_data('./china-people-daily-ner-corpus/example.test')

# 建立分词器
tokenizer = Tokenizer(dict_path, do_lower_case=True)

# 类别映射
labels = ['PER', 'LOC', 'ORG']
id2label = dict(enumerate(labels))
```

```python
label2id = {j: i for i, j in id2label.items()}
num_labels = len(labels) * 2 + 1


class data_generator(DataGenerator):
    """数据生成器
    """
    def __iter__(self, random=False):
        batch_token_ids, batch_segment_ids, batch_labels = [], [], []
        for is_end, item in self.sample(random):
            token_ids, labels = [tokenizer._token_start_id], [0]
            for w, l in item:
                w_token_ids = tokenizer.encode(w)[0][1:-1]
                if len(token_ids) + len(w_token_ids) < maxlen:
                    token_ids += w_token_ids
                    if l == 'O':
                        labels += [0] * len(w_token_ids)
                    else:
                        B = label2id[l] * 2 + 1
                        I = label2id[l] * 2 + 2
                        labels += ([B] + [I] * (len(w_token_ids) - 1))
                else:
                    break
            token_ids += [tokenizer._token_end_id]
            labels += [0]
            segment_ids = [0] * len(token_ids)
            batch_token_ids.append(token_ids)
            batch_segment_ids.append(segment_ids)
            batch_labels.append(labels)
            if len(batch_token_ids) == self.batch_size or is_end:
                batch_token_ids = sequence_padding(batch_token_ids)
                batch_segment_ids = sequence_padding(batch_segment_ids)
                batch_labels = sequence_padding(batch_labels)
                yield [batch_token_ids, batch_segment_ids], batch_labels
                batch_token_ids, batch_segment_ids, batch_labels = [], [], []


"""
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：
model = build_transformer_model(
    config_path,
    checkpoint_path,
    model='albert',
)
output_layer = 'Transformer-FeedForward-Norm'
output = model.get_layer(output_layer).get_output_at(bert_layers - 1)
"""

model = build_transformer_model(
    config_path,
    checkpoint_path,
)

output_layer = 'Transformer-%s-FeedForward-Norm' % (bert_layers - 1)
output = model.get_layer(output_layer).output
output = Dense(num_labels)(output)
CRF = ConditionalRandomField(lr_multiplier=crf_lr_multiplier)
output = CRF(output)

model = Model(model.input, output)
model.summary()

model.compile(
    loss=CRF.sparse_loss,
    optimizer=Adam(learing_rate),
    metrics=[CRF.sparse_accuracy]
)


class NamedEntityRecognizer(ViterbiDecoder):
    """命名实体识别器
    """
    def recognize(self, text):
        tokens = tokenizer.tokenize(text)
        while len(tokens) > 512:
            tokens.pop(-2)
        mapping = tokenizer.rematch(text, tokens)
        token_ids = tokenizer.tokens_to_ids(tokens)
        segment_ids = [0] * len(token_ids)
        nodes = model.predict([[token_ids], [segment_ids]])[0]
        labels = self.decode(nodes)
        entities, starting = [], False
        for i, label in enumerate(labels):
            if label > 0:
                if label % 2 == 1:
                    starting = True
                    entities.append([[i], id2label[(label - 1) // 2]])
                elif starting:
                    entities[-1][0].append(i)
                else:
                    starting = False
            else:
                starting = False

        return [(text[mapping[w[0]][0]:mapping[w[-1]][-1] + 1], l)
                for w, l in entities]


