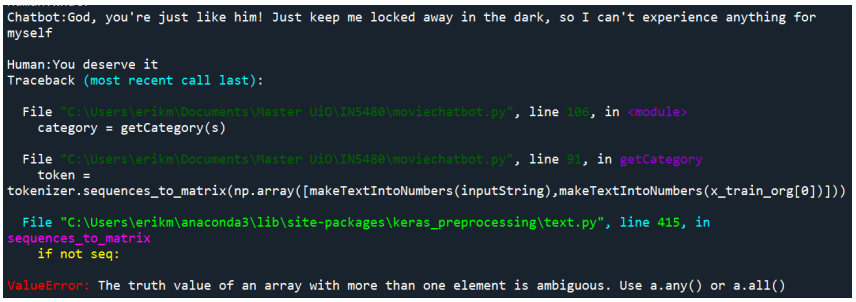
Appendix 2 - Machine learning task

We spent a lot of time initially trying to understand the code to figure out which numbers we could manipulate in order to see any changes in the training of the chatbot. While the code was not very descriptive, we found out that changes in the batch number and epoch values seemed to have the biggest impact, while changing the “Dense” value in the model also had some minor effects. Throughout this working with this task, we have been very confused by the output values of the neural network’s training. We still don’t really have a good understanding of what the loss and accuracy values actually mean and how they correlate to how the bot responds to our input. The difference between val_loss and loss was also not apparent. We ran into some issues where the script would randomly crash after no more than 20 inputs from the user:



Chatbot crashing

As the chatbot replied with the movie lines, we were confused by whether it had any correlation to what we wrote to the bot. At some point the replies indicated that the chatbot had understood what was written by us, however we were quickly disappointed when the next line seemed to be completely random. We are therefore left with the feeling that it doesn't matter what we write to the chatbot.

Learning outcomes

It seems to take a very high amount of iterations for the chatbot to exhibit any form of intelligence. We have not yet seen any signs of this. We change the batch size to 512 and later to 1000, and change the epochs to 10000. Still the accuracy was 0.17 and the interaction with the chatbot was confusing.

It’s very hard to tell what actually makes a difference and what doesn’t. This might be connected to using too few iterations. However we didn't find what we were supposed to increase or do differently to get a better chatbot.

When we added a dropout to the model of 0.3 the accuracy number seemed to fluctuate up and down a bit more. The number changed from 0.15 to 0.18 after every epoch. Without dropout the accuracy was consistently 0.15 until it changes to 0.17.

```
val_accuracy: 0.0000e+00
Epoch 9995/10000
2/2 [==============================] - 0s 15ms/step - loss: 2.6205 - accuracy: 0.1722 - val_loss: 12.4610 -
val_accuracy: 0.0000e+00
Epoch 9996/10000
2/2 [==============================] - 0s 15ms/step - loss: 2.6227 - accuracy: 0.1722 - val_loss: 12.4459 -
val_accuracy: 0.0000e+00
Epoch 9997/10000
2/2 [==============================] - 0s 15ms/step - loss: 2.6192 - accuracy: 0.1722 - val_loss: 12.4459 -
val_accuracy: 0.0000e+00
Epoch 9998/10000
2/2 [==============================] - 0s 15ms/step - loss: 2.6220 - accuracy: 0.1722 - val_loss: 12.4244 -
val_accuracy: 0.0000e+00
Epoch 9999/10000
2/2 [==============================] - 0s 16ms/step - loss: 2.6198 - accuracy: 0.1722 - val_loss: 12.4110 -
val_accuracy: 0.0000e+00
Epoch 10000/10000
2/2 [==============================] - 0s 15ms/step - loss: 2.6248 - accuracy: 0.1722 - val_loss: 12.4279 -
val_accuracy: 0.0000e+00
Finished training
ready
WARNING:tensorflow:6 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function
at 0x000001863926CA0> triggered tf.function retracing. Tracing is expensive and the excessive number of
tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different
shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the
loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can
avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
C:\Users\erikm\Documents\Master UiO\IN5480\moviechatbot.py:92: VisibleDeprecationWarning: Creating an ndarray
from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or
shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  token =
tokenizer.sequences_to_matrix(np.array([makeTextIntoNumbers(inputString),makeTextIntoNumbers(x_train_org[0])]))
Chatbot:Just once.  Afterwards, I told him I didn't want to anymore.  I wasn't ready. He got pissed.  Then he
broke up with me.

Human:What an asshole!
Traceback (most recent call last):

  File "C:\Users\erikm\Documents\Master UiO\IN5480\moviechatbot.py", line 107, in <module>
    category = getCategory(s)

  File "C:\Users\erikm\Documents\Master UiO\IN5480\moviechatbot.py", line 93, in getCategory
    token =
tokenizer.sequences_to_matrix(np.array([makeTextIntoNumbers(inputString),makeTextIntoNumbers(x_train_org[0])]))

  File "C:\Users\erikm\anaconda3\lib\site-packages\keras_preprocessing\text.py", line 415, in
sequences_to_matrix
    if not seq:
```

High validation loss (12.4) after 10 000 epochs