Convolutinal Neural Network for Twitter Sentiment Analysis

As Kim suggested in his 2014 paper, Convolutional neural networks for sentence classification, a simple CNN with a single convolutional layer can produce excellent results for sentence classification. Therefore, we decided to test their effectiveness for our task. In this notebook, we will train our own word embeddings on the training dataset. The embedding layer is then followed by a convolution layer.

In [1]:

```
import pandas as pd
import re
from keras.preprocessing.text import Tokenizer
import torch
import torch.nn as nn
import torch.nn.functional as F
from sklearn.model_selection import train_test_split
import glob, os
```

```
Using TensorFlow backend.
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarni
ng: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorflow/python/framework/dtypes.py:517: FutureWarni
ng: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorflow/python/framework/dtypes.py:518: FutureWarni
ng: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorflow/python/framework/dtypes.py:519: FutureWarni
ng: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
```

```
site-packages/tensorflow/python/framework/dtypes.py:520: FutureWarni
ng: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarni
ng: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: Futu
reWarning: Passing (type, 1) or '1type' as a synonym of type is depr
ecated; in a future version of numpy, it will be understood as (type
, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: Futu
reWarning: Passing (type, 1) or '1type' as a synonym of type is depr
ecated; in a future version of numpy, it will be understood as (type
, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: Futu
reWarning: Passing (type, 1) or '1type' as a synonym of type is depr
ecated; in a future version of numpy, it will be understood as (type
, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: Futu
reWarning: Passing (type, 1) or '1type' as a synonym of type is depr
ecated; in a future version of numpy, it will be understood as (type
, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: Futu
reWarning: Passing (type, 1) or '1type' as a synonym of type is depr
ecated; in a future version of numpy, it will be understood as (type
, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/shivaomrani/opt/anaconda3/envs/neural_networks/lib/python3.7/
site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: Futu
reWarning: Passing (type, 1) or '1type' as a synonym of type is depr
ecated; in a future version of numpy, it will be understood as (type
, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:

```
os.chdir("data/")
```

Helper functions for reading training and test datasets into dataframes and for pre-processing tweets.

In [3]:

```
from nltk.stem.wordnet import WordNetLemmatizer


# inspired from https://www.kaggle.com/rahulvv/bidirectional-lstm-glove200d

def read_tsv(file_path):
    df = pd.read_table(file_path)
    return df


def remove_urls(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'',text)


def remove_html(text):
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)


def split_text(text):
    text = text.split()
    return text

def lower(text):
    text = [word.lower() for word in text]
    return str(text)

def remove_punct(text):
    text = ''.join([char for char in text if char not in string.punctuation])
    text = re.sub('[0-9]+', '', str(text))
    return text

def remove_stopwords(text):
    pattern = re.compile(r'\b('+r'|'.join(stopwords.words('english')) + r')\b\s*')
    text = pattern.sub(' ', text)
    return text

lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    text = lemmatizer.lemmatize(text)
    return text
```

```
def clean_tweet(text):
    t0 = remove_urls(text)
    t1 = remove_html(t0)
    t2 = split_text(t1)
    t3 = lower(t2)
    t4 = remove_punct(t3)
    t5 = remove_stopwords(t4)
    t6 = lemmatize_words(t5)
    return t6
```

In [4]:

```
tweet_df = pd.DataFrame(columns=['tweet', 'sentiment','NA'])
df_test = pd.DataFrame(columns=['tweet', 'sentiment','NA'])

for file in glob.glob("*.tsv"):
        if 'final_test' in file:
            df_test_cur = read_tsv(file)
            df_test = pd.concat([df_test, df_test_cur])
        else:
            df_train_cur = read_tsv(file)
            tweet_df = pd.concat([tweet_df, df_train_cur])
```

We will print some of the training and test datasets tweets to get an idea of how they look before pre-processing.

In [5]:

