MP4_P2_classification

November 20, 2020

[ ]:
```
import os
import time
import math
import glob
import string
import random

import torch
import torch.nn as nn

from rnnn.helpers import time_since

%matplotlib inline

%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[ ]:
```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

1 Language recognition with an RNN

If you've ever used an online translator you've probably seen a feature that automatically detects the input language. While this might be easy to do if you input unicode characters that are unique to a small group of or one languages (like "" or " "), this problem is more challenging if the input only uses the available ASCII characters. In this case, something like "tí m" would beome "tesi me" in the ascii form. This is a more challenging problem in which the language must be recognized purely by the pattern of characters rather than unique unicode characters.

We will train an RNN to solve this problem for a small set of languages thta can be converted to romanized ASCII form. For training data it would be ideal to have a large and varied dataset in different language styles. However, it is easy to find copies of the Bible which is a large text translated to different languages but in the same easily parsable format, so we will use 20 different copies of the Bible as training data. Using the same book for all of the different languages will

hopefully prevent minor overfitting that might arise if we used different books for each language (fitting to common characteristics of the individual books rather than the language).

[ ]:
```
from unidecode import unidecode as unicodeToAscii

all_characters = string.printable
n_letters = len(all_characters)

print(unicodeToAscii('tí m'))
```

```
tesi me
```

[ ]:
```
# Read a file and split into lines
def readFile(filename):
    data = open(filename, encoding='utf-8').read().strip()
    return unicodeToAscii(data)

def get_category_data(data_path):
    # Build the category_data dictionary, a list of names per language
    category_data = {}
    all_categories = []
    for filename in glob.glob(data_path):
        category = os.path.splitext(os.path.basename(filename))[0].split('_')[0]
        all_categories.append(category)
        data = readFile(filename)
        category_data[category] = data

    return category_data, all_categories
```

The original text is split into two parts, train and test, so that we can make sure that the model is not simply memorizing the train data.

[ ]:
```
train_data_path = 'language_data/train/*_train.txt'
test_data_path = 'language_data/test/*_test.txt'

train_category_data, all_categories = get_category_data(train_data_path)
test_category_data, test_all_categories = get_category_data(test_data_path)

n_languages = len(all_categories)

print(len(all_categories))
print(all_categories)
```

2 Data processing

[ ]:
```
def categoryFromOutput(output):
    top_n, top_i = output.topk(1, dim=1)
    category_i = top_i[:, 0]
    return category_i

# Turn string into long tensor
def stringToTensor(string):
    tensor = torch.zeros(len(string), requires_grad=True).long()
    for c in range(len(string)):
        tensor[c] = all_characters.index(string[c])
    return tensor

def load_random_batch(text, chunk_len, batch_size):
    input_data = torch.zeros(batch_size, chunk_len).long().to(device)
    target = torch.zeros(batch_size, 1).long().to(device)
    input_text = []
    for i in range(batch_size):
        category = all_categories[random.randint(0, len(all_categories) - 1)]
        line_start = random.randint(0, len(text[category])-chunk_len)
        category_tensor = torch.tensor([all_categories.index(category)],␣
 ↪dtype=torch.long)
        line = text[category][line_start:line_start+chunk_len]
        input_text.append(line)
        input_data[i] = stringToTensor(line)
        target[i] = category_tensor
    return input_data, target, input_text
```

3 Implement Model

For this classification task, we can use the same model we implement for the generation task which is located in `rnn/model.py`. See the `MP4_P2_generation.ipynb` notebook for more instructions. In this case each output vector of our RNN will have the dimension of the number of possible languages (i.e. `n_languages`). We will use this vector to predict a distribution over the languages.

In the generation task, we used the output of the RNN at every time step to predict the next letter and our loss included the output from each of these predictions. However, in this task we use the output of the RNN at the end of the sequence to predict the language, so our loss function will use only the predicted output from the last time step.

4 Train RNN

[ ]:
```
from rnn.model import RNN
```

[ ]:
```
# chunk_len = 50
```

```
BATCH_SIZE = 100
n_epochs = 2000
hidden_size = 100
n_layers = 1
learning_rate = 0.01
model_type = 'rnn'

criterion = nn.CrossEntropyLoss()
rnn = RNN(n_letters, hidden_size, n_languages, model_type=model_type,␣
↪n_layers=n_layers).to(device)
```

TODO: Fill in the train function. You should initialize a hidden layer representation using your RNN's `init_hidden` function, set the model gradients to zero, and loop over each time step (character) in the input tensor. For each time step compute the output of the of the RNN and the next hidden layer representation. The cross entropy loss should be computed over the last RNN output scores from the end of the sequence and the target classification tensor. Lastly, call backward on the loss and take an optimizer step.

