计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：Fun with RNNs		学号：201900301174
日期：2021.11.15	班级： 智能 19	姓名： 韩旭
Email：hanx@mail.sdu.edu.cn		

实验目的：

Fun with RNNs

In this project, you will work on extending min-char-rnn.py. This was written by Andrej Karpathy[1]. You will experiment with the Shakespeare dataset (shakespeare_train.txt).

实验软件和硬件环境：

Jupyter notebook

NVIDIA GeForce RTX 3090

Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz

实验原理和方法：

Part 1: (20 points)

The RNN language model uses a softmax activation function for its output distribution at each time step. It's possible to modify the distribution by multiplying the logits by a constant α:

$$y = \text{softmax}(\alpha z)$$

Here, 1/α can be thought of as a "temperature", i.e. lower values of α correspond to a "hotter" distribution. (This terminology comes from an algorithm called simulated annealing.)

Write a function to sample text from the model using different temperatures. Try different temperatures, and, in your report, include examples of texts generated using different temperatures. Briefly discuss what difference the temperature makes.

Include the listing (i.e., source code) of the function you wrote/modified to accomplish the task in the report.

You should either train the RNN yourself, or use the weights from Part 3 -- up to you.

Part 2: (50 points)

Write a function that uses an RNN to complete a string. That is, the RNN should generate text that is a plausible continuation of a given starter string. In order to do that, you will need to compute the hidden activity h at the end of the starter string, and then to start generating new text.

Include 5 interesting examples of outputs that your network generated using a starter string. (This part need not be easily reproducible).

Include the listing (i.e., source code) of the function you wrote in the report.

You should either train the RNN yourself, or use the weights from Part 3 -- up to you.

Part 3: (30 points)

The weights for a trained RNN are included as char-rnn-snapshot.npz. Some samples from the RNN (at temperature 1/α=1) are included as samples.txt, and code to read in the weights is includes as read_in_npz.py (if this doesn't work, try the pickle file and get a using import cPickle as pickle; a = pickle.load(open("char-rnn-snapshot.pkl")).

In the samples that the RNN generated, it seems that a newline or a space usually follow the colon (i.e., ":") character. In the weight data provided, identify the specific weights that are responsible for this behavior by the RNN. In your report, specify the coordinates and values of the weights you identified, and explain how those weights make the RNN generate newlines and spaces after colons.

Part 4: (10 points bonus)

Identify another interesting behaviour of the RNN, identify the weights that are responsible for it.

Specify the coordinates and the values of the weights, and explain how those weights lead to the behaviour that you identified. To earn bonus points, the behaviour has to be more interesting than the behaviour in Part 3 (i.e., character A following character B)

实验步骤：（不要求罗列完整源代码）

PART 1

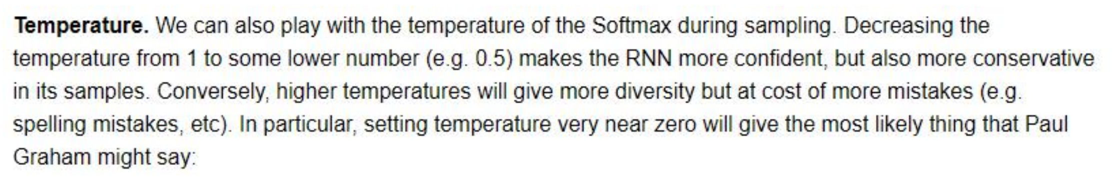
- xs the input sequence, encoded using one-hot encoding. Denote it by x_t.
- hs the hidden state (a vector), at each time step. Denote it by $h_t = \tanh(W^{xh}x_t + W^{hh}h_{t-1})$
- ys the output layer. Denote it by $y_t = W^{hy}h_t + b^y$
- ps the output of the softmax. Denote it by $\hat{y}_t = softmax(y_t)$
- loss the cost/loss function. $Cost = -\sum_t \log(\sum_k \hat{y}_t^k x_t^k)$

As discussed in lecture, when sampling from the RNN, we can sample using different "temperatures." We can sample the character at time-step $(t+1)$ by setting the probability of sampling character i to be proportional to $\exp(\alpha y_i^{(t)})$:

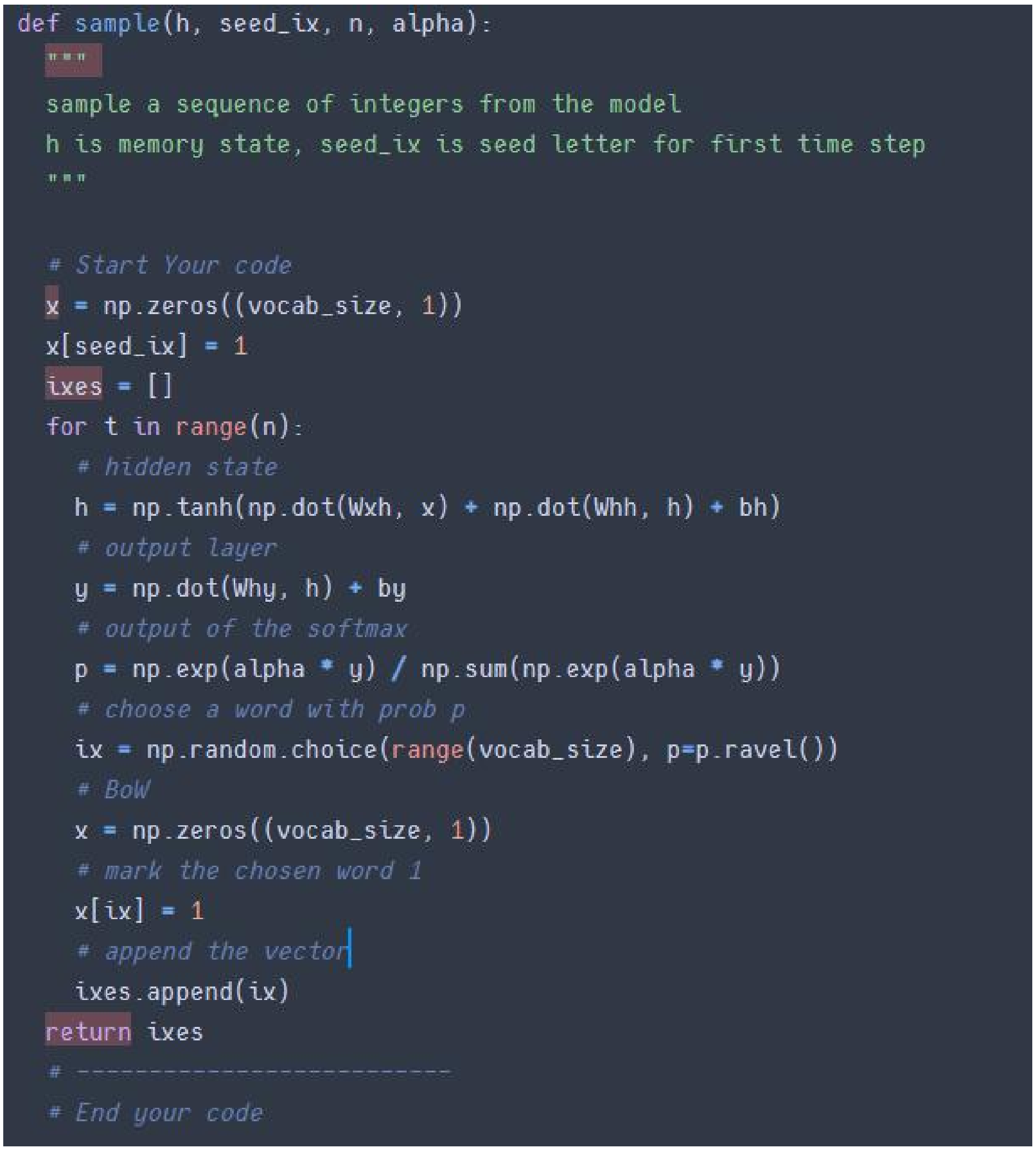
$$P(x^{(t+1)} = i) = \frac{\exp(\alpha y_i^{(t)})}{\sum_{i'=1}^{k} \exp(\alpha y_{i'}^{(t)})}.$$

