

基于简单 RNN 单元的恐龙名字生成

1. 实验目的

利用简单的 RNN 单元，实现恐龙名字生成，了解 RNN 单元的使用方法

2. 实验环境

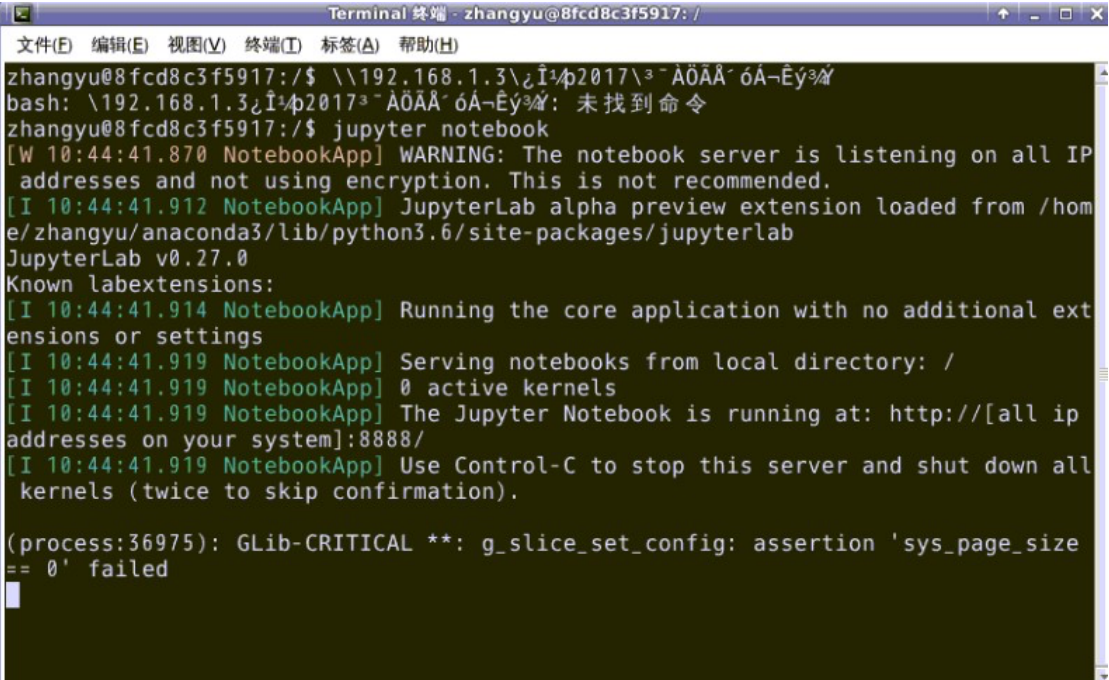
Linux Ubuntu 16.04

Python 3.6.1

Jupyter

3. 实验步骤

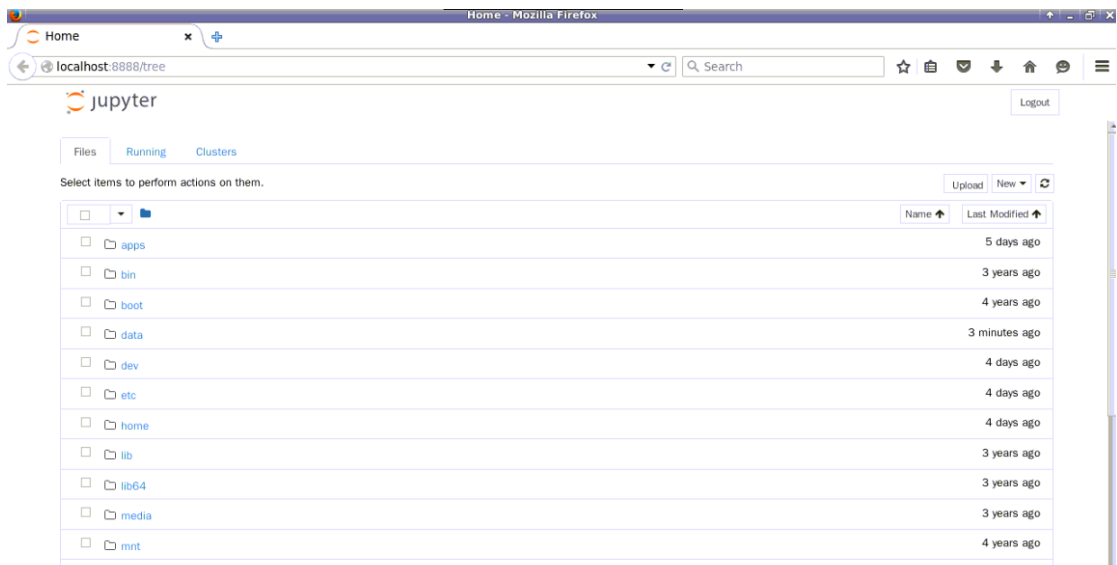
1. 首先打开终端模拟器，输入下面命令：jupyter notebook -ip= '127.0.0.1'