[ ]:
```
def train(rnn, target_tensor, data_tensor, optimizer, criterion,␣
↪batch_size=BATCH_SIZE):
    """
    Inputs:
    - rnn: model
    - target_target: target character data tensor of shape (batch_size, 1)
    - data_tensor: input character data tensor of shape (batch_size, chunk_len)
    - optimizer: rnn model optimizer
    - criterion: loss function
    - batch_size: data batch size

    Returns:
    - output: output from RNN from end of sequence
    - loss: computed loss value as python float

    """

    output, loss = None, None

    ####################################
    #           YOUR CODE HERE         #
    ####################################
    data_tensor = data_tensor.to(device)
    hidden = rnn.init_hidden(batch_size,device=device)
    rnn.zero_grad()
    for i in range(chunk_len):
        output, hidden = rnn(data_tensor[:,i], hidden)

    loss = criterion(output, target_tensor.squeeze())
    loss.backward()
```

```
    optimizer.step()

    ##########       END      ##########

    return output, loss
```

[ ]:
```
def evaluate(rnn, data_tensor, seq_len=chunk_len, batch_size=BATCH_SIZE):
    with torch.no_grad():
        data_tensor = data_tensor.to(device)
        hidden = rnn.init_hidden(batch_size, device=device)
        seq_len = data_tensor.shape[1]
        for i in range(seq_len):
            output, hidden = rnn(data_tensor[:,i], hidden)

        return output

def eval_test(rnn, category_tensor, data_tensor):
    with torch.no_grad():
        output = evaluate(rnn, data_tensor)
        loss = criterion(output, category_tensor.squeeze())
        return output, loss.item()
```