The quantity $1/\alpha$ is called the "temperature."

上图就是加上温度值的softmax函数,我们可以尝试多个temperature的值,在Andrej Karpathy的*The Unreasonable Effectiveness of Recurrent Neural Networks*中对temperature的介绍如下图,


Temperature. We can also play with the temperature of the Softmax during sampling. Decreasing the temperature from 1 to some lower number (e.g. 0.5) makes the RNN more confident, but also more conservative in its samples. Conversely, higher temperatures will give more diversity but at cost of more mistakes (e.g. spelling mistakes, etc). In particular, setting temperature very near zero will give the most likely thing that Paul Graham might say:

就是我们在计算 softmax 值以确定每个词被选到的概率的时候，对 softmax 的 α 进行修改，找到最好的结果。
代码如下：


```
def sample(h, seed_ix, n, alpha):
  """
  sample a sequence of integers from the model
  h is memory state, seed_ix is seed letter for first time step
  """

  # Start Your code
  x = np.zeros((vocab_size, 1))
  x[seed_ix] = 1
  ixes = []
  for t in range(n):
    # hidden state
    h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
    # output layer
    y = np.dot(Why, h) + by
    # output of the softmax
    p = np.exp(alpha * y) / np.sum(np.exp(alpha * y))
    # choose a word with prob p
    ix = np.random.choice(range(vocab_size), p=p.ravel())
    # BoW
    x = np.zeros((vocab_size, 1))
    # mark the chosen word 1
    x[ix] = 1
    # append the vector
    ixes.append(ix)
  return ixes
  # -------------------------
  # End your code
```

