Homework 4 Final Models (Text Classification)

Adam Kiehl
5/11/2023

```python
# import analysis packages
import keras
from keras.callbacks import EarlyStopping
from keras.layers import Dense, Dropout, Embedding, Flatten, SimpleRNN, TextVectorization
from keras.models import Sequential
from keras.regularizers import l2
from keras.utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import random
import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow_addons.metrics import F1Score
```

```
/Users/akiehl/miniconda3/envs/dsci/lib/python3.8/site-packages/tensorflow_addons/utils/tfa_e

TensorFlow Addons (TFA) has ended development and introduction of new features.
TFA has entered a minimal maintenance and release mode until a planned end of life in May 20
Please modify downstream libraries to take dependencies from other repositories in our Tenso

For more information see: https://github.com/tensorflow/addons/issues/2807

  warnings.warn(
/Users/akiehl/miniconda3/envs/dsci/lib/python3.8/site-packages/tensorflow_addons/utils/ensur
 The versions of TensorFlow you are currently using is 2.9.2 and is not supported.
Some things might work, some things might not.
If you were to encounter a bug, do not file an issue.
If you want to make sure you're using a tested and supported configuration, either change th
You can find the compatibility matrix in TensorFlow Addon's readme:
https://github.com/tensorflow/addons
  warnings.warn(
```

Data Cleaning

```
# read data from .csv files
trainDF = pd.read_csv('./ibotta_train.csv')
testDF = pd.read_csv('./ibotta_test.csv')

# combine data sets for preprocessing
trainDF['origin'] = 'train'
testDF['origin'] = 'test'
fullDF = pd.concat([trainDF, testDF])

# combine name and brand name fields
fullDF['Brand_name'].where(-fullDF['Brand_name'].isna(), '', inplace = True)
fullDF['Full_text'] = fullDF['Brand_name'] + ' ' + fullDF['Name']

# seed random seed
random.seed(542023)

# split data
trainDF = pd.DataFrame(fullDF.loc[fullDF['origin'] == 'train'].drop('origin', axis = 1))
testDF = pd.DataFrame(fullDF.loc[fullDF['origin'] == 'test'].drop(['origin', 'Category'],
```

Text Vectorization

```
# train integer index tokenizer
intTokenizer = TextVectorization()
intTokenizer.adapt(fullDF['Full_text'])

# vectorize text data
intVecDF = pd.DataFrame(intTokenizer(fullDF['Full_text']))
trainDFintVec = intVecDF.loc[0:7999]
testDFintVec = intVecDF.loc[8000:9999]
```

```
Metal device set to: Apple M1

systemMemory: 8.00 GB
maxCacheSize: 2.67 GB

2023-05-11 13:25:43.408047: W tensorflow/core/platform/profile_utils/cpu_utils.cc:128] Failed
```

```
# train bag of words tokenizer
countTokenizer = TextVectorization(output_mode = 'multi_hot')
countTokenizer.adapt(fullDF['Full_text'])

# vectorize text data
countVecDF = pd.DataFrame(countTokenizer(fullDF['Full_text']))
trainDFcountVec = countVecDF.loc[0:7999]
testDFcountVec = countVecDF.loc[8000:9999]
```

```
# train tfidf tokenizer
tfidfTokenizer = TextVectorization(output_mode = 'tf_idf')
tfidfTokenizer.adapt(fullDF['Full_text'])

# vectorize text data
tfidfVecDF = pd.DataFrame(tfidfTokenizer(fullDF['Full_text']))
trainDFtfidfVec = tfidfVecDF.loc[0:7999]
testDFtfidfVec = tfidfVecDF.loc[8000:9999]
```

Model Fitting

```
# set random seeds
np.random.seed(542023)
tf.random.set_seed(542023)

# define model architecture
model1 = Sequential([
    Dense(512, activation = 'relu'),
    Dense(256, activation = 'relu'),
    Dense(128, activation = 'relu'),
    Dense(64, activation = 'relu'),
    Dense(7, activation = 'softmax')
])

# define F1 metric
f1_score_metric = F1Score(num_classes = 7, average = 'weighted')

# compile model
model1.compile(optimizer = 'rmsprop',