```
print(tweet_df[['tweet', 'sentiment']] )
```

```
                                                  tweet sentiment
0     05 Beat it - Michael Jackson - Thriller (25th ...   neutral
1     Jay Z joins Instagram with nostalgic tribute t...  positive
2     Michael Jackson: Bad 25th Anniversary Edition ...   neutral
3     I liked a @YouTube video http://t.co/AaR3pjp2P...  positive
4     18th anniv of Princess Diana's death. I still ...  positive
...                                                 ...       ...
1137                     Maybe it was - his - fantasy ?  positive
1138  It was ok , but they always just seem so nervo...  negative
1139  It is streamable from YepRoc -- matter of fact...  positive
1140  comment telling me who you are , or how you fo...  positive
1141  im on myspace ... ill try and find you and add...   neutral

[53368 rows x 2 columns]
```

```
In [6]: print(df_test[['tweet', 'sentiment']] )
```

```
                                                   tweet sentiment
0      #ArianaGrande Ari By Ariana Grande 80% Full ht...   neutral
1      Ariana Grande KIIS FM Yours Truly CD listening...  positive
2      Ariana Grande White House Easter Egg Roll in W...  positive
3      #CD #Musics Ariana Grande Sweet Like Candy 3.4...  positive
4      SIDE TO SIDE 😘 @arianagrande #sidetoside #aria...   neutral
...                                                  ...       ...
11901  @dansen17 update: Zac Efron kissing a puppy ht...  positive
11902  #zac efron sex pic skins michelle sex https://...   neutral
11903  First Look at Neighbors 2 with Zac Efron Shirt...   neutral
11904  zac efron poses nude #lovely libra porn https:...   neutral
11905  #Fashion #Style The Paperboy (NEW Blu-ray Disc...   neutral

[11906 rows x 2 columns]
```

We will then pre-process the tweets and select a subset of the tweets so that they work with our specified number of batches.

In [7]:

```
import string
from nltk.corpus import stopwords

clean_tweets = []
for tweet in tweet_df['tweet']:
    clean_tweets.append(clean_tweet(tweet))

clean_tweets_1 = clean_tweets[:53000]



clean_tweets_test = []
for tweet in df_test['tweet']:
    clean_tweets_test.append(clean_tweet(tweet))

clean_tweets_test_1 = clean_tweets_test[:11900]


print("original length of train tweets: ", len(clean_tweets))
print("original length of test tweets: ", len(clean_tweets_test))

print("custom length of train tweets: ", len(clean_tweets_1))
print("custom length of test tweets: ", len(clean_tweets_test_1))
```

```
original length of train tweets:  53368
original length of test tweets:  11906
custom length of train tweets:  53000
custom length of test tweets:  11900
```

In [8]:
```
print(clean_tweets_1[:10])
print(clean_tweets_test_1[:10])
```

```
[' beat  michael jackson  thriller th anniversary edition hd', 'jay
z joins instagram  nostalgic tribute  michael jackson jay z apparent
ly joined instagram  saturday  ', 'michael jackson bad th anniversar
y edition picture vinyl  unique picture disc vinyl includes  origina
l ', ' liked  youtube video one direction singing man   mirror  mich
ael jackson  atlanta ga june ', 'th anniv  princess dianas death  st
ill want  believe   living   private island away   public  michael j
ackson', 'oridaganjazz  st time  heard michael jackson sing   honolu
lu hawaii   restaurant  radio   abc    loved   ', 'michael jackson ap
peared  saturday    th place   top  miamis trends trndnl', '  old en
ough  remember michael jackson attending  grammys  brooke shields  w
ebster sat   lap   show', 'etbowser  u enjoy  nd rate michael jackso
n bit honest ques like  cant feel face song  god   obvious  want mj
', ' weeknd   closest thing  may get  michael jackson   long timeesp
ecially since  damn near mimics everything']
['arianagrande ari  ariana grande  full singer actress', 'ariana gra
nde kiis fm  truly cd listening party  burbank arianagrande', 'arian
a grande white house easter egg roll  washington arianagrande', 'cd
musics ariana grande sweet like candy  oz  ml sealed  box  authenic
new', 'side  side 😘 arianagrande sidetoside arianagrande musically
comunidadgay lgbt🌈 lotb…', 'hairspray live previews   macys thanksg
iving day parade arianagrande televisionbc', 'lindsaylohan  ‘feelin
g thankful’  blasting arianagrande  wearing ‘toomuch…', ' hate    lo
ve  songs dammit arianagrande', 'ariana grande 【right  ft big sean】
アリアナ arianagrande', ' one would  prefer  listen    whole day 😍🤘🏼
could never choose arianagrande intoyou sidetoside songs poll']
```

Then, we will convert our data to integer sequences (where each tweet is represented as a sequence of numbers and the numbers are the index of the words in the whole vocabulary). This prepared our data to be fed into the embedding layer.

In [9]:
```
# converting tweets to integer sequences
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean_tweets_1)
tweet_sequences = tokenizer.texts_to_sequences(clean_tweets_1)
word_index = tokenizer.word_index
print('Found %s unique words in train tweets.' % len(word_index))


