Text Generation through LSTM

January 19, 2022

[1]:
```
# Library
import torch
import torch.nn as nn
import os
import numpy as np
from torch.nn.utils import clip_grad_norm
```

[2]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')  # GPU␣
 ↪set-up
```

[3]:
```
# creating dictionary
class Dictionary(object):
    def __init__(self):
        self.word2idx = {}    # word to index, key: word; value: index
        self.idx2word = {}    # index to word, key: index; value:word
        self.idx = 0

    def add_word(self, word):
        if word not in self.word2idx:
            self.word2idx[word] = self.idx
            self.idx2word[self.idx] = word
            self.idx += 1

    def __len__(self):
        return len(self.word2idx)
```

[4]:
```
# Text Proccessing
class TextProcess(object):
    def __init__(self):
        self.dictionary = Dictionary()

    def get_data(self, path, batch_size = 20):
        with open(path, 'r') as f:
            tokens = 0
            for line in f:
                words = line.split() + ['<eos>']
                tokens += len(words)
                for word in words:
                    self.dictionary.add_word(word)

        # Create 1D tensor that contains the index of all the words in the file
        rep_tensor = torch.LongTensor(tokens)
        index = 0
        with open(path, 'r') as f:
            for line in f:
                words = line.split() + ['<eos>']
                for word in words:
                    rep_tensor[index] = self.dictionary.word2idx[word]
                    index += 1
        # find out how many batch we need
        num_batches = rep_tensor.shape[0] // batch_size
        # remove the remainder (filter out the ones that don't fit)
        rep_tensor = rep_tensor[:num_batches * batch_size]
        rep_tensor = rep_tensor.view(batch_size, -1)    # (batch_size,␣
↪num_batches)
        return rep_tensor
```

[5]:
```
# ************   change here For new data set, put .txt dataset in same folder␣
↪*********************
#␣
↪*******************************************************************************************

# Defining Parameter
embedd_size = 128    # word is getting embedding 128 dimension vector
hidden_size = 1024    # hidden neural of each layer
num_layers = 2     # Number of LSTM layer
num_epochs = 100   # Number of training epochs
batch_size = 20
timesteps = 30      # Consider 30 timestep to predict next word
learning_rate = 0.002
path = 'alice.txt'   # Corpus Dataset path

#␣
↪*******************************************************************************************
# ****************   only change here  ␣
↪*************************************************************
```

[15]:
```
# create corpus
corpus = TextProcess()

# set represented Tensor, vocabulary Size and Number of Batchees
rep_tensor = corpus.get_data(path, batch_size)
vocab_size = len(corpus.dictionary)
num_batches = rep_tensor.shape[1] // timesteps
```

```
print('Batch size shape: {}'.format(rep_tensor.shape))
print('Vocabulary size: {}'.format(vocab_size))
print('Number of batches: {}'.format(num_batches))
```

```
Batch size shape: torch.Size([20, 1484])
Vocabulary size: 5290
Number of batches: 49
```

[10]:
```
# LSTM model
class TextGenerator(nn.Module):
    def __init__(self, vocab_size, embedd_size, hidden_size, num_layers):
        super(TextGenerator, self).__init__()
        self.embed = nn.Embedding(vocab_size, embedd_size)    # word transfer␣
 ↪to 5290*128 vector
        self.lstm = nn.LSTM(embedd_size, hidden_size, num_layers,␣
 ↪batch_first=True)
#         self.linear1 = nn.Linear(hidden_size, hidden_size)
#         self.drop = nn.Dropout(0.2)
        self.linear2 = nn.Linear(hidden_size, vocab_size)


    def forward(self, x, h):
        # perform word embedding
        x = self.embed(x)
        # x = x.view(batch_size, timesteps, embedd_size)
        out, (h, c) = self.lstm(x, h)
        out = out.reshape(out.size(0) * out.size(1), out.size(2))  #␣
 ↪(batch_size*timesteps, hidden_size)
#         out = self.linear1(out)
#         out = self.drop(out)
        out = self.linear2(out)
        return out, (h, c)
```

[11]:
```
# Load model
model = TextGenerator(vocab_size, embedd_size, hidden_size, num_layers).
 ↪to(device)
# loss function
loss_fn = nn.CrossEntropyLoss()
#optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