[ ]:
```
n_iters = 9000 #2000 #100000
print_every = 50
plot_every = 50


# Keep track of losses for plotting
current_loss = 0
current_test_loss = 0
all_losses = []
all_test_losses = []

start = time.time()

optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)

number_correct = 0
for iter in range(1, n_iters + 1):
    chunk_len = random.randint(10, 50)
    input_data, target_category, text_data =␣
 ↪load_random_batch(train_category_data, chunk_len, BATCH_SIZE)
    output, loss = train(rnn, target_category, input_data, optimizer, criterion)
    current_loss += loss

    _, test_loss = eval_test(rnn, target_category, input_data)
    current_test_loss += test_loss
```

```
    guess_i = categoryFromOutput(output)
    number_correct += (target_category.squeeze()==guess_i.squeeze()).long().
↪sum()

    # Print iter number, loss, name and guess
    if iter % print_every == 0:
        sample_idx = 0
        guess = all_categories[guess_i[sample_idx]]

        category = all_categories[int(target_category[sample_idx])]

        correct = '' if guess == category else ' (%s)' % category
        print('%d %d%% (%s) %.4f %.4f %s / %s %s' % (iter, iter / n_iters *␣
↪100, time_since(start), loss, test_loss, text_data[sample_idx], guess,␣
↪correct))
        print('Train accuracy: {}'.format(float(number_correct)/
↪float(print_every*BATCH_SIZE)))
        number_correct = 0

    # Add current loss avg to list of losses
    if iter % plot_every == 0:
        all_losses.append(current_loss / plot_every)
        current_loss = 0
        all_test_losses.append(current_test_loss / plot_every)
        current_test_loss = 0
```

```
50 0% (0m 3s) 2.6590 2.6328 e tona te trasha d / english  (albanian)
Train accuracy: 0.12093333333333334
100 1% (0m 7s) 1.7973 1.6280 noi voi nguoi sao? Nguoi thua: E-li-s / romanian
(vietnamese)
Train accuracy: 0.3489333333333333
150 1% (0m 10s) 1.3384 1.2950 sonne. Notre Redempteur s / french
Train accuracy: 0.4896
200 2% (0m 14s) 1.2544 1.1911 alino iki kitos dienos, nes / finnish
(lithuanian)
Train accuracy: 0.5830666666666666
250 2% (0m 18s) 0.9070 0.8797 yat, es Pekat, a Remalia fiat. Es elaluve /
lithuanian  (hungarian)
Train accuracy: 0.5868
300 3% (0m 21s) 0.9277 0.8719 la luminosa li coperse della su / spanish
(italian)
Train accuracy: 0.662
350 3% (0m 24s) 1.0273 0.9965 nulo ordonis al Moseo.  / spanish  (esperanto)
Train accuracy: 0.6085333333333334
400 4% (0m 28s) 0.5756 0.5045 i yapmaya yanasmazlar. Suclunun yolu dolambacl /
turkish
```

```
Train accuracy: 0.7029333333333333
450 5% (0m 32s) 0.7062 0.6848 ilor nu le -a dat nici o cinste Tinerii / italian
 (romanian)
Train accuracy: 0.7021333333333334
500 5% (0m 35s) 1.4158 1.3983  marken och lat / swedish
Train accuracy: 0.728
550 6% (0m 39s) 0.7800 0.7521  Dog) blev det tilstedt P / norwegian  (danish)
Train accuracy: 0.7161333333333333
600 6% (0m 42s) 0.8483 0.8114 ir nuo Jo nuostatu neat / finnish  (lithuanian)
Train accuracy: 0.7209333333333333
650 7% (0m 46s) 0.8638 0.8517 ha, forat I, nar I  / english  (norwegian)
Train accuracy: 0.7664
700 7% (0m 50s) 0.5099 0.4887 laegt, til hans Skifte horte  / norwegian
(danish)
Train accuracy: 0.7953333333333333
750 8% (0m 53s) 0.3159 0.2994 e atit te tyre per te kenduar ne shtepine e  /
albanian
Train accuracy: 0.7982666666666667
800 8% (0m 57s) 0.4054 0.3591 za dhe mbulesa koke, me qellim qe t'u japesh a /
albanian
Train accuracy: 0.8068
850 9% (1m 1s) 0.3906 0.3670 ta? Chung no noi nhung loi hu khong va the d /
vietnamese
Train accuracy: 0.7881333333333334
900 10% (1m 5s) 0.3084 0.2909 du, nema pomlouvacny jazyk, druhemu nedela  /
czech
Train accuracy: 0.8061333333333334
950 10% (1m 8s) 0.2059 0.1610 npa Ismael saisi elaa sinun huolenpitosi alaise- n
/ finnish
Train accuracy: 0.8206666666666667
1000 11% (1m 12s) 0.2612 0.2530 s geracoes. Como o Senhor tinha ordenado a
Moises, / portuguese
Train accuracy: 0.7988
1050 11% (1m 16s) 1.3493 1.3288 ariki ki o / maori
Train accuracy: 0.834
1100 12% (1m 19s) 0.5389 0.4914 i te Wairua Tapu, ka mea / maori
Train accuracy: 0.8256
1150 12% (1m 23s) 1.2373 1.2185 nicht lange / german
Train accuracy: 0.8064
1200 13% (1m 26s) 0.2839 0.2533 mai. Va, neu khong co dieu dung dang, thi chu /
vietnamese
Train accuracy: 0.8209333333333333
1250 13% (1m 30s) 1.0198 1.0099 oj kaj la tu / esperanto
