

# 基于 RNN 的莎士比亚式语句生成

## 1. 实验目的

训练多层 RNN 网络，实现莎士比亚式语句的生成，加深对 RNN 相关基础模型层的了解

## 2. 实验环境

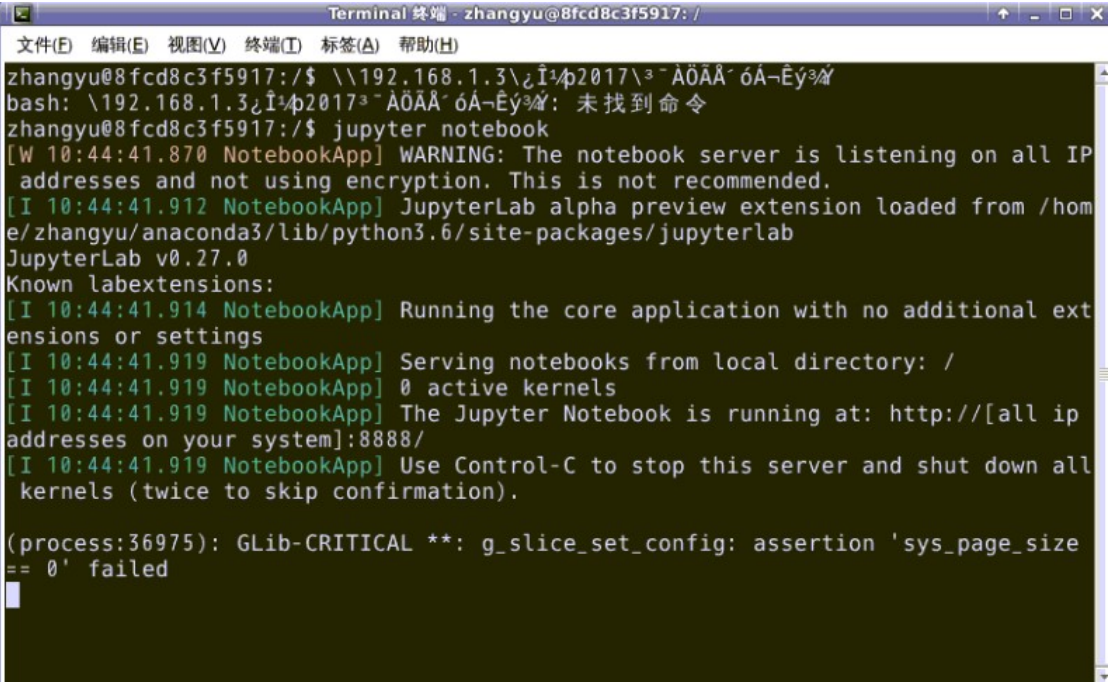
Linux Ubuntu 16.04

Python 3.6.1

Jupyter

## 3. 实验步骤

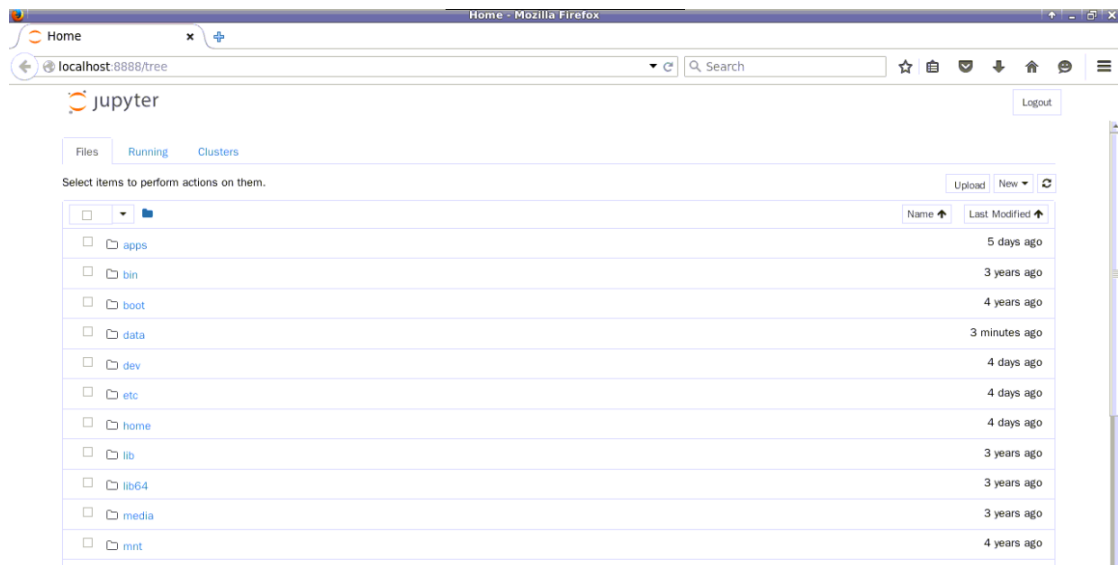
1. 首先打开终端模拟器，输入下面命令：jupyter notebook -ip= '127.0.0.1'

A terminal window titled 'Terminal 终端 - zhangyu@8fcd8c3f5917: /' showing the execution of 'jupyter notebook'. The output includes a warning about listening on all IP addresses, the loading of the JupyterLab alpha preview extension, and the server starting on port 8888. The terminal text is as follows:

```
zhangyu@8fcd8c3f5917:/$ \192.168.1.3\2017\3~A0AA~6A-Ey3W
bash: \192.168.1.3\2017\3~A0AA~6A-Ey3W: 未找到命令
zhangyu@8fcd8c3f5917:/$ jupyter notebook
[W 10:44:41.870 NotebookApp] WARNING: The notebook server is listening on all IP