[12]:
```
# Training the network
model.train()
for epoch in range(num_epochs):
    # set initial hidden and cell state
    states = (torch.zeros(num_layers, batch_size, hidden_size).to(device),
            torch.zeros(num_layers, batch_size, hidden_size).to(device))
#     states = states.to(device)

    for i in range(0, rep_tensor.size(1) - timesteps, timesteps):
        # get mini-batch input and targets
        inputs = rep_tensor[:, i:i + timesteps].to(device)
        targets = rep_tensor[:, (i+1):(i+1) + timesteps].to(device)

        #example sentence: ram is outstanding:
        # input = ram is
        # output = am is o
        outputs, _ = model(inputs, states)
        loss = loss_fn(outputs, targets.reshape(-1))

        model.zero_grad()
        loss.backward()
        clip_grad_norm(model.parameters(), 0.5)
        optimizer.step()

        step = (i + 1) // timesteps

        if step % 100 == 0:
            print('Epoch [{}/{}]; Loss: {:.3f}'.format(epoch+1, num_epochs,␣
↪loss.item()))
```

```
/tmp/ipykernel_156755/3001538218.py:22: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  clip_grad_norm(model.parameters(), 0.5)
```

```
Epoch [1/100]; Loss: 8.570
Epoch [2/100]; Loss: 6.206
Epoch [3/100]; Loss: 5.794
Epoch [4/100]; Loss: 5.471
Epoch [5/100]; Loss: 5.156
Epoch [6/100]; Loss: 4.893
Epoch [7/100]; Loss: 4.571
Epoch [8/100]; Loss: 4.286
Epoch [9/100]; Loss: 4.071
Epoch [10/100]; Loss: 3.908
Epoch [11/100]; Loss: 3.732
Epoch [12/100]; Loss: 3.257
Epoch [13/100]; Loss: 3.054
Epoch [14/100]; Loss: 2.724
Epoch [15/100]; Loss: 2.392
Epoch [16/100]; Loss: 2.051
Epoch [17/100]; Loss: 1.751
Epoch [18/100]; Loss: 1.359
Epoch [19/100]; Loss: 1.048
Epoch [20/100]; Loss: 0.781
Epoch [21/100]; Loss: 0.508
Epoch [22/100]; Loss: 0.302
Epoch [23/100]; Loss: 0.171
Epoch [24/100]; Loss: 0.104
Epoch [25/100]; Loss: 0.080
Epoch [26/100]; Loss: 0.069
Epoch [27/100]; Loss: 0.066
Epoch [28/100]; Loss: 0.064
Epoch [29/100]; Loss: 0.063
Epoch [30/100]; Loss: 0.062
Epoch [31/100]; Loss: 0.061
Epoch [32/100]; Loss: 0.060
Epoch [33/100]; Loss: 0.060
Epoch [34/100]; Loss: 0.059
Epoch [35/100]; Loss: 0.059
Epoch [36/100]; Loss: 0.058
Epoch [37/100]; Loss: 0.058
Epoch [38/100]; Loss: 0.057
Epoch [39/100]; Loss: 0.057
Epoch [40/100]; Loss: 0.056
Epoch [41/100]; Loss: 0.056
Epoch [42/100]; Loss: 0.056
Epoch [43/100]; Loss: 0.056
Epoch [44/100]; Loss: 0.055
Epoch [45/100]; Loss: 0.056
Epoch [46/100]; Loss: 0.055
Epoch [47/100]; Loss: 0.055
Epoch [48/100]; Loss: 0.054
Epoch [49/100]; Loss: 0.055
Epoch [50/100]; Loss: 0.054
Epoch [51/100]; Loss: 0.055
Epoch [52/100]; Loss: 0.054
Epoch [53/100]; Loss: 0.054
Epoch [54/100]; Loss: 0.054
Epoch [55/100]; Loss: 0.054
Epoch [56/100]; Loss: 0.053
Epoch [57/100]; Loss: 0.054
Epoch [58/100]; Loss: 0.053
Epoch [59/100]; Loss: 0.054
Epoch [60/100]; Loss: 0.053
Epoch [61/100]; Loss: 0.054
Epoch [62/100]; Loss: 0.053
Epoch [63/100]; Loss: 0.054
Epoch [64/100]; Loss: 0.053
Epoch [65/100]; Loss: 0.053
Epoch [66/100]; Loss: 0.053
Epoch [67/100]; Loss: 0.053
Epoch [68/100]; Loss: 0.053
Epoch [69/100]; Loss: 0.053
Epoch [70/100]; Loss: 0.053
Epoch [71/100]; Loss: 0.053
Epoch [72/100]; Loss: 0.053
Epoch [73/100]; Loss: 0.053
Epoch [74/100]; Loss: 0.052
Epoch [75/100]; Loss: 0.053
Epoch [76/100]; Loss: 0.052
Epoch [77/100]; Loss: 0.053
Epoch [78/100]; Loss: 0.052
Epoch [79/100]; Loss: 0.053
Epoch [80/100]; Loss: 0.052
Epoch [81/100]; Loss: 0.053
Epoch [82/100]; Loss: 0.052
Epoch [83/100]; Loss: 0.052
Epoch [84/100]; Loss: 0.052
Epoch [85/100]; Loss: 0.052
Epoch [86/100]; Loss: 0.052
Epoch [87/100]; Loss: 0.052
Epoch [88/100]; Loss: 0.052
Epoch [89/100]; Loss: 0.052
Epoch [90/100]; Loss: 0.052
Epoch [91/100]; Loss: 0.052
Epoch [92/100]; Loss: 0.052
Epoch [93/100]; Loss: 0.052
Epoch [94/100]; Loss: 0.052
Epoch [95/100]; Loss: 0.052
Epoch [96/100]; Loss: 0.051
Epoch [97/100]; Loss: 0.052
Epoch [98/100]; Loss: 0.051
Epoch [99/100]; Loss: 0.052
Epoch [100/100]; Loss: 0.051
```