# converting tweets to integer sequences
tweet_sequences_test = tokenizer.texts_to_sequences(clean_tweets_test_
1)
```

```
Found 66956 unique words in train tweets.
```

We split the training data into train and validation so we have a better idea of how our model is doing on the validation set in each epoch.

```
In [10]: #preparing train lables
         tweet_df.loc[tweet_df.sentiment == "positive", "sentiment"] = 2
         tweet_df.loc[tweet_df.sentiment == "neutral", "sentiment"] = 1
         tweet_df.loc[tweet_df.sentiment == "negative", "sentiment"] = 0

         labels = tweet_df["sentiment"].tolist()
         labels = [ int(x) for x in labels ]

         labels_1 = labels[:53000]

         # Split the train set into train and validation set
         x_train, x_val, y_train, y_val = train_test_split(tweet_sequences, lab
         els_1, test_size = 0.2, random_state=17)

         #preparing test labels
         df_test.loc[df_test.sentiment == "positive", "sentiment"] = 2
         df_test.loc[df_test.sentiment == "neutral", "sentiment"] = 1
         df_test.loc[df_test.sentiment == "negative", "sentiment"] = 0

         labels_test = df_test["sentiment"].tolist()
         labels_test = [ int(x) for x in labels_test ]

         labels_test_1 = labels_test[:11900]

         print("length of x train ", len(x_train))
         print("length of y train ", len(y_train))
         print("length of x val: ", len(x_val))
         print("length of y val: ", len(y_val))
```

```
length of x train  42400
length of y train  42400
length of x val:  10600
length of y val:  10600
```

To further prepare our data to be fed into the embedding layer, we convert the list of integer sequences to tensors and we will pad 0s and the end of each sequence so that our samples are all of equal size.

In [11]:

```
# Training batch size
batch_size = 100

# Put into tensors
x_train = [torch.tensor(x) for x in x_train]
X_train = nn.utils.rnn.pad_sequence(x_train, batch_first=True, padding_value=0).long()
X_train = X_train.view(-1, batch_size, X_train.shape[1])

x_val = [torch.tensor(x) for x in x_val]
X_val = nn.utils.rnn.pad_sequence(x_val, batch_first=True, padding_value=0).long()
X_val = X_val.view(-1, batch_size, X_val.shape[1])


x_test = [torch.tensor(x) for x in tweet_sequences_test]
X_test = nn.utils.rnn.pad_sequence(x_test, batch_first=True, padding_value=0,).long()
X_test = X_test.view(-1, batch_size, X_test.shape[1])

y_train = torch.tensor(y_train).view(-1, batch_size)
y_val = torch.tensor(y_val)
y_test = torch.tensor(labels_test_1)


# Apply the same length of X_train on X_val and X_test
len_voc = int((X_train.max()+1).item())
```

In [12]:

```
# double checking the shapes

print("shape of X train ", X_train.shape)
print("shape of y train ", y_train.shape)

print("shape of X val ", X_val.shape)
print("shape of y val ", y_val.shape)

print("shape of X test ", X_test.shape)
print("shape of y test ", y_test.shape)
```