A terminal window titled "Terminal 终端 - zhangyu@8fcd8c3f5917: /" showing the execution of the command 'jupyter notebook'. The output includes a warning about listening on all IP addresses, the loading of the JupyterLab alpha preview extension, and the successful start of the Jupyter Notebook server at 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).

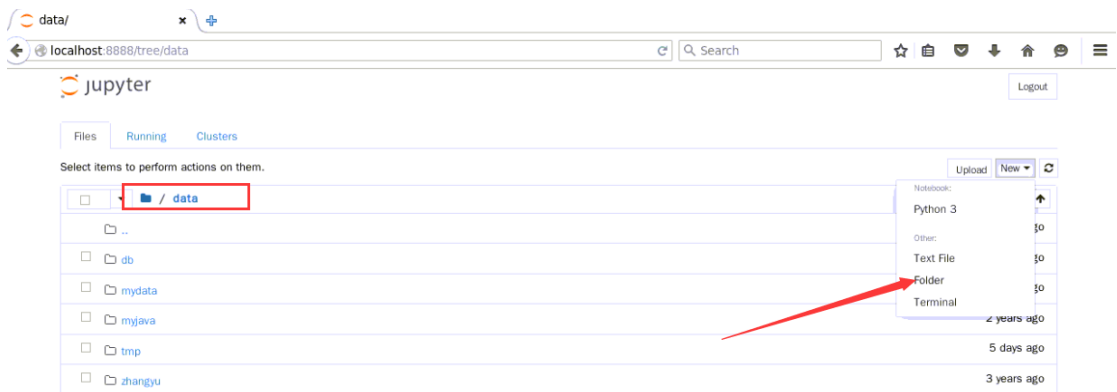
(process:36975): GLib-CRITICAL **: g_slice_set_config: assertion 'sys_page_size
== 0' failed
```

