**Assignment 4: Text and Sequence:**

Steven Kalinoff  
4/07/2024

Adv. Machine Learning 64061

Baseline performance

A graph with blue dots

Description automatically generatedA graph with blue dots

Description automatically generated

Top 150 words 100 samples

from tensorflow.keras.datasets import imdb

(train\_data, train\_labels), (test\_data, test\_labels) = imdb.load\_data(

    num\_words=150)

A graph with blue dots

Description automatically generatedA graph with blue dots

Description automatically generated

Embedding Layer @ 150/100

from tensorflow import keras

from tensorflow.keras import layers

max\_tokens = max([max(sequence) for sequence in train\_data]) + 3

embedding\_layer = layers.Embedding(input\_dim=max\_tokens, output\_dim=256)

inputs = keras.Input(shape=(None,), dtype="int64")

embedded = layers.Embedding(input\_dim=max\_tokens, output\_dim=256)(inputs)

x = layers.Bidirectional(layers.LSTM(32))(embedded)

x = layers.Dropout(0.5)(x)

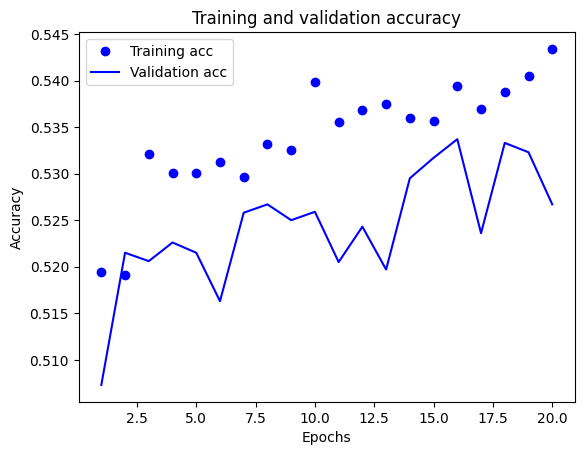
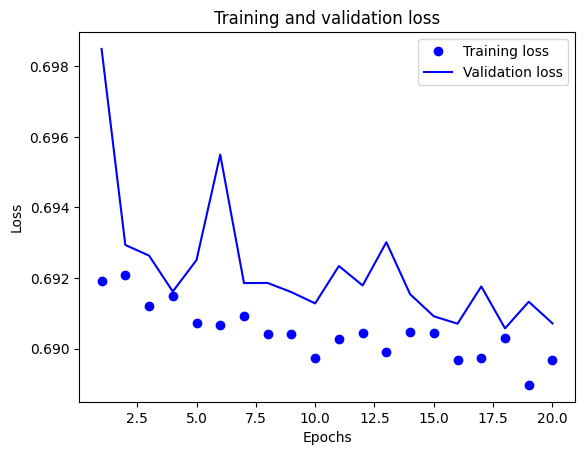
outputs = layers.Dense(1, activation="sigmoid")(x)

model = keras.Model(inputs, outputs)

model.compile(optimizer="rmsprop",

              loss="binary\_crossentropy",

              metrics=["accuracy"])



Pretrained word embedding

#pretrained

import tensorflow\_hub as hub

text\_vectorization = TextVectorization(

    ngrams=2,

    max\_tokens=20000,

    output\_mode="multi\_hot",

)

import numpy as np

path\_to\_glove\_file = "glove.6B.100d.txt"

embeddings\_index = {}

with open(path\_to\_glove\_file) as f:

    for line in f:

        word, coefs = line.split(maxsplit=1)

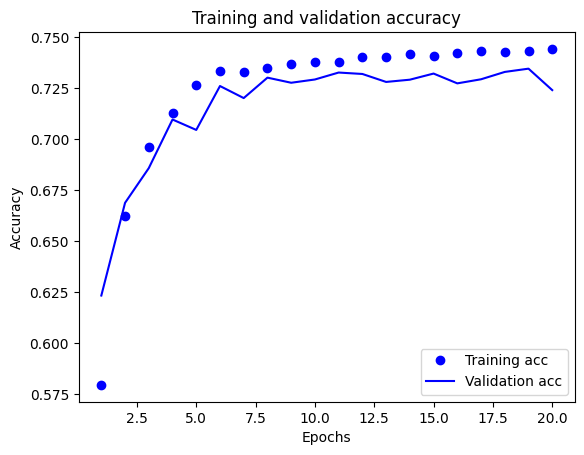
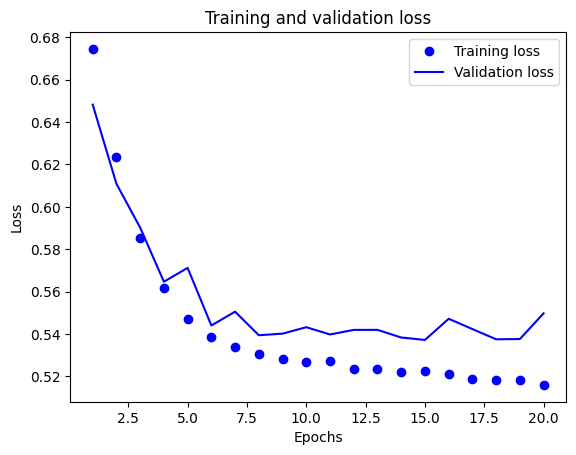
        coefs = np.fromstring(coefs, "f", sep=" ")

        embeddings\_index[word] = coefs

print(f"Found {len(embeddings\_index)} word vectors.")

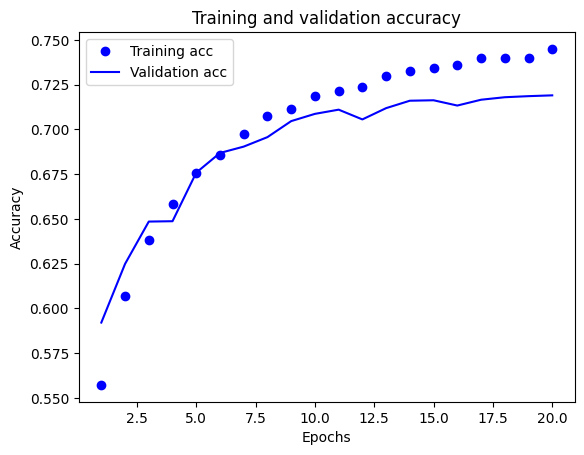
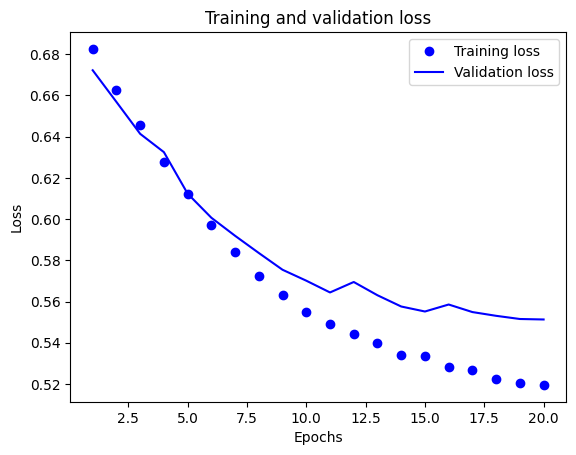
from tensorflow import keras

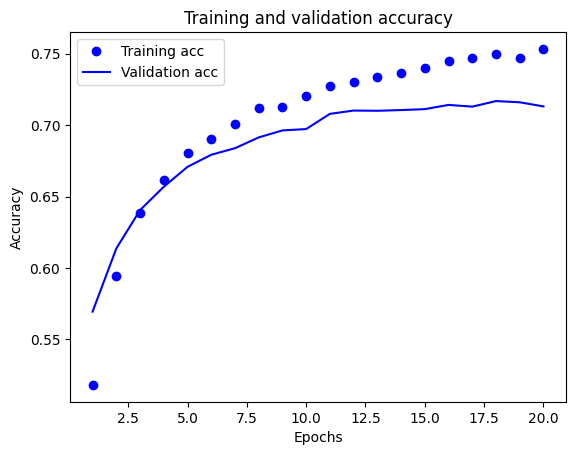
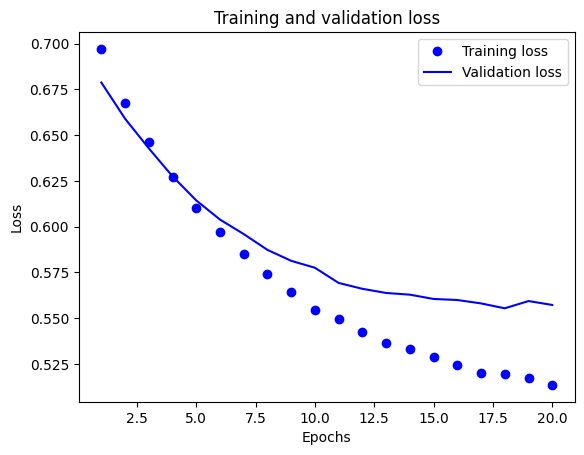
from tensorflow.keras import layers



20,000 training samples

Embedded vs pretrained





Summary/Findings

Our baseline performance on the IMDB model is very high 97-98% that is using the full training samples and this model’s performance is optimized quite well given the parameters. For the rest of the modeling, we will restrict both the words the model can use to train and the number of entries it can learn from.

We then look at our new baseline and as expected with less data our validation accuracy drops by about 20% to the low 70’s. Both exercises are meant to draw a relationship between a review score and the words used in that review, words like bad, dull, bland, are meant to draw a negative correlation with the score. In this basic case the model has less data to make sense of down to the point of being as good as a random heads or tails guess. While we are artificially reducing the data to make the results worse we can now use other tools to gleam more accuracy with the same amount of data.

We aim to make more meaningful relationships from the word’s and phrases that we have access to by word embedding. This is a vectorization process that codes each unique word to a integer. We are aiming to embed words like Good, Great, must see as positive attributes in our training data that can then be used get more accurate findings in our validation data. In terms how a human would associate something like a pitcher, first basemen, etc. to baseball and a goalie for soccer the algorithm learns these relations with the testing phase.

The other method that is used in this text and sequence project is the pretrained source. By using a nearest neighbor approach the GloVe algorithm looks at the vector length of words for determining their similarity it also uses probability tables it creates to looks for word pairs that may help derive context.

With small datasets and limited reviews that pretrained systems works better and converges much more stably than the embedded “learning” system. It is interesting to note however in this case that because it is a general model that at some point our reviews should be able to beat the pretrained system due to the amount of times common words like “great” or “forgettable” would be seen in the training data. In this case 20,000 reviews are needed to be observed by the embedded system to beat the off the shelf GloVe data.

I think that typically with limited data a pretrained embedded can provide useful, in certain cases a specific or specialized pretrained embedding may be even better but given enough data the embedding layer can surpass the pretrained one if the training data is concise and descriptive enough to capture the natural idea of whatever is being trained.