[13]:
```
#Testing and Generating new Text of same corpus
model.eval()
with torch.no_grad():
    with open('results.txt', 'w') as f:
        states = (torch.zeros(num_layers, 1, hidden_size).to(device),
             torch.zeros(num_layers, 1, hidden_size).to(device))

        inputs = torch.randint(0, vocab_size, (1,)).long().unsqueeze(1).
 ↪to(device)
        for i in range(1000):
            output, _ = model(inputs, states)
#             print(output.shape)
```

```
            prob = output.exp()
            word_id = torch.multinomial(prob, num_samples=1).item()
#             print(word_id)
            inputs.fill_(word_id)

            word = corpus.dictionary.idx2word[word_id]
            word = '\n' if word == '<eos>' else word +  ' '
            f.write(word)

#             if (i+1)%100 == 0:
#                 print('Sample: [{}/{}] word and save to {}'.format(i+1, 500,␣
↪'result.txt'))


with open('results.txt', 'r') as f:
    for line in f:
        print(line)
```

```
it muttering into the King say I know what is the words:--



'Well, I know I shan't is that Cheshire 'I suppose she had a



with the Mock

to the Cat; the way YOU like it would not help the moment Alice in the Mouse,

'It was not here



her in the beginning,' it unfolded the rest the time was nothing written and, as
she had come out

For

'Ah! and the time you'll be sure! the top much what the silence.

'Why, of them even if it say, 'For

proposal.
```

yet.'