```
shape of X train  torch.Size([424, 100, 555])
shape of y train  torch.Size([424, 100])
shape of X val  torch.Size([106, 100, 626])
shape of y val  torch.Size([10600])
shape of X test  torch.Size([119, 100, 653])
shape of y test  torch.Size([11900])
```

In [13]:

```
# pytorch model layout inspired from https://towardsdatascience.com/co
nvolutional-neural-network-in-natural-language-processing-96d67f91275c

import torch.optim as optim

# The shape of our embeddings will be the size of words in the vocab,
followed by the dimension of embedding (50)
embeddings = nn.Embedding(len_voc, 50)

# Build CNN model
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.embeddings = nn.Embedding(len_voc, 50)
        #creating a convolution with 1 inout channel, 100 output chann
els, and 3x 50 kernel size
        self.cnn = nn.Conv2d(1, 100, (3, 50))

        # we will have three neurons at the last layer before the acti
vation function for our three classes.
        self.clf = nn.Linear(100, 3)

    def forward(self, x):
        x = self.embeddings(x)
        # Add an extra dimension for CNN
        x = x.unsqueeze(1)
        # Apply CNN
        x = self.cnn(x)
        # Choose the maximum value of each filter and delete the extra
dimension
        x = x.max(2)[0].squeeze(2)
        # Choose the most important features for the classification
        x = F.relu(x)
        #  Apply linear nn for classification
        x = self.clf(x)
        return x


model = Model()
optimizer = optim.Adam(model.parameters(), lr=1e-3)
criterio  = nn.CrossEntropyLoss()
```

In [14]:

```
from sklearn import metrics
from collections import Counter

# Function for evaluating returns f1 followed by accuracy
def get_f1(X, y_real):
  y_pred = []
  for x in X:
      # Choose the value (class label) with higher probability
      predicted = model(x).argmax(1).cpu().detach()
      y_pred.append(predicted)

  y_pred_torch = torch.cat(y_pred)
  acc = metrics.accuracy_score(y_true=y_real, y_pred= y_pred_torch)
  return metrics.f1_score(y_true=y_real, y_pred=y_pred_torch, average=
'micro'), acc, y_pred
```

In [16]:

```
# Training steps
epochs = 20
LOSS = []
for e in range(epochs):
    for i, (x, y) in enumerate(zip(X_train, y_train)):

        # Delete the prvious values of the gradient
        optimizer.zero_grad()
        x, y = x, y

        y_pred = model(x)
        loss = criterio(y_pred, y)

        # Compute the gradient
        loss.backward()

        # Apply the optimization method for one step
        optimizer.step()

        LOSS.append(loss.item())
        if i%200==0:
            with torch.no_grad():
                f1, acc, y_pred = get_f1(X_val, y_val)
            print('Epoch: %d \t Batch: %d \t Loss: %.10f \t F1_val: %.
10f \t Accuracy: %.10f'%(e,i, torch.tensor(LOSS[-100:]).mean(), f1, ac
c))
```