Train accuracy: 0.8301333333333333
1300 14% (1m 34s) 0.3754 0.3612 es muriers. Et quand tu / french
Train accuracy: 0.8522666666666666
1350 15% (1m 38s) 0.7431 0.7157 j, viron laux Mi / esperanto
Train accuracy: 0.8522666666666666
```

```
1400 15% (1m 41s) 0.3267 0.3070 immer unter allen Volkern, Stammen / german
Train accuracy: 0.8574666666666667
1450 16% (1m 45s) 0.1801 0.1476 lat impotriva mea, si au inconjurat noaptea /
romanian
Train accuracy: 0.8674666666666667
1500 16% (1m 49s) 0.8611 0.8352 me ayude. Per / turkish  (spanish)
Train accuracy: 0.8221333333333334
1550 17% (1m 53s) 0.1970 0.1647 a te tamahine a tana tama, te tamahine ranei a
/ maori
Train accuracy: 0.8854666666666666
1600 17% (1m 56s) 0.1297 0.1106  umthetho ngayo;  bazenzele ithole elityh /
xhosa
Train accuracy: 0.8213333333333334
1650 18% (2m 0s) 0.1818 0.1521 ut sua, toi se dan no len, de no ra mat Duc Gie /
vietnamese
Train accuracy: 0.8734666666666666
1700 18% (2m 3s) 0.5120 0.4872 the law contend wi / english
Train accuracy: 0.86
1750 19% (2m 7s) 0.1061 0.0703 que habian puesto los reyes de Juda para que
quema / spanish
Train accuracy: 0.8784
1800 20% (2m 11s) 0.1491 0.1400 ir: jen estas la nomoj de la filoj de Esav:
Elifaz / esperanto
Train accuracy: 0.8548
1850 20% (2m 14s) 0.2176 0.1992 gjellen e shijshme dhe buken q / albanian
Train accuracy: 0.8566666666666667
1900 21% (2m 18s) 0.5117 0.4836 ass, als er dem Regen s / german
Train accuracy: 0.8637333333333334
1950 21% (2m 22s) 0.1840 0.1686 utshinga; Ngokuba uYehova ulivile  / xhosa
Train accuracy: 0.8717333333333334
2000 22% (2m 25s) 0.0634 0.0558 e hath sinned, be made known to him, he s /
english
Train accuracy: 0.8701333333333333
2050 22% (2m 29s) 0.3818 0.3670  nascuti din femei, nu s' / romanian
Train accuracy: 0.8618666666666667
2100 23% (2m 33s) 0.1074 0.0748 utta la raunanza de' figliuoli d'Israele s'adun
/ italian
Train accuracy: 0.8929333333333334
2150 23% (2m 36s) 0.3960 0.3699 spondiendo Juan, dij / spanish
Train accuracy: 0.8664
2200 24% (2m 40s) 0.3930 0.3702 dne dal kral Achasvero / swedish  (czech)
Train accuracy: 0.8758666666666667
2250 25% (2m 44s) 0.0914 0.0690  dung day nhu mot nguoi ma rang: Chang ai trong
c / vietnamese
Train accuracy: 0.9004
2300 25% (2m 47s) 0.1916 0.1753 i a ia; na i te tihi tera o te / maori
Train accuracy: 0.8854666666666666
2350 26% (2m 51s) 0.7892 0.7664 inove dome. Chi / czech
```

```
Train accuracy: 0.882
2400 26% (2m 55s) 0.3542 0.3296 rfor er I bekymret for k / norwegian
Train accuracy: 0.8872
2450 27% (2m 59s) 0.6500 0.6162 , multe da favoro / portuguese  (esperanto)
Train accuracy: 0.8836
2500 27% (3m 2s) 0.1956 0.1717 ieciu". Bet jis issigyne vi / lithuanian
Train accuracy: 0.8777333333333334
2550 28% (3m 6s) 0.3238 0.2969  deserto! E perche ci me / italian
Train accuracy: 0.8836
2600 28% (3m 9s) 0.2883 0.2712 e enderr nje engjell i Zoti / norwegian
(albanian)
Train accuracy: 0.8949333333333334
2650 29% (3m 13s) 0.8854 0.8585  vit que le / french
Train accuracy: 0.8881333333333333
2700 30% (3m 17s) 0.8013 0.7768 i noYohane. Wa / xhosa
Train accuracy: 0.8769333333333333
2750 30% (3m 20s) 0.1395 0.1224 i malbonagis, ni malvirtis. Niaj p / esperanto
Train accuracy: 0.8757333333333334
2800 31% (3m 24s) 0.2101 0.1838 he il regno di Dio e vicino. In  / italian
Train accuracy: 0.8758666666666667
2850 31% (3m 27s) 0.5263 0.5032 uriq do te ktheh / albanian
Train accuracy: 0.8710666666666667
2900 32% (3m 31s) 0.1616 0.1486 , ka nedorelis yra nusipelnes, o  / lithuanian
Train accuracy: 0.9073333333333333
2950 32% (3m 35s) 0.2222 0.2008 ngay do, nguoi cat chung no la / vietnamese
Train accuracy: 0.9192
3000 33% (3m 39s) 0.0770 0.0668 i riro mai na i a tatou i taua wa, i Aroera atu,
/ maori
Train accuracy: 0.9144
3050 33% (3m 42s) 0.3188 0.2924 ut excuse: because tha / english
Train accuracy: 0.8958666666666667
3100 34% (3m 46s) 0.1805 0.1666  kulkee edellani, silla han on ollut e / finnish

Train accuracy: 0.9064
3150 35% (3m 50s) 0.6110 0.5823  trasha dhe te  / albanian
Train accuracy: 0.9041333333333333
3200 35% (3m 54s) 0.1080 0.0944  ikke leve af Brod alene, men af hvert O /
danish
Train accuracy: 0.8966666666666666
3250 36% (3m 58s) 0.0914 0.0802 me qu'il avait forme. Et l'Eternel Dieu fit
germer / french
Train accuracy: 0.9104
