2.PyTorch Text-CNN with GloVe(F1-Score)

November 4, 2020

Acknowledgments: https://www.kaggle.com/ziliwang/pytorch-text-cnn

Acknowledgments: I would also like to thank Youwen Wang for helping me out through out for technicalities and places where I was stuck.

[ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

[ ]:
```
import pandas as pd
import numpy as np
import torch
from torch import nn
# from sklearn.metrics import fbeta_score, make_scorer
# f2score=make_scorer(fbeta_score, beta=2)
from sklearn.metrics import f1_score
import torchtext
from tqdm import tqdm, tqdm_notebook
from nltk import word_tokenize
import random
from torch import optim
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

[ ]:
```
True
```

[ ]:
```
!pip install -q kaggle
```

[ ]:
```
from google.colab import files
```

[ ]:
```
files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle.json
```

```
[ ]: {'kaggle.json':
     b'{"username":"nabhsanjaymehtautd","key":"ee0e2e2e8b50d345f23e44404b090088"}'}
```

```
[ ]: !mkdir ~/.kaggle/
```

```
[ ]: !cp kaggle.json ~/.kaggle/
```

```
[ ]: !chmod 60 ~/.kaggle/kaggle.json
```

```
[ ]: !kaggle competitions download -c transferlearning-dl-spring2020
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix
this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Warning: Looks like you're using an outdated API Version, please consider
updating (server 1.5.9 / client 1.5.4)
Downloading sample_submission.csv to /content
  0% 0.00/34.2k [00:00<?, ?B/s]
100% 34.2k/34.2k [00:00<00:00, 30.1MB/s]
Downloading train.csv.zip to /content
  0% 0.00/527k [00:00<?, ?B/s]
100% 527k/527k [00:00<00:00, 74.4MB/s]
Downloading test.csv to /content
  0% 0.00/506k [00:00<?, ?B/s]
100% 506k/506k [00:00<00:00, 70.1MB/s]
```

```
[ ]: !unzip train.csv.zip -d train
```

```
Archive:  train.csv.zip
  inflating: train/train.csv
```

```
[ ]: data = pd.read_csv('/content/train/train.csv', encoding = "ISO-8859-1")
     testdata = pd.read_csv('/content/test.csv', encoding="ISO-8859-1")
```

```
[ ]: data.head()
```

```
[ ]:       id                                                text  target
     0  86426  @USER She should ask a few native Americans wh…       1
     1  16820  Amazon is investigating Chinese employees who …       0
     2  62688  @USER Someone should'veTaken" this piece of sh…       1
     3  43605  @USER @USER Obama wanted liberals &amp;amp; illega…       0
     4  97670                  @USER Liberals are all Kookoo !!!       1
```

```
[ ]: text = torchtext.data.Field(lower=True, batch_first=True,␣
      ↪tokenize=word_tokenize, fix_length=70)
     id = torchtext.data.Field()
     target = torchtext.data.Field(sequential=False, use_vocab=False, is_target=True)
```

[ ]:
```
train = torchtext.data.TabularDataset(path='/content/train/train.csv',
                                      format='csv',
                                      fields={'text': ('text',text),
                                              'target': ('target',target)})

test = torchtext.data.TabularDataset(path='/content/test.csv',
                                     format='csv',
                                     fields={'id': ('id', id),
                                             'text': ('text', text)})
```

[ ]:
```
#Build Vocabulary
text.build_vocab(train, test, min_freq=3)
id.build_vocab(test)
```

[ ]:
```
!wget http://nlp.stanford.edu/data/glove.6B.zip
```

```
--2020-11-04 05:04:51--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)… 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80…
connected.
HTTP request sent, awaiting response… 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2020-11-04 05:04:51--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443…
connected.
HTTP request sent, awaiting response… 301 Moved Permanently
Location: http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2020-11-04 05:04:51--  http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)… 171.64.64.22
Connecting to downloads.cs.stanford.edu
(downloads.cs.stanford.edu)|171.64.64.22|:80… connected.
HTTP request sent, awaiting response… 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  2.18MB/s    in 6m 26s

2020-11-04 05:11:17 (2.13 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

[ ]:
```
!unzip glove*.zip
```

```
Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
```