addresses and not using encryption. This is not recommended.
[I 10:44:41.912 NotebookApp] JupyterLab alpha preview extension loaded from /hom
e/zhangyu/anaconda3/lib/python3.6/site-packages/jupyterlab
JupyterLab v0.27.0
Known labextensions:
[I 10:44:41.914 NotebookApp] Running the core application with no additional ext
ensions or settings
[I 10:44:41.919 NotebookApp] Serving notebooks from local directory: /
[I 10:44:41.919 NotebookApp] 0 active kernels
[I 10:44:41.919 NotebookApp] The Jupyter Notebook is running at: http://[all ip
addresses on your system]:8888/
[I 10:44:41.919 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).

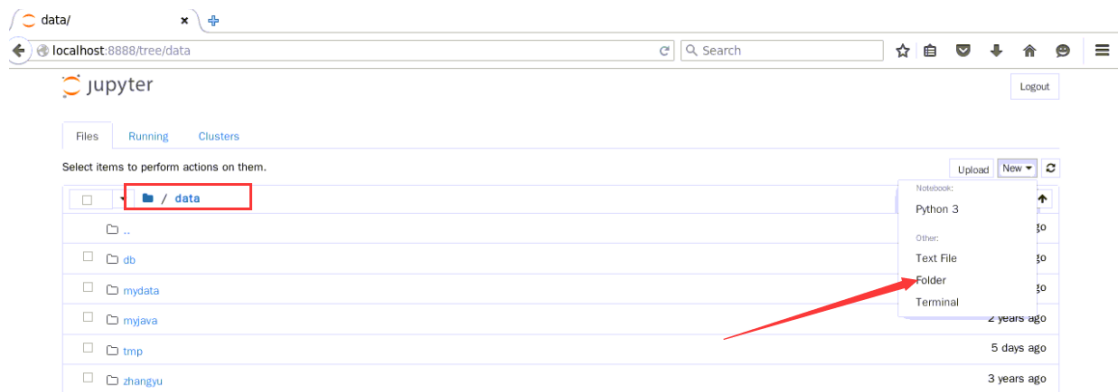
(process:36975): GLib-CRITICAL **: g_slice_set_config: assertion 'sys_page_size
== 0' failed
```