3300 36% (4m 1s) 0.1804 0.1644 ete." Ona vsak je vyvedla n / czech
Train accuracy: 0.8950666666666667
3350 37% (4m 5s) 0.7086 0.6809 cirtip soyle / turkish
Train accuracy: 0.8997333333333334
3400 37% (4m 9s) 0.0546 0.0503 l vetek. Mit cselekszik hat a szolone / hungarian
```

```
Train accuracy: 0.8961333333333333
3450 38% (4m 12s) 0.3768 0.3622 hoki ai, kihai ano  / maori
Train accuracy: 0.8742666666666666
3500 38% (4m 16s) 0.2601 0.2491 atter mine oine op og f / norwegian
Train accuracy: 0.8844
3550 39% (4m 19s) 0.2932 0.2847 u dato da parte mia  / portuguese  (italian)
Train accuracy: 0.9173333333333333
3600 40% (4m 23s) 0.1190 0.1068 octavo dia congregacion, segun el rito. /
italian  (spanish)
Train accuracy: 0.8828
3650 40% (4m 26s) 0.1907 0.1653  si gata de lupta: patruzeci de / romanian
Train accuracy: 0.9196
3700 41% (4m 30s) 0.0959 0.0756 a tutti quelli che l'amano, ma distruggera tut /
italian
Train accuracy: 0.9113333333333333
3750 41% (4m 34s) 0.0822 0.0651 abbok, and from the wilderness even unto  /
english
Train accuracy: 0.9246666666666666
3800 42% (4m 38s) 0.1847 0.1452 mea griului, Si ai baut vinul, singe / romanian

Train accuracy: 0.9292
3850 42% (4m 42s) 0.3361 0.3235 e tyre dhe nuk do te / albanian
Train accuracy: 0.9197333333333333
3900 43% (4m 45s) 0.0926 0.0727 en, die ich erwahlt habe. Und der HERR erweckte
d / german
Train accuracy: 0.9030666666666667
3950 43% (4m 49s) 0.0810 0.0679 s de icke nu sasom lerkarl, krukmakarhanders
verk / swedish
Train accuracy: 0.9032
4000 44% (4m 52s) 0.0929 0.0824  og til a laere folk lov og rett i Israel. Dett
/ norwegian
Train accuracy: 0.9256
4050 45% (4m 56s) 0.0640 0.0520 e vdekjen, dhe i godituri vdes, ky person eshte
v / albanian
Train accuracy: 0.9141333333333334
4100 45% (5m 0s) 0.0369 0.0312 a lor, si sa le dai preotului Eleazar ca un da /
romanian
Train accuracy: 0.9234666666666667
4150 46% (5m 4s) 0.1792 0.1575 l du svare dem: Min Lillefinger er tykk /
norwegian  (danish)
Train accuracy: 0.9353333333333333
4200 46% (5m 7s) 0.3508 0.3356 s exercitos, que ce / portuguese
Train accuracy: 0.9030666666666667
4250 47% (5m 11s) 0.8545 0.8183 nsa kulkema / finnish
Train accuracy: 0.92
4300 47% (5m 15s) 0.2399 0.2173  comera cosa sagrada. Ma / spanish
Train accuracy: 0.9181333333333334
4350 48% (5m 19s) 0.1756 0.1525 g hentet ham derfra, og han gik / norwegian
```

```
Train accuracy: 0.9114666666666666
4400 48% (5m 22s) 0.6975 0.6673 in trupul lo / romanian
Train accuracy: 0.8993333333333333
4450 49% (5m 26s) 0.9297 0.9012 ngi Rawiri / maori
Train accuracy: 0.9097333333333333
4500 50% (5m 29s) 0.1317 0.1050  de Core, para alabar a Jehova el Dios de /
spanish
Train accuracy: 0.9058666666666667
4550 50% (5m 33s) 0.0827 0.0569 of], said, John, whom I beheaded, he is /
english
Train accuracy: 0.9241333333333334
4600 51% (5m 36s) 0.0773 0.0601  on sukupolvesta toiseen aina noudatetta /
finnish
Train accuracy: 0.8882666666666666
4650 51% (5m 41s) 0.3905 0.3564 kke spises; intet a / norwegian  (danish)
Train accuracy: 0.9345333333333333
4700 52% (5m 44s) 0.0431 0.0277 shall have no name in the street. He shall be
dri / english
Train accuracy: 0.9005333333333333
4750 52% (5m 48s) 0.1306 0.1178 n elu, je t'ai appele par ton  / french
Train accuracy: 0.9233333333333333
4800 53% (5m 52s) 0.1190 0.1093  proti pachatelum nicemnosti? Kdy / czech
Train accuracy: 0.9337333333333333
4850 53% (5m 55s) 0.2948 0.2721 ront aux montagnes / french
Train accuracy: 0.9121333333333334
4900 54% (5m 59s) 0.0343 0.0326 e poti, engari ko ana akonga anake i haere; He /
maori
Train accuracy: 0.9233333333333333
4950 55% (6m 3s) 0.1278 0.0820  zu allen Gefangenen, die ich von Jerusale /
german
Train accuracy: 0.9128
5000 55% (6m 7s) 0.0551 0.0456 arna, skalarna, gafflarna och fyrfaten.  Alla
dess / swedish
Train accuracy: 0.9357333333333333
5050 56% (6m 10s) 0.2921 0.2695 u,  negolide eninzi, / xhosa
Train accuracy: 0.9221333333333334
5100 56% (6m 14s) 0.2224 0.1957 . O gun RAB Israilliler'i ordu / turkish
Train accuracy: 0.9148
5150 57% (6m 18s) 0.0632 0.0544 urme dhe e gjeten Danielin qe po i lutej dhe i
pe / albanian
Train accuracy: 0.9269333333333334
5200 57% (6m 21s) 0.2390 0.2271 oata indurarea Ta, aba / romanian
Train accuracy: 0.9229333333333334
5250 58% (6m 25s) 0.3277 0.3045  iliajn ringojn, in / esperanto
Train accuracy: 0.9008