NER = NamedEntityRecognizer(trans=K.eval(CRF.trans), starts=[0], ends=[0])


def evaluate(data):
    """评测函数
    """
    X, Y, Z = 1e-10, 1e-10, 1e-10
    for d in tqdm(data):
        text = ''.join([i[0] for i in d])
        R = set(NER.recognize(text))
        T = set([tuple(i) for i in d if i[1] != 'O'])
        X += len(R & T)
        Y += len(R)
        Z += len(T)
    f1, precision, recall = 2 * X / (Y + Z), X / Y, X / Z
    return f1, precision, recall


class Evaluate(keras.callbacks.Callback):
    def __init__(self):
        self.best_val_f1 = 0

    def on_epoch_end(self, epoch, logs=None):
        trans = K.eval(CRF.trans)
        NER.trans = trans
        print(NER.trans)
        f1, precision, recall = evaluate(valid_data)
        # 保存最优
        if f1 >= self.best_val_f1:
            self.best_val_f1 = f1
            model.save_weights('./best_model.weights')
        print(
            'valid:  f1: %.5f, precision: %.5f, recall: %.5f, best f1: %.5f\n' %
            (f1, precision, recall, self.best_val_f1)
        )
        f1, precision, recall = evaluate(test_data)
        print(
            'test:  f1: %.5f, precision: %.5f, recall: %.5f\n' %
            (f1, precision, recall)
        )


if __name__ == '__main__':

    evaluator = Evaluate()
    train_generator = data_generator(train_data, batch_size)

    model.fit_generator(
        train_generator.forfit(),
        steps_per_epoch=len(train_generator),
        epochs=epochs,
        callbacks=[evaluator]
    )

else:

    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]
__________________________________________________________________________________
________________
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 [ ]:

```
!git clone -q https://github.com/imcaspar/gpt2-ml
%cd /content/gpt2-ml
!mkdir -p /content/gpt2-ml/models/mega

!perl 3rd/gdown.pl/gdown.pl https://drive.google.com/open?id=1mT_qCQg4AWnAXTwKfsyyRWCRpgPrBJS3
!wget -q --show-progress https://github.com/imcaspar/gpt2-ml/releases/download/v1.0/model.ckp
!wget -q --show-progress https://github.com/imcaspar/gpt2-ml/releases/download/v1.0/model.ckp
!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 [ ]:

```
!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 [ ]:

```
!ls configs
```

```
base.json  large.json  mega.json
```

In [ ]:

```
!ls dataset
```

```
prepare_data.py  prepare_data.sh  README.md
```

In [ ]:

```
!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 [ ]:

```
!ls tokenization
```

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

In [ ]:

```python
import numpy as np
from bert4keras.models import build_transformer_model
from bert4keras.tokenizers import Tokenizer
from bert4keras.snippets import AutoRegressiveDecoder
from bert4keras.snippets import uniout

config_path = './configs/mega.json'
checkpoint_path = './models/mega/model.ckpt-220000'
dict_path = './tokenization/clue-vocab.txt'

tokenizer = Tokenizer(
    dict_path, token_start=None, token_end=None, do_lower_case=True
)  # 建立分词器

model = build_transformer_model(
    config_path=config_path, checkpoint_path=checkpoint_path, model='gpt2_ml'
)  # 建立模型，加载权重


class ArticleCompletion(AutoRegressiveDecoder):
    """基于随机采样的文章续写
    """
    #@AutoRegressiveDecoder.set_rtype('probas')
    def predict(self, inputs, output_ids, states=None, rtype='probas'):
        token_ids = np.concatenate([inputs[0], output_ids], 1)
        return model.predict(token_ids)[:, -1]

    def generate(self, text, n=1, topk=5):
        token_ids, _ = tokenizer.encode(text)
        results = self.random_sample([token_ids], n, topk)  # 基于随机采样
        return [text + tokenizer.decode(ids) for ids in results]


