读取数据

In [1]:

```
import jieba

review_data=[]
labels=[]

f_pos=open('./data/pos.txt','r',encoding='utf8')
for line in f_pos:
    if line.strip('\n')!=None:
        tokens=list(jieba.cut(line))
        processed_sent=" ".join(tokens)
        review_data.append(processed_sent)
        labels.append(1)
    else:
        pass

f_neg=open('./data/neg.txt','r',encoding='utf8')
for line in f_neg:
    if line.strip('\n')!=None:
        tokens=list(jieba.cut(line))
        processed_sent=" ".join(tokens)
        review_data.append(processed_sent)
        labels.append(0)
    else:
        pass


print(len(review_data))
print(review_data[3],labels[3])
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\fengl\AppData\Local\Temp\jieba.cache
Loading model cost 0.344 seconds.
Prefix dict has been built successfully.
3089
首先 说 一下 物流 ， 中午 十一点左右 下单 ， 下午 5 点多 就 到 了 ， 速度 真的 是 快 。 在 买 之前 ， 纠结 了 好久 ， 不 知道 到底 买多大 的 ， 最后 决定 买 32G 的 ， 毕竟 只是 作为 娱乐 工具 ， 够用 就 好 了 。 买回来 之后 用 了 几天 才 来 评价 ， 说实话 真的 是 很 好 。 反应速度 非常 快 ， 没有 卡顿 现象 ， 至于 屏幕 ， 个人感觉 并 没有 网上 的 那些 问题 ， 色泽 清晰 ， 鲜明 。
 1
```

在开始前，先对所用数据做个初步探索。特别地，我们需要知道数据中有多少个不同的单词，每句话由多少个单词组成。

In [2]:

```
import numpy as np
avglen = 0  #句子最大长度

len_list=[]
for sent_str in review_data:
    words=list(jieba.cut(sent_str))
    length = len(words)
    len_list.append(length)

avglen=np.sum(np.array(len_list))/len(len_list)
print('avg_len:',avglen)
```

```
avg_len: 61.081579799287795
```

将句子进行数字向量化表示

In [3]:

```
from tensorflow.keras.preprocessing.text import Tokenizer
import numpy as np

tokenizer = Tokenizer() # 创建一个Tokenizer对象，将一个词转换为正整数
tokenizer.fit_on_texts(review_data)  #将词编号，词频越大，编号越小

word2index = tokenizer.word_index

vocab_size=len(word2index)

#print(vocab,len(vocab))

index2word = {word2index[word]:word for word in word2index}
x_word_ids = tokenizer.texts_to_sequences(review_data) #将句子中的每个词转换为数字
print(x_word_ids[1])
from tensorflow.keras.preprocessing.sequence import pad_sequences
x_padded_seqs = pad_sequences(x_word_ids,truncating='post',maxlen=100)#将每个句子设置

x_padded_seqs=np.array(x_padded_seqs)

print(x_padded_seqs[1])
#print(vocab)

#print(x_padded_seqs[2])
```

```
[25, 14, 20, 1, 9, 365, 2, 123, 1, 1044, 94, 234, 2, 3520, 43, 100, 40, 305, 10, 2495,
3521, 2, 388, 1, 1138, 27, 377, 110, 3522, 1045, 1, 779, 267, 780, 2, 20, 23, 1, 1657,
38, 78, 23, 40, 2, 234, 7, 474, 31, 1, 6, 50, 29, 23, 781, 3523, 295, 1, 228, 3524, 2,
1, 7, 8, 3, 50, 337, 2, 1, 601, 1436, 15, 3525, 966, 1, 69, 2496, 1045, 329, 446, 2,
1, 1276, 29, 1437, 2497, 421, 1045, 43, 9, 66, 3526, 397, 174, 1658, 229, 8, 548, 2,
1, 149, 6, 7, 156, 3, 632, 295, 74, 3527, 7, 330, 1659, 2019, 107, 1660, 3, 4]
[  25   14   20    1    9  365    2  123    1 1044   94  234    2 3520
   43  100   40  305   10 2495 3521    2  388    1 1138   27  377  110
 3522 1045    1  779  267  780    2   20   23    1 1657   38   78   23
   40    2  234    7  474   31    1    6   50   29   23  781 3523  295
    1  228 3524    2    1    7    8    3   50  337    2    1  601 1436
   15 3525  966    1   69 2496 1045  329  446    2    1 1276   29 1437
 2497  421 1045   43    9   66 3526  397  174 1658  229    8  548    2
    1  149]
```