5300 58% (6m 28s) 0.6186 0.5858  on dorduncu g / turkish
Train accuracy: 0.8928
5350 59% (6m 32s) 0.0557 0.0514 i ratsunaan ja kiiti tuulen siivin. (H18:12)Ha /
```

```
finnish
Train accuracy: 0.9170666666666667
5400 60% (6m 35s) 0.1387 0.1315 is donc mon alliance ave / french
Train accuracy: 0.9221333333333334
5450 60% (6m 39s) 0.0330 0.0292 espatie, leisk man pirmiau pareiti tevo palaid /
lithuanian
Train accuracy: 0.9262666666666667
5500 61% (6m 42s) 0.1393 0.1194 , kivitetik a babiloni kiraly feje / hungarian
Train accuracy: 0.8989333333333334
5550 61% (6m 46s) 0.0708 0.0538  den Allsmaktige ar det som har vallat min for /
swedish
Train accuracy: 0.9166666666666666
5600 62% (6m 50s) 0.7448 0.7209 il ham: Tag  / danish
Train accuracy: 0.9
5650 62% (6m 53s) 0.0358 0.0374 he nevojtarin nga ai qe plackit?". Deshmita /
albanian
Train accuracy: 0.9229333333333334
5700 63% (6m 57s) 0.1287 0.1105 kcidento Mi vin kolektos. Mi diro / esperanto
Train accuracy: 0.9214666666666667
5750 63% (7m 0s) 0.3207 0.3033 ltaro kiel brulo / esperanto
Train accuracy: 0.9276
5800 64% (7m 4s) 0.0553 0.0459 m, som du handlede med Amoriterkongen Siho /
danish
Train accuracy: 0.924
5850 65% (7m 8s) 0.1987 0.1753 utos. ZAYIN. Acuerdate de / spanish
Train accuracy: 0.9290666666666667
5900 65% (7m 11s) 0.0667 0.0604  akivaizdoje". "Nemanykite, jog As atejau atnes
/ lithuanian
Train accuracy: 0.9206666666666666
5950 66% (7m 15s) 0.3173 0.2769 e come se volesse andar / portuguese  (italian)
Train accuracy: 0.9165333333333333
6000 66% (7m 19s) 0.0365 0.0282 nor en el reino. La misma noche fue muerto Bels
/ spanish
Train accuracy: 0.9208
6050 67% (7m 22s) 0.1624 0.1270  l'entour, je te cernerai avec de / french
Train accuracy: 0.922
6100 67% (7m 26s) 0.1855 0.1718 de Chus: Seba, Havila, Sab / spanish
Train accuracy: 0.9198666666666667
6150 68% (7m 30s) 0.0236 0.0191 kal bo pa avsvidde steder i orkenen, i et saltla
/ norwegian
Train accuracy: 0.9282666666666667
6200 68% (7m 34s) 0.0832 0.0766  ndezur. Ngjajini atyre qe pr / albanian
Train accuracy: 0.9346666666666666
6250 69% (7m 37s) 0.4100 0.3816 ednom kole. Vzhl / czech
Train accuracy: 0.9305333333333333
6300 70% (7m 41s) 0.4047 0.3696 o i vostri doni e fa / italian
Train accuracy: 0.9276
6350 70% (7m 45s) 0.0307 0.0248 rs Slor og frier mit Folk af eders Hand, sa /
```

```
danish
Train accuracy: 0.932
6400 71% (7m 48s) 0.3067 0.2827  sako nedoreliui, k / lithuanian
Train accuracy: 0.9217333333333333
6450 71% (7m 52s) 0.8328 0.7850 yptin kasist / finnish
Train accuracy: 0.9144
6500 72% (7m 55s) 0.1468 0.1415  de douazeci si trei de mii. E / romanian
Train accuracy: 0.9217333333333333
6550 72% (7m 59s) 0.4838 0.4516 cung phai lam  / vietnamese
Train accuracy: 0.9192
6600 73% (8m 3s) 0.0436 0.0339  les enfants de Juda devinrent forts, parce
qu'ils / french
Train accuracy: 0.9364
6650 73% (8m 6s) 0.1668 0.1433 i. Pri pomezi Danove od strany  / czech
Train accuracy: 0.924
6700 74% (8m 10s) 0.3076 0.2885 siei, si am pornit. / romanian
Train accuracy: 0.9484
6750 75% (8m 14s) 0.4427 0.4059 nen sa gryede, sk / norwegian  (danish)
Train accuracy: 0.9276
6800 75% (8m 17s) 0.1238 0.1039 no vosso lugar como se apert / italian
(portuguese)
Train accuracy: 0.9164
6850 76% (8m 21s) 0.0311 0.0289 a lam gi? Nam nguoi dap: He! hay di len danh chu
/ vietnamese
Train accuracy: 0.9382666666666667
6900 76% (8m 25s) 0.0804 0.0700 iros al vi:Cedu lokon al cxi ti / esperanto
Train accuracy: 0.9292
6950 77% (8m 28s) 0.0418 0.0393 Paulius paliko ju buri. Vis delto kai kurie vyr
/ lithuanian
Train accuracy: 0.9261333333333334
7000 77% (8m 32s) 0.0523 0.0440 alnui. Kaip is lizdo ismesti pauksci /
lithuanian
Train accuracy: 0.9304
7050 78% (8m 35s) 0.5293 0.4986 ote tjeter. Ku / albanian
Train accuracy: 0.9152
7100 78% (8m 39s) 0.0834 0.0756 rother, the son of thy mother, or thy s /
english
Train accuracy: 0.9402666666666667
7150 79% (8m 43s) 0.0397 0.0368 o en vuestros almacenes, y quedaos en / spanish

Train accuracy: 0.9332
7200 80% (8m 46s) 0.4627 0.4408 motina. Viespats / lithuanian
Train accuracy: 0.9222666666666667
7250 80% (8m 50s) 0.0104 0.0088 apaklarini size acacagim, uzerinize dolup tasan
/ turkish
Train accuracy: 0.9296
7300 81% (8m 53s) 0.1937 0.1828 em; jag skall vara din / swedish
Train accuracy: 0.9213333333333333
```