[ ]:
```
!ls
!pwd
```

```
gdrive              glove.6B.300d.txt  kaggle.json            test.csv
glove.6B.100d.txt   glove.6B.50d.txt   sample_data            train
glove.6B.200d.txt   glove.6B.zip       sample_submission.csv  train.csv.zip
/content
```

[ ]:
```
glove = glove = torchtext.vocab.GloVe(name='840B', dim=300)#torchtext.vocab.
↪Vectors('/content/glove.6B.200d.txt')
text.vocab.set_vectors(glove.stoi, glove.vectors, dim=300)
```

```
.vector_cache/glove.840B.300d.zip: 2.18GB [16:53, 2.15MB/s]
100%|      | 2195480/2196017 [04:11<00:00, 9036.46it/s]
```

[ ]:
```
class TextCNN(nn.Module):

    def __init__(self, lm, padding_idx, static=True, kernel_num=128,␣
↪fixed_length=50, kernel_size=[2, 5, 10], dropout=0.2):
        super(TextCNN, self).__init__()
        self.dropout = nn.Dropout(p=dropout)
        self.embedding = nn.Embedding.from_pretrained(lm)
        if static:
            self.embedding.weight.requires_grad = False
        self.embedding.padding_idx = padding_idx
        self.conv = nn.ModuleList([nn.Conv2d(1, kernel_num, (i, self.embedding.
↪embedding_dim)) for i in kernel_size])
        self.maxpools = [nn.MaxPool2d((fixed_length+1-i,1)) for i in␣
↪kernel_size]
        self.fc = nn.Linear(len(kernel_size)*kernel_num, 1)

    def forward(self, input):
        x = self.embedding(input).unsqueeze(1)  # B X Ci X H X W
        x = [self.maxpools[i](torch.tanh(cov(x))).squeeze(3).squeeze(2) for i,␣
↪cov in enumerate(self.conv)]  # B X Kn
        x = torch.cat(x, dim=1)  # B X Kn * len(Kz)
        y = self.fc(self.dropout(x))
        return y
```

[ ]:
```
def search_best_f1(true, pred):
    tmp = [0,0,0] # idx, cur, max
    delta = 0
    for tmp[0] in np.arange(0.1, 0.501, 0.01):
        tmp[1] = f1_score(true, np.array(pred)>tmp[0])
        if tmp[1] > tmp[2]:
            delta = tmp[0]
            tmp[2] = tmp[1]
```

```
    return tmp[2], delta
```

[ ]:
```
def training(epoch, model, loss_func, optimizer, train_iter):
    e = 0

    while e < epoch:
        train_iter.init_epoch()
        losses, preds, true = [], [], []
        for train_batch in tqdm(list(iter(train_iter)), 'epcoh {} training'.
 ↪format(e)):
            model.train()
            x = train_batch.text.cuda()
            y = train_batch.target.type(torch.Tensor).cuda()
            true.append(train_batch.target.numpy())
            model.zero_grad()
            pred = model.forward(x).view(-1)
            loss = loss_function(pred, y)
            preds.append(torch.sigmoid(pred).cpu().data.numpy())
            losses.append(loss.cpu().data.numpy())
            loss.backward()
#             clip_grad_norm_(model.parameters(), 2)
            optimizer.step()
        train_f1, alpha_train = search_best_f1([j for i in true for j in i], [j␣
 ↪for i in preds for j in i])
        print('epcoh {:02} - train_loss {:.4f} - train f1 {:.4f} - delta {:.
 ↪4f}'.format(
                            e, np.mean(losses), train_f1, alpha_train))

        e += 1
    return alpha_train
```

[ ]:
```
random.seed(1234)
batch_size = 512
train_iter = torchtext.data.BucketIterator(dataset=train,
                                           batch_size=batch_size,
                                           shuffle=True,
                                           sort=False)
```

[ ]:
```
def init_network(model, method='xavier', exclude='embedding', seed=123):
    torch.manual_seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed_all(seed)
    for name, w in model.named_parameters():
        if not exclude in name:
            if 'weight' in name:
                if method is 'xavier':
                    nn.init.xavier_normal_(w)
```

```
                elif method is 'kaiming':
                    nn.init.kaiming_normal_(w)
                else:
                    nn.init.normal_(w)
            elif 'bias' in name:
                nn.init.constant_(w, 0.0)
            else:
                pass
```

