RNNs (part 2)

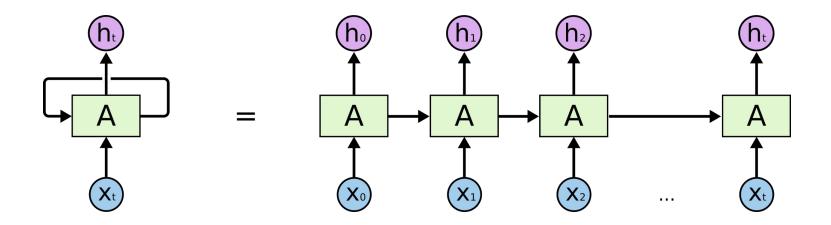
RNN

- Along with CNNs, RNNs are the most important class of models in DL
- Applicable in speech recognition(https://research.googleblog.com/2015/09/google-voice-search-faster-and-more.html) machine translation(https://research.googleblog.com/2016/09/a-neural-network-for-machine.html) and many other task especially in NLP

Overview

- Basic RNN recap
- Language modelling example character level RNN
- LSTM recap
- RNN extensions bidirectional RNN, multilayer RNNs, RvNN
- Using RNNs with Tensorflow
- Case studies
- Quora Kaggle competition
- Lab instructions

Basic RNN



$$RNN: h_{t-1}, x_t \to h_t$$

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$

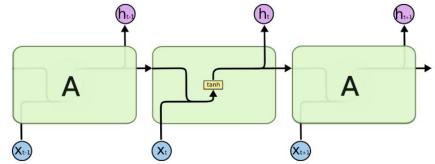
 W_h, W_x are the same across all timesteps

Basic RNN implementation

for t in xrange(steps_n):
 x t = x[:, t, :]

input = tf.concat([x t, h[t - 1]], axis=1)

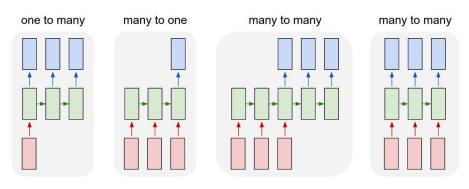
h[t] = tf.tanh(tf.matmul(input, self.W) + self.bias)



The repeating module in a standard RNN contains a single layer.

What to do with h_ts

- We have h t for each t how do we use it?
- In "many to one" case we could just take h_last, and process it further to arrive at our final prediction
- In "many to many" case, we would have some other net take h_t and produce y_t, we would apply same net at
 one to one
 each timestep
- other cases possible



Basic RNN

There are few important technical issues with this code

- it requires steps_n to be constant, at the time of graph creation
- sometimes we would like different number of steps for each example in a minibatch
- what to do with very long inputs?

Language modelling

Train RNN to model

$$p(w_{n+1}|[w_1, w_2, w_3, \dots, w_n])$$
$$p("you"|["I", "like"]) > p("like"|["I", "like"])$$

- A lot of training data
- We can use such model to choose more probable sentences, ex. in speech recognition system, translation systems
- Or we could let it run and generate some text

Language modelling

 Give our RNN sequence of words and ask for the probability distribution of the next word

$$p("you"|["I","like"]) > p("like"|["I","like"])$$

 Give our RNN sequence of characters and ask for the probability distribution of the next character

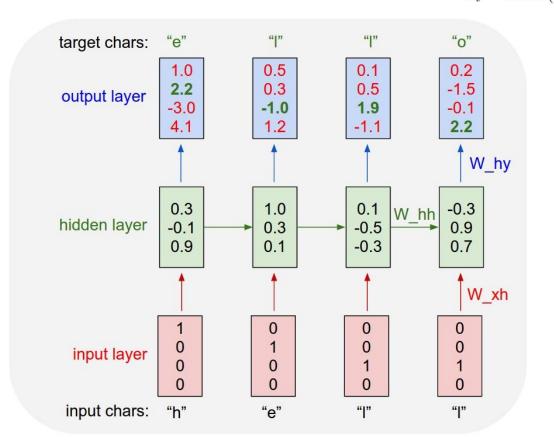
$$p("o"|["h", "e", "l", "l"]) > p("w"|["h", "e", "l", "l"])$$

Language modelling char rnn

$$RNN: h_{t-1}, x_t \to h_t$$

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$





hidden state size = 3

min-char-rnn.py gist: 112 lines of Python

```
Minimal character-level Vanilla RNN model, Written by Andrej Karpathy (@karpathy)
    BSD License
 5 import numpy as np
 7 # data I/O
 8 data = open('input.txt', 'r').read() # should be simple plain text file
chars = list(set(data))
data_size, vocab_size = len(data), len(chars)
print 'data has %d characters, %d unique.' % (data_size, vocab_size)
char_to_ix = { ch:i for i,ch in enumerate(chars) }
ix_to_char = { i:ch for i,ch in enumerate(chars) }
in hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RMN for
10 learning_rate = 1e-1
21 Wxh = np.random.randn(hidden_size, vocab_size)*8.81 # input to hidden
22 Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
23 Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
ph = np.zeros((hidden_size, 1)) # hidden bias
us by = np.zeros((vocab size, 1)) # output blas
27 def lossFun(inputs, targets, hprev):
      inputs, targets are both list of integers.
      hprev is Hx1 array of initial hidden state
      returns the loss, gradients on model parameters, and last hidden state
      xs, hs, ys, ps = {}, {}, {}, {}
      hs[-1] = np.copy(hprev)
      loss = 0
      # forward pass
      for t in xrange(len(inputs)):
        xs[t] = np.zeros((vocab_size,1)) = encode in 1-of-k representation
        xs[t][inputs[t]] = 1
        hs[t] = np.tanh(np.dot(wxh, xs[t]) + np.dot(whh, hs[t-1]) + bh) = hidden state
        ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
        ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) = probabilities for next chars
43 loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
44 # backward pass: compute gradients going backwards
duch, dwhh, dwhy = np.zeros_like(wxh), np.zeros_like(whh), np.zeros_like(why)
      dbh, dbv = np.zeros like(bh), np.zeros like(bv)
      dhnext = np.zeros_like(hs[0])
      for t in reversed(xrange(len(inputs))):
        dy = np.copy(ps[t])
        dy[targets[t]] -= 1 # backprop into y
51 dwhy += np.dot(dy, hs[t].T)
dh = np.dot(Why.T, dy) + dhnext # backprop into h
34 dhraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
        dwxh += np.dot(dhraw, xs[t].T)
        dWhh += np.dot(dhraw, hs[t-1].T)
        dhnext = np.dot(Whh.T, dhraw)
      for dparam in [dwxh, dwhh, dwhy, dbh, dby]:
       np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
      return loss, dwxh, dwhh, dwhy, dbh, dby, hs[len(inputs)-1]
```

```
43 def sample(h, seed_ix, n):
  65 sample a sequence of integers from the model
        h is memory state, seed_ix is seed letter for first time step
       x = np.zeros((vocab_size, 1))
        x[seed_ix] = 1
        ixes = []
        for t in xrange(n):
        h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
 74 p = np.exp(y) / np.sum(np.exp(y))
 ix = np.random.choice(range(vocab_size), p=p.ravel())
         x = np.zeros((vocab_size, 1))
          x[ix] = 1
          ixes.append(ix)
        return ixes
  82 mWxh, mWhh, mWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
 mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
 84 smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
 an # prepare inputs (we're sweeping from left to right in steps seg length long)
 if p+seq length+1 >= len(data) or n == 0:
 88 hprey = np.zeros((hidden_size,1)) # reset RNN memory
gg p = 0 # go from start of data
 inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
        targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
       # sample from the model now and then
         sample_ix = sample(hprev, inputs[0], 200)
         txt = ''.join(ix_to_char[ix] for ix in sample_ix)
         print '----\n %s \n---- % (txt, )
so # forward seg length characters through the net and fetch gradient
loss, dwxh, dwhh, dwhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
ini smooth_loss = smooth_loss * 0.999 + loss * 0.001
if n % 100 == 0; print 'iter %d, loss: %f' % (n, smooth loss) = print progress
       a perform parameter update with Adaprad
        for param, dparam, mem in zip([wxh, whh, why, bh, by],
                                   [dwxh, dwhh, dwhy, dbh, dby],
                                   [mixh, mixhh, mixhy, mbh, mby]):
          men += dparam * dparam
          param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
 111 p += seq_length # move data pointer
 112 n += 1 # iteration counter
```

(https://gist.github. com/karpathy/d4dee566867f8291f086)

```
minimal character level vanilla man model, written by madre; xaruathy (@karpathy)
855 Licemie
Separt humpy as no
data = open("imput.tet", "r").recd() = simulai be simple plate test file
data_size, vecus_size + len(data), len(chars)
grint 'data has ad characters, ad unique.' a (data_size, vocab_size)
char_to_ix = C chis for i.ch in organizate/charal l
is to char = { i:ch for i,ch in enswerate(chara) }
#38dem_size = 380 # Size of Midden idear of Heuross
seq length = 25 + number of stans to serall the PANK for
learning_rate = 1e-1
WR - ID random rando(hidden size, vocab size)*0.81 + input to hidden
With a rep render renderlybidden attre. hidden attre150 of a hidden on hidden
Why a mp. random. rando(vocab_size, hidden_size)*8.81 + hidden in narput
$6 = np.zeros((Aldden_size, 3)) = hinter that
by = np.reros((woodb_size, 33) = minut bias
def loostin(inputs, targets, Agrav):
  imputs, targets are both list of integers.
   harey is Hel array of initial hidden state
  returns the loss, gradients on model parameters, and last hidden state
   as, he, we as = \Omega_1 \cdot \Omega_2 \cdot \Omega_3 \cdot \Omega_4
  hs[-1] = sp.cogy(horev)
  Seen of H
   for I in arange(lendinguts))
    safet = eg.perce((vocab_abre, b)) = counds to 1.07-% representation
    \label{eq:target} \exists x[t] = \eta y. \tanh[\eta p. \det(wsh, \ xx[t]) + \eta y. \det(whh, \ \exists x[t-1]) + bh) + \forall i. dien \ itself = 0.
     walth I me dativity, bafely + by a assermation in a communities for each chara-
     usit; = no.enu(ys(t)) / no.sum(np.enu(ys(t))) = promonitizing for most chura
      Loss -- -sp.log(ps[t](tergets[t],0]) + orfines (trass-entropy loss)
   duch, dath, duty = np.zeros_like(abh), np.zeros_like(ath), np.zeros_like(ath)
   din, dby = np.zeros_like(bb), np.zeros_like(by)
   dispert a op. seros like/ballity
   for t in reversed(scenes(lex(topotal)))
     dyltargets[t]] -= 1 + backgrop tree a
    deby += to_det(dy, fol(t).T)
    di = mp. det Dahy. T. dy) + dhyear = hackaras leno h
    dhram = (1 - hs[t] * hs[t]) * dh = naciprou inrough two conditionally
    dech == rp.dot[dress, volt3.7]
     debh -- np.dot(dhraw, hs[t-1].T)
     dissect = sp.dot(whh.t, diraw)
  for operan in [mon, men, deny, obn, sty]:
    en clip/degree. -5. 5. subscherved a clip to middely evaluates areatterin.
   return loss, dech, debt, deby, dob, dby, hs[lan(inputs)-1]
ser soughts, seed by its
  sample a sequence of integers from the model.
 It is memory state, seed in is seed letter for first time stay
  s[seed.ix] = t
   h \in \operatorname{rig}_{-1}\operatorname{Sarb}(\operatorname{rig}_{-1}\operatorname{Bit}(\operatorname{bot}_{1}, \ v) \to \operatorname{rig}_{-2}\operatorname{Bit}(\operatorname{bot}_{1}, \ h) \to \operatorname{Bit}_{1}
   g = mp_dot(why, h) = by
g = np_dot(why) / np_num(np_dop(y))
    in - ep. random. chatcol rocard/vocah.nize), pro. ravel(1)
    isse.append(ix)
  return here
manh, mark, maky - my perce like(blob), my perce like(blob), my perce like(blob)
Man, May " No. Deron, Like(DE), op. Toron, Like(Dy) = mentry variables for a
secon, loss = op. Lagi. Stocom, size() beg. length = loss of interesting &
   *prev = sp.zeros((%180es_clas,3)) = /elet mon leaving
p.= 8 = an from start of bits.
  treams - loter to inich! for ch in details a-see Tenerhill
    hample is I cample(turne, imputs(s), 200)
   tot = "'.jein[is.to.cher[ix] for ix in sample.is]
grant '-----'a (txt. )
  Jose, don. 869, day, 66, 69, horey i lossPerclands, terrers, forest
  amouth_lous = amouth_lous = 0.000 + lous = 0.000
af n w com com or grant 'lise wa, lose: of w ps, amouth_lous = often property
  for param, operam, see in mischael, who, why, bit, by
                                [deck, detc. day, ste, sty],
[max, max, may, ste, sty]);
```