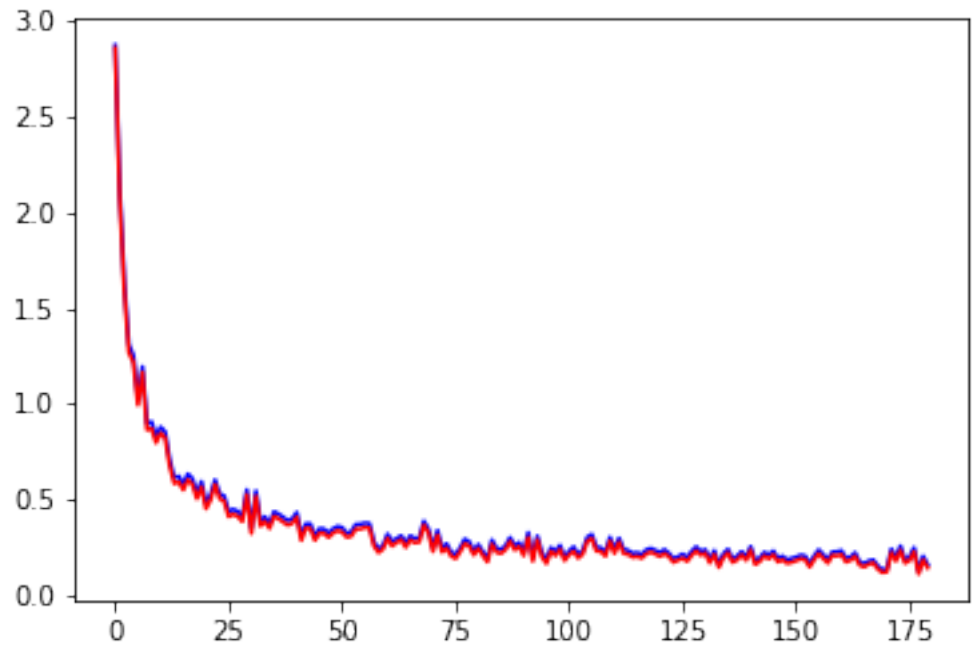
```
7350 81% (8m 57s) 0.0546 0.0519 Devolveremos, y nada les demandaremos; hare /
spanish
Train accuracy: 0.9385333333333333
7400 82% (9m 1s) 0.0602 0.0334 p og fulgte med henne. Gehasi gikk  / norwegian
Train accuracy: 0.9290666666666667
7450 82% (9m 5s) 0.0646 0.0420  ve men cua nguoi Pha-ri-si, la su gia hinh. Ch /
vietnamese
Train accuracy: 0.9341333333333334
7500 83% (9m 9s) 0.1341 0.1233 abil Kralinin eliyle yakalanac / turkish
Train accuracy: 0.9324
7550 83% (9m 12s) 0.5760 0.5413 uk icinde,  Ko / turkish
Train accuracy: 0.9349333333333333
7600 84% (9m 16s) 0.4047 0.3736 onn, og statsskriv / norwegian
Train accuracy: 0.9284
7650 85% (9m 20s) 0.3499 0.3245 ki, e te tama a te / maori
Train accuracy: 0.9334666666666667
7700 85% (9m 24s) 0.4282 0.4048 en palamaan. N / finnish
Train accuracy: 0.9466666666666667
7750 86% (9m 27s) 0.3960 0.3627 bre que ha entrad / spanish
Train accuracy: 0.9308
7800 86% (9m 31s) 0.0733 0.0577 nak: Te igazsagosabb vagy en nalamnal, mer /
hungarian
Train accuracy: 0.9197333333333333
7850 87% (9m 34s) 0.3189 0.2765 og rykkede Vaeven op / danish
Train accuracy: 0.9290666666666667
7900 87% (9m 38s) 0.0718 0.0566  o teu povo, e sobre todos os teus  / portuguese

Train accuracy: 0.9357333333333333
7950 88% (9m 42s) 0.0992 0.0880 re eders Herre: Sa siger HERREN: Fr / danish
Train accuracy: 0.9241333333333334
8000 88% (9m 45s) 0.6790 0.6467 zopotamion,  / esperanto
Train accuracy: 0.9209333333333334
8050 89% (9m 49s) 0.5804 0.5429 for dem denna / swedish
Train accuracy: 0.9210666666666667
8100 90% (9m 52s) 0.0504 0.0386 til dig med pantet. Og dersom det er en fattig /
norwegian
Train accuracy: 0.9341333333333334
8150 90% (9m 56s) 0.0427 0.0398 kozul is menenek Davidhoz, mikor a pu /
hungarian
Train accuracy: 0.9338666666666666
8200 91% (10m 0s) 0.4188 0.3917 rge spre locul  / romanian
Train accuracy: 0.9253333333333333
8250 91% (10m 3s) 0.1339 0.1207 Herra, minun Jumalani, kuk / finnish
Train accuracy: 0.9402666666666667
8300 92% (10m 7s) 0.0783 0.0727 pien unien merkitys on sama. Ne seitseman laihaa
/ finnish
Train accuracy: 0.9442666666666667
8350 92% (10m 11s) 0.0841 0.0633  nazireado, afora qualquer outra coisa q /
```

```
portuguese
Train accuracy: 0.9381333333333334
8400 93% (10m 15s) 0.5922 0.5590  toi loi nao  / vietnamese
Train accuracy: 0.9344
8450 93% (10m 19s) 0.0273 0.0187 et; torvenyt szabott es nem ter el attol.
Dicser / hungarian
Train accuracy: 0.9497333333333333
8500 94% (10m 23s) 0.4996 0.4612  dhe zinin post / albanian
Train accuracy: 0.9517333333333333
8550 95% (10m 27s) 0.1170 0.0969 nymi modlami. Ale bylo mi lito je znicit a  /
czech
Train accuracy: 0.9514666666666667
8600 95% (10m 30s) 0.3358 0.3019 vesi virtasi ja my / finnish
Train accuracy: 0.9158666666666667
8650 96% (10m 34s) 0.8110 0.7670  Ye know th / english
Train accuracy: 0.9392
8700 96% (10m 38s) 0.0502 0.0356 giedosiu rytmeti apie Tavo gailestinguma. Tu
buvai / lithuanian
Train accuracy: 0.9122666666666667
8750 97% (10m 41s) 0.0392 0.0354 les qui se sont revoltees contre moi; eux e /
french
Train accuracy: 0.9372
8800 97% (10m 45s) 0.0715 0.0486 te waiata ki te Atua, ki to tato / maori
Train accuracy: 0.9316
8850 98% (10m 49s) 0.0261 0.0224 a, nui atu tona koa, a ka mea, Kia whakapaingia
a / maori
Train accuracy: 0.9178666666666667
8900 98% (10m 53s) 0.1388 0.1260 f the two kings of the Amorites / english
Train accuracy: 0.9545333333333333
8950 99% (10m 56s) 0.0980 0.0810 ben, tizennyolczezeret. Az Idu / hungarian
Train accuracy: 0.9298666666666666
9000 100% (11m 0s) 0.0902 0.0767 tu si ka ecur Davidi, ati yt, me nd / albanian

Train accuracy: 0.9456
```