               loss = 'categorical_crossentropy',
               metrics = ['accuracy', f1_score_metric])
```

```python
# train deep learning model
trained1 = model1.fit(trainDFcountVec,
                      to_categorical(trainDF['Cat_code']),
                      epochs = 10,
                      batch_size = 128,
                      verbose = 1)

# predict on test set
pred1 = model1.predict(testDFcountVec)

# create submission data frame
submission = pd.DataFrame({'Id': testDF['Id'], 'Cat_code': np.argmax(pred1, axis = 1).resh

# export submission
submission.to_csv('./submission1.csv', index = False)
```

```
Epoch 1/10
63/63 [==============================] - 2s 22ms/step - loss: 0.6611 - accuracy: 0.7745 - f1_
Epoch 2/10
63/63 [==============================] - 1s 20ms/step - loss: 0.1063 - accuracy: 0.9709 - f1_
Epoch 3/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0474 - accuracy: 0.9847 - f1_
Epoch 4/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0291 - accuracy: 0.9895 - f1_
Epoch 5/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0188 - accuracy: 0.9936 - f1_
Epoch 6/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0134 - accuracy: 0.9945 - f1_
Epoch 7/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0089 - accuracy: 0.9959 - f1_
Epoch 8/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0083 - accuracy: 0.9961 - f1_
Epoch 9/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0050 - accuracy: 0.9981 - f1_
Epoch 10/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0042 - accuracy: 0.9981 - f1_
63/63 [==============================] - 0s 3ms/step
```

```python
# set random seeds
np.random.seed(542023)
```

```
tf.random.set_seed(542023)

# define model hyperparameters
PENALTY = 0.001
RATE = 0.2

# define model architecture
model2 = Sequential([
    Dense(2048, kernel_regularizer = l2(PENALTY), activation = 'relu'),
    Dense(1024, kernel_regularizer = l2(PENALTY), activation = 'relu'),
    Dropout(RATE),
    Dense(512, kernel_regularizer = l2(PENALTY), activation = 'relu'),
    Dense(256, kernel_regularizer = l2(PENALTY), activation = 'relu'),
    Dense(128, kernel_regularizer = l2(PENALTY), activation = 'relu'),
    Dense(64, kernel_regularizer = l2(PENALTY), activation = 'relu'),
    Dense(7, activation = 'softmax')
])

# define F1 metric
f1_score_metric = F1Score(num_classes = 7, average = 'weighted')

# compile model
model2.compile(optimizer = 'rmsprop',
               loss = 'categorical_crossentropy',
               metrics = ['accuracy', f1_score_metric])

# train deep learning model
trained2 = model2.fit(trainDFtfidfVec,
                      to_categorical(trainDF['Cat_code']),
                      epochs = 10,
                      batch_size = 128,
                      verbose = 1)

# predict on test set
pred2 = model2.predict(testDFtfidfVec)

# create submission data frame
submission = pd.DataFrame({'Id': testDF['Id'], 'Cat_code': np.argmax(pred2, axis = 1).resh

# export submission
submission.to_csv('./submission2.csv', index = False)
```

```
Epoch 1/10
63/63 [==============================] - 4s 46ms/step - loss: 2.5798 - accuracy: 0.6991 - f1_
Epoch 2/10
63/63 [==============================] - 3s 46ms/step - loss: 0.7858 - accuracy: 0.9496 - f1_
Epoch 3/10
63/63 [==============================] - 3s 46ms/step - loss: 0.5069 - accuracy: 0.9599 - f1_
Epoch 4/10
63/63 [==============================] - 3s 44ms/step - loss: 0.3944 - accuracy: 0.9615 - f1_
Epoch 5/10
63/63 [==============================] - 3s 44ms/step - loss: 0.3275 - accuracy: 0.9645 - f1_
Epoch 6/10
63/63 [==============================] - 3s 46ms/step - loss: 0.2782 - accuracy: 0.9705 - f1_