article_completion = ArticleCompletion(
    start_id=None,
    end_id=511,  # 511是中文句号
    maxlen=256,
    minlen=128
)
```

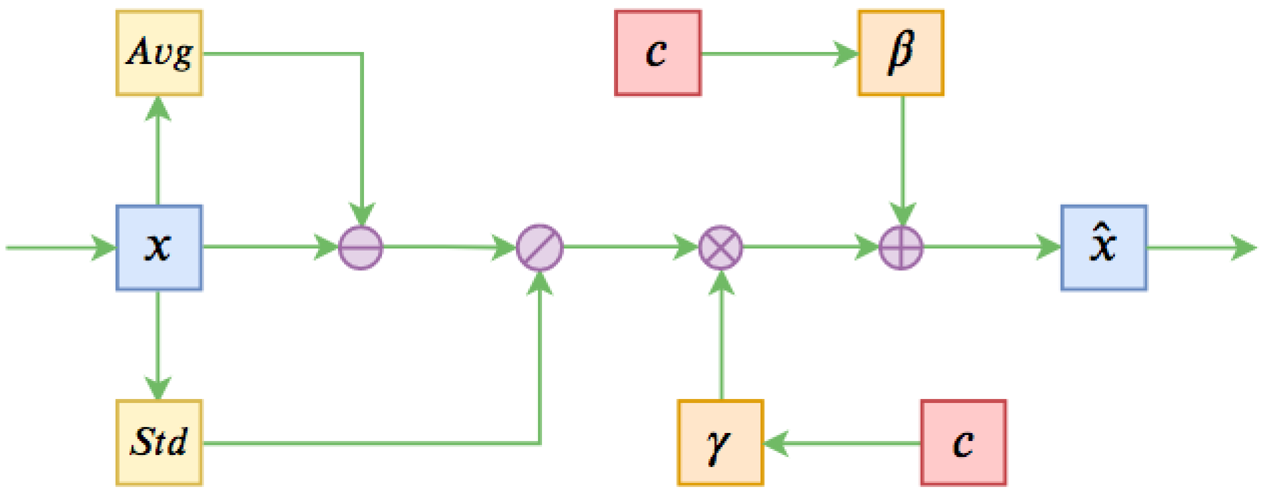
In [ ]:

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

In [ ]:

```python
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)