如上图所示，该终端不要关闭，在浏览器中会打开下面界面，

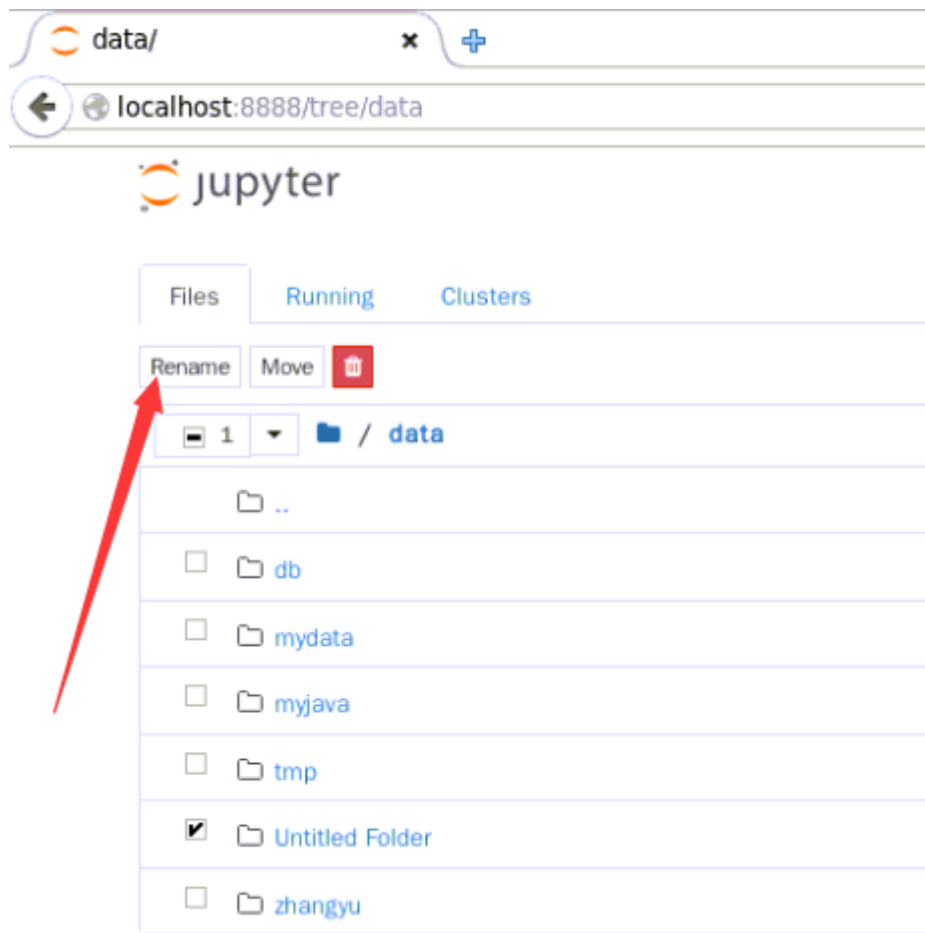


如果是第一次打开，浏览器界面会要求输入密码，密码为 zhangyu

2. 切换到/data 目录下，点击 New，在其下拉框中选择 folder

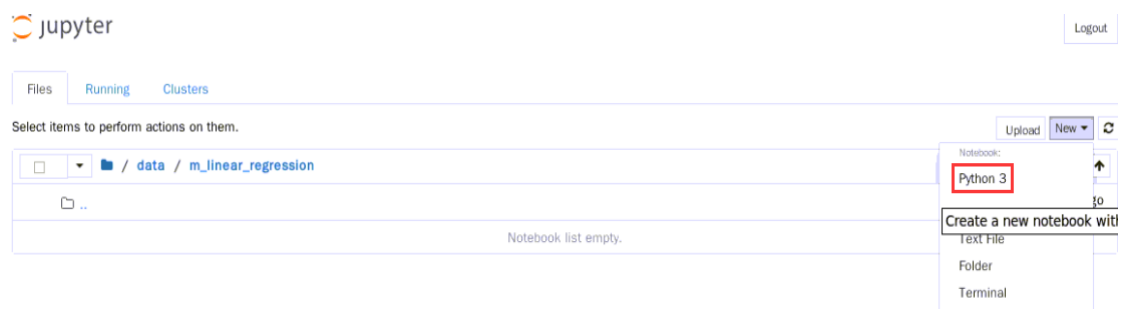


选中刚才创建的文件夹，点击页面左上角的【Rename】

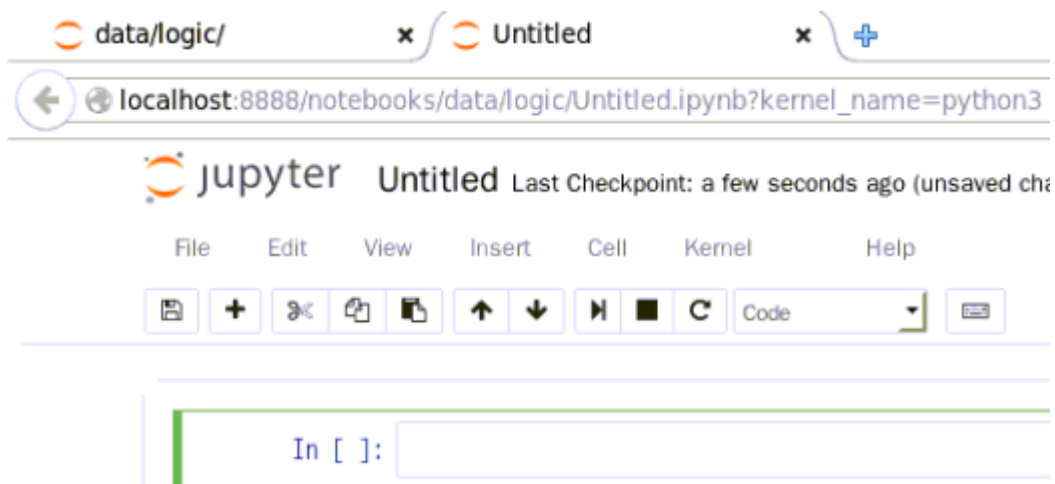


重命名为 logic（命名无要求）

3. 切换到 myapp 目录下，新建一个 logic 文件，用于编写并执行代码。点击页面右上角的 New，选中【Python3】



新建 ipynb 文件如下所示，在此可以编写代码了



## 4. python 包导入

载入各类程序需要的库和包（pycharm 和 jupyter 对一些包的版本要求可能不同）

---

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers, callbacks
import tensorflow.keras.utils as ku
import numpy as np
import matplotlib.pyplot as plt
```

---

没有的包可以使用 `pip install` 命令安装。

## 5. 数据集处理

获取莎士比亚数据集，方法是进入该网站后，将内容全选复制，并保存到一个新.txt 文档中保存。保存后先全部转换小写，后按行分割到列表中。

---

```
# 数据集处理
# 进入该网站后，全选复制到一个新的.txt 文档中保存
# https://storage.googleapis.com/laurencemoroney-blog.appspot.com/sonnets.txt
data = open('./datasets/sonnets.txt').read()
corpus = data.lower().split("\n")
```

---

```
FROM fairest creatures we desire increase,  
That thereby beauty's rose might never die,  
But as the ripper should by time decease,  
His tender heir might bear his memory:  
But thou, contracted to thine own bright eyes,  
Feed'st thy light'st flame with self-substantial fuel,  
Making a famine where abundance lies,  
Thyself thy foe, to thy sweet self too cruel.  
Thou that art now the world's fresh ornament  
And only herald to the gaudy spring,  
Within thine own bud buriest thy content  
And, tender churl, makest waste in niggarding.  
Pity the world, or else this glutton be,  
To eat the world's due, by the grave and thee.  
When forty winters shall beseege thy brow,  
And dig deep trenches in thy beauty's field,  
Thy youth's proud livery, so gazed on now,  
Will be a tatter'd weed, of small worth held:  
Then being ask'd where all thy beauty lies,  
Where all the treasure of thy lusty days,  
To say, within thine own deep-sunken eyes,  
Were an all-eating shame and thriftless praise.  
How much more praise deserved thy beauty's use,  
If thou couldst answer 'This fair child of mine  
Shall sum my count and make my old excuse,'
```

## 6. 数据预处理

由于原始数据是以英文单词形式存储，故需要先将词 token 进行转换。完成转换后，再将不同长度的句子（序列）统一填充到相同长度，便于后续的训练。

---

```
# 使用 token 列表创建输入序列  
tokenizer = Tokenizer()  
tokenizer.fit_on_texts(corpus)  
total_words = len(tokenizer.word_index) + 1  
  
input_sequences = []  
for line in corpus:  
    token_list = tokenizer.texts_to_sequences([line])[0]  
    for i in range(1, len(token_list)):  
        n_gram_sequence = token_list[:i + 1]  
        input_sequences.append(n_gram_sequence)  
  
# padding 输入序列  
max_sequence_len = max([len(x) for x in input_sequences])  
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len,  
padding='pre'))
```

---

## 7. 数据集特征标签处理

针对原数据，设置不同的标签，方便进行训练。

---

```
# 创建预测值与标签  
predictors, label = input_sequences[:, :-1], input_sequences[:, -1]  
label = ku.to_categorical(label, num_classes=total_words)
```

---

## 8. 创建模型

设置网络模型、优化器和损失函数。网络在通过嵌入层后，进入双向 LSTM (BiLSTM)，进行 0.2 的结点随机失活，再进入一个普通 LSTM 中，最后通过两层全连接层输出。使用 Adam 方法，分类交叉熵损失函数进行训练。

---

```
# 构建网络模型
model = Sequential([
    Embedding(total_words, 100, input_length=max_sequence_len - 1),
    Bidirectional(LSTM(150, return_sequences=True)),
    Dropout(0.2),
    LSTM(100),
    Dense(total_words / 2, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dense(total_words, activation='softmax')
])

# 设置网络优化器与损失函数
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

---

## 9. 训练模型

设置回调函数，用于存储最佳效果的模型，后开始训练。

---

```
callback = callbacks.ModelCheckpoint(filepath='./saved_models/shakespeare.h5',
                                    monitor='loss',
                                    save_weights_only=True,
                                    save_best_only=True,
                                    verbose=1,
                                    period=1)

history = model.fit(predictors, label, epochs=100, verbose=1, callbacks=callback)
```