param +C _learning_rate * dparam / sp.sgrt(mem + 1e-t) = schijted police

Data I/O

```
Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
BSD License
minimal import numpy as np

# data I/O
data = open('input.txt', 'r').read() # should be simple plain text file
chars = list(set(data))
data_size, vocab_size = len(data), len(chars)
print 'data has %d characters, %d unique.' % (data_size, vocab_size)
char_to_ix = { ch:i for i,ch in enumerate(chars) }
ix_to_char = { i:ch for i,ch in enumerate(chars) }
```

min-char-rnn.py gist Minimal character-level vanilla mon model, written by Andrei Karpathy (Bhorpathy) SECURIT PLEEDY AS THE data - opend 'imput.txt', 'r'), read() = should be size in claim test file phers - list(set(date() data_size, vocab_size = les(data), les(chars) print 'data has no characters, no unique." % (data_size, vocab_size) shar to be a Compt for Lab to engagerateleharst.) is to other = (1:ch for 1,ch in seamerate(chars)) \$150m alte I 100 a vice of bidden lawer of concurs seq.length = 25 = number of staps to smeall the Abb for learning_rate = le-1 web = np.random.randn(hiddem.mico, vocab_aico)*0.01 + Ensut in hidden with - np.random.randojhiddem_size, hiddem_size}'0.01 = ninten to hidden why a no random random scott tipe. Nicotes size Parata a Australia to cornel by - np.zeres((weeds,alfe, 1)) = compan him def Louseun(Inputs, tarpets, horev); imputs, targets are both list of integers herey to But array of intrial higher stars returns the loss, gradients on model parameters, and Last hidden state #8, #8, y8, #8 = (), (), (), () Ma[-1] = np.cogy(hprev)loss = 0 for t in wrange(lentingers)); #SITI = mp. cerosi(vocab_size_11) = enone in 1-15-4 representation fis[t] = rg.tanh(rp.det(wit, xe[t]) + rg.det(wit, fis[t-1]) + bh) + hinter starswaith a me.datings, builth + by a uncornalized les groupediffices for ones there us[t] = nu.exu(ys[t]) / nu.sum(np.exu(ys[t])) / probabilities for most others loss ++ -mp.log(px[t][targets[t],0]) + softmax (cross-mirror loss) doch, date, daty = np.zeros_like(ach), np.zeros_like(ath), np.zeros_like(athy) dbh, dby = sp.zeros_like(bh), sp.zeros_like(by) sheert - ep.zeros_like(hs[80]) For I in reversed(xrange(len(inputs))): dy = np.copy(gs[t]) dy[targets[t]] = 1 + backgrop loss s deby += rp.dot(dy, Ps(t).T) $ds = np.det(Why, Y_p, dy) + dhesext + tackgroup into be$ dherms of (1 + hx[t] + hx[t]) + ah + sackgroup through two conditionaritydon in the sw desh -- rp.dot(dhrew, xs[t].T debh -- rp.dot(dhraw, hs[t-1].T) direct = as dot(whit. T. direct) for operan to [dech. chin, duty, dbh, dby]: *p.clip(duerom, -5, 5, set-duerom) = clip to extigeto explosing gradients return loss, dwin, dwhe, dwhy, dbh, dby, hs[len(inputs)-i] mer named oth, need to, not named a sequence of Integers from the mosel. h is memory state, seed in is send letter for first time stap a - mg permaticularity size, 131 for t in stange(n): b c mp. Carb(mp. doc(sub, x) = np. buc(sub, z) = bb) y = np. dot(hhy, h) + by $p = np. \exp(y) / np. xam(np. exp(y))$ return tres mach, make, make = mp percon [Simpleshi], mp percon [Simpleshi], mp percon [Simpleshi] abs, mby * mg.marce_like(bb), rg.marms_like(by) * memory variables for m smooth_look = -mg_log().Streeth_size('eng_lemph' = look of tiresting b if p-seq length-; -- lenddata) or s -- s; Spray 1 op peros[(Scases_size, 1)) / root too mount imputs = [char_to_is[th] for ch in deta[p:s+seq_isqth]] targets = [thar_ts_is[th] for th in deta[p:s+seq_isqth:]] timple_is T cample(figres, injute(s), 200) tot = ''.injuis.to.charity! for is in sample_is) print '.....'s as 's' a (fat,) Dock, dash, dath, daty, doc, doc, burev = DockFactiques, targets, hores) smooth_Docs = smooth_Docs =

For person, operior, see in cloc(both, set, alsy, 64, by).

a er seg langth's more only military

parks at liberroing rate * Sparks / Np. sprijess + 1e.8) / mingrad applica

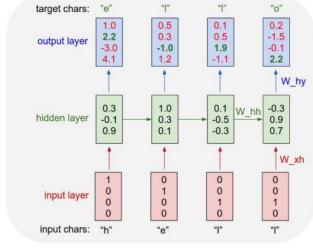
Initializations

```
hidden_size = 100 # size of hidden layer of neurons
seq_length = 25 # number of steps to unroll the RNN for
learning_rate = 1e-1

# model parameters

wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
bh = np.zeros((hidden_size, 1)) # hidden bias
by = np.zeros((vocab_size, 1)) # output bias
```

recall:



```
minimal character. Level vanilla mun model, written by andrei sacuatty (discounts)
850 License
DEPOT NAMED AS THE
 data + open('imput.txt', 'r').read() < steald be simple glade test file.
 chars - list(set(data))
 data_size, worsh_size = les(data), len(chars)
 print 'data has no obseracters, no unique." n (data_size, vocab_size)
 char_to_is = f ch:1 for 1.ch in enumerate(chars) }
 is to cher = ( i:ch for i,ch in www.merate(chers) )
%1559en_size = 100 = size of hidden layer of merces
 seq.length = 25 a mader of stops to servil the 666 For
 learning race - be-1
Web = np.random.randombiddem.sipe, vocab.sipe1'6.81 = broot to bidden
 with = np.random.rando(hiddem.size, hiddem.size)"0.01 = hidden to hidden
 why m np. random rando (vocab_size, hidden_size) *0.01 o hidden to running
 88 = np.zerosi(Nidden_size, 13) = hidles biss
by - mp.ceres((weesb.mice, 1)) - notice that
 def loneFun(imputs, targets, hyrav):
  imputs, targets are both list of integers.
    Agrey is Wel array of initial hidden state
   returns the lose, gradients on model parameters, and last himsen state
   #5, 85, 95 = (), (), (), ()
  Suns - a
   for t in orange(len(inputs)):
     xs[t] = sp.peros((vecab_size,t)) = emode in 1-ef-A representation
      ha[t] = rg.tanh[rp.dot(wsh, xa[t]) + sg.dot(whh, ha[t-x]) + bh) + hidden inits
     ys[t] = np.40t[sty, hs[t]) + by + incormalized log probabilities for rest chara
     as[t] = re.exp(vs[t]) / re.sum(re.exp(vs[t])) = probabilities for ment obers
     loss -- -sp.log[ps[t][tergets[t].0]] # softmax (cross-outropy loss)
   dwin, dath, daty = op.zerss_like(wch), np.zeros_like(wch), np.zeros_like(wchy)
    dah, dby = my.zeros_like(bh), np.zeros_like(by)
    diment - op. zeros_like(ts[0])
   for t in reversed(presented(invitation))
     dy = np.copyros[t])
     dy[tergets[t]] == 1 = terminop inta s
     deby == ng.det(dy, hs[t].T)
     dies on the
     dh = np.dot(Why.T, dy) + dheect = berkgrog into h
     where \tau (a - No[t] * ha[t]) * Wh w backgroup through task conditionarity
     dish += dishlaw
     dich += rp.dst(dhraw, x1[t].T)
     dshh \leftarrow rp.dst(dhrww, hs[t-x].T)
     dispers to the detable to dispart
   for operan in [dech, date, daty, obt, styl]
     ep.clip(Sparse, -5, 5, ext-doorse) - clip to entigety explosing gradients
  return loss, desh, shehr, shery, dob, shy, hs[len(isputs)-i]
def smalleth, seed in, all
  nample a sequence of integers from the model
  % is momory state, excells is used latter for first time step
 a - ap perced (worst, size, 131
  For t in image(n):

6 = np.tamb(sp.dot(sen, s) - np.dot(sen, n) + bb)
    p = np.dot(Mhy, h) - by
    Ix w ms.ramdom.chatce[range(yonah.atze), mss.resel[]]
    Sons, appendite)
 much, make, make - no perce likelikahi, op perce likelikahi, no perce likelikay)
 sworth_loss x -np_lept(.6/words_size)/seq_length x (one at tirrition to
    Aprel 7 No. seros[[512004_3124_1]] = Ferrit from record
g = 0.6 pc from report of data
  leads + timer to Latest for the in datafalanesses length?)
  isrgets - [char_ta_is]shi for ch is deta[poi:goseq_length:i]]
    If n a new ne d:

seepie_ix v sample(typrev, inputs[2], new)

tox = ''.jmin(ix.c.sher[ix) For ix ix sample_ix)

print '----'An Na Na---' N (IXI, )
 Does, dash, dafu, deby, die, Bby, Norve : Instructions a fargets, horsely
amount. Jose + amount. Lasa * 0.000 + Lasa * 0.001
£7 n a 200 to 8 prior later No. Lasa 15° n (n. smooth_lasa) o $100 priories
  For parent operate, seen in conditions, while why, bit, but
                               [dot, den, dey, str, sty],
[mot, men, men, str, sty];
   param +1 - bearing_rate + sparam / sp. oprignes + 10.0) + obsychi spinos
  \mu or meg.langth e once data pairway
```

Main loop

```
n, p = 0, 0
    mWxh, mWhh, mWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
    mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
    smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
    while True:
      # prepare inputs (we're sweeping from left to right in steps seg length long)
      if p+seq length+1 >= len(data) or n == 0:
        hprev = np.zeros((hidden_size,1)) # reset RNN memory
        p = 0 # go from start of data
      inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
      targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
      # sample from the model now and then
94
      if n % 100 == 0:
        sample_ix = sample(hprev, inputs[0], 200)
        txt = ''.join(ix to char[ix] for ix in sample ix)
        print '----\n %s \n----' % (txt, )
      # forward seq_length characters through the net and fetch gradient
      loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
      smooth loss = smooth loss * 0.999 + loss * 0.001
      if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth loss) # print progress
      # perform parameter update with Adagrad
      for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
                                     [dWxh, dWhh, dWhy, dbh, dby],
                                     [mWxh, mWhh, mWhy, mbh, mby]):
        mem += dparam * dparam
        param += -learning rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
      p += seq_length # move data pointer
      n += 1 # iteration counter
```

```
Minimal character-level vanilla MON model. Written by Andrej Karpathy (Sharpathy)
850 License
INDUST HAMPY AS NO
 date = opend'imput, txt', 'r'), read() = unould be pisale plane test file
 data size, worsh size t lesignts). Terrotars)
 scient "data bay his characters, his unique," h idata size, wheah size)
 char_to_ix = { ch:1 for 1,ch in memorate(chars) ;
 is to cher - { ich for i ch in mumerate(chars) }
#1600m_wize = 100 = mire of Midden layer of neurosc
 seq.length = 25 = master of stages to usrail the RSA For
 learning rate * 56-1
With a np. random, randolhidden_size, vocab_size1*0.01 > input to hidden
 with a np.random.rando(hidden size, hidden size)'0.81 a hidden to hidden
 why m np.random.rando(vocab_size, hidden_size)*0.81 v 356500 to dutput
8h I on zevestrhidden size. 155 - modes stas
by = np.zeres((vecab_bize, 2)) = output time
def DoneFundinguts, targets, horevic
  imputs, targets are both list of integers
   barrey to Rel array of initial higher state
   returns the loss, gradients on model parameters, and last hidden state
   88, 86, 98, 98 = (). (). (). ()
   hal-11 = np. copy(horey)
  for t in krampe(len(imputs)):
     so[t] = sp.zerosi(vocah_size,)]) = escoto in I-of-b representation
     hair! = no.tanhino.dot(wah, xx[t]) + no.dot(whh, hair-1]) + bh) + hidden whats
     ys[t] = mp.dot(why, bs[t]) + by w unnarmalized log grobabilities for next share
      gs[t] = np.exp[ys[t]] / np.sum[np.exp[ys[t]]] + profonilities for sent offers
     loss 44 -eg.log(ps[t][targets[t],0]) 4 softens (tress-entrope loss)
    dash, dabb, daby = np.zeros_like(abb), np.zeros_like(abb), np.zeros_like(aby)
    dth, dby 2 mg. zeros like(bb), np. zeros like(by)
    divers = re.peros_like(hsf01)
    for t in reversel(xrange(len(imputs))):
     dy = np.copy(gs[t])
     deby -= rg.dst(dy, hs[t].T)
     di = np.dat(Why.t. dy) + dhreat + hackaray into h.
     deraw = (1 - hs[t] + hs[t]) , on a sectors through two nucleostity
     don += ohraw
     desh so on darliferant safety. Th
     dahh += rg.dot(dhraw, hs[t-1].T)
     distract = ep.dot(whit.t, disraw)
   for operan in [deck, dwth, dwty, dbt, dby]:
     ep.clip(Gearge, -5, 5, satisfactors) + clip to mitiagra exploites professors
    return loss, dech, debb, deby, dbb, dby, hs[len(inputs)-i]
 mer complete, seed in, still
  namely a decision of interest from the social
  5 is memory state, weed_ix to send letter for first time stap
  r = (xz. beselv
    b = ng_1 \cos b(ng_1 \sin ng_1 \sin ng_2) + ng_2 \sin ng_2 \sin ng_2 + ng_3
    y = np.601(hhy, h) + by

p = np.600(y) > np.cum(np.400(y))
  return lives
much, made, early a so revox line (and), so revox line (abh), so revox line (aby)
sen, say = no.zeros_like(bh), np.zeros_like(by) = monry variable: for a
smooth_loss = -np.lng()_S/moned_size()+mp_lemph = loss = literation &
 if proof largebox == largdate) or e == e:

*prov T mp_leros((%120ex_k12e,1)) = const but names)
 import * [char_to_is[ch] for ch in data[s:p-seq_isqth]] targets - [char_ts_is[ch] for ch is data[p:lorseq_length:[]]
    sample_is = sample[tyres, inputs[t], 200)

tex = ''.jein[in.co.chu/[ix] for ix in sample_ix]
    print "------ be by 'so---- " w (but, )
 loss, doh, ddb. ddb. dh, db. by, brev - lossPar(irada, fargeta, harev)
amoch_loss - wooth_loss * 0.000 - loss * 0.000
dr x 300 to 0 yells 'life th. loss 'n' 0 (6, amost_loss) - pilo proprio
 for parent sparent see in similate, who, who, by, by), (shoth, sales, carry, dot, doy),
                                (mark, matri, marry, min, mary)
    parae +1 -inercong_rate * sparser / sp.agrijaee + 38-8] + onigner sonice
```

Main loop

```
n, p = 0, 0
    mWxh, mWhh, mWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
    mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
    smooth loss = -np.log(1.0/vocab size)*seq length # loss at iteration 0
    while True:
      # prepare inputs (we're sweeping from left to right in steps seg length long)
87
      if p+seq length+1 >= len(data) or n == 0:
        hprev = np.zeros((hidden_size,1)) # reset RNN memory
        p = 0 # go from start of data
      inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
      targets = [char to ix[ch] for ch in data[p+1:p+seq length+1]]
      # sample from the model now and then
      if n % 100 == 0:
        sample_ix = sample(hprev, inputs[0], 200)
        txt = ''.join(ix_to_char[ix] for ix in sample_ix)
        print '---- \n %s \n----' % (txt. )
      # forward seg length characters through the net and fetch gradient
      loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
      smooth loss = smooth loss * 0.999 + loss * 0.001
      if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress
      # perform parameter update with Adagrad
      for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
                                     [dWxh, dWhh, dWhy, dbh, dby],
                                     [mWxh, mWhh, mWhy, mbh, mby]):
        mem += dparam * dparam
        param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
      p += seq_length # move data pointer
      n += 1 # iteration counter
```

```
Minimal character-level vanilla mon model, written by Andrej Karpathy (@Karpathy)
850 License
INSETT MARRY ST 70
data - cornd imput. Dat', 'r's, read() a should be simile elsis test file
 data size, worsh size o lenddata), lendcharsh
ories 'data has no characters, no unique,' in conta size, warm size)
 char_to_ix = { ch:1 for 1,ch in enamerate(chars) }
 is to ther - ( tich for i, ch in enserate(chars) ]
Aldden, size = 100 # size of Aldden layer of Hoursess
 sea Sereth + 25 a number of trans on strail the Shi for
learning_rate = 18-1
Web - np. random rando/hidden_sign, yoush_sign)*0.01 + input to hidden
 With - mp.random.randn(hidden_mize, hidden_mize)'0.01 + hidden to hidden
 why t no range rando/socab size, bidden size) a se higher to cannot
Bit I on recognitioned size. 153 a higher hims
by - np.reres((wecob_bize, I)) - conput bins
def lossPan(inputs, tarpets, hurev)
  imputs, targets are both list of integers.
  herey in this array of initial hidden state
   returns the loss, gradients on model parameters, and last hidden state
  85, 85, 95, 85 = (), (), (), ()
  hs[-1] = np.cogy(hprev)
  for t in arange(len(inputs)))
     esfallimputafith - 1
    hs[t] = ep.tanh(np.dot(with, xs[t]) + np.dot(with, hs[t-1]) + bh) = finisen state
     ys[\tau] = rp.det(shy, ts[\tau]) + by a suscriptions be probabilities for rest others
     ps[t] = ps.esp(ys[t]) \wedge ps.sue(op.esp(ys[t])) + probabilities for most chara-
     Into as .- we looked titterestairs oft a cotton (come.company look)
   doon, dath, they = np.zeros_like(son), np.zeros_like(son), np.zeros_like(sey)
   don, dby = np. zeros_like(bh), np. zeros_like(by)
   shnext = ep.peros_like(haf80)
   for t in reversed(srange(len(inputs)))
    dy = np.cosy(os[t])
     dyltargets[t]] -= 1 + taragrou none 1
     dwhy -- re.det(dy, ho[t].T)
     day -- dy
     dn = np.det(Why.T. dy) = dhnext = barkorna inno h
     div_{RMM} = (1 - hs[t] + hs[t]) + dh + beckgrop through their nonlinearity
     district district
     dight to eq.det(dives, safe).Ti
     dahh -- rp.dot(diras, bs[t-1].Y)
    dhosext = np.dot(whb.Y. dbraw)
  for docume in Edwar, duth, day, day, day, day):
    no.clip(Goardw, -5. 5. set-Abbrew) + file (0-militate excissing gradients
  return loss, desh, sheh, shey, dob, shy, hs[len(inputs)-i]
def unulrib, seed,ix, n)
 sample a sequence of inteners from the appeal
 s = es.peres(feocab.sipe. 13)
  tites it II
  for t in arange(n):

n i n_0 tanh(n_0 acc; n_0, n) n_0 duction, n) n to
   y = np. dot(My, h) + by

p = np. exp(y) / np. sam(np. exp(y))
   ty - an random chatcelrapper(wood, steel, con reset[1])
    Dots appendits)
 much, make, maky = no rerox like(mak), no rerox like(ado), no rerox like(ado)
esh. shy + cu.zeros_lixe(id), rp.zerps_liée(by) + moury vo
 prooft, loss = -rg-leg()_0/week_size)'wes_length = line of lightline s
 Spread to Descript (NASSEC, SIDE, SIDE FOR THE SHEET)

# + 0 + ye from start of Asia

Doputs + [star_to_ta[c5] for on in data[s:p+sec_length]]
 if it is 200 or it:
somple_la it sample;tures, impacingly, 200)
   tex = "'/pit(is.to.cher(ix) for ix is mample.is) 
print '----'us ma 'vo---' n (ixi, )
 is forward any lampit obstactors through the ser and front product bins, shok, shin, ship, ship, ship, ship if loss functionals, largers, byrev).
 smooth_loss = smooth_loss = 0.000 + loss = 0.001

If a h loss to bi print 'list hd, loss; hf h [a, smooth_loss] = print printers.
  for param, operam, wen in riselland, who, who, bit, but,
```

param of -bearing_rate * sparam r sprogrames * 10-8) is adopted below

a or see height a new cuts and an in-

Main loop

```
n, p = 0, 0
mWxh, mWhh, mWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
smooth loss = -np.log(1.0/vocab_size)*seq length # loss at iteration 0
while True:
  # prepare inputs (we're sweeping from left to right in steps seg length long)
  if p+seq length+1 >= len(data) or n == 0:
    hprev = np.zeros((hidden_size, 1)) # reset RNN memory
    p = 0 # go from start of data
  inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
  targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
  # sample from the model now and then
  if n % 100 == 0:
    sample ix = sample(hprev, inputs[0], 200)
    txt = ''.join(ix_to_char[ix] for ix in sample_ix)
    print '----\n %s \n----' % (txt, )
  # forward seq_length characters through the net and fetch gradient
  loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
  smooth loss = smooth loss * 0.999 + loss * 0.001
  if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth loss) # print progress
  # perform parameter update with Adagrad
  for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
                                [dWxh, dWhh, dWhy, dbh, dby],
                                [mWxh, mWhh, mWhy, mbh, mby]):
    mem += dparam * dparam
    param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
  p += seq length # move data pointer
```

n += 1 # iteration counter

```
Minimal (Paracter-level vanilla MON model: Written by Andre) Karpathy (@Karpathy)
855 License
Depart mapy as no
data = count('imput.txt', 'r').read() = should be simple glain tout file
share = list(set(data))
data nire, yoush nire v len(data), len(chara)
print 'data has nd characters, nd unique." % (data_size, vocab_size)
char_to_ix = { ch:1 for i,ch in cramerate(chars) }
is to cher = { ich for i ch in enumerate(chers) }
Andden size = 100 o size of higher layer of marries
seq.length = 1% + number of stage to previl the Bask for
web = np.rendom.rendo(hiddom.mize, vocab.mize)*0.01 = input to hidden
White - op. random, random hidden size, hidden size) 0.01 x hidden to hidden
why m np. random. randolyocab size, hidden size)*8.91 * hidden to output
Sh = np.zeros((hidden_size, 1)) = hidden bias
by - mp.zeros((weesb_size, 13) - compat time
def limitin(inputs, targets, horev):
  lagues, targets are both list of integers.
  Herey is Hel array of initial hidden state
  returns the loss, gradients on model parameters, and last hidden state
  #8, 86, 98, #8 = (), (), (), ()
  ha[-i] = np.copy[horev]
  less - 0
    xs[t] = eq.zeros](vocet.size,[]) = county Dr 1-ef.k representation
     \text{fu}[t] = \text{rp.tamh}(\text{rp.det}(\text{woh.} \times \text{s}[t]) + \text{sp.det}(\text{whh.} \text{fu}[t-1]) + \text{bh}) + \text{bidden sinks}
     ys[t] = np.dot(why, hs[t]) + by w unnormalized log grababilities for next unare
     usit) = no member(1) / no sumble repres(1)) a pronoutilities for meet store
    loss += -ep_log[pu[t][tergers[t],0]] = softnex (trans-entropy loss)
  much, made, damy - np.zerzs_like(moh), sp.zeroo_like(moh), sp.zeroo_like(moy)
  shin, dby = sp.peros_like(th), sp.peros_like(by)
  for t in reversed[prescut[len(imputat]));
    dy = np.copy(pa[t])
    dy[rargets[t]] -= 1 + bacagroup into a
     deby += ep.dot(dy, hs(t).T)
    offer an ele-
     dn + np.dot(Why.f, dy) + dhewaxt + backgross inco h
     deraw = (s - hs[t] * hs[t]) * dh = tactoring through them socialmently
    didy += chraw
     dick -- rp.dot[dhraw, xo[t].T]
    dishh \leftarrow rp.dot(dhrea, hs[t-1].T)
    droom our day (who it drown
  for operas in [don, date, daty, die, sty]:
     sp.clip(dures, -5, 5, set-downes) + till to mitigate exploding gradients
  veturn loss, dark, dath, daty, dob, sty, hs[len(inputs)-1]
per constett, seed in, co-
  sample a sequence of lategory from the model.
 % to memory state, need in is send letter for first time step
 * - ma.peres([vecab.size, 13]
  lives of II
   b = n_0 tarajag dotjana, s0 + n_0 dotjana, b0 + b0
  g = np.dot(why. h) = by
p = np.exp(y) / np.cumcnp.exp(y))
tx = np.rendom.chatce(rompnCwcosh.nitm), pwp.rendim.
  i = rg.20782((vec80_kl20, 1))
e[14] = L
   Does appending
 more, made, maky - on recon_like(abb), on recon_like(abb), on recon_like(abb)
     th. Dos 4 -re-legit -blescab.size2'seq.leepth + loss of iteration b
  Norwy T mp. zeroe((%188ex_612e, 2)) = result miss security
p = 0.4 gs from start of data
  imputs + (ther to im(th) for th in deta(v.a-eeg length))
  of n x 200 -= A:
sample is t sample(torev, legacia)s), 200)
  tot = ''.jelelia.to.mar(ix) for is is nemale.is)
print '----on my 'm----' x (tot, )
  - firstern and larger connectors through the out and fatch gradient loss, dash, dash, dash, dash, dash, day, dash, day, day, are a loss functionals, largers, horses
  amount love - amount later * 0.000 + late * 0.000 
if n % amount in prior *ster bd, lead off @ (A, amount later) + prior progress
```

for param, dearen, men in ripClash, wan, way, ba, byl.

parament -bearing_rate." Maramer squarecomm = 16.62 x structures

Main loop

```
n, p = 0, 0
mWxh, mWhh, mWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
mbh, mby = np.zeros like(bh), np.zeros like(by) # memory variables for Adagrad
smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
while True:
 # prepare inputs (we're sweeping from left to right in steps seq_length long)
 if p+seq length+1 >= len(data) or n == 0:
    hprev = np.zeros((hidden_size,1)) # reset RNN memory
    p = 0 # go from start of data
 inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
 targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
 # sample from the model now and then
 if n % 100 == 0:
    sample_ix = sample(hprev, inputs[0], 200)
    txt = ''.join(ix to char[ix] for ix in sample ix)
    print '----\n %s \n----' % (txt. )
 # forward seq_length characters through the net and fetch gradient
  loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
 smooth loss = smooth loss * 0.999 + loss * 0.001
  if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth loss) # print progress
 # perform parameter update with Adagrad
 for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
                                [dWxh, dWhh, dWhy, dbh, dby],
                                [mWxh, mWhh, mWhy, mbh, mby]):
    mem += dparam * dparam
    param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
 p += seq length # move data pointer
  n += 1 # iteration counter
```

a -c see length - more data satisfied

```
Sisimal (Paracter-level vancila MON model: written by Andre] Karpathy (@Karpathy)
 850 License
 Literat makey as no
 data = count'imput.txt', 'F'), read() = should be missis about town file
 data_sire, vocah_sire v len(data), len(chara)
 scint 'data has no characters, no unique,' n (data size, worsh size)
 thar_to_ix = { chil for i,ch in cramerate(chars) }
 is to cher = { such for 1, ch in enumerate(chers) }
 Addler.aire = 100 o size of Abbien layer of marora
 seq.length = 28 4 master of steam to invell the same for
 learning_rate = 18-1
 Web = rg.rendom.rendo(higher.size, vocab.size)*0.01 = brook in higher
 Whit = np.rancom.rawdn[hidden_size, hidden_size]'6.81 = hidden in hidden
 why - np.random.randojvocab_wire, hidden_strej*m.mi + hidden to output
 SN I mo reconfibilities size. 155 a higher bias
 by = np.zeros((wood_nize, 1)) = campat time
 def linifications, targets, horavit
   imputs, targets are both list of integers.
   Berry to Kel array of tetrial higher state
   returns the loss, gradients on model parameters, and last hidden state
   49, 86, 55, 85 = (), (), (), ()
   hal-if = np.cody[hprey]
   for t in prosperientingers));
     sufilizemutafill - 1
     hu[t] = np.tem(np.det(wsh, xu[t]) + np.det(wsh, hu[t-1]) + bh) + histori ninte
     yx[t] = np.dot(why, he[t]) + by + unnernalized log probabilities for next share
      as[t] = m.emiys[t]1 / mp.sw(mp.emiys[t])) = probabilities for each store
     loss += -ep_log[ps[t][tergers[t],0]] + softnex (trans-entropy loss)
   much, date, daty = np.zerzs_like(wch), sy.zeros_like(wcn), np.zeros_like(wcy)
   stin, dby = sp.peros_like(bh), sp.perus_like(by)
   dheext = sp.geros_like(hs[0])
   for t in reversed[xrange(lan(inputs))):
     dy = rp.copy(px[t])
dy[tacpets[t]] == 1 + backgrop into y
      deby += mp.dot(dy, fo(t).T)
     dity += dix
     di = np.dot(Why.f. dy) + dhrext + backgros inco h
      deraw = (1 - As[t] * hs[t]) * dh = tactoring through their confinentity
     did to draw
     dight on my datisforms, saft1.T1
      dishh \leftarrow rp.dot(dhraw, hs[t-1].T)
     direct = sp.dot(abb.T. dbraw)
   for govern in Ideal, data, data, day, day, day.
     mp.clip(dearem, -5, 5, met-observe) + tilly to mitigate excluding gradients
   vaturn loss, desh, shirb, duby, dbh, sby, hs[len(inputs)-1]
 ster sample(b, seek_bc, 40)
  casals a sequence of integers from the model.
  A is memory state, weedlik is send letter for first time step
  e's estrement (worst size, 13)
   lives - II
  For t in prange(n):

b = np.tamb(sp.dotpoh., x) + np.dotpom., h) + bb)
   g = ng.dot(why. n) = by
p = ng.sep(y) / ng.com(ng.emp(y))
px = ng.rendom.chetce(rongn(woodh.nite), pro.texel[])
   i + rg.20781((vecab_8220, 1))
e[14] = E
    Does, appendige
  more, made, maky - on recon. likelishti, on recon. likelishti, on recon. likelishti
smooth_loos = .rg. legit.e/recom_size1'seq_length = less at tiretion t
smalle from:
    forey, it mp. percentification alize, bit is residently assessed
   inquire + (ther to in[th] for th in data(w.e-eeu length))
   targets - Johan on Island for the is data[pol/prises_length:[]]
  of n x 200 -- x:
sample_is t sample(three, legacity), 200)
   tot + "'.jeislis.to.mar[ix] for is in semile.is)
print "-----he me "m-----" x (tst., )
   + firmers and largit convectors through the set and finish product loss, deek, debt, debt, debt, debt, dept and largets, horself
  smooth_loss = smooth_loss = 0.000 = loss = 0.0001
If n w loss to an arise "last wit, loss wit w (n, amouth_loss) = arise propress
   for param, dearen, men in cip()and, who, why, bit, by
    parament - Dearning years 1 squares / squares + squares + squares + squares squares
```

Main loop

n, p = 0, 0

```
mWxh, mWhh, mWhy = np.zeros like(Wxh), np.zeros like(Whh), np.zeros like(Why)
    mbh, mby = np.zeros like(bh), np.zeros like(by) # memory variables for Adagrad
    smooth loss = -np.log(1.0/vocab size)*seq length # loss at iteration 0
    while True:
      # prepare inputs (we're sweeping from left to right in steps seg length long)
      if p+seq length+1 >= len(data) or n == 0:
        hprev = np.zeros((hidden_size, 1)) # reset RNN memory
        p = 0 # go from start of data
      inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
      targets = [char to ix[ch] for ch in data[p+1:p+seq_length+1]]
      # sample from the model now and then
      if n % 100 == 0:
94
        sample ix = sample(hprev, inputs[0], 200)
        txt = ''.join(ix_to_char[ix] for ix in sample_ix)
        print '----\n %s \n----' % (txt, )
      # forward seq_length characters through the net and fetch gradient
      loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
      smooth loss = smooth loss * 0.999 + loss * 0.001
      if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth loss) # print progress
      # perform parameter update with Adagrad
      for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
                                     [dWxh, dWhh, dWhy, dbh, dby],
                                     [mWxh, mWhh, mWhy, mbh, mby]):
        mem += dparam * dparam
        param += -learning rate * dparam / np.sgrt(mem + 1e-8) # adagrad update
      p += seq_length # move data pointer
      n += 1 # iteration counter
```

```
winised character, level vanidia was model, written by andre) surpathy (discounts)
850 License
SMEST NAMEY AS NO
data = open('imput.txt', 'r').read() = steeld by missis plain test file
 data_sire, vocat_sire = les(data), les(chars)
griot "data has no characters, no unique." In coata size, vocab size)
char_to_ix = f chil for i_ch in enumerate(chars) :
is_to_cher = { i:ch for i,ch in wessersts(chars) }
Madden, size = 100 = size of Midden layer of Hearters
 seq_length + 25 # number of steps to shrell the NAW For
learning rate + 18-1
with a ret random randoffstoder size, worsh attest 0.01.4 hours on builden
With = np.random.rando(hidden_size, hidden_size)*0.61 = hidden to hidden
why = np.random.rando(vocab_size, hidden_size)*m.ml = midden to mutgict
86 = np.zeros((6186en_632e, 3)) = figures $1.65
be a my zeroof (wocob_mize, 13) a mating blas-
def lossmun(deputs, targets, horse):
```

```
imputs, targets are both list of integers.
 returns the loss, gradients on model parameters, and last hidden state
ht[-1] = rp.cepy(hprev)
for this prangetlendingstable
  exit) + rm. zeroxi(vocah. size. 1)) = connor in 1. of 4 representation
   na[1] = np.tanh(np.dot(wsh, xa[t]) + np.dot(wsh, ha[t-1]) + bh) = nimins state
  well to be detined, being + by a unique billion in accomplished for next char-
   asit) = re-exptysiti) / no.sum(re-exp(ysiti)) > probabilities for meat others
    loss 40 -ep.log(pa[t](targeta[t],0]) 0 softmax (traca-entropy loss)
 desh, dath, daty = rp.zeros_like(soh), sp.zeros_like(shh), rp.zeros_like(shy)
 Stolect - sp. peros. like(taff))
  dy(targets(t)) of t + macron line a
   deby == re.def(dy, hs211.T)
  dh = rp.det(why.T. dy) + dhrext = lactures into h
  dhraw t (1 - ha[t] + ha[t]) + ah = naceprop strongs now continuously
  g(n) += ehrme
   |T.[1]cs .wawfoltcb.un == thisb
  den - re-detidores, haft-iT.TI
   dyorox + ep.dot(Web.+, dhraw)
for operate in [death, date, date, date, day]:
 op.clip(deares, -5, 5, est-openes) = (ii) to excipt excitating gratients return loss, dech. debt. deby, skh. dey, he[len(ispets)-1]
```

```
It is momory state, need in is seed latter for first time step
  s o na reveni (tracely stee, 17)
  x[seed,ix] = 1
  ives = []
for t in arange(n):
   k = n\phi canding solution, n\rangle + n\phi solution, n\rangle + k\phi

\phi = n\phi solution, n\rangle + b\gamma
    a t recessivit / recession.emptyti
    s = rg.zerax((vocab.xize, 1))
    Does append(1x)
mint, mich, miles o en verse l'épalisable, en cares l'épalishie, en cares l'épalishie,
assert tons a .ne. legit frames size! See I see to a loss at the otten it
while from:

o prepare inputs (serve sampling from left to right to single and length long)
  if peas lengther = lenders) or a == 6:
    Aprex - No. Jeros (Clades, Size, 1) - 10161 from money
  tracts = (ther to la(ch) for on in Sta(s:p-ess_length))
targets = [ther is_sa(ch) for sh is Seta(s=turses_length=Cl)
    paspin_is = sample(tures, imputs(0), 200)
    est a "" tain(in.to.marlis) for in in sample is)
  loss, dash, dath, dath, dath, dbt, dby, harev = lossPunclepath, targets, farrer) amount loss = amount lass = 0.000 + lass = 0.000
  of m to come to be proved 'other tod. Leads the' to the amount because a print progress
                                     Treat, men, mov. etc., etc.)
    param of liberroom_rate * dparam r up.oprignes + 1e.4) = wingrat option
```

Loss function

- forward pass (compute loss)
- backward pass (compute param gradient)

```
def lossFun(inputs, targets, hprev):
 inputs, targets are both list of integers.
 hprev is Hx1 array of initial hidden state
  returns the loss, gradients on model parameters, and last hidden state
 xs, hs, ys, ps = {}, {}, {}, {}, {}
 hs[-1] = np.copy(hprev)
 loss = 0
 # forward pass
  for t in xrange(len(inputs)):
   xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
   xs[t][inputs[t]] = 1
   hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state
   ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
   ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
   loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
  dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
  dbh, dby = np.zeros_like(bh), np.zeros_like(by)
  dhnext = np.zeros_like(hs[0])
  for t in reversed(xrange(len(inputs))):
   dy = np.copy(ps[t])
   dy[targets[t]] -= 1 # backprop into y
    dWhy += np.dot(dy, hs[t].T)
   dbv += dv
   dh = np.dot(Why.T, dy) + dhnext # backprop into h
   dhraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
   dbh += dhraw
    dWxh += np.dot(dhraw, xs[t].T)
   dWhh += np.dot(dhraw, hs[t-1].T)
   dhnext = np.dot(Whh.T, dhraw)
  for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
```

np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients

return loss, dwxh, dwhh, dwhy, dbh, dby, hs[len(inputs)-1]

```
min-char-rnn.py gist
                                                                                                                            def lossFun(inputs, targets, hprev):
  Minimal character-level vanilla MMB model. Written by Andrej Kurpathy (@Kurpathy)
  import numpy as no
                                                                                                                                inputs, targets are both list of integers.
                                                                                                               29
  data - compliance test, inth read() - checks by simple whate test fills
                                                                                                                                hprev is Hx1 array of initial hidden state
  data sire, vocab sire = les(data), les(chars)
  print 'data has no characters, no unique,' to cours size, worst size;
  char_to_ix = { ch:1 for i_ch in coumerate(chars) }
                                                                                                                                 returns the loss, gradients on model parameters, and last hidden state
  is_to_cher + ( i:ch for i,ch in enamerate(chars) )
  Abdden, blize = 100 / size of blocks layer of manyons
                                                                                                                                 11 11 11
  seq length = 28 a master of atom to serall the SSS for
                                                                                                                                xs, hs, ys, ps = {}, {}, {}, {}, {}
  With 4 op. Faridom, rando/hiddem, alice, vocab, alice? 0.01 + legal to hidden
  with - rp. random.randn(hiddem_size, hidden_size) *0.01 = hidden to hidden
  why m no random random vocah size, hidden sizethe es a hidden to outsut
                                                                                                                                hs[-1] = np.copy(hprev)
  Bh I no zeros/(hidden size. Til a hidden bias
  by = np.zeres((wacab.size, 1)) = output Size
  def losseur/impara, targets, horse)
                                                                                                                                 loss = 0
    liquits, targets are both list of integers.
   between to had strong of leiting budden state
                                                                                                                                # forward pass
    returns the loss, gradients on model parameters, and last hidden state
                                                                                                                                for t in xrange(len(inputs)):
                                                                                                                                     xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
     \text{Re}[x] = \text{rg.tarm[rg.dec(with, se[x])} = \text{rg.doc(whh, ha[x-1])} + \text{hh}) = \text{Nintern stress}
                                                                                                                                     xs[t][inputs[t]] = 1
   dem, dem, dwhy - rp.zerss_like(soh), sp.zeros_like(whn), rp.zeros_like(shy)
                                                                                                                                    hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state
   din, dby = up.zeros_like(bb), up.zeros_like(by)
   For t in reserved(xrange(len(imputs))):
                                                                                                                                   ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
                                                                                                               41
     dy * np.copyras[1])
     dery += mp.dot(dy. hu(t).7)
                                                                                                               42
                                                                                                                                     ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
     drose = \{x - he[x] + he[x]\} + dh + harkgroun through twon conditionally.
     don += dhraw
                                                                                                                                     loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
     dem - rp.det(draw, sa[t].T
                                                                                                                43
     didn += rp.dot(dhraw, hs[t-1].Y)
     displace of the doct (white it displace)
   for sparse in [deck, date, daty, obe, sty]
   return loss, don, deb. dev. do., dov. hallentimputs)-il-
   It is memory state, used is in seed latter for first time atem
   a = mg.peronilwscab.size, ill
                                                                                   h_t = 	anh(W_{hh}h_{t-1} + W_{xh}x_t)
   e[med_ix] = 1
ixes = []
for t in prange(n):
    B = p_{\phi} to be p_{\phi} and p_{\phi} and p_{\phi} and p_{\phi} and p_{\phi} and p_{\phi} and p_{\phi}
    w = top. ext(Mey., N) + bu-
    p = np. exp(y) / np. exp(y), exp(y))

IX = np. rendem.che(ce(range(vocab_nize), pro_renel())
    a = np. zerwariyaceh. xize. 135
   efod + 5
zors.append(1c)
                                                                                   y_t = W_{hy}h_t
  mach, makh, maky = mp.zeroz_like(who), mp.zeroz_like(who), mp.zeroz_like(who)
odh, mby = mp.zeroz_like(bb), mp.zeroz_like(by) = mmory verigilize for mingrar
  search loss o the leaft diseast appethes from a loss of franchis of
   If p-seq_length-1 = len(deta) or $ == 6;
Aprex = sp_lencs((NLBSeq_klap_1)) = reset NNL semir
                                                                                                                                                                       Softmax classifier
   inquity - fifter to infich! for on thi details:p-ess length!
   targets - [thar_ts_is[ch] for ch is data[p-1-p-seq_length-1]]
    umple_is = umple(tprev, imputage), 200)
tot = ''.jeln(is_to_oher[ix] for ix in semple_ix)-
    print formed to be and a citie. It
   loss, dock, date, date, date, day, havev - lossPunctionals, targets, haves)
   insc. dev. dem. dev. deb. dby, tore : instructionet, targets, turner;
imsoch_loss : imseth_loss ' v.500 * loss ' 0.001
if n h dem to 0 print 'lier nd, loss' by to 0 meeth_loss) = print programs
   for earner, degrees, were in storthads, also also let, but
                     limak, mate, mate, see, storil-
```

paras < - deeming_rate * deares / sp-sprt(see = 1e-t) = mingrad motion

min-char-rnn.py gist # backward pass: compute gradients going backwards 44 dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why) 45 epaleal character-level vanilla men model, written by andrey corpathy (@karpathy) 850 License dbh, dby = np.zeros_like(bh), np.zeros_like(by) 46 SWEET PLREN AS RE data = spen('imput.txt', 'y').read() = should be simple state (set file dhnext = np.zeros_like(hs[0]) 47 chars a Dischartidatell data_sire, vocab_sire = lem(data), lem(chars) print 'data has 'nd characters, 'nd unique.' 'n (data_size, vocab_size) for t in reversed(xrange(len(inputs))): char to in a f chit for 1 sh in enserate(chars) I 48 is to cher = { sich for i,ch in ensmerate(chers) } dy = np.copy(ps[t])\$386ex.512e = 180 * Size of States layer of concress 49 seq. length = 15 r marker of steps to orrail the AMA Fac learning_rate = 1e-1 dy[targets[t]] -= 1 # backprop into y 50 With a mp. random. random(hilden_size, yeash_size)*0.01 + Input in hilden With a no random resolutions also bidder also bidder also be a bidder to bidder Why = np.random.rando(vocab_size, hidden_size)*s.st + hidden to subject dWhy += np.dot(dy, hs[t].T)\$6 = np.zeros((%1dden_size, 1)) = %100es \$15 by a mo. reves((wscab. nice. 13) a compar hims def lacemontispace, targets, Agrav): dby += dy imports, targets are both list of integers. dh = np.dot(Why.T, dy) + dhnext # backprop into h returns the loss, gradients on model parameters, and last hidden state 53 45, 76, 55, 85 = Cl. (), (), () 54 dhraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity less - 0 dbh += dhraw hu[t] = np.turb[np.sec(wih, xx[t]] + np.det[wih, he[t-t]] + bb) + filler restaurantsys[t] = sy. set(aby, be[t]) + by a associalized leg probabilities for sext charsdWxh += np.dot(dhraw, xs[t].T) 56 loss -- -sg.log(m(t)[targets[t],0]) - orfone farmes-moreous loss) dan, dby = ap.geros_like(\$8), op.perss_like(by) dWhh += np.dot(dhraw, hs[t-1].T)57 dy = rp.cngy(px[t]) dhnext = np.dot(Whh.T, dhraw) 58 deby += ne.detloy, Reft1.T1 59 for dparam in [dWxh, dWhh, dWhy, dbh, dby]: shows a (3 - he[t] . he[t]) . sh a bacuaran through them conditionally dich -- ip.dot)dires, xs[t].T) debt -- ne.detidhres. haft-17.71 np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients 60 diversit + ep.dot(whh.f. dhraw) for donras to (dech, then, day, one, day) return loss, dwxh, dwhh, dwhy, dbh, dby, hs[len(inputs)-1] 61 s - sp.peres((wocati.s)(es, 1)) target chars: 70 de "0" 0.5 0.1 0.2 1.0 $b = rg. the blue, bottom, <math>x_1 + rg. sottom, h_1 + bb_2$ $y = ng. sottom, h_1 + bs$ -1.5 2.2 0.3 0.5 p T re-weety) / ne-samon-weety)) output layer -3.0-1.0 1.9 -0.1 s - vp.perso((vocab_size, 1)) 4.1 1.2 -1.1 2.2 ixes.append(is) W hy mich, mich, michy = op.cercs.like(with), op.cercs.like(with), op.cercs.like(wity) mbs, mby = 60.20ros_like(ch), np.zerss_like(by) = money variables for entered smooth love is one legit Armount size? See length a loss of transition & 0.3 0.1 -0.3 1.0 W hh if p-seq_length-1 = lengthta) or x = x; %prev = np_lennon((tidden_tide, 1)) = reset for reset) hidden layer -0.1 0.3 -0.5 0.9 recall: rputs = [char_to_is[ch] for ch in data[p:p-keq_length]] 0.9 0.1 -0.3 0.7 targets a Johan Sa culcky for on in datafact occurs length (1)

comple_In I compleyfores, imputato], 2003

but + " interior but for in in semin_tel artest "cooks as become a case, h

For param, operam, mem in rip([sech, web, why, bh, by],

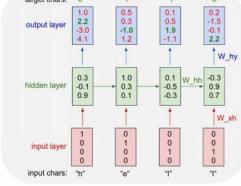
p or seq length a new only outstood

loss, dark, dahr, dahr, dah, day, harev = lossPunctionits, targets, harev) smooth loss * smooth loss * s. 600 + loss * 0.001

param of cheering rate " maram / malagrithms - 1e-65 o minuted update

AT N % 300 At 21 print "liter hd, lane: M" h (n, smooth_lens) a print progress

I more, mate, make, etc., stoy I i



For seres, dearen, see in stoclass, who, who, bit, by?

y or seq length a more data patrons

[Good, dath, dath, day), 504, 509].
[Mark, make, make, sake, ske]])

person -- Sparroon, Yate - Sparroon - Sparroon -- Sparroo

```
*Inimal character-level vanilla mos model. Written by Andrej Karpathy (@karpathy)
880 License
Depart mapy as no
 data = open('imput.txt', 'r').read() = should be simple stain test file
 chars - List(set[data])
 data_size, vocab_size = lew(data), leegchars)
 print 'data has he characters, he unique.' h (data_size, vocab_size)
                                                                                                                                                                                                                                        def sample(h, seed_ix, n):
 char_to_ix = { ch:1 for i_ch in conserate(chars) }
is_to_cher = { 1:ch for 1,ch in ensmerate(chers) }
                                                                                                                                                                                                                  64
$30000.0120 = 100 = 1120 of States Layer of supress
 seq.length = 25 # minter of staps to orrall the Abb for
 learning rate - 18-1
                                                                                                                                                                                                                                                   sample a sequence of integers from the model
                                                                                                                                                                                                                  65
Web = op.random.rando(hiddem.size, vocab.size)*0.01 + logot to hidden
With a no random rendnihipdem size, hiddem size) 9.91 a hidden to hidden
 Why - np.random.randn(vocab_size, hidden_size)*s.st + tidden to satural
                                                                                                                                                                                                                                                   h is memory state, seed_ix is seed letter for first time step
 $6 = np.zeros((%ldden_size, 3)) = hinden black
by - op.geres((woosb.size, 13) - output him
 def loosfun(inputs, targets, Aprav):
                                                                                                                                                                                                                                                   11 11 11
  legate, targets are both list of integers.
   turey in And array of initial hidden state
   returns the loss, gradients on model parameters, and last hidden state
                                                                                                                                                                                                                                                  x = np.zeros((vocab size, 1))
   85, 86, 95, 85 = (), (), (), ()
  Sees - s
                                                                                                                                                                                                                  69
                                                                                                                                                                                                                                                  x[seed_ix] = 1
   for t in wrange(imn(impets)):
    raft) = op. peroal(yeegh_Alpe_1)) = evenin to 1.07.4 venumentarise
     hu[t] = np. \tanh(np. dot(wit, sx[t]) + np. dot(wit, hu[t-1]) + hh) + ntcosm units
                                                                                                                                                                                                                                                  ixes = []
    ys[t] = sq.det(why, hs[t]) + by = secondalized leg probabilities for next observe
     ps[t] = no.exp(vs[t]] / no.sum(np.exp(vs[t]]) + prononlilities for most where
                                                                                                                                                                                                                                                  for t in xrange(n):
   mon, men, men, meny = np.zersa_like(soh), sp.zersa_like(seh), np.zersa_like(sehy)
   dan, dby = sp.zeros_like(bb), sp.zeros_like(by)
   Sheert - op.peros_like(fo[0])
   For 1 in reversed[xrange(len[inputs])]:
                                                                                                                                                                                                                                                            h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
     dy = rp.cspy(ps[t])
    dy[terpets[t]] == 1 + heropros into a
     deby == ep.dot(dy. hsft1.T)
                                                                                                                                                                                                                                                            y = np.dot(Why, h) + by
     dn = np.dst(Why.T. dy) + dhrext = harkurus iron h.
     deraw \tau (1 - Ms[t] * Ms[t]) * dh + balayong through term nonlinearity
     don += oticev
     dish = np.dot(diram, so[t],T)
                                                                                                                                                                                                                                                            p = np.exp(y) / np.sum(np.exp(y))
                                                                                                                                                                                                                  74
     debt ++ re.dot/dhrew. ts[t-1],Ti
     divext = sp.dot(whi.Y, direw)
   for operas in [dect, date, day, dor, day]:
    no.clip(sparse, -5, 5, set-doorem) = clip in militare contoning products
                                                                                                                                                                                                                                                            ix = np.random.choice(range(vocab_size), p=p.ravel())
   return loss, dech, debt, deby, deby, dby, hs[len[ispate]-i]
  sample a sequence of integers from the most
                                                                                                                                                                                                                                                           x = np.zeros((vocab_size, 1))
 % Is memory state, weed in is need latter for first time stay
                                                                                                                                                                                                                  76
 s[med_ix] = t
                                                                                                                                                                                                                                                           x[ix] = 1
   b = n_0. Lending doclain, x_1 + n_0. boc(den, h) + (b)

y = n_0. boc(dey, h) + boc

y = n_0. cop(y) \neq n_0. cop(x_0, exp(y))
   is = no.remdom.chetce(range(vecah_mits), pro.remtl(1) s v np.lermix(vecah_mits, 1))
                                                                                                                                                                                                                                                            ixes.append(ix)
                                                                                                                                                                                                                                                  return ixes
                                                                                                                                                                                                                  79
much, math, midy - no reros likelishi), no reros likelishi), no reros likelishi
 smooth, Jose - -rp. log() #0most_size('seq_length < loss of transliss #
 if p-seq_largit-1 = largidata) or x == 8;
  p = 0 it go from start of data
imputs = (char to is(ch) for ch in data(o:p-seq length))
  targets - [char_ta_sa]ch] for ch in data[pri:pricq_length:j]]
    used to 7 capaternary, inputatel, 2005
   print from the tal Month in (the, )
 Inc., darb. dabo, dabo, dab, day, have - DosFrentingsts, tergets, forey) smooth_loss = smooth_less * 0.000 + loss * 0.001 to n loss * 0.001 to n loss * 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100 + 100
```

at first:

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e plia tklrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

train more

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

train more

"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftened him.

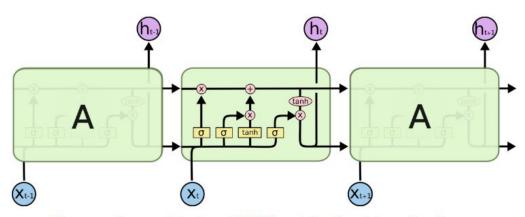
Pierre aking his soul came to the packs and drove up his father-in-law women.

How to improve this code?

- GPU
- Add minibatches
- LSTM
- More powerful model(multilayer LSTM)
- Regularization (Recurrent Neural Network Regularization, https://arxiv.org/abs/1409.2329)
- https://github.com/karpathy/char-rnn

Recap architectures LSTM - Long short term memory

$$\begin{split} & \operatorname{LSTM}: \ \mathcal{X}_t \ , h_{t-1}, c_{t-1} \to h_t, c_t \\ & i = sigm(W_i \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_i) \\ & f = sigm(W_f \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_f) \\ & o = sigm(W_o \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_o) \\ & g = tanh(W_g \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_g) \\ & c_t = f \odot c_{t-1} + i \odot g \\ & h_t = o \odot \tanh(c_t) \end{split}$$



The repeating module in an LSTM contains four interacting layers.

LSTM implementation

LSTM implementation

```
c, h = \{\}, \{\}
  c[-1] = tf.tile(self.c 0, [tf.shape(x)[0], 1])
  h[-1] = tf.tile(self.h 0, [tf.shape(x)[0], 1])
for t in xrange(steps n):
      x t = x[:, t, :]
      input = tf.concat([x t, h[t - 1]], axis=1)
      z = tf.matmul(input, self.W) + self.bias
      i = tf.sigmoid(z[:, 0 * hidden n: 1 * hidden n])
      f = tf.sigmoid(z[:, 1 * hidden_n: 2 * hidden_n])
      o = tf.sigmoid(z[:, 2 * hidden n: 3 * hidden n])
      g = tf.tanh(z[:, 3 * hidden n: 4 * hidden n])
      c[t] = f * c[t - 1] + i * g
      h[t] = o * tf.tanh(c[t])
```

$$\begin{split} & \operatorname{LSTM}: \ \mathcal{X}_t \ , h_{t-1}, c_{t-1} \to h_t, c_t \\ & i = sigm(W_i \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_i) \\ & f = sigm(W_f \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_f) \\ & o = sigm(W_o \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_o) \\ & g = tanh(W_g \cdot \binom{\mathbf{h}_{t-1}}{\mathbf{x}_t} + b_g) \\ & c_t = f \odot c_{t-1} + i \odot g \\ & h_t = o \odot \tanh(c_t) \end{split}$$

Bidirectional RNN

- Nothing special about them
- Just run one RNN from left to right
- The other one from right to left
- Concatenate their respective hidden states

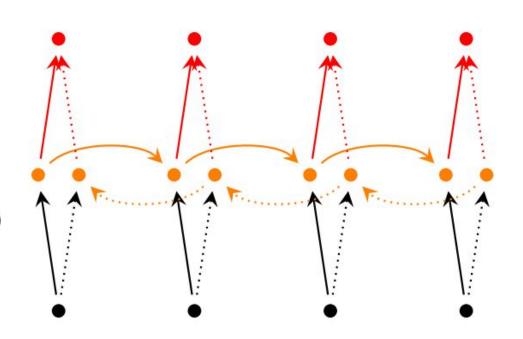
$$h_{bidirect}[t] = concact(h_{forward}[t], h_{backward}[t]) \label{eq:hbidirect}$$

Or just take

$$h = concact(h_{forward}[T], h_{backward}[0])$$

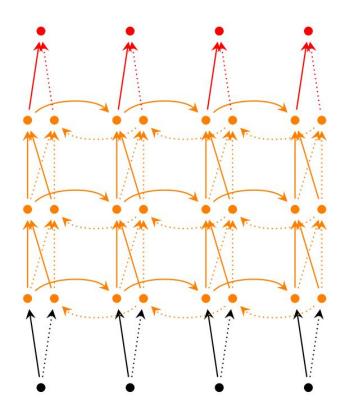
when doing classification

 Used when you want h_t to depend not only on the "past", but also the "future"



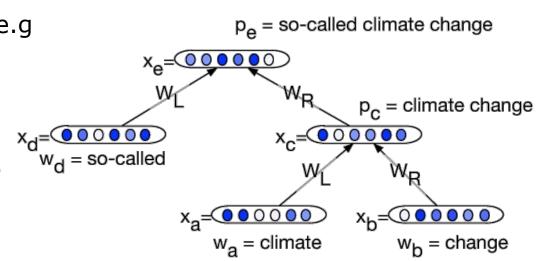
MultiLayer RNN

- Just run one RNN on the outputs of another one
- We can combine this with bidirectional RNN and make Deep Bidirectional RNN
- Or Deep Bidirectional LSTM



Recursive neural networks

- Works on a tree structure, e.g suppose you want to do do sentiment analysis and have parse tree of the sentence
- A RNN is just a special case of RvNN, a linear tree



Tensorflow RNN API

Using RNNs/LSTMs is very easy using tensorflow.

```
# Unstack to get a list of 'n_steps' tensors of shape (batch_size, n_input)
x = tf.unstack(x, n_steps, 1)

# Define a lstm cell with tensorflow
lstm_cell = rnn.BasicLSTMCell(n_hidden, forget_bias=1.0)

# output, state = lstm_cell(x_t, state)

# Get lstm cell output
outputs, states = rnn.static_rnn(lstm_cell, x, dtype=tf.float32)
```

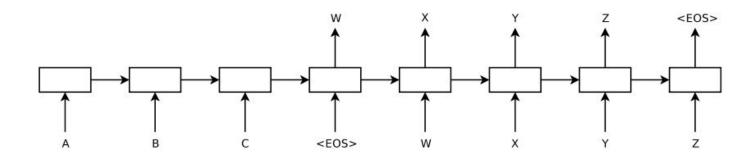
https://www.tensorflow.org/tutorials/recurrent

Case studies

- seq2seq
- Machine translation with attention
- Caption generation
- Neural Turing Machine

seq2seq

- We want to translate from English to French
- WMT'14 dataset, 12M sentences consisting of 348M French words and 304M English words
- Basic idea: train some RNN to take source sentence one word at a time(encoder), and other RNN to take this encoded state and produce one word of translation at a time.



Sequence to Sequence Learning with Neural Networks, Ilya Sutskever et al.

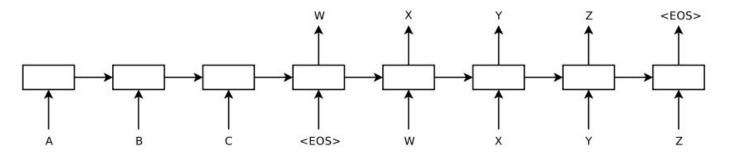
https://arxiv.org/abs/1409.3215

How to treat words? - Word embeddings

- We give each word in a vocabulary a d-dimensional real vector
- We learn those vectors jointly with our whole model
- The word embedding of a word can learn to represent important "things" about this word, e.g. if this is verb/noun?, is it positive/negative? and many other things that are not easily described in words
- Often we initialize these word embedding with some precomputed vectors, trained in unsupervised manner, like word2vec
- 1000 dimensional word embeddings used in this paper

seq2seq model details

- Words embeddings at input each timestep
- Two different LSTMs, one for encoding the source sentence and one for decoding
- "we found it extremely valuable to reverse the order of the words of the input sentence"
- "we found that deep LSTMs significantly outperformed shallow LSTMs, so we chose an LSTM with four layers"
- Beam search for selecting best translation
- Training was not easy, had to split the model to many gpus
- Trained on 8 GPU, 4 of them just for computing softmax



seq2seq

WMT '14 English-to-French translation task

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	34.81

Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
State of the art [9]	37.0
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	36.5
Oracle Rescoring of the Baseline 1000-best lists	~45

Table 2: Methods that use neural networks together with an SMT system on the WMT'14 English to French test set (ntst14).

- Did not beat state of the art in 2014
- But in 2016 DL base methods dominated the WMT 2016

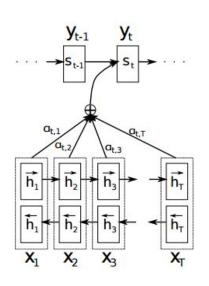
Attention in neural networks

- focusing on part of the information
- e.g. we will see, RNN attending to certain parts of other RNN's output
- Andrej Karpathy: The concept of **attention** is the most interesting recent architectural innovation in neural networks.

Machine translation with attention

- Same task: machine translation.
- High level idea:
 In seq2seq encoder RNN encoded the source task into some fixed dimesional hidden vector.
 Then the decoder RNN decoded this state one word at a time.
- Now we want the RNN outputting the words of translation to be able to attend to parts of the source sentence it "thinks" are useful for generating the output word.

Machine translation with attention



$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

$$h_j = \left[\overrightarrow{h}_j^{\top}; \overleftarrow{h}_j^{\top}\right]^{\top}$$

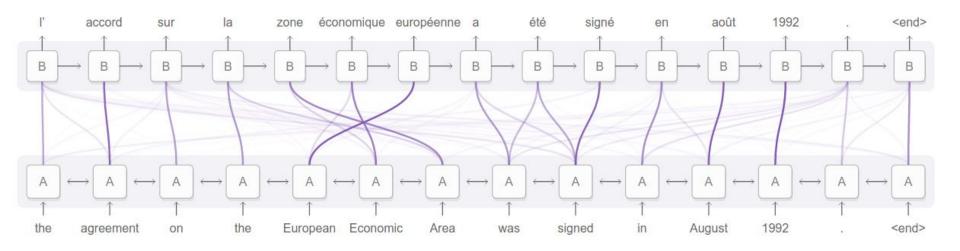
- Use bidirectional LSTM to compute hidden states h_j
- When generating translation at each timestep i compute attention score e_ij
- c_i is the context generated by attention mechanism. Use this context, with current state in LSTM to generate output
- The attention used here is soft, the whole model is differentiable

lacktriangle

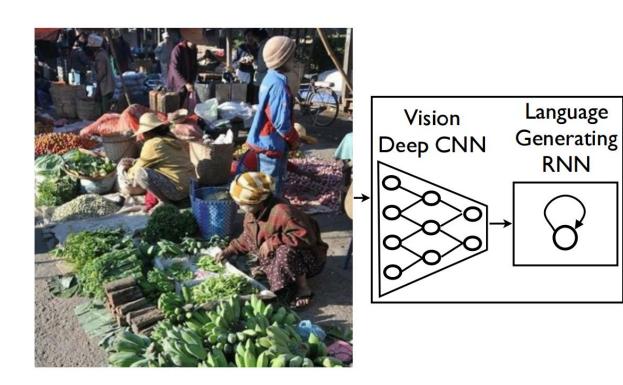
Neural machine translation by jointly learning to align and translate.

https://arxiv.org/abs/1409.0473

Machine translation with attention



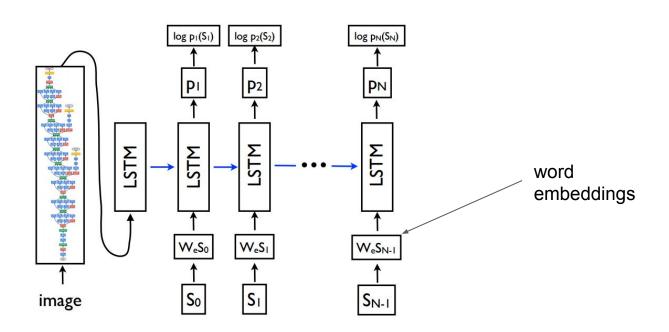
Caption generation



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

Caption generation



Show, tell

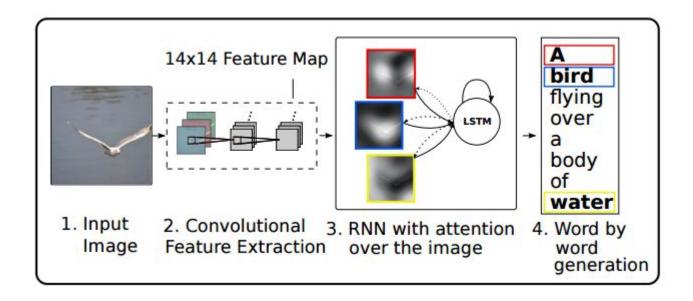
Not easy to automatically evaluate the model



Figure 5. A selection of evaluation results, grouped by human rating.

Show, attend, tell

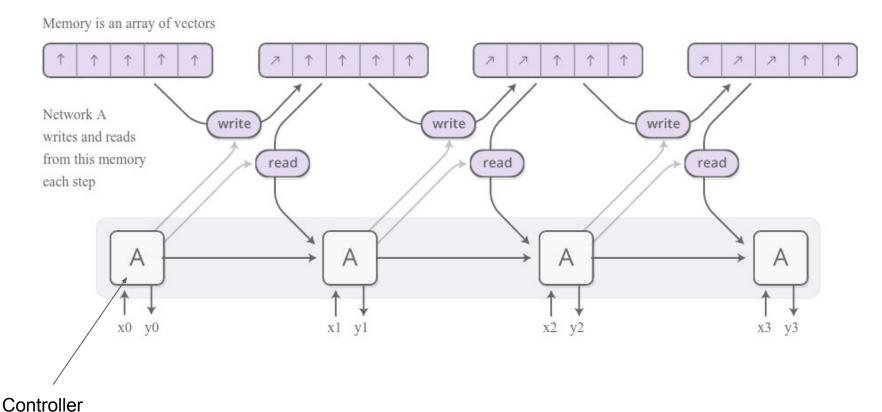
The extractor produces L vectors, each of which is a D-dimensional representation corresponding to a part of the image.



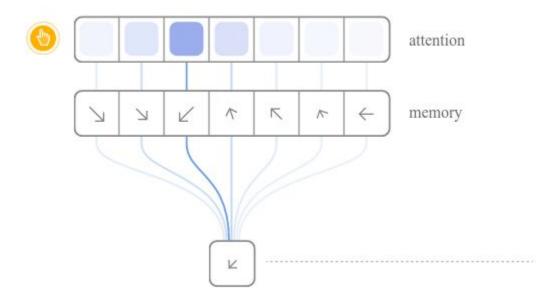
Neural Turing Machine

- Add explicit memory to the RNN
- Use attention to read and write from the memory
- Trained on simple "algorithmic" task like, copying input sequence the important results is that the NTM often

Neural Turing Machine



Reading from memory

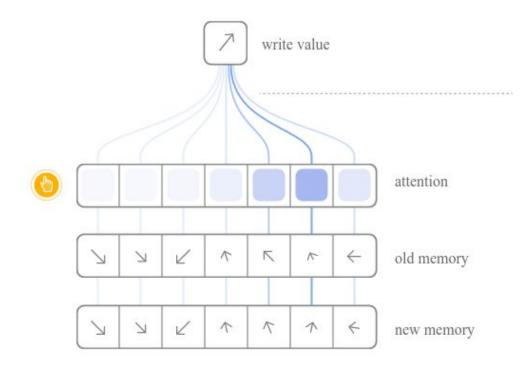


The RNN gives an attention distribution which describe how we spread out the amount we care about different memory positions

The read result is a weighted sum.

$$r \leftarrow \sum_i a_i M_i$$

Writing to memory



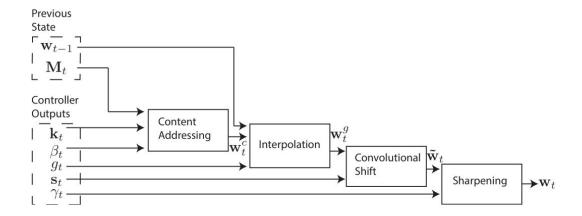
Instead of writing to one location, we write everywhere, just to different extents.

The RNN gives an attention distribution, describing how much we should change each memory position towards the write value.

$$M_i \leftarrow a_i w + (1-a_i)M_i$$

Where to read/write

- Content-based addressing (similar to attention in the translation model) compare some query vector with every cell in the memory
- Location-based addressing do some operation on the attention weighting, like shifting it to the right(this is realised using a convolution). Enables the NTM controller to iterate through memory cells
- Pretty complicated and other ways to do also possible



Generalization LSTM vs NTM

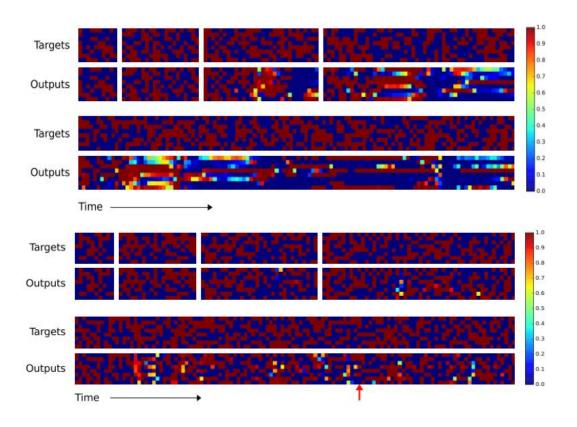
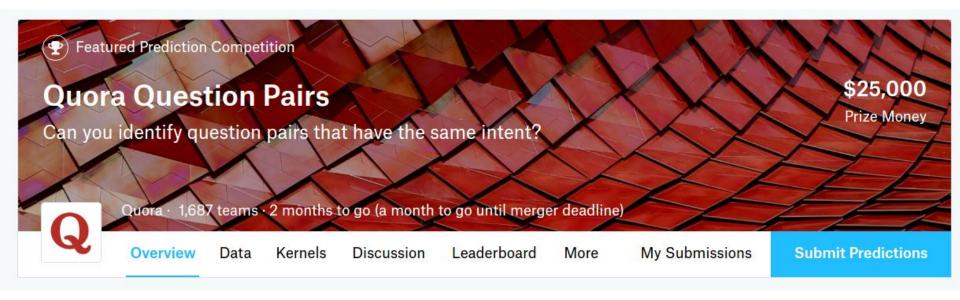


Figure 4: NTM Generalisation on the Copy Task. The four pairs of plots in the top row

NTM

- Nice explanation in http://distill.pub/2016/augmented-rnns/
- Lots of follow up papers exploring similar directions. Adding some external memory to a network.
 Sometime called neural calculators.

Quora competition



https://www.kaggle.com/c/quora-question-pairs/

https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Quora competition

same intent: "What are the best ways to lose weight?", "How can a person reduce weight?", "What are effective weight loss plans?"

not same intent: "Who is Darth Vader's father?", "Does Darth Vader know Yoda is still alive?"

- ~400k sentence pairs in the training set
- Ends on June 6, 2017

Quora competition

- What enginners at Quora tried: <u>https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning</u>
- Currently using some random forest approach
- Experimenting w DL approaches

RNN approach 1

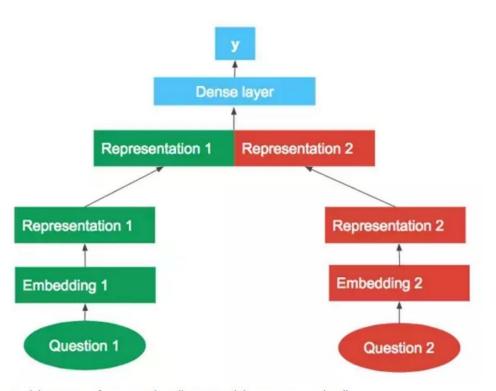


Figure 1: Architecture of approach 1, "LSTM with concatenation"

Lab instructions

```
Week 8 folder on Dropbox, mnist_rnn.py:
Task 1. Write simple RNN to recognize MNIST digits.
The image is 28x28. Flatten it to a 784 vector.
Pick some divisior d of 784, e.g. d = 28.
At each timestep the input will be d bits of the image.
Thus the sequence length will be 784 / d
You should be able to get over 93% accuracy
Write your own implementation of RNN, you can look at the one from the slide, but do not copy it blindly.
```

Task 2.

Same, but use LSTM instead of simple RNN.

What accuracy do you get.

Experiment with choosing d, compare RNN and LSTM.

Again do not use builtin Tensorflow implementation. Write your own:)

Task 3*.

Make LSTM a deep bidirectional, multilayer LSTM.

Links

- http://cs224d.stanford.edu/ Great NLP course from Stanford
- http://distill.pub/2016/augmented-rnns/
- https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning
- Show and Tell: A Neural Image Caption Generator, https://arxiv.org/abs/1411.4555
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, https://arxiv.org/abs/1502.03044
- Neural Turing Machine, https://arxiv.org/abs/1410.540
- Sequence to Sequence Learning with Neural Networks, https://arxiv.org/abs/1409.3215
- min-char-rnn https://gist.github.com/karpathy/d4dee566867f8291f086
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://www.tensorflow.org/tutorials/recurrent