Spam-Email-Classification

Group Mean:

Cagatay Duygu

Ahmet Ata Ersoy

What Is the Dataset?

We used a email dataset which has approximately 5000 sample as labeled spam or not. We tried to classify dataset as spam or real. Dataset can be found in the link:

https://github.com/Balakishan77/Spam-Email-Classifier

Metric and Explanation: Recall

Recall accuracy is a metric that is often used in classification tasks. It measures the proportion of positive examples that were correctly classified by the model. In other words, it measures how well the model was able to identify all of the positive examples in the dataset.

Recall accuracy is particularly useful when working with imbalanced datasets, where there are a relatively small number of positive examples compared to negative examples. In these cases, it is important to ensure that the model is able to identify as many of the positive examples as possible, even if it means that there are more false positives.

Some examples of scenarios where recall accuracy might be a useful metric include medical diagnosis, fraud detection, and spam filtering. In all of these cases, it is important to identify as many of the positive examples as possible, even if it means that there may be some false positives.

Overall, recall accuracy is a useful metric for evaluating the performance of a model when working with imbalanced datasets and when it is important to identify as many of the positive examples as possible.

For our case it makes sense to use 'Recall' since it is important for us to identify spam emails.

Data Separation Method

Stratified sampling is used to ensure that each class in the dataset is represented in the sample in the same proportion as it is in the entire dataset. This is particularly important in unbalanced datasets, where one class may be underrepresented. By using stratified sampling, we can ensure that the split between the classes in the training and test sets is representative of the overall distribution of classes in the dataset. This can help to prevent bias in the model, and can improve its performance on the minority class. Because of all these reasons, we chose using stratified shuffle split.

We also used undersampling. Undersampling is typically used when the dataset is unbalanced, meaning that one class is much more prevalent than the other. In this case, the majority class can dominate the model, causing it to be biased towards the majority class. Undersampling can be used to reduce the number of samples in the majority class to be more in line with the minority class, making the model more balanced and less biased.

In general, it is recommended to use stratified k-fold cross-validation to evaluate machine learning models, particularly when the target variable is imbalanced. However, whether or not to use stratified k-fold cross-validation depends on the specific context and goals of your machine learning project. Therefore, we decided to compare both methods.

Pre-Processing and Tokenize

In [126…
```
#Categorizing given email is spam or ham
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt


dataset = pd.read_csv('spamham.csv')
dataset.head()
```

Out[126]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

In [127…
```
dataset.columns
```

Out[127]:
```
Index(['text', 'spam'], dtype='object')
```

In [128…
```
dataset.shape
```

Out[128]:
```
(5728, 2)
```

In [129…
```
#Checking for duplicates and removing them
dataset.drop_duplicates(inplace = True)
dataset.shape  #(5695, 2)
```

Out[129]:
```
(5695, 2)
```

In [130…
```
dataset
```

Out[130]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1
...	...	...
5723	Subject: re : research and development charges...	0
5724	Subject: re : receipts from visit jim , than...	0
5725	Subject: re : enron case study update wow ! a...	0
5726	Subject: re : interest david , please , call...	0
5727	Subject: news : aurora 5 . 2 update aurora ve...	0

5695 rows × 2 columns

In [131…

```
#Checking for any null entries in the dataset
print (pd.DataFrame(dataset.isnull().sum()))
```

```
      0
text  0
spam  0
```

In [132…

```
#Checking class distribution
dataset.groupby('spam').count()
```

Out[132]:

spam	text
0	4327
1	1368

In [133…

```
len(dataset.text)
```

Out[133]: 5695

In [134…

```
import numpy as np

# display a random document and label
idx = round(np.random.rand()*len(dataset.text))
print('--------Random Document---------')
print('================================')
print('Document Label: ',[dataset.spam[idx]])
print('================================')
print("\n".join(dataset.text[idx].split("\n")))
```