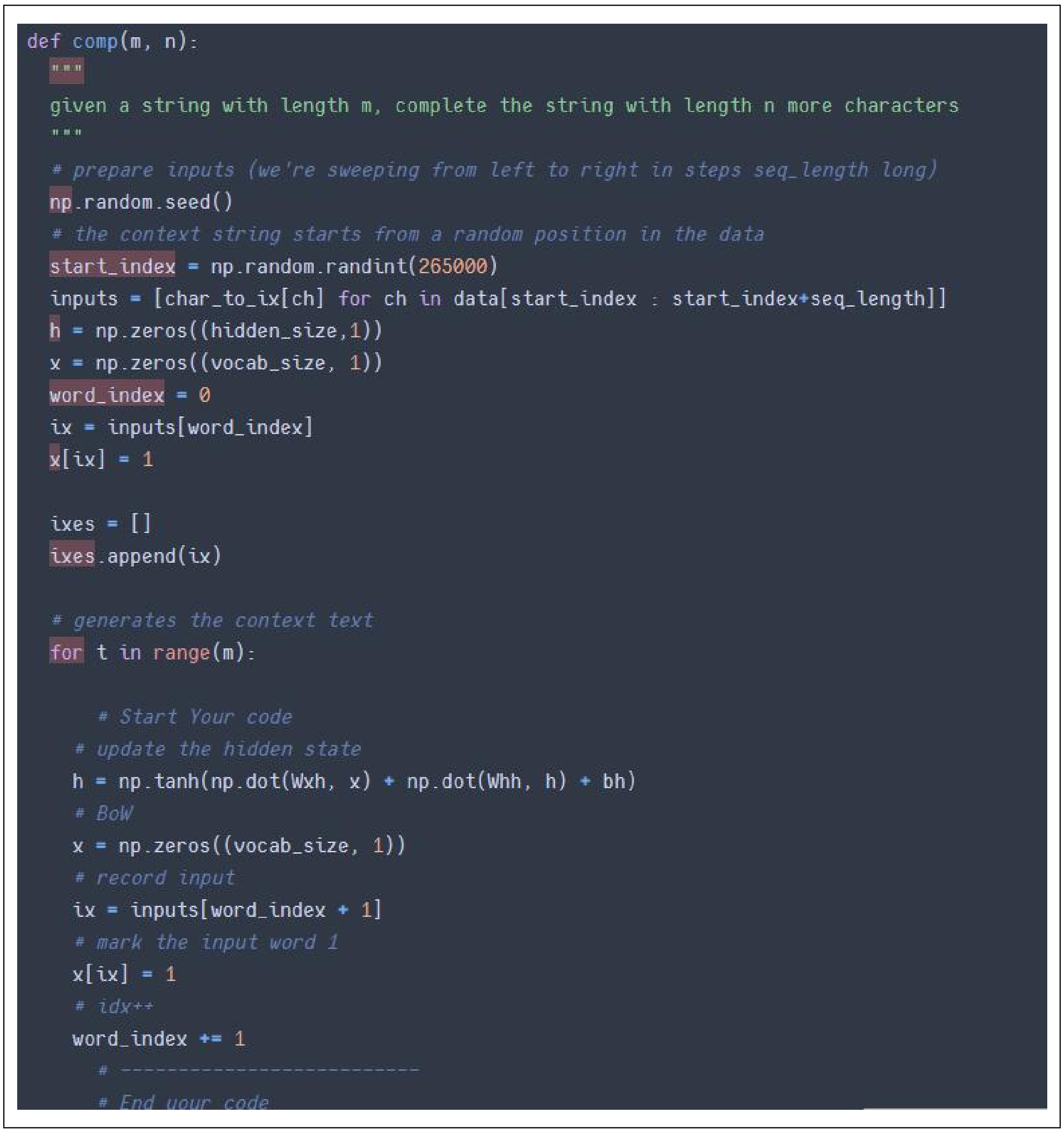
输出结果：

我们会发现，温度越高，概率分布会更广，每一个字母被选中的概率都可能以很大概率选中，更加多样化但是结果错误率升高，会输出一些无法理解的词语；温度越低，就更容易选择概率最高的字母，会使模型输出重复的字母，但是温度选择合适会使结果更精确。

```
Part 1:
------alpha=5------
----
irst Senater the the the the the she he the the con the the the the the stand he the the the the the the the the who he the the the the the the the the the so the consul the word the with and the the
----
------alpha=1------
----
irst Cithwe
And dost Tadainath gale omer the on an end sey tull you, on my penteilizes man,
To stoks, is, I coulshery,
Ie he, at of
Juth have amp haps! a walown atbe as say, Cates lut he ghe the von:

----
------alpha=0.1------
----
aznydZqSHCun,b'weN;oler.'YhuybuZumRlt?
 ge?d
d!aQeoC,O?Vsk:R!mr?-qdnbhapoMs
CjQysyP!cm bao'wa'oxHybbIMpwzwu,?.s,bmgg
JopnlR,BmeooksdeBysp!ks Ebii!
VCa-MsroHfk,
dC-f G'm;:;Wodh?',Gk&Uv'-n;iu:a?thohyma
----
```

PART 2

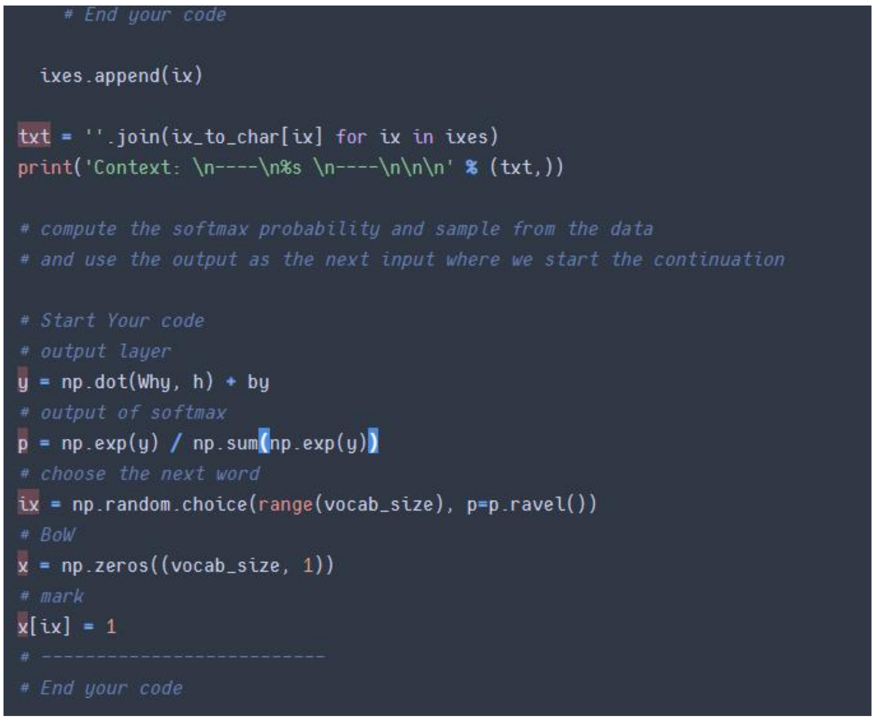


```
def comp(m, n):
  """
  given a string with length m, complete the string with length n more characters
  """
  # prepare inputs (we're sweeping from left to right in steps seq_length long)
  np.random.seed()
  # the context string starts from a random position in the data
  start_index = np.random.randint(265000)
  inputs = [char_to_ix[ch] for ch in data[start_index : start_index+seq_length]]
  h = np.zeros((hidden_size,1))
  x = np.zeros((vocab_size, 1))
  word_index = 0
  ix = inputs[word_index]
  x[ix] = 1

  ixes = []
  ixes.append(ix)

  # generates the context text
  for t in range(m):

      # Start Your code
    # update the hidden state
    h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
    # BoW
    x = np.zeros((vocab_size, 1))
    # record input
    ix = inputs[word_index + 1]
    # mark the input word 1
    x[ix] = 1
    # idx++
    word_index += 1
      # ----------------------------
      # End your code
```



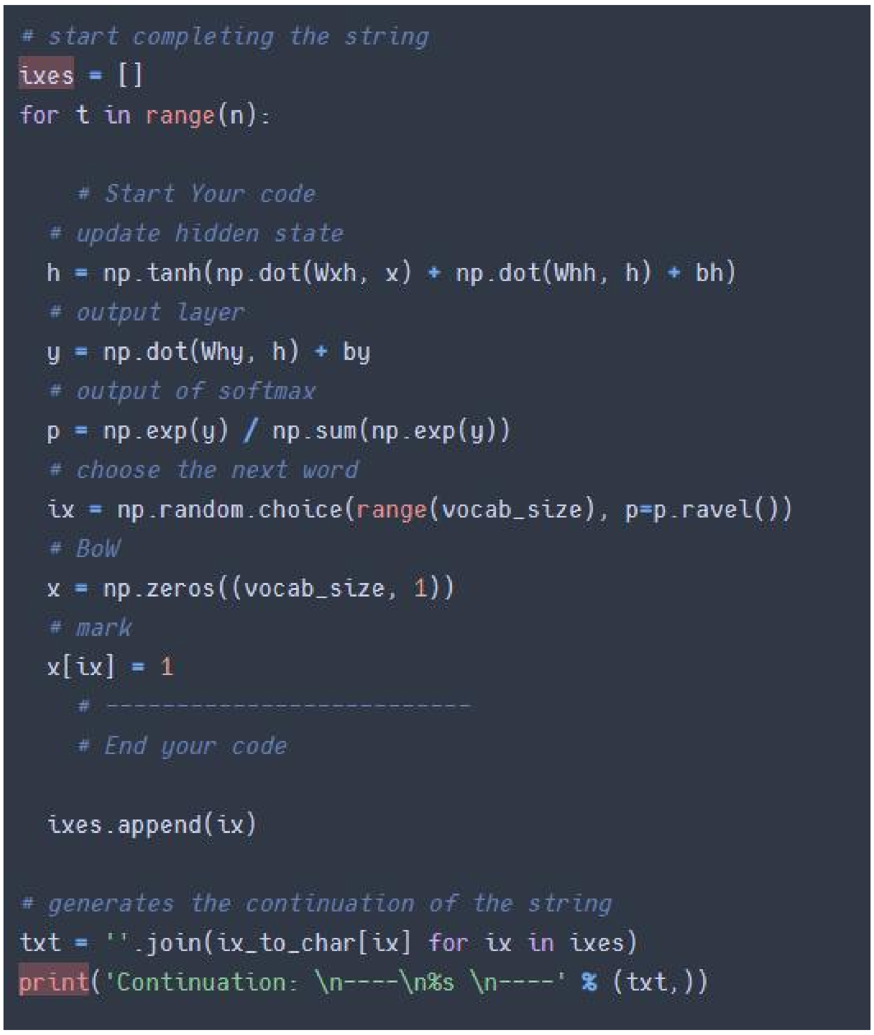
```
    # End your code

  ixes.append(ix)

txt = ''.join(ix_to_char[ix] for ix in ixes)
print('Context: \n----\n%s \n----\n\n\n' % (txt,))

# compute the softmax probability and sample from the data
# and use the output as the next input where we start the continuation

# Start Your code
# output layer
y = np.dot(Why, h) + by
# output of softmax
p = np.exp(y) / np.sum(np.exp(y))
# choose the next word
ix = np.random.choice(range(vocab_size), p=p.ravel())
# BoW
x = np.zeros((vocab_size, 1))
# mark
x[ix] = 1
# -------------------------
# End your code
```



```
# start completing the string
ixes = []
for t in range(n):

    # Start Your code
  # update hidden state
  h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
  # output layer
  y = np.dot(Why, h) + by
  # output of softmax
  p = np.exp(y) / np.sum(np.exp(y))
  # choose the next word
  ix = np.random.choice(range(vocab_size), p=p.ravel())
  # BoW
  x = np.zeros((vocab_size, 1))
  # mark
  x[ix] = 1
    # --------------------------
    # End your code

  ixes.append(ix)

# generates the continuation of the string
txt = ''.join(ix_to_char[ix] for ix in ixes)
print('Continuation: \n----\n%s \n----' % (txt,))
```