如上图所示，该终端不要关闭，在浏览器中会打开下面界面，

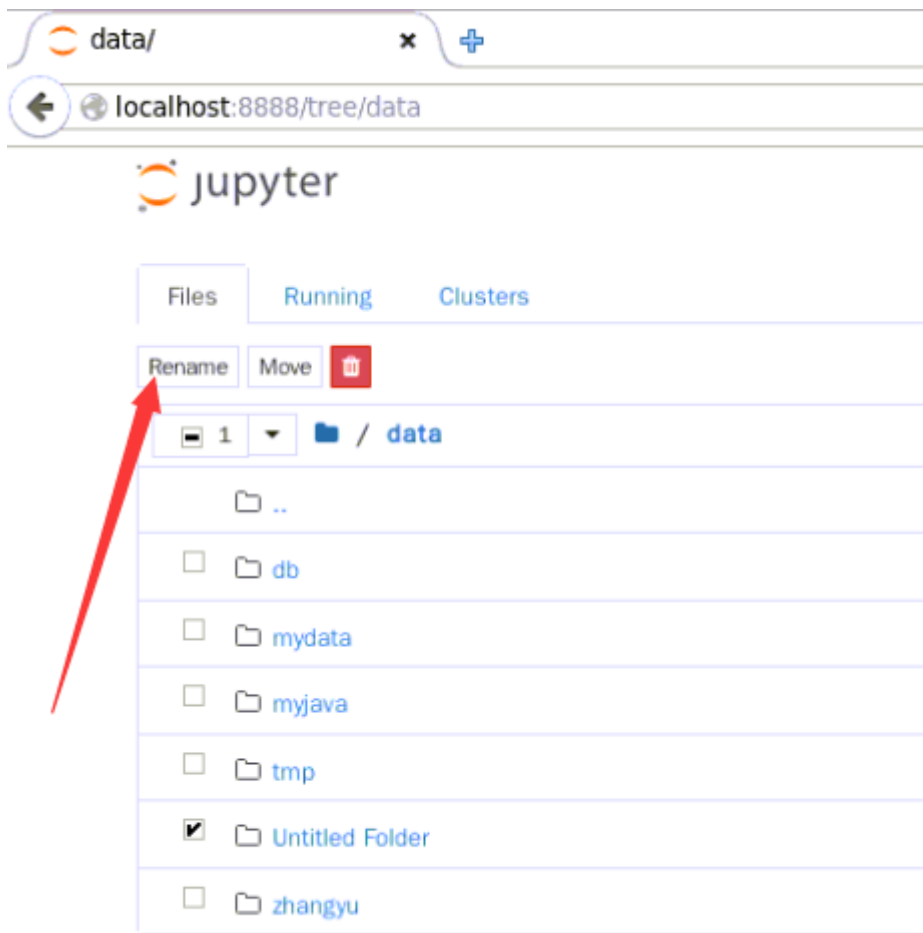


如果是第一次打开，浏览器界面会要求输入密码，密码为 zhangyu

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

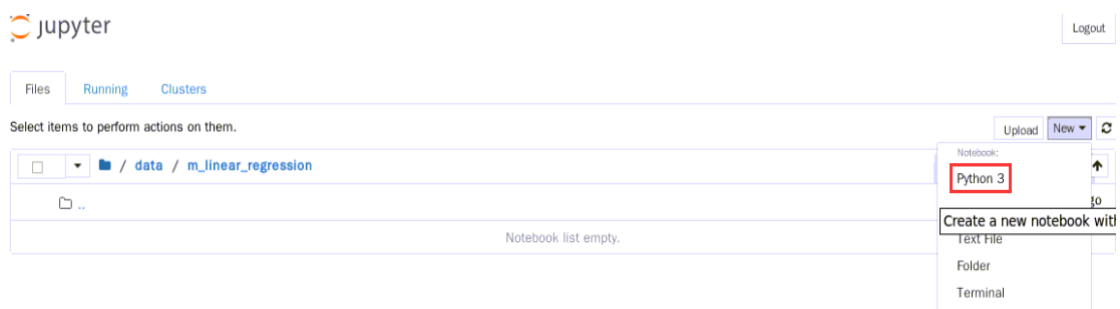


选中刚才创建的文件夹，点击页面左上角的【Rename】

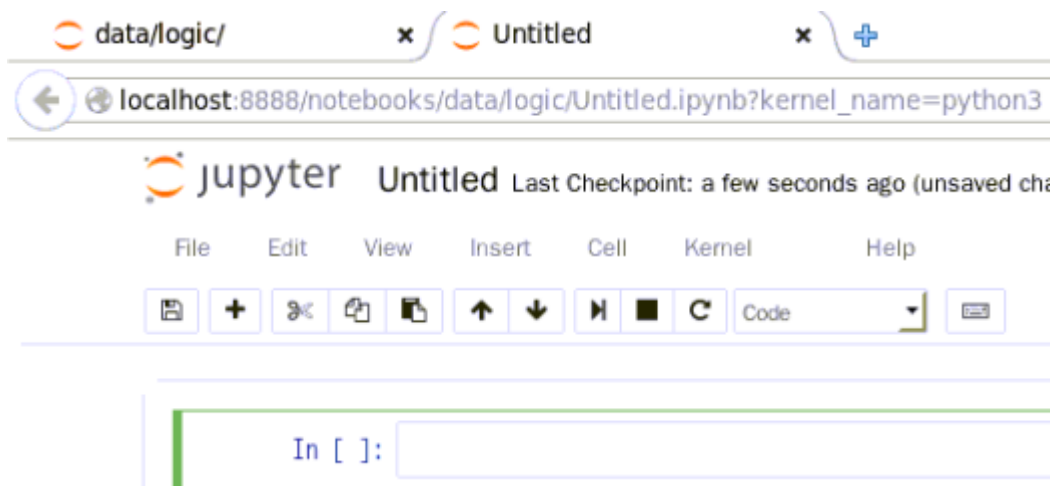


重命名为 logic（命名无要求）

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



1. 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.layers import Dense, SimpleRNN, Embedding
import tensorflow as tf
import numpy as np
import random
```