Avg
c
β
x
x̂
Std
γ
c

In [ ]:

```python
from __future__ import print_function
import re
import numpy as np
from bert4keras.backend import keras, K
from bert4keras.layers import Loss
from bert4keras.models import build_transformer_model
from bert4keras.tokenizers import Tokenizer, load_vocab
from bert4keras.optimizers import Adam
from bert4keras.snippets import sequence_padding, open
from bert4keras.snippets import text_segmentate
from bert4keras.snippets import DataGenerator, AutoRegressiveDecoder
from bert4keras.snippets import uniout  # 打印中文
from keras.layers import Input, Embedding, Reshape
from keras.models import Model

# 模型配置
maxlen = 128
batch_size = 32
num_classes = 2
epochs = 20

# bert配置
config_path = './chinese_L-12_H-768_A-12/bert_config.json'
checkpoint_path = './chinese_L-12_H-768_A-12/bert_model.ckpt'
dict_path = './chinese_L-12_H-768_A-12/vocab.txt'

# 加载并精简词表，建立分词器
token_dict, keep_tokens = load_vocab(
    dict_path=dict_path,
    simplified=True,
    startswith=['[PAD]', '[UNK]', '[CLS]', '[SEP]'],
)
tokenizer = Tokenizer(token_dict, do_lower_case=True)


def load_data(filenames):
    """加载数据，并尽量划分为不超过maxlen的句子
    """
    D = []
    seps, strips = u'\n。！？!?；;，, ', u'；;，, '
    for filename in filenames:
        with open(filename, encoding='utf-8') as f:
            for l in f:
                text, label = l.strip().split('\t')
                for t in text_segmentate(text, maxlen - 2, seps, strips):
                    D.append((t, int(label)))
    return D


# 加载数据集
data = load_data([
    './sentiment/sentiment.train.data',
    './sentiment/sentiment.valid.data',
    './sentiment/sentiment.test.data',
])


class data_generator(DataGenerator):
    """数据生成器
    """
    def __iter__(self, random=False):
        batch_token_ids, batch_segment_ids, batch_labels = [], [], []
        for is_end, (text, label) in self.sample(random):
            token_ids, segment_ids = tokenizer.encode(text, maxlen=maxlen)
            batch_token_ids.append(token_ids)
            batch_segment_ids.append(segment_ids)
            batch_labels.append([label])
            if len(batch_token_ids) == self.batch_size or is_end:
                batch_token_ids = sequence_padding(batch_token_ids)
                batch_segment_ids = sequence_padding(batch_segment_ids)
                batch_labels = sequence_padding(batch_labels)
                yield [batch_token_ids, batch_segment_ids, batch_labels], None
                batch_token_ids, batch_segment_ids, batch_labels = [], [], []


class CrossEntropy(Loss):
    """交叉熵作为loss，并mask掉padding部分
    """
    def compute_loss(self, inputs, mask=None):
        y_true, y_pred = inputs
        if mask[1] is None:
            y_mask = 1.0
        else:
            y_mask = K.cast(mask[1], K.floatx())[:, 1:]
        y_true = y_true[:, 1:]  # 目标token_ids
        y_pred = y_pred[:, :-1]  # 预测序列，错开一位
        loss = K.sparse_categorical_crossentropy(y_true, y_pred)
        loss = K.sum(loss * y_mask) / K.sum(y_mask)
        return loss


c_in = Input(shape=(1,))
c = Embedding(2, 128)(c_in)
c = Reshape((128,))(c)

# Bert模型
model = build_transformer_model(
    config_path,
    checkpoint_path,
    application='lm',
    keep_tokens=keep_tokens,  # 只保留keep_tokens中的字，精简原字表
    layer_norm_cond=c,
    additional_input_layers=c_in,
)

output = CrossEntropy(1)([model.inputs[0], model.outputs[0]])

model = Model(model.inputs, output)
model.compile(optimizer=Adam(1e-5))
model.summary()


class RandomSentiment(AutoRegressiveDecoder):
    """根据情感标签（0:负，1:正）随机生成一批句子
    """
    @AutoRegressiveDecoder.wraps(default_rtype='probas')
    def predict(self, inputs, output_ids, states):
        token_ids = output_ids
        segment_ids = np.zeros_like(token_ids)
        return model.predict([token_ids, segment_ids, inputs[0]])[:, -1]

    def generate(self, label, n=1, topk=5):
        results = self.random_sample([[label]], n, topk)  # 基于随机采样
        return [tokenizer.decode(ids) for ids in results]


random_sentiment = RandomSentiment(
    start_id=tokenizer._token_start_id,
    end_id=tokenizer._token_end_id,
    maxlen=maxlen
)


def just_show():
    print(u'正面采样:')
    print(random_sentiment.generate(1, 5, 5), '\n')
    print(u'负面采样:')
    print(random_sentiment.generate(0, 5, 5), '\n')


class Evaluate(keras.callbacks.Callback):
    def __init__(self):
        self.lowest = 1e10

    def on_epoch_end(self, epoch, logs=None):
        # 保存最优
        if logs['loss'] <= self.lowest:
            self.lowest = logs['loss']
            model.save_weights('./best_model.weights')
        # 演示效果
        just_show()


if __name__ == '__main__':

    evaluator = Evaluate()
    train_generator = data_generator(data, batch_size)

    model.fit_generator(
        train_generator.forfit(),
        steps_per_epoch=len(train_generator),
        epochs=epochs,
        callbacks=[evaluator]
    )

else:

    model.load_weights('./best_model.weights')
"""
正面采样:
[
    u'外观时尚、漂亮、性价比高。',
    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。',
    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
    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
    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
]
负面采样:
[
    u'不知道是不是因为电池不太好，不是我不喜欢。',
    u'看了评论才买的. 结果发现不是那么便宜, 价格也不便宜.',
    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',
    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
    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
]
"""
```