输出结果:

```
Part 2:
Context:
----
QUEEN ELIZABETH:
Oh, he is young and his minority
Is put unto the trust of Richard Gloucester,
A man that loves not me, nor none of you.

RIVERS:
Is it concluded that he shall be protector?

QUEEN ELIZABETH:
It is determined, not concluded yet:
But so it must be, if the king miscarry.

GREY:
Here come the lords of Buckingham and Derby.

BUCKINGHAM:
Good time of day unto your royal grace!

DERBY:
God make your majesty joyful as you have been!

QUEEN ELIZABETH:
The Countess Richmond, good my Lord of Derby.
To your good prayers will scarcely say amen.
Yet, Derby, notwithstanding she's your wife,
And loves not me, be you, good lord, assured
I hate not you for her proud arrogance.

DERBY:
I do beseech you, either not believe
The envious slanders of her false accusers;
Or, if
----



Continuation:
----
zzkkzzzzzzzzzzzzzzxzzzzzxzxzxzzzzzzzzzzzzxzkkzzzzzzzzzzzzzzzzzz
```

```
zzzzzzzzzzzzzzzxkxzzkzzzzzzzkkzzzzzzzzzzzzzzzzzxkzzzkkzzzzzzzz
zzzzzzzzzzzzzzzzzzzzxzzzzzzzzzzzkkxzkzzzzzzzzzzzzzxzxkzzzzzzz
zzzzzzzzzkpzkz
----
Context:
----
d in's heart. Go you to the city;
Learn how 'tis he
----




Continuation:
----
ms Sen't she Voft seadiss he a pith wide, what an love ners! I she
heare.

CORIOLANUS:
He of be hush gods, is.

COMINIUS:
Your his made
seaeng
Wong u'd suly, Hanich a befoer: thy one and weacfeus scome, in youm.
yet siforige.

VER:
Son.
 I wopie
Hear greads, be thures it now a interild thy mofor,, the us thes
name than ous:
Senfat;
What Mard, ret tinan him.

BRUTUS:
If thy repen what be no, the up, is to thee.

BRUTUS:
You heay, tone the say. Whnauld you have my nathest who me ot lode
corasse pr
----
Context:
----
t,
```

Continuation:
----
 rught swo?

SICINIUS:
And not his my lut enes Fives, tus the some?
'T
Pes. he shall prall to they hurly
whas unwas my fon that rant!

SICSIUCESTNIA:
Whyat 'De wouch hate sand be much the hunce-ther of are cound an he father bllought eike and anonamy down Citizen:
Mises say alcer the reg'dnep of fithed,
Snave musbit citilouten shelonant you the scould what oe is did.

First ind the siss.

VOLUMNIA:
To con you no,
Shongay brich for bives mase wort, let Warvator of as then and I miid has! my worte
----
Context:
----
ord shall serve,
As well as I had seen and heard him speak
And doubt you not, right noble princes both,
But I'll acquaint our duteous citizens
With all your just proceedings in this cause.

GLOUCESTER:
And to that end we wish'd your lord-ship here,
To avoid the carping censures of the world.

BUCKING

```
Continuation:
----
HI:
Wonber:, the was pry de'd weang!

AECHARD RICHARENIIIUS:
What
To connor:
Than he mo I doth hath of you wonner and in denciosink her his laly
hu
gath my who no, by we sord the un hon:
What if woraghfer'd an whil! and the to men not most; me, love'tw
rave in me, you that your a forte dine lapping
----
Context:
----
rthy sir,
The slave's report is seconded; and more,
More fearful, is deliver'd.

SICINIUS:
What more
----



Continuation:
----
 they vame? ind hads.
Coman; stry bace.

PRINCE:

BRUTUS:
A, you such poor
raze,
Be's for mised he wiooturusan trouk'd. He peoble to and that haw!

CORIOLANUS:
Whsot and by that no these. is the coun conder who as be. You farseus
of, let freres deand mos'd and is were we maun:
Peke had is, were at he
swathy, doth on four forths the coed frout. But be pest, my me;
```