Epoch 7/10
63/63 [==============================] - 3s 45ms/step - loss: 0.2450 - accuracy: 0.9710 - f1_
Epoch 8/10
63/63 [==============================] - 3s 44ms/step - loss: 0.2180 - accuracy: 0.9761 - f1_
Epoch 9/10
63/63 [==============================] - 3s 44ms/step - loss: 0.2087 - accuracy: 0.9737 - f1_
Epoch 10/10
63/63 [==============================] - 3s 43ms/step - loss: 0.1890 - accuracy: 0.9772 - f1_
63/63 [==============================] - 0s 3ms/step
```

```python
# set random seeds
np.random.seed(542023)
tf.random.set_seed(542023)

# define model architecture
model3 = Sequential([
    Embedding(4880, 64, input_shape = (4880, )),
    Flatten(),
    Dense(128, activation = 'relu'),
    Dense(64, activation = 'relu'),
    Dense(7, activation = 'softmax')
])

# define F1 metric
f1_score_metric = F1Score(num_classes = 7, average = 'weighted')

# compile model
model3.compile(optimizer = 'rmsprop',
               loss = 'categorical_crossentropy',
               metrics = ['accuracy', f1_score_metric])

# train deep learning model
trained3 = model3.fit(trainDFcountVec,
                      to_categorical(trainDF['Cat_code']),
                      epochs = 12,
                      batch_size = 32,
                      verbose = 1)

# predict on test set
pred3 = model3.predict(testDFcountVec)

# create submission data frame
submission = pd.DataFrame({'Id': testDF['Id'], 'Cat_code': np.argmax(pred3, axis = 1).resh

# export submission
submission.to_csv('./submission3.csv', index = False)
```

```
Epoch 1/12
250/250 [==============================] - 200s 799ms/step - loss: 1.5559 - accuracy: 0.6158
Epoch 2/12
250/250 [==============================] - 175s 701ms/step - loss: 0.1838 - accuracy: 0.9436
Epoch 3/12
250/250 [==============================] - 151s 605ms/step - loss: 0.1256 - accuracy: 0.9632
Epoch 4/12
250/250 [==============================] - 142s 570ms/step - loss: 0.0933 - accuracy: 0.9704
Epoch 5/12
250/250 [==============================] - 138s 551ms/step - loss: 0.0753 - accuracy: 0.9797
Epoch 6/12
250/250 [==============================] - 129s 516ms/step - loss: 0.0667 - accuracy: 0.9804
Epoch 7/12
250/250 [==============================] - 128s 510ms/step - loss: 0.0616 - accuracy: 0.9846
Epoch 8/12
250/250 [==============================] - 125s 500ms/step - loss: 0.0537 - accuracy: 0.9858
Epoch 9/12
250/250 [==============================] - 119s 474ms/step - loss: 0.0396 - accuracy: 0.9884
Epoch 10/12
250/250 [==============================] - 108s 433ms/step - loss: 0.0397 - accuracy: 0.9893
Epoch 11/12
250/250 [==============================] - 111s 444ms/step - loss: 0.0333 - accuracy: 0.9908
Epoch 12/12
250/250 [==============================] - 105s 418ms/step - loss: 0.0319 - accuracy: 0.9915
```

```
63/63 [==============================] - 2s 37ms/step
```

```
# set random seeds
np.random.seed(542023)
tf.random.set_seed(542023)

# define model architecture
model4 = Sequential([
    Embedding(4880, 64, input_shape = (4880, )),
    Flatten(),
    Dense(128, activation = 'relu'),
    Dense(64, activation = 'relu'),
    Dense(7, activation = 'softmax')
])

# define F1 metric
f1_score_metric = F1Score(num_classes = 7, average = 'weighted')

# compile model
model4.compile(optimizer = 'rmsprop',
               loss = 'categorical_crossentropy',
               metrics = ['accuracy', f1_score_metric])

# train deep learning model
trained4 = model4.fit(trainDFtfidfVec,
                      to_categorical(trainDF['Cat_code']),
                      epochs = 12,
                      batch_size = 32,
                      verbose = 1)