构建模型

In [4]:

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding,LSTM,Dense,Activation,Dropout
from tensorflow.keras.utils import to_categorical
import numpy as np


# 创建深度学习模型，  Embedding + LSTM + Softmax.
def create_LSTM(n_units, input_size, output_dim,vocab_size):
    model = Sequential()
    model.add(Embedding(input_dim=vocab_size + 1, output_dim=output_dim,
                        input_length=input_size, mask_zero=True))
    model.add(LSTM(n_units, input_shape=(None,input_size)))
    model.add(Dropout(0.2))
    model.add(Dense(output_dim , activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accura
    model.summary()
```

```
    return model
```

模型训练

In [5]:

```
# 将数据集分为训练集和测试集，占比为9:1


x_train,x_test,y_train,y_test=train_test_split(x_padded_seqs,labels,test_size=0.2,shu

x_train=np.array(x_train)
x_test=np.array(x_test)
y_train=np.array(y_train)
y_test=np.array(y_test)

y_train_onehot=to_categorical(y_train)
y_test_hot=to_categorical(y_test)
# 模型输入参数，需要自己根据需要调整
input_size = x_padded_seqs.shape[1]

print(input_size)
n_units = 100
batch_size = 32
epochs = 5
output_dim = 2

# 模型训练
lstm_model = create_LSTM(n_units, input_size, output_dim,vocab_size=vocab_size)
lstm_model.fit(x_train, y_train_onehot, epochs=epochs, batch_size=batch_size, verbose
```

```
100
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 2)            14480
_________________________________________________________________
lstm (LSTM)                  (None, 100)               41200
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 2)                 202
=================================================================
Total params: 55,882
Trainable params: 55,882
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
78/78 [==============================] - 2s 26ms/step - loss: 0.5627 - accuracy: 0.700
9
Epoch 2/5
78/78 [==============================] - 3s 37ms/step - loss: 0.3872 - accuracy: 0.874
1
Epoch 3/5
78/78 [==============================] - 3s 39ms/step - loss: 0.2217 - accuracy: 0.938
9
Epoch 4/5
78/78 [==============================] - 3s 38ms/step - loss: 0.1536 - accuracy: 0.961
```

```
1
Epoch 5/5
78/78 [==============================] - 3s 38ms/step - loss: 0.1174 - accuracy: 0.973
3
```

Out[5]: <tensorflow.python.keras.callbacks.History at 0x2ec43fdae80>

模型效果测试

In [6]:

```
from sklearn.metrics import accuracy_score
results=lstm_model.predict(x_test)
result_labels = np.argmax(results, axis=-1)  # 获得最大概率对应的标签
#print(result_labels)
print('准确率', accuracy_score(y_test, result_labels))
```

准确率 0.9320388349514563

对新的文本进行预测

In [7]:

```
new_reviews=['体验不是很好，信号差还发烫。手机第一次充电就发烫的要死。热点总是自动打开

new_sents=[]
for sent_str in new_reviews:
    tokens=jieba.cut(sent_str)
    sent=' '.join(tokens)
    new_sents.append(sent)



x_new_ids = tokenizer.texts_to_sequences(new_sents) #将句子中的每个词转换为数字
print(x_new_ids[0])

x_new_padseqs = pad_sequences(x_new_ids,truncating='post',maxlen=100)#将每个句子设置为

print(x_new_padseqs)

probs=lstm_model.predict(x_new_padseqs)



new_labels=np.argmax(probs,axis=-1)

print(new_labels)
```

```
[187, 53, 14, 20, 1, 1088, 68, 12, 808, 4, 241, 94, 213, 7, 808, 2, 628, 4, 2815, 809,
692, 259, 1, 52, 692, 259, 4]
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0  187   53   14   20    1 1088   68   12  808    4  241
    94  213    7  808    2  628    4 2815  809  692  259    1   52  692
   259    4]]
[0]
```