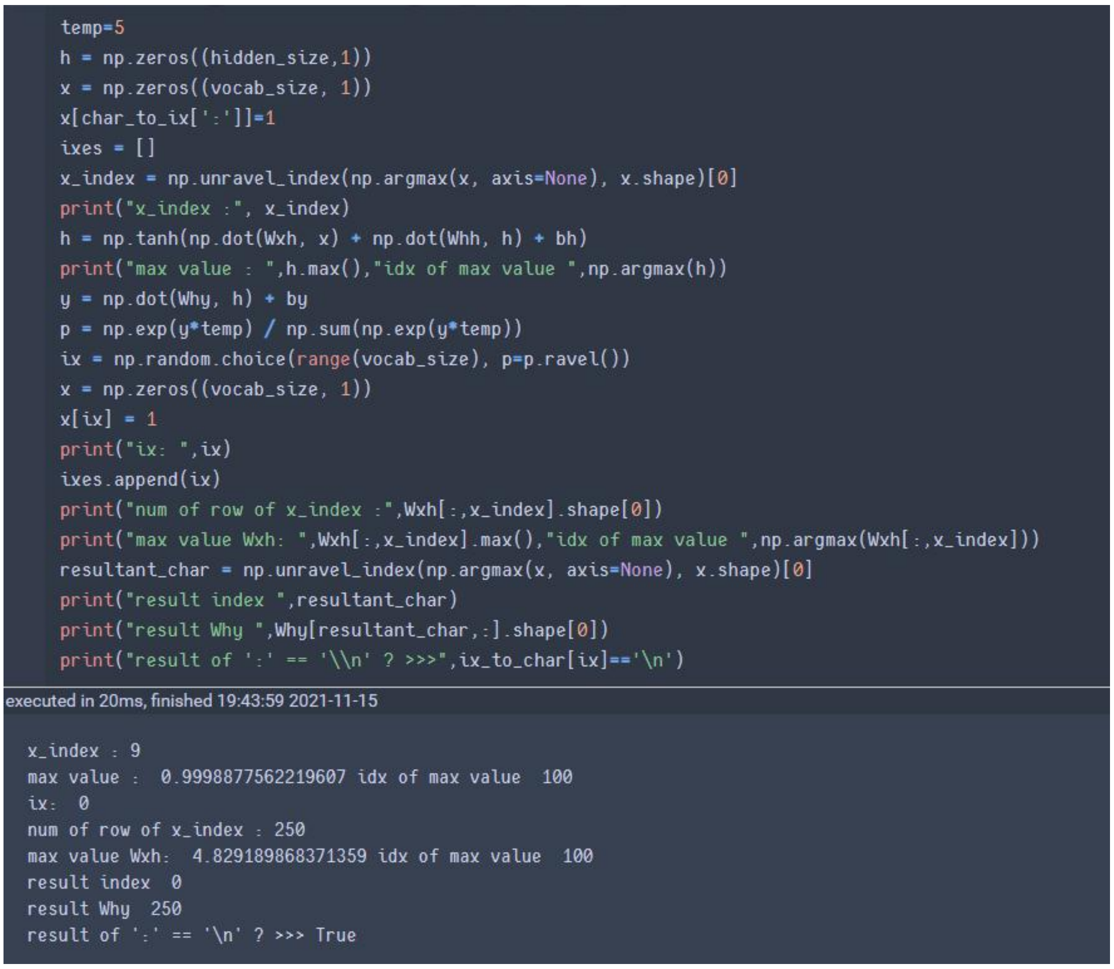
```
Hold must be's;
Ovee plancies on we bove I
Eefe,
In with momanver wortch.
here a knowhest is hos
This he?
Stir his then?

SINIUS:
And
----
```

PART 3

我们发现结果中在冒号后面通常跟着换行符或者空格，我们要找到造成这种结果的原因：

我们通过找到影响最大的因素发现，**Wxh[100][9]，Why[0][100]，Why[2][100]**是主要原因，因为 Wxh[100][9]很大造成 h[100]非常大，达到了 0.999，这样的话会使得这个占比非常大，概率也会大大增加，使得其他的 h 值以及 Whh 没有用了，然后再输出的时候，h 乘以 Why 得到 y，Why[0][100] 和 Why[2][100]很大，使得 y[0](换行符)和 y[2](空格)的值很大，所以激活后也很大，选中的概率就会很大。于是结果中在冒号后面通常跟着换行符或者空格。



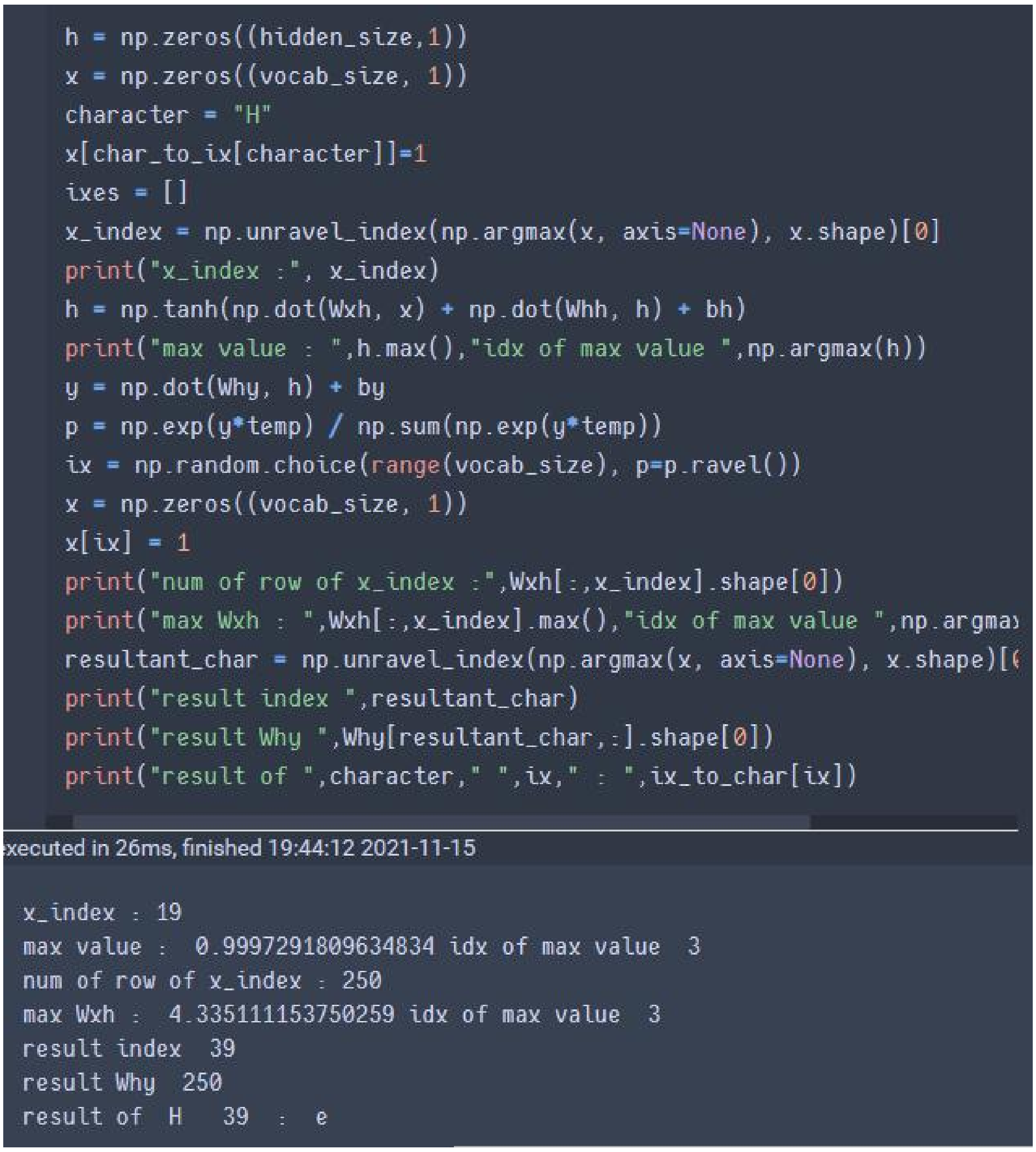
```python
temp=5
h = np.zeros((hidden_size,1))
x = np.zeros((vocab_size, 1))
x[char_to_ix[':']]=1
ixes = []
x_index = np.unravel_index(np.argmax(x, axis=None), x.shape)[0]
print("x_index :", x_index)
h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
print("max value : ",h.max(),"idx of max value ",np.argmax(h))
y = np.dot(Why, h) + by
p = np.exp(y*temp) / np.sum(np.exp(y*temp))
ix = np.random.choice(range(vocab_size), p=p.ravel())
x = np.zeros((vocab_size, 1))
x[ix] = 1
print("ix: ",ix)
ixes.append(ix)
print("num of row of x_index :",Wxh[:,x_index].shape[0])
print("max value Wxh: ",Wxh[:,x_index].max(),"idx of max value ",np.argmax(Wxh[:,x_index]))
resultant_char = np.unravel_index(np.argmax(x, axis=None), x.shape)[0]
print("result index ",resultant_char)
print("result Why ",Why[resultant_char,:].shape[0])
print("result of ':' == '\\n' ? >>>",ix_to_char[ix]=='\n')
```