# predict on test set
pred4 = model4.predict(testDFtfidfVec)

# create submission data frame
submission = pd.DataFrame({'Id': testDF['Id'], 'Cat_code': np.argmax(pred4, axis = 1).resh

# export submission
submission.to_csv('./submission4.csv', index = False)
```

```
Epoch 1/12
250/250 [==============================] - 210s 840ms/step - loss: 1.6101 - accuracy: 0.5831
```

```
Epoch 2/12
250/250 [==============================] - 58s 230ms/step - loss: 1.5591 - accuracy: 0.3799 -
Epoch 3/12
250/250 [==============================] - 32s 129ms/step - loss: 1.7687 - accuracy: 0.2940 -
Epoch 4/12
250/250 [==============================] - 32s 128ms/step - loss: 1.7684 - accuracy: 0.2940 -
Epoch 5/12
250/250 [==============================] - 32s 126ms/step - loss: 1.7686 - accuracy: 0.2940 -
Epoch 6/12
250/250 [==============================] - 32s 130ms/step - loss: 1.7686 - accuracy: 0.2940 -
Epoch 7/12
250/250 [==============================] - 32s 127ms/step - loss: 1.7684 - accuracy: 0.2936 -
Epoch 8/12
250/250 [==============================] - 32s 130ms/step - loss: 1.7686 - accuracy: 0.2940 -
Epoch 9/12
250/250 [==============================] - 32s 127ms/step - loss: 1.7684 - accuracy: 0.2940 -
Epoch 10/12
250/250 [==============================] - 32s 126ms/step - loss: 1.7684 - accuracy: 0.2934 -
Epoch 11/12
250/250 [==============================] - 32s 128ms/step - loss: 1.7685 - accuracy: 0.2940 -
Epoch 12/12
250/250 [==============================] - 32s 126ms/step - loss: 1.7682 - accuracy: 0.2939 -
63/63 [==============================] - 2s 37ms/step
```

```python
# set random seeds
np.random.seed(542023)
tf.random.set_seed(542023)

# define model architecture
model5 = Sequential([
    Dense(512, activation = 'relu'),
    Dense(256, activation = 'relu'),
    Dense(128, activation = 'relu'),
    Dense(64, activation = 'relu'),
    Dense(7, activation = 'softmax')
])

# define F1 metric
f1_score_metric = F1Score(num_classes = 7, average = 'weighted')

# compile model
```

```
model5.compile(optimizer = 'rmsprop',
               loss = 'categorical_crossentropy',
               metrics = ['accuracy', f1_score_metric])

# train deep learning model
trained5 = model5.fit(trainDFtfidfVec,
                      to_categorical(trainDF['Cat_code']),
                      epochs = 10,
                      batch_size = 128,
                      verbose = 1)

# predict on test set
pred5 = model5.predict(testDFtfidfVec)

# create submission data frame
submission = pd.DataFrame({'Id': testDF['Id'], 'Cat_code': np.argmax(pred5, axis = 1).resh

# export submission
submission.to_csv('./submission5.csv', index = False)
```

```
Epoch 1/10
63/63 [==============================] - 2s 21ms/step - loss: 0.5937 - accuracy: 0.8036 - f1_
Epoch 2/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0741 - accuracy: 0.9793 - f1_
Epoch 3/10
63/63 [==============================] - 1s 20ms/step - loss: 0.0327 - accuracy: 0.9902 - f1_
Epoch 4/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0186 - accuracy: 0.9930 - f1_
Epoch 5/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0135 - accuracy: 0.9952 - f1_
Epoch 6/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0091 - accuracy: 0.9974 - f1_
Epoch 7/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0064 - accuracy: 0.9974 - f1_
Epoch 8/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0040 - accuracy: 0.9985 - f1_
Epoch 9/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0038 - accuracy: 0.9986 - f1_
Epoch 10/10
63/63 [==============================] - 1s 19ms/step - loss: 0.0028 - accuracy: 0.9991 - f1_
63/63 [==============================] - 0s 3ms/step
```