```
Epoch: 0         Batch: 0       Loss: 0.9495213032      F1_val: 0.4
471698113        Accuracy: 0.4471698113
Epoch: 0         Batch: 200     Loss: 0.9902819991      F1_val: 0.5
126415094        Accuracy: 0.5126415094
Epoch: 0         Batch: 400     Loss: 0.9411697388      F1_val: 0.5
493396226        Accuracy: 0.5493396226
Epoch: 1         Batch: 0       Loss: 0.9342650771      F1_val: 0.5
516981132        Accuracy: 0.5516981132
Epoch: 1         Batch: 200     Loss: 0.8722500801      F1_val: 0.5
641509434        Accuracy: 0.5641509434
Epoch: 1         Batch: 400     Loss: 0.8265760541      F1_val: 0.5
750000000        Accuracy: 0.5750000000
Epoch: 2         Batch: 0       Loss: 0.8226707578      F1_val: 0.5
768867925        Accuracy: 0.5768867925
Epoch: 2         Batch: 200     Loss: 0.7616884112      F1_val: 0.5
855660377        Accuracy: 0.5855660377
Epoch: 2         Batch: 400     Loss: 0.7139935493      F1_val: 0.5
846226415        Accuracy: 0.5846226415
Epoch: 3         Batch: 0       Loss: 0.7108892798      F1_val: 0.5
871698113        Accuracy: 0.5871698113
Epoch: 3         Batch: 200     Loss: 0.6455029249      F1_val: 0.5
943396226        Accuracy: 0.5943396226
Epoch: 3         Batch: 400     Loss: 0.6001831293      F1_val: 0.5
927358491        Accuracy: 0.5927358491
Epoch: 4         Batch: 0       Loss: 0.5976411700      F1_val: 0.5
944339623        Accuracy: 0.5944339623
Epoch: 4         Batch: 200     Loss: 0.5271141529      F1_val: 0.5
950943396        Accuracy: 0.5950943396

------------------------------------------------------------------
-------
KeyboardInterrupt                         Traceback (most recent cal
l last)
<ipython-input-16-73c4c72a6f97> in <module>
      9         x, y = x, y
     10 
---> 11         y_pred = model(x)
     12         loss = criterio(y_pred, y)
     13 

~/opt/anaconda3/envs/neural_networks/lib/python3.7/site-packages/tor
ch/nn/modules/module.py in _call_impl(self, *input, **kwargs)
    725             result = self._slow_forward(*input, **kwargs)
    726         else:
--> 727             result = self.forward(*input, **kwargs)
    728         for hook in itertools.chain(
    729                 _global_forward_hooks.values(),

<ipython-input-13-fd4d2e4c7af0> in forward(self, x)
     22         x = x.unsqueeze(1)
     23         # Apply CNN
---> 24         x = self.cnn(x)
     25         # Choose the maximum value of each filter and delete the extra dimension
     26         x = x.max(2)[0].squeeze(2)

~/opt/anaconda3/envs/neural_networks/lib/python3.7/site-packages/torch/nn/modules/module.py in _call_impl(self, *input, **kwargs)
    725             result = self._slow_forward(*input, **kwargs)
    726         else:
--> 727             result = self.forward(*input, **kwargs)
    728         for hook in itertools.chain(
    729                 _global_forward_hooks.values(),

~/opt/anaconda3/envs/neural_networks/lib/python3.7/site-packages/torch/nn/modules/conv.py in forward(self, input)
    421 
    422     def forward(self, input: Tensor) -> Tensor:
--> 423         return self._conv_forward(input, self.weight)
    424 
    425 class Conv3d(_ConvNd):

~/opt/anaconda3/envs/neural_networks/lib/python3.7/site-packages/torch/nn/modules/conv.py in _conv_forward(self, input, weight)
    418                             _pair(0), self.dilation, self.groups)
    419         return F.conv2d(input, weight, self.bias, self.stride,
--> 420                         self.padding, self.dilation, self.groups)
    421 
    422     def forward(self, input: Tensor) -> Tensor:

KeyboardInterrupt: 
```

In [17]:

```
f1, acc, y_pred = get_f1(X_test, y_test)
print(" f1 is ", f1, " accuracy is ", acc)
```

```
 f1 is  0.5085714285714286  accuracy is  0.5085714285714286
```

In [18]:

```
from sklearn.metrics import classification_report
predicted_labels_list = []

flat_list = [item for sub in y_pred for item in sub]

for elem in flat_list:
    predicted_labels_list.append(elem.tolist())

print(len(predicted_labels_list))
print(classification_report(labels_test[:11900], predicted_labels_list
))
```

```
11900
              precision    recall  f1-score   support

           0       0.55      0.28      0.37      3811
           1       0.55      0.66      0.60      5739
           2       0.39      0.52      0.45      2350

    accuracy                           0.51     11900
   macro avg       0.50      0.48      0.47     11900
weighted avg       0.52      0.51      0.49     11900
```

In [ ]: