```
%read the data

files = dir('Data/trump/*.json');
load('trump_chars.mat')
no_tweets = 0;

end_of_tweet='^';
tweet_cell =cell(35319,1); %without RT's
i=1;
count = 0;
unique_global = '';

for file = files'
   tweets = jsondecode(fileread(file.name))
   no_tweets = no_tweets +length(tweets);
   for tweet = tweets'

      if  (startsWith(tweet.text, 'RT') == 0)
         %no RT's

         text = tweet.text;
         text = text(regexp(text,trump_chars));
         tweet_cell{i} = text;
         count = count +1;

         unique_tweet = unique(text);
         unique_global = unique(strcat(unique_global, unique_tweet));
      end
      i = i+1;
   end
```

```
end

tweet_chars = strcat(unique_global, end_of_tweet);

char_to_ind_bonus = CreateCharToInd(trump_chars);
ind_to_char_bonus = CreateIndToChar(trump_chars);
X= EncodeInput(tweet_cell, trump_chars);

% s='picture'  % Example
% s(regexp(s,'[t,i,x,y]'))=[]

function char_to_ind = CreateCharToInd(book_chars)
char_to_ind=containers.Map('KeyType','char','ValueType','int32');
d = numel(book_chars);
for i= 1:d
   char_to_ind(book_chars(i)) = i;
end
end

function ind_to_char = CreateIndToChar(book_chars)
ind_to_char=containers.Map('KeyType','int32','ValueType','char');
d=numel(book_chars);
for i= 1:d
   ind_to_char(i) = book_chars(i);
end
end

function X = EncodeInput(tweet_cell, trump_chars)

X = cell(size(tweet_cell));
```

```
C = trump_chars;
d = numel(C);
char_to_ind=CreateCharToInd(C);

for j=1:length(tweet_cell)
   tweet=tweet_cell{j};
   length_tweet = length(tweet);
   tweet_vec = sparse(d, length_tweet);
   for k=1:length_tweet
      char = tweet(k);
      CharIndex=char_to_ind(char);
      tweet_vec(CharIndex, k) = 1;
   end
   X{j} = tweet_vec;
end
end
```

--

```
load('bonusData.mat') %load book_data, char to ind etc
X_unshuffled = X; %save a copy

m = 100;
K = length(char_to_ind_bonus); % 87

sig = 0.01;

rng(400); % 400 for life
%rng('shuffle')
```

```
%s?tt till 1 om bara vill k?ra 1 n?tverk
number_of_networks = 2;
RNN_cell = cell(number_of_networks,1);

RNN_cell{1} = CreateNetwork(400, 400, K, sig, 0.01);
RNN_cell{2} = CreateNetwork(400, 300, K, sig, 0.01);

result_cell = cell(number_of_networks,6); %1st is loss, 2nd is RNN, 3rd is hprev,
%4th is bestRNN, 5h is besthprev, 6th is cell of  tweets


no_tweets = size(X,1);
n_epochs = 10;


chars_to_synt = 200;


first_dp = X{1};



for k=1:length(RNN_cell)
   %to test multiple RNNs
   k
   RNN =  RNN_cell{k};

   %init adagrad.
   adagrad.b = zeros(size(RNN.b));
   adagrad.c = zeros(size(RNN.c));
```

```
adagrad.U = zeros(size(RNN.U));
adagrad.W = zeros(size(RNN.W));
adagrad.V = zeros(size(RNN.V));

iter=1;
save_iter=1;

%graph stuff
smooth_loss_vector = zeros(50*no_tweets,1);
smooth_loss = 0;

%to save the best RNN
min_loss = 60;
best_RNN = RNN;
best_hprev = 0;

%to save the texts
text_cell = cell(n_epochs+1,1);

m = size(RNN.W,1);
hprev = zeros(m,1);
syn_text = synt_text(RNN, hprev, first_dp(:,1), randi([40 140]));
the_text = DecodeString(syn_text, ind_to_char_bonus)
text_cell{1} = the_text;



for i=1:n_epochs
   i
   %shuffle the tweets to avoid spikes
   X = shuffle_tweets(X_unshuffled);
   for j=1:no_tweets
```

```
m = size(RNN.W,1);
hprev = zeros(m,1);
h0 = hprev;

tweet = X{j};
tweet_length = size(tweet,2);

if (isempty(tweet) == 0 && tweet_length > 20) %ngn bugg, tom matris

   %randomize seq_length
   max_seq_len = round(tweet_length/2);
   min_seq_len = max(5, round(tweet_length/10));
   seq_length = randi([min_seq_len max_seq_len]);
   e = 1;

   while (e < tweet_length-seq_length-1) %slutet?
      X_chars_one_hot = tweet(:,e:e+seq_length-1);
      Y_chars_one_hot = tweet(:,e+1:e+seq_length);
      [RNN, adagrad, loss, hprev] = MiniBatchGD(X_chars_one_hot, Y_chars_one_hot, RNN,
adagrad, hprev);

      loss;
      if smooth_loss == 0
         smooth_loss = full(loss);
      end

      smooth_loss = 0.999*smooth_loss + 0.001*(full(loss));
      smooth_loss_vector(iter) = smooth_loss;
      if (smooth_loss < min_loss)
         min_loss = smooth_loss;
         best_RNN = RNN;
```

```
                best_hprev = hprev;

            end

            h0=hprev;
            iter= iter+1;
            e = e + seq_length;
          end
        end
      end
      syn_text = synt_text(RNN, hprev, first_dp(:,1), randi([40 140]));
      the_text = DecodeString(syn_text, ind_to_char_bonus)
      text_cell{i+1} = the_text;
    end

    %remove 0s from smooth vec first
    indices = find(smooth_loss_vector ~= 0);
    final_loss_vec = smooth_loss_vector(indices);
    final_loss_vec(end)
    result_cell{k,1} = final_loss_vec;
    result_cell{k,2} = RNN;
    result_cell{k,3} = hprev;
    result_cell{k,4} = best_RNN;
    result_cell{k,5} = best_hprev;
    result_cell{k,6} = text_cell;

end



function sh_X_cell = shuffle_tweets(X_cell)
%shuffles the tweets
```

```
ny = size(X_cell,1) ;
sh_X_cell = cell(ny,1);
shuffle = randsample(1:ny,ny);

for j=1:ny
   sh_X_cell{j}=X_cell{shuffle(j)};
end

end
```

```
function [RNN, adagrad, loss, hprev] = MiniBatchGD(X_chars_one_hot, Y_chars_one_hot, RNN,
adagrad, hprev)
[loss, a, h, o, p] = ComputeLoss(X_chars_one_hot, Y_chars_one_hot, RNN, hprev);
grads = ComputeGradients(X_chars_one_hot, Y_chars_one_hot, RNN, a, h, p);
hprev = h(:,end);

%sgd

eps = 1e-9;
for f = fieldnames(grads)'
   %clipping
   grads.(f{1}) = max(min(grads.(f{1}), 5), -5);

   %adagrad
   adagrad.(f{1}) = adagrad.(f{1}) + grads.(f{1}).^2;
   RNN.(f{1}) = RNN.(f{1}) - (RNN.eta*grads.(f{1})./((adagrad.(f{1})+eps).^(0.5));
end
```

```
end



function Y = synt_text(RNN, h0, x0, n)
%0.3
%assuming n is like a timestep

K  =size(RNN.c,1);
Y = zeros(K,n);

h_t = h0;
xt = x0;



for t=1:n
   a_t = RNN.W*h_t + RNN.U*xt + RNN.b; %mx1
   h_t = tanh(a_t); %mx1
   o_t = RNN.V * h_t +RNN.c; % Kx1
   p_t = softmax(o_t);
   xnext = sample_char(p_t, K);
   Y(:,t) = xnext;
   xt = xnext;
end



end
```

```
function xnext = sample_char(p, K)
cp = cumsum(p);
a = rand;
ixs = find(cp -a >0);
ii = ixs(1);
xnext = one_hot_encode_char(ii, K);

end
```

```
function [loss, a, h, o, p] = ComputeLoss(X, Y , RNN, h0)
%h(1) is hprev
%fwd pass
%assuming n is seq_length

K = size(RNN.c,1);
m = size(RNN.b,1);
n = size(X,2);

ht = h0;

a=zeros(m,n);
h=zeros(m,n);
o=zeros(K,n);
p=zeros(K,n);

for t=1:n
   a(:,t) = RNN.W*ht + RNN.U*X(:,t) + RNN.b; %mx1
```

```
    h(:,t) = tanh(a(:,t)); %mx1
    ht = h(:,t); %needed for next iteration
    o(:,t) = RNN.V * h(:,t) +RNN.c; % Kx1
    p(:,t) = softmax(o(:,t));
end

h= [h0 h]; %insert h0 at the first place

loss = sum(-log(dot(double(Y),p)));
end



function grads = ComputeGradients(X, Y, RNN, a, h, p)
%hprev should be h(1)

m = size(RNN.U,1);
n = size(X,2);

grads.b = zeros(size(RNN.b)); %mx1
grads.c = zeros(size(RNN.c)); %kx1
grads.U = zeros(size(RNN.U)); %mxK
grads.W = zeros(size(RNN.W)); %mxm
grads.V = zeros(size(RNN.V)); %Kxm

g_batch = -(Y-p)'; %NxK, this is dL/do_t

grads.c = g_batch'*ones(n,1); %Kx1

%h is mxn
%only use h_t != 0. therefore start at 2. h(1) is h0.
```

```
grads.V = g_batch'*h(:,2:end)'; % KxN * NxM = KxM



%now grads for a & h
dL_dA = zeros(n,m); % one per time step?

dL_dh_tao = g_batch(end,:)*RNN.V; %1xK * KxM = 1xM

dL_dA_tao = dL_dh_tao*diag(1-tanh(a(:,end)).^2); %1xM * MxM = 1xM

dL_dA(end,:) = dL_dA_tao;

for i=n-1:-1:1
   dL_dh_t = g_batch(i,:)* RNN.V + dL_dA(i+1,:)*RNN.W; % 1xM + MxM = 1xM
   dL_dA(i,:) = dL_dh_t*diag(1-tanh(a(:,i)).^2); % 1xM *MxM = 1xM

end

%gradW

%dL_dA_t * h_t-1. this will work because h(1) = h0. never use h_tao?
% for t=1:n
%     grads.W = grads.W + dL_dA(t,:)'*h(:,t)'; % Mx1 * 1xM = MxM
% end

grads.W = dL_dA'*h(:,1:end-1)';

grads.U = dL_dA'*X';
grads.b = dL_dA'*ones(n,1); % Mx1

end
```

```
function one_hot_char = one_hot_encode_char(index, K)
one_hot_char = zeros(K,1);
one_hot_char(index)=1;

end



function one_hot_matrix = one_hot_encode_string(char_string, char_to_ind)
%K = 79;% hardcode 4 life
N = length(char_string);
one_hot_matrix = zeros(K,N);
for i=1:N
   index = char_to_ind(char_string(i));
   one_hot_matrix(index,i)=1;
end
end

function string = DecodeString(Y, ind_to_char)
string=[];
[~,I] = max(Y, [], 1); %argmax

for i=1:length(I)
   string = [string ind_to_char(I(i))];
end
end

function RNN = CreateNetwork(seed, m, K, sig, eta)
```

```
rng(seed); % 400 for life
RNN.eta = eta;
RNN.b = zeros(m,1);
RNN.c = zeros(K,1);
RNN.U = randn(m, K)*sig;
RNN.W = randn(m, m)*sig;
RNN.V = randn(K, m)*sig;

end
```