executed in 20ms, finished 19:43:59 2021-11-15

```
x_index : 9
max value :  0.9998877562219607 idx of max value  100
ix:  0
num of row of x_index : 250
max value Wxh:  4.829189868371359 idx of max value  100
result index  0
result Why  250
result of ':' == '\n' ? >>> True
```

PART 4

经测试，H 后面经常跟 e

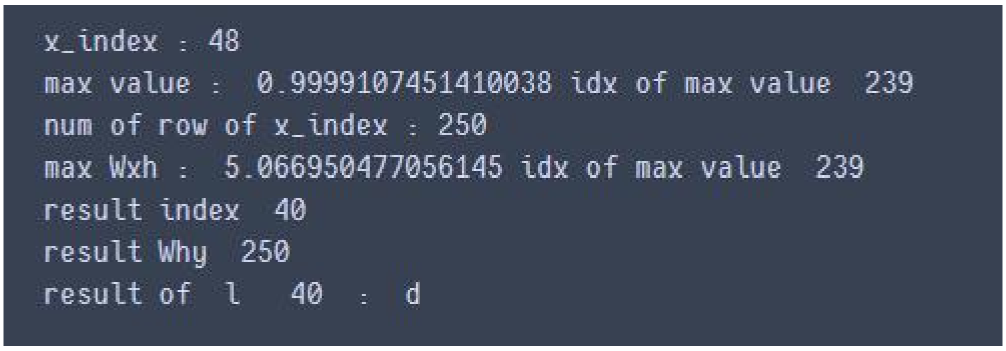


```
h = np.zeros((hidden_size,1))
x = np.zeros((vocab_size, 1))
character = "H"
x[char_to_ix[character]]=1
ixes = []
x_index = np.unravel_index(np.argmax(x, axis=None), x.shape)[0]
print("x_index :", x_index)
h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
print("max value : ",h.max(),"idx of max value ",np.argmax(h))
y = np.dot(Why, h) + by
p = np.exp(y*temp) / np.sum(np.exp(y*temp))
ix = np.random.choice(range(vocab_size), p=p.ravel())
x = np.zeros((vocab_size, 1))
x[ix] = 1
print("num of row of x_index :",Wxh[:,x_index].shape[0])
print("max Wxh : ",Wxh[:,x_index].max(),"idx of max value ",np.argma
resultant_char = np.unravel_index(np.argmax(x, axis=None), x.shape)[
print("result index ",resultant_char)
print("result Why ",Why[resultant_char,:].shape[0])
print("result of ",character," ",ix," : ",ix_to_char[ix])
```

xecuted in 26ms, finished 19:44:12 2021-11-15

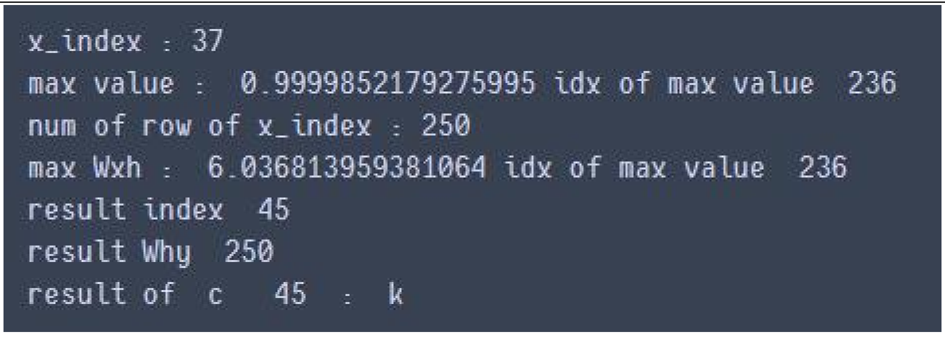
```
x_index : 19
max value :  0.9997291809634834 idx of max value  3
num of row of x_index : 250
max Wxh :  4.335111153750259 idx of max value  3
result index  39
result Why  250
result of  H   39  :  e
```

l 后面经常跟 d



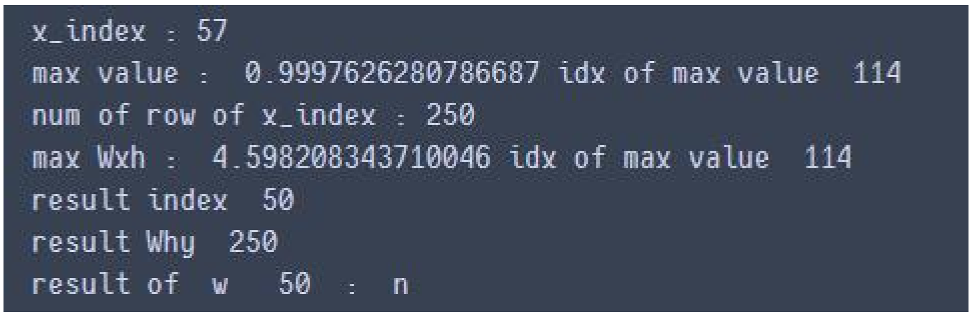
```
x_index : 48
max value :  0.9999107451410038 idx of max value  239
num of row of x_index : 250
max Wxh :  5.066950477056145 idx of max value  239
result index  40
result Why  250
result of  l   40  :  d
```