没有的包可以使用 `pip install` 命令安装。

2. 数据集处理

加载文本获取恐龙名字。创建字符表，计算样本与字符表的长度(大小写不区分)，进行 id 与 char 的双向表示操作。

```
# 数据预处理
# 加载文本获取恐龙名字。创建字符表，计算样本与字符表的长度。(大小写不区分)
data = open('./datasets/dinos.txt').read()
data = data.lower()
char = sorted(set(data))
char_num = len(char)
print(f'样本长度{len(data)},字符数量{char_num}')

# 创建对照列表。char2id 表示字符映射到数字。id2char 表示数字映射到字符。
# '\n'表示<EOS>，对应 0.
char2id = {i: u + 1 for u, i in enumerate(char)}
id2char = {u + 1: i for u, i in enumerate(char)}
char2id, id2char

# 创建训练集
with open('./datasets/dinos.txt') as f:
    examples = f.readlines()
```

```
examples = [x.lower().strip() for x in examples]
maxlen = max([len(i) for i in examples])
examples[0]
```

```
# 将训练集的字符变为数字编码
X, Y = [], []
for index in range(len(examples)):
    x = [char2id[ch] for ch in examples[index]]
    y = x[1:] + [char2id["\n"]]
    X.append(x)
    Y.append(y)
X[0], Y[0]
```

恐龙名示例：

| | |
|----|-------------------------|
| 1 | <u>Aachenosaurus</u> |
| 2 | <u>Aardonyx</u> |
| 3 | <u>Abdallahsaurus</u> |
| 4 | <u>Abelisaurus</u> |
| 5 | <u>Abrictosaurus</u> |
| 6 | <u>Abrosaurus</u> |
| 7 | <u>Abydosaurus</u> |
| 8 | <u>Acanthopholis</u> |
| 9 | <u>Achelousaurus</u> |
| 10 | <u>Acheroraptor</u> |
| 11 | <u>Achillesaurus</u> |
| 12 | <u>Achillobator</u> |
| 13 | <u>Acristavus</u> |
| 14 | <u>Acrocanthosaurus</u> |

3. 数据预处理

将输入的名字 pad 统一到相同长度，并将训练集打乱。同时将训练数据变成可迭代的数据，便于后续进行训练。

```
# 将输入 padding 为同一长度
X = np.array(X)
Y = np.array(Y)
padded_X = tf.keras.preprocessing.sequence.pad_sequences(X, maxlen=maxlen,
padding='post', value=0)
padded_Y = tf.keras.preprocessing.sequence.pad_sequences(Y, maxlen=maxlen,
padding='post', value=0)
print(padded_X.shape, padded_Y.shape)
```

```
# 将训练集随机打乱
```

```

np.random.seed(3)
np.random.shuffle(X)
np.random.seed(3)
np.random.shuffle(Y)
X[3], Y[3]
print(type(padded_X[0]))

train_db = tf.data.Dataset.from_tensor_slices((padded_X, padded_Y))
train_db = train_db.batch(32, drop_remainder=True)

train_iter = iter(train_db)
# next() 返回迭代器的下一个项目
sample = next(train_iter)
print('batch:', sample[0].shape, sample[1].shape)
print(sample[0][0], sample[1][0])