---

## 10. 绘制训练曲线

使用 plt 对训练正确率和损失进行绘图，可视化训练过程。

---

```
# 可视化训练过程
acc = history.history['accuracy']
loss = history.history['loss']

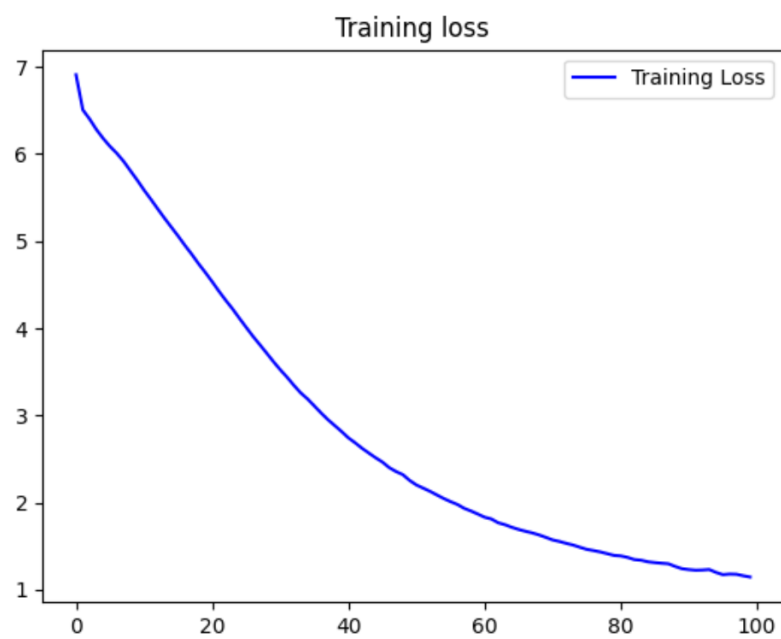
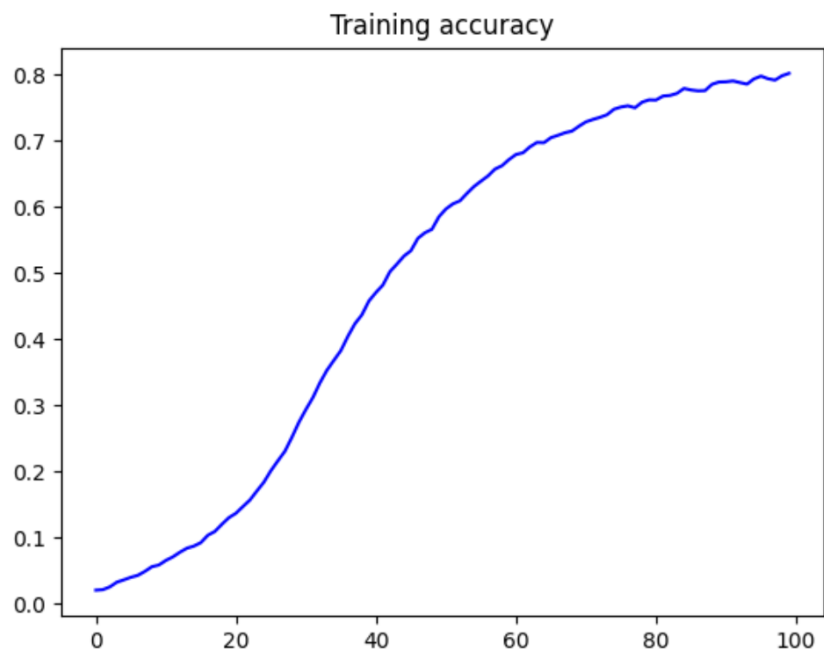
epochs = range(len(acc))

plt.plot(epochs, acc, 'b', label='Training accuracy')
```

```
plt.title('Training accuracy')
plt.figure()
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.title('Training loss')
plt.legend()

plt.show()
```

---



## 11. 进行自定义预测

自行设置语句，放入到网络中进行预测与输出。

---

```
seed_text = "Help me Obi Wan Kenobi, you're my only hope"
next_words = 100

for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len - 1, padding='pre')
    predicted = model.predict_classes(token_list, verbose=0)
    output_word = ""
    for word, index in tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    seed_text += " " + output_word
print(seed_text)
```

---

```
Help me Obi Wan Kenobi, you're my only hope set the spite of bath bow bow way pleasure give cross tell dead,
reeks dead end kill'd kill'd hour hour slain tend confounds weeds bright near rotten ward open about ,
greater lips strong painted cold leaves increase dead near burn'd lies twain cross near burn'd free lies ,
told me young days bearing thee tongue lived more leaves 'greeing hate lies bright live rolling bright in ,
thought hate hate strong pride still thee can tell thee with approve another pride well bright alone lies ,
buried state to thee free as now lies live made bad bad well made a very lays
```