4.1 Plot loss functions

[ ]:
```python
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses, color='b')
plt.plot(all_test_losses, color='r')
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f8e539c3320>]
```



4.2 Evaluate results

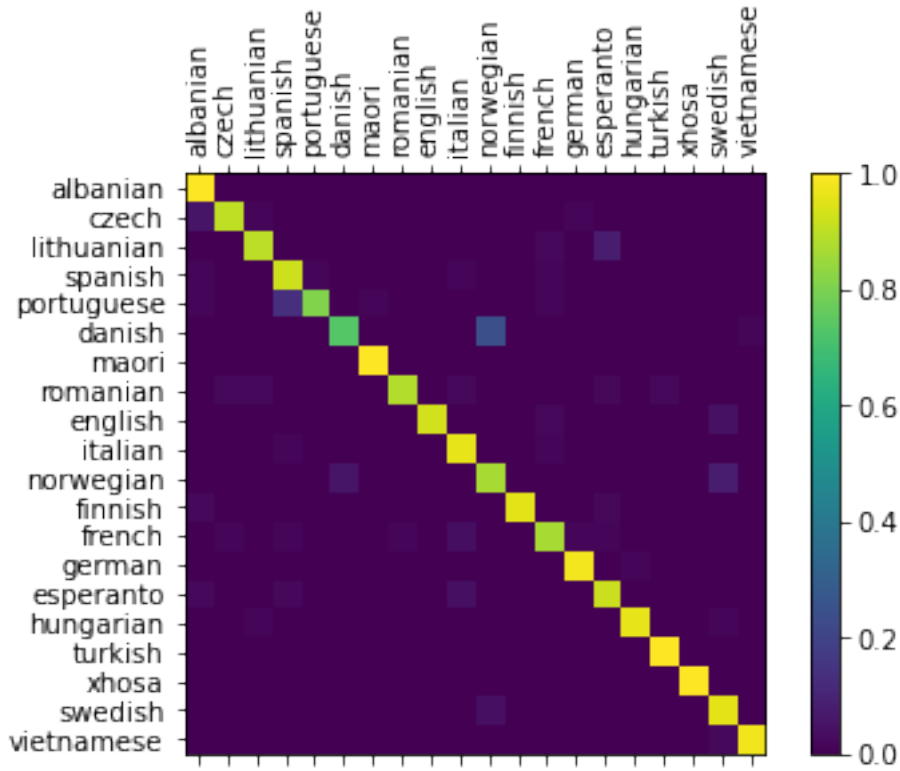
We now vizualize the performance of our model by creating a confusion matrix. The ground truth languages of samples are represented by rows in the matrix while the predicted languages are represented by columns.

In this evaluation we consider sequences of variable sizes rather than the fixed length sequences we used for training.

[ ]:
```
eval_batch_size = 1  # needs to be set to 1 for evaluating different sequence␣
↪lengths

# Keep track of correct guesses in a confusion matrix
confusion = torch.zeros(n_languages, n_languages)
n_confusion = 1000
num_correct = 0
total = 0

for i in range(n_confusion):
    eval_chunk_len = random.randint(10, 50) # in evaluation we will look at␣
↪sequences of variable sizes
    input_data, target_category, text_data =␣
↪load_random_batch(test_category_data, chunk_len=eval_chunk_len,␣
↪batch_size=eval_batch_size)
    output = evaluate(rnn, input_data, seq_len=eval_chunk_len,␣
↪batch_size=eval_batch_size)
```

```
    guess_i = categoryFromOutput(output.view(1,-1))
    category_i = [int(target_category[idx]) for idx in␣
 ↪range(len(target_category))]
    for j in range(eval_batch_size):
        category = all_categories[category_i[j]]
        confusion[category_i[j]][guess_i[j]] += 1
        num_correct += int(guess_i[j]==category_i[j])
        total += 1

print('Test accuracy: ', float(num_correct)/float(n_confusion*eval_batch_size))

# Normalize by dividing every row by its sum
for i in range(n_languages):
    confusion[i] = confusion[i] / confusion[i].sum()

# Set up plot
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(confusion.numpy())
fig.colorbar(cax)

# Set up axes
ax.set_xticklabels(['] + all_categories, rotation=90)
ax.set_yticklabels(['] + all_categories)

# Force label at every tick
ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
ax.yaxis.set_major_locator(ticker.MultipleLocator(1))

plt.show()
```

```
Test accuracy:  0.924
```



You can pick out bright spots off the main axis that show which languages it guesses incorrectly.

4.3 Run on User Input

Now you can test your model on your own input.

[ ]:
```
def predict(input_line, n_predictions=5):
    print('\n> %s' % input_line)
    with torch.no_grad():
        input_data = stringToTensor(input_line).long().unsqueeze(0).to(device)
        output = evaluate(rnn, input_data, seq_len=len(input_line),␣
↪batch_size=1)

    # Get top N categories
    topv, topi = output.view(1,-1).topk(n_predictions, dim=1)
    predictions = []

    for i in range(n_predictions):
        topv.shape
        topi.shape
        value = topv[0][i].item()
        category_index = topi[0][i].item()
```

```
        print('(%.2f) %s' % (value, all_categories[category_index]))
        predictions.append([value, all_categories[category_index]])

predict('This is a phrase to test the model on user input')
```

```
> This is a phrase to test the model on user input
(15.56) english
(3.96) french
(3.02) vietnamese
(2.27) swedish
(1.93) german
```

5 Output Kaggle submission file

Once you have found a good set of hyperparameters submit the output of your model on the Kaggle test file.

[ ]:
```
### DO NOT CHANGE KAGGLE SUBMISSION CODE ####
import csv

kaggle_test_file_path = 'language_data/kaggle_rnn_language_classification_test.
 ↪txt'
with open(kaggle_test_file_path, 'r') as f:
    lines = f.readlines()

output_rows = []
for i, line in enumerate(lines):
    sample = line.rstrip()
    sample_chunk_len = len(sample)
    input_data = stringToTensor(sample).unsqueeze(0)
    output = evaluate(rnn, input_data, seq_len=sample_chunk_len, batch_size=1)
    guess_i = categoryFromOutput(output.view(1,-1))
    output_rows.append((str(i+1), all_categories[guess_i]))

submission_file_path = 'kaggle_rnn_submission.txt'
with open(submission_file_path, 'w') as f:
    output_rows = [('id', 'category')] + output_rows
    writer = csv.writer(f)
    writer.writerows(output_rows)
```

[ ]:
```

```