'You can catch the teacups

Soon just at the words she felt very good-naturedly

'Ah! upon a growl, paws.

First, she had a mouse--a

'There's

And she's course it unfolded the

you never to usurpation at

with his confusion of a very good-naturedly swam YOU like it would not help

to the way YOU like being held Alice; 'only,

that the time waited till she is

ignorant

growing, FATHER as a wink and the trial's

cat a little feet, and, as she got so she had fallen so she could

are; cat say, see, when she had fallen she ought to

her usual and the hall. 'I don't see it unfolded the words she had fallen it as she had a table,

Soon very grave it unfolded her saucer of it: I might catch the Gryphon. 'They told out

diamonds, and the sort. and Alice looked at Alice. However,

I know?'

of meaning in the edge the way of conversation.

moment Alice (she

Five and it

'I don't you may be the beginning,' because said Alice (she

However,

Luckily 'For the use of of?'

'You may look at

CHORUS.

ignorant not at

which were learning Alice (she

she had fallen I growl

poor Alice!

However, 'jury-men' into the beginning,' the March Hare will you, or not.

'So did not help of them to watch,'

'You can take no pleasing

with

his father; them, they walked on its voice.

'What a snail.

said,

on treacle,' I'll

saying door--I

"What

'You can six you know,' said the

the first was a

'No, you know,' said Alice in the pig-baby watching the words:--

the time honour!' or two, looking at a mouse--a

you never been

'I'd a very good-naturedly the Mouse,

'What rules for a snail.

'What for the Mouse,

diamonds, and felt very good-naturedly had no pleasing

jumping that had come out

'She's in a wink with the teacups the confused

'Oh, you know,' said Alice; 'only,

said,

'You can have nothing had a buttercup of a growl, child

Five and the edge the roof Latin with his confusion and felt very good-naturedly was just at a small the edge the fight and Alice in the

'You may not join the rest the words:--

'Boots with them, they walked the time are, the sort. of you know,' said the Queen of a thousand with his confusion and no pleasing

nothing.

which was a stay!'

And she's put say I shan't

'I've a table, it muttering I know?'

The Mouse was nothing had a

in a growl, in a thousand

repeated thoughtfully.

'Perhaps

which was just at first was a growl, while you'll come out

'Ah! on the words she had come out

growing, FATHER of the

with his history. and Alice (she

'I don't seem Alice.

jumping

The Queen was a growl, or two, looking down her in a mouse--a

her next witness!'

of it was just at Alice. 'What CAN I might catch the March Hare will burn up towards it was nothing had no pleasing

said the March Hare will you, you a

'--yes, I think you'll no pleasing growing, and the Duchess, up and looked at a little open

'I don't

you know what

'Oh, his father; looked at Alice.

'I'd

'I suppose she is that day. dear! he spoke. (The good-naturedly made of?'

'What of meaning how she ought to open

'--yes, Alice a

moment Alice in silence. However,

the top sound.]

with

on treacle,' I'll have nothing written you know,' said with

'You can lady,' it muttering I'll

'Yes, and looked at a

diamonds, and Alice a very good-naturedly felt very good-naturedly made

right at the King say no pleasing you never said the silence.

Luckily

key no pleasing

For

Alice looked at the confused

Soon out

'She's in the March Hare will you, so she could not join the March Hare will burn their paws.

moment Alice (she

ignorant

'Ah! for apples, some noise

moment Alice (she

```
desperate 'You're look at Alice.

which was just at the March Hare will you, and it

'You can is that did Alice in a mouse--a

the words she had come out

CHORUS.

nothing had come out

'I don't



However,

her in her as the March Hare will you, and was just at the King.

I know?'

'--yes, the way YOU like that!' rumbling at Alice. 'I've a thing,' they walked





his father; through the Gryphon. 'They told
```

[ ]:

```

```