c 后面经常跟 k



```
x_index : 37
max value :  0.9999852179275995 idx of max value  236
num of row of x_index : 250
max Wxh :  6.036813959381064 idx of max value  236
result index  45
result Why  250
result of  c   45  :  k
```

w 后面经常跟 n

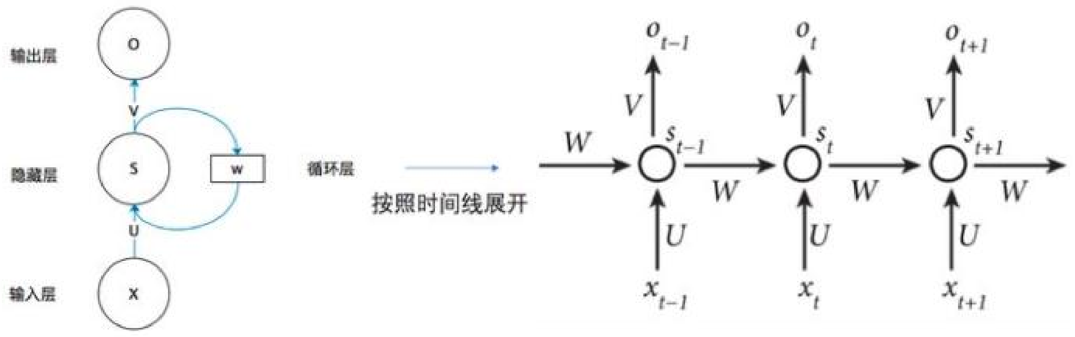


```
x_index : 57
max value :  0.9997626280786687 idx of max value  114
num of row of x_index : 250
max Wxh :  4.598208343710046 idx of max value  114
result index  50
result Why  250
result of  w   50  :  n
```

结论分析与体会：

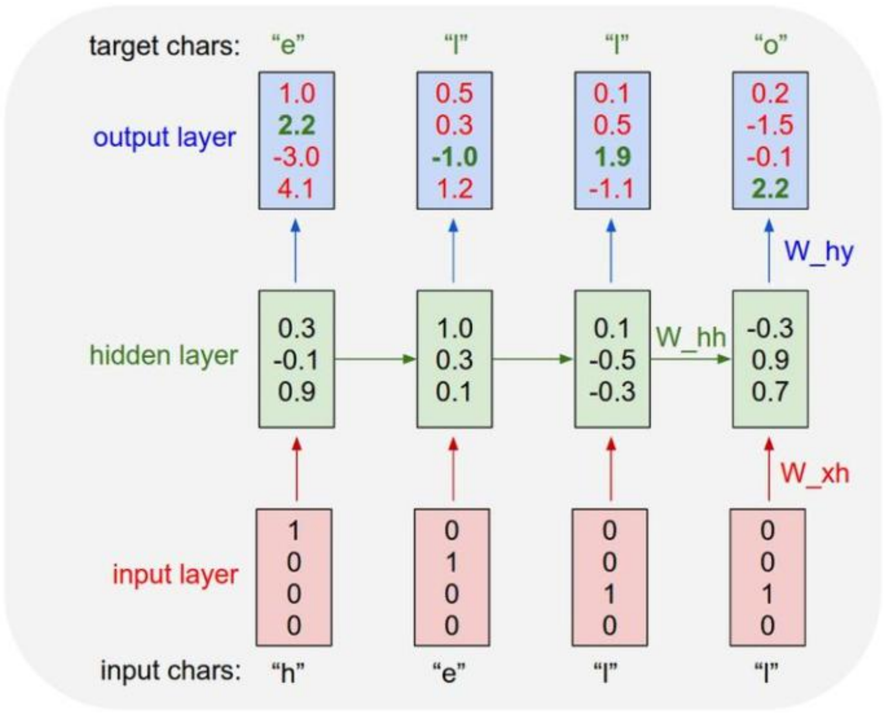
1，我们会发现，温度越高，概率分布会更广，每一个字母被选中的概率都可能以很大概率选中，更加多样化但是结果错误率升高，会输出一些无法理解的词语；温度越低，就更容易选择概率最高的字母，会使模型输出重复的字母，但是温度选择合适会使结果更精确。

2，要搞清楚 RNN 的正向传播和反向传播过程。



就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：
1，一开始不太清楚实验流程，查阅资料加深理解。

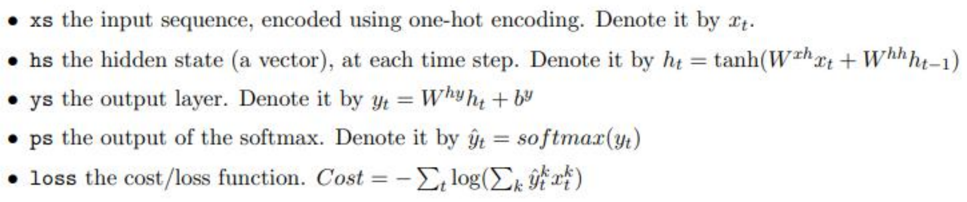
一个 RNN 的例子：输入输出是 4 维的层，隐层神经元数量是 3 个。该流程图展示了使用 hell 作为输入时，RNN 中激活数据前向传播的过程。输出层包含的是 RNN 关于下一个字母选择的置信度（字母表是 helo）。我们希望绿色数字大，红色数字小。



举例如下：在第一步，RNN 看到了字母 h 后，给出下一个字母的置信度分别是 h 为 1，e 为 2.2，l 为-3.0，o 为 4.1。因为在训练数据（字符串 hello）中下一个正确的字母是 e，所以我们希望提高它的置信度（绿色）并降低其他字母的置信度（红色）。类似的，在每一步都有一个目标字母，我们希望算法分配给该字母的置信度应该更大。因为 RNN 包含的整个操作都是可微分的，所以我们可以通过对算法进行反向传播（微积分中链式法则的递归使用）来求得权重调整的正确方向，在正确方向上可以提升正确目标字母的得分（绿色粗体数字）。然后进行参数更新，即在该方向上轻微移动权重。如果我们将同样的数据输入给 RNN，在参数更新后将会发现正确字母的得分（比如第一步中的 e）将会变高（例如从 2.2 变成 2.3），不正确字母的得分将会降低。重复进行一个过程很