```

4. 创建模型

创建模型，注意 embedding 层是 vocab_size+1，应为加入了 padding 0。最后的 softmax 也是 vocab_size+1，注意是 return_sequences=True, 应为每一个时刻我们都要产生输出。在定义优化器和损失函数时，我们要将 padding 0 位置上产生的损失 mask 掉。

```

class My_model(tf.keras.Model):
    def __init__(self, vocab_size, rnn_units):
        super(My_model, self).__init__()
        self.embedding = Embedding(vocab_size + 1, 5, name='emb')
        self.rnn = SimpleRNN(rnn_units, return_sequences=True, name='rnn')
        # self.d1=Dense(64,activation='relu',name='d1')
        self.d2 = Dense(vocab_size + 1, activation='softmax', name='d2')

    def call(self, x):
        x = self.embedding(x)
        x = self.rnn(x)
        # x=self.d1(x)
        x = self.d2(x)
        return x

model = My_model(char_num, 16)

# 定义优化器和损失函数
loss_object = tf.keras.losses.SparseCategoricalCrossentropy()
optimizer = tf.keras.optimizers.Adam(1e-3)

```

```
def loss_function(y_true, y_pred):
    # 我们将 0mask 掉，不计算 0 的损失
    mask = tf.math.logical_not(tf.math.equal(y_true, 0))
    loss = loss_object(y_true, y_pred)
    mask = tf.cast(mask, dtype=loss.dtype)
    loss *= mask
    return tf.reduce_mean(loss)
```

5. 取样查看

我们在一个时刻会得到一个预测，然后我们需要将这个预测结果作为下一个时间点的输入，然后进行下一次预测。我们输出的 `yt` 是 softmax 之后的结果，代表我们预测下一个单词的概率，然后我们需要依照概率进行抽样（注意不像往常依照取 `argmax`，因为这样我们很有可能产生死循环）。

```
def sample(model):
    seed = 0
    name = []
    for i in range(5):
        a = [random.randint(1, 27)]
        b = tf.expand_dims(a, 0)
        ans = [id2char[a[0]].upper()]
        for i in range(20):
            pred = model(b)
            pred = tf.squeeze(pred)
            pred = np.array(pred)

            # for grading purposes
            np.random.seed(i + seed)

            idx = np.random.choice(list(range(28)), p=pred.ravel())
            if idx == 0 or idx == 1:
                break
            next_word = id2char[idx]
            ans.append(next_word)
            a = [char2id[next_word]]
            b = tf.expand_dims(a, 0)
            seed += 1

    ans = ''.join(ans)
    name.append(ans)
    for n in name:
        if n is not None:
            print(n)
```

6. 训练模型

训练的时候，我们不像预测进行采样，因为训练的时候，我们预测的结果很有可能是错的，然后我们传入错误的结果进行预测，那么产生的下一个结果就更加糟糕了。所以我们使用教师强制（teaching force）的方式。将下一个正确的答案输入到模型，然后进行下一次的预测。此外，我们需要对模型进行梯度裁剪，避免梯度爆炸。

在每一次训练周期进行输出，可以查看一开始生成的名字乱七八糟，后来的名字逐渐有规律了。

```
@tf.function
def train_step(inp, targ):
    loss = 0

    with tf.GradientTape() as tape:
        # 教师强制-将目标词作为下一个输入
        model_input = inp[:, 0]
        model_input = tf.expand_dims(model_input, 1)
        for t in range(1, targ.shape[1]):
            # 将编码器输出传到解码器
            predictions = model(model_input)
            loss += loss_function(targ[:, t], predictions)

            # 使用教师强制
            model_input = tf.expand_dims(targ[:, t], 1)

    batch_loss = (loss / int(targ.shape[1]))
    variables = model.variables

    # 对每一个变量计算梯度
    gradients = tape.gradient(loss, variables)
    gradients, _ = tf.clip_by_global_norm(gradients, 3)

    # 对变量更新梯度
    optimizer.apply_gradients(zip(gradients, variables))
    return batch_loss

EPOCHS = 100
steps_per_epoch = len(X) // 32
for epoch in range(EPOCHS):

    total_loss = 0
```



```
for (batch, (inp, targ)) in enumerate(train_db.take(steps_per_epoch)):
    batch_loss = train_step(inp, targ)
    total_loss += batch_loss

print('Epoch {} Loss {:.4f}'.format(epoch + 1, total_loss))
print()

sample(model)
print()
```