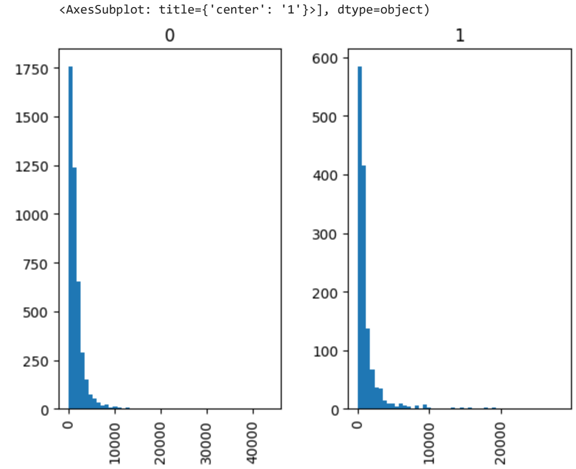
```
--------Random Document---------
================================
Document Label:  [0]
================================
Subject: mid - year 2000 performance feedback  note : you will receive this message each time you are selected as a reviewer .  you have been selected to participate in the mid - year 2000 performance  management process by providing meaningful feedback on specific employee ( s )  that have been identified for you . your feedback plays an important role in  the per formance management process , and your participation is very critical  to the success of enron ' s performance management goals .  please provide feedback on the employee ( s ) listed below by accessing the  performance management system ( pep ) and completing an online feedback form as  described in the " performance management quick reference guide " . you may  b egin your feedback input immediately . please have all feedback forms  completed by the date noted below .  if you have any questions regarding pep or your responsibility in the  proc ess , please call the pep help desk at the following numbers :  in the u . s . : 1 - 713 - 853 - 4777 , option 4  in europe : 44 - 207 - 783 - 4040 , option 4  in canada : 1 - 403 - 9 74 - 6724 ( canada employees only )  or e - mail your questions to : perfmgmt @ enron . com  thank you for your participation in this important process .  the following list of employ ees is a cumulative list of all feedback  requests , by operating company , that have an " open " feedback status . an  employee ' s name will no longer appear once you have completed the feedback  form and select the " submit " button in pep .  review group : enron  feedback due date : jun 16 , 2000  employee name supervisor name date selected - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  ahmad , anjam dale surbey may 22 , 2000  carson , margaret m james d steffes may 26 , 2000  carson , richard l richard b buy may 22 , 2000  crenshaw , shirley j wincenty j . kaminski may 24 , 2000  ghosh , soma timothy davies may 31 , 2000  kaminski , wincenty j . david w delainey jun 05 , 2000  peyton , john a randal t maffett jun 05 , 2000  thuraisingham , ravi vasant shanbhogue may 30 , 2000  vernon , clayton j vasant shanbhogue may 26 , 2000  yuan , ding richard l carson jun 02 , 2000  zipter , rudi c theodore r murphy may 25 , 2000
```

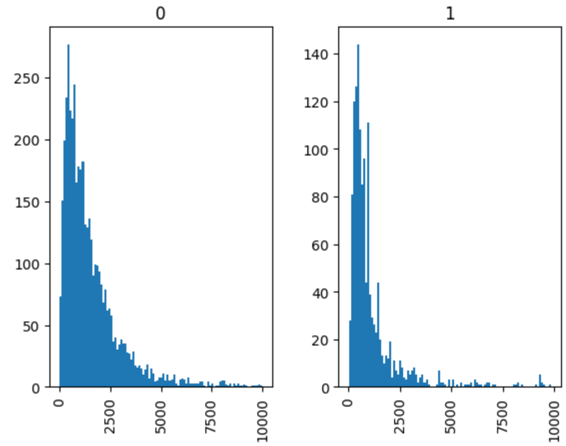
In [135…

```
dataset['length'] = dataset['text'].map(lambda text: len(text))
#Let's plot histogram for length distribution by spam
dataset.hist(column='length', by='spam', bins=50)
#we can see some extreme outliers, we'll set a threshold for text length and plot the histogram again
dataset[dataset.length < 10000].hist(column='length', by='spam', bins=100)
#Using Natural Language Processing to cleaning the text to make one corpus
```

Out[135]:

```
array([<AxesSubplot: title={'center': '0'}>,
       <AxesSubplot: title={'center': '1'}>], dtype=object)
```





Tokenize

In [136…

```
%%time
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

NUM_TOP_WORDS = None # use entire vocabulary!
MAX_ART_LEN = 250 # maximum and minimum number of words

#tokenize the text
tokenizer = Tokenizer(num_words=NUM_TOP_WORDS)
tokenizer.fit_on_texts(