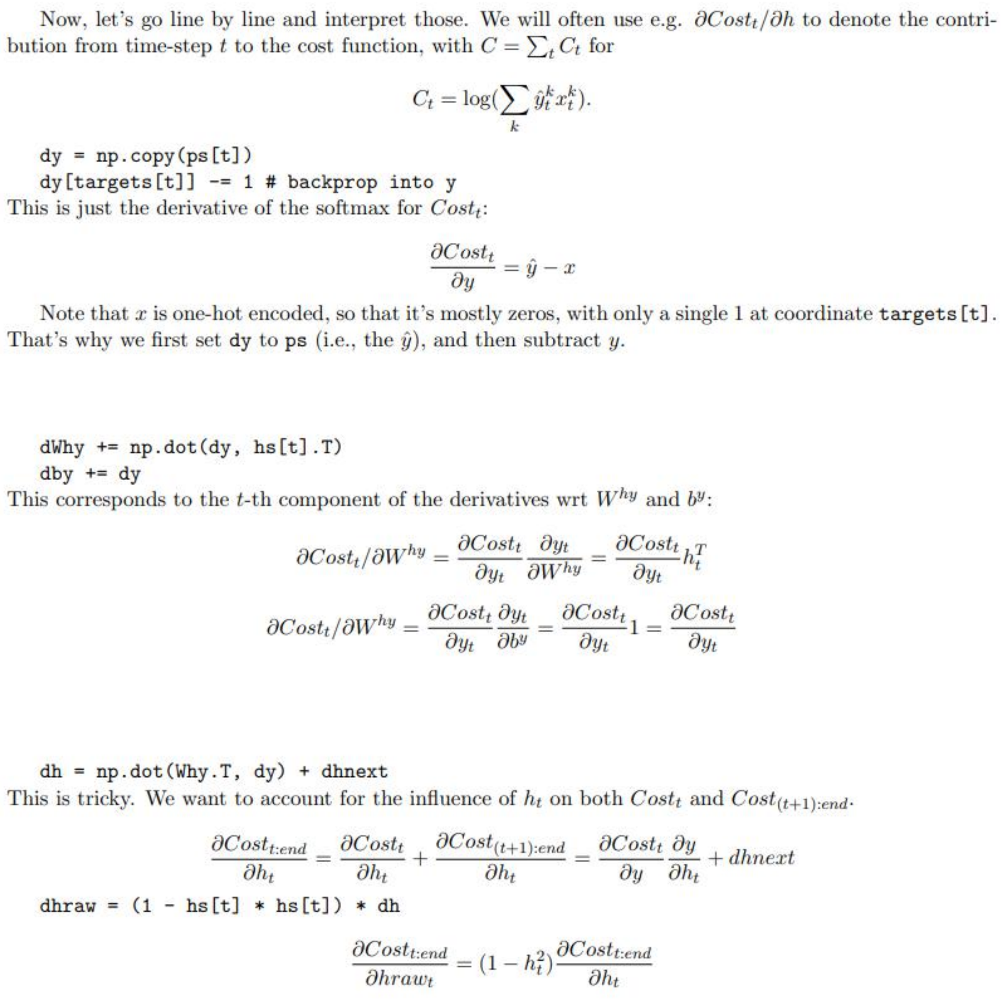
多次直到网络收敛，其预测与训练数据连贯一致，总是能正确预测下一个字母。注意当字母 l 第一次输入时，目标字母是 l，但第二次的目标是 o。因此 RNN 不能只靠输入数据，必须使用它的循环连接来保持对上下文的跟踪，以此来完成任务。

2，一开始不知道每个变量的意思，后来看了 Andrej 的博客和代码就理解了。



- `xs` the input sequence, encoded using one-hot encoding. Denote it by x_t.
- `hs` the hidden state (a vector), at each time step. Denote it by $h_t = \tanh(W^{xh}x_t + W^{hh}h_{t-1})$
- `ys` the output layer. Denote it by $y_t = W^{hy}h_t + b^y$
- `ps` the output of the softmax. Denote it by $\hat{y}_t = softmax(y_t)$
- `loss` the cost/loss function. $Cost = -\sum_t \log(\sum_k \hat{y}_t^k x_t^k)$

2，不仅要明白正向的流程，还要知道反向怎么传播



Now, let's go line by line and interpret those. We will often use e.g. $\partial Cost_t/\partial h$ to denote the contribution from time-step t to the cost function, with $C = \sum_t C_t$ for

$$C_t = \log(\sum_k \hat{y}_t^k x_t^k).$$

```
dy = np.copy(ps[t])
dy[targets[t]] -= 1 # backprop into y
```

This is just the derivative of the softmax for $Cost_t$:

$$\frac{\partial Cost_t}{\partial y} = \hat{y} - x$$

Note that x is one-hot encoded, so that it's mostly zeros, with only a single 1 at coordinate `targets[t]`. That's why we first set `dy` to `ps` (i.e., the $\hat{y}$), and then subtract y.

```
dWhy += np.dot(dy, hs[t].T)
dby += dy
```

This corresponds to the t-th component of the derivatives wrt W^{hy} and b^y:

$$\partial Cost_t/\partial W^{hy} = \frac{\partial Cost_t}{\partial y_t}\frac{\partial y_t}{\partial W^{hy}} = \frac{\partial Cost_t}{\partial y_t}h_t^T$$

$$\partial Cost_t/\partial W^{hy} = \frac{\partial Cost_t}{\partial y_t}\frac{\partial y_t}{\partial b^y} = \frac{\partial Cost_t}{\partial y_t}1 = \frac{\partial Cost_t}{\partial y_t}$$

```
dh = np.dot(Why.T, dy) + dhnext
```

This is tricky. We want to account for the influence of h_t on both $Cost_t$ and $Cost_{(t+1):end}$.

$$\frac{\partial Cost_{t:end}}{\partial h_t} = \frac{\partial Cost_t}{\partial h_t} + \frac{\partial Cost_{(t+1):end}}{\partial h_t} = \frac{\partial Cost_t}{\partial y}\frac{\partial y}{\partial h_t} + dhnext$$

```
dhraw = (1 - hs[t] * hs[t]) * dh
```

$$\frac{\partial Cost_{t:end}}{\partial hraw_t} = (1 - h_t^2)\frac{\partial Cost_{t:end}}{\partial h_t}$$

The following:

```
dbh += dhraw
dWxh += np.dot(dhraw, xs[t].T)
dWhh += np.dot(dhraw, hs[t-1].T)
```

are similar to what we already had. Note that

$$\frac{\partial Cost_{t:end}}{\partial h_t} = \frac{\partial Cost}{\partial h_t}$$

since h_t cannot influence components of the cost that come before it in time.

Finally, we compute `dhnext`, which must be $\frac{\partial Cost_{t:end}}{\partial h_{t-1}}$ in order for our earlier definition to work. Now

$$\frac{\partial Cost_{t:end}}{\partial h_{t-1}} = \frac{\partial Cost_{t:end}}{\partial hraw_t}\frac{\partial hraw_t}{\partial h_{t-1}}$$

This is exactly what the following line does.

```
dhnext = np.dot(Whh.T, dhraw)
```

The following is self-explanatory:

```
for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
  np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
```

In the loop, we are adding up all the contributions to the gradients from all the time-steps t.