[ ]:
```
def print_model(model, ignore='embedding'):
    total = 0
    for name, w in model.named_parameters():
        if not ignore or ignore not in name:
            total += w.nelement()
            print('{} : {}  {} parameters'.format(name, w.shape, w.nelement()))
    print('-------'*4)
    print('Total {} parameters'.format(total))
```

[ ]:
```
text.fix_length = 70
model = TextCNN(text.vocab.vectors,
                padding_idx=text.vocab.stoi[text.pad_token],
                kernel_size=[1, 2, 3, 5], kernel_num=128,
                static=False, fixed_length=text.fix_length,
                dropout=0.1).cuda()

init_network(model)

optimizer = optim.Adam(params=model.parameters(), lr=1e-3)

loss_function = nn.MSELoss()

print_model(model, ignore=None)
```

```
embedding.weight : torch.Size([6484, 300])  1945200 parameters
conv.0.weight : torch.Size([128, 1, 1, 300])  38400 parameters
conv.0.bias : torch.Size([128])  128 parameters
conv.1.weight : torch.Size([128, 1, 2, 300])  76800 parameters
conv.1.bias : torch.Size([128])  128 parameters
conv.2.weight : torch.Size([128, 1, 3, 300])  115200 parameters
conv.2.bias : torch.Size([128])  128 parameters
conv.3.weight : torch.Size([128, 1, 5, 300])  192000 parameters
conv.3.bias : torch.Size([128])  128 parameters
fc.weight : torch.Size([1, 512])  512 parameters
fc.bias : torch.Size([1])  1 parameters
----------------------------
Total 2368625 parameters
```

[ ]:
```
alpha = training(3, model, loss_function, optimizer, train_iter)
```

```
epcoh 0 training:   0%|          | 0/19 [00:00<?, ?it/s]
epcoh 0 training:   5%|          | 1/19 [00:00<00:07,  2.47it/s]
epcoh 0 training:  32%|        | 6/19 [00:00<00:03,  3.44it/s]
epcoh 0 training:  58%|       | 11/19 [00:00<00:01,  4.77it/s]
epcoh 0 training: 100%|     | 19/19 [00:00<00:00, 23.37it/s]

epcoh 1 training:   0%|          | 0/19 [00:00<?, ?it/s]
epcoh 00 - train_loss 0.4193 - train f1 0.5021 - delta 0.4700

epcoh 1 training:  26%|         | 5/19 [00:00<00:00, 48.56it/s]
epcoh 1 training:  58%|       | 11/19 [00:00<00:00, 48.85it/s]
epcoh 1 training: 100%|     | 19/19 [00:00<00:00, 46.88it/s]

epcoh 2 training:   0%|          | 0/19 [00:00<?, ?it/s]
epcoh 01 - train_loss 0.1632 - train f1 0.5250 - delta 0.5000

epcoh 2 training:  26%|         | 5/19 [00:00<00:00, 45.40it/s]
epcoh 2 training:  53%|       | 10/19 [00:00<00:00, 45.00it/s]
epcoh 2 training: 100%|     | 19/19 [00:00<00:00, 45.87it/s]
epcoh 02 - train_loss 0.1384 - train f1 0.5279 - delta 0.5000
```

[ ]:
```
def predict(model, test_list):
    pred = []
    with torch.no_grad():
        for test_batch in test_list:
            model.eval()
            x = test_batch.text.cuda()
            pred += torch.sigmoid(model.forward(x).view(-1)).cpu().data.numpy().
 ↪tolist()
    return pred
```

[ ]:
```
test_list = list(torchtext.data.BucketIterator(dataset=test,
                                    batch_size=batch_size,
                                    sort=False,
                                    train=False))
```

[ ]:
```
preds = predict(model, test_list)
sub = pd.DataFrame()
sub['id'] = [id.vocab.itos[j] for i in test_list for j in i.id.view(-1).numpy()]
sub['prediction'] = (preds > alpha).astype(int)
sub.head()
```

[ ]:

	id	prediction
0	90194	1
1	77444	1
2	13384	1
3	54920	1
4	56117	1

[ ]:
```
pd.DataFrame({'Id': sub.id, 'Target': sub.prediction}).to_csv('submission1.
↪csv', index =False)
```

[ ]:
```
pwd
```

[ ]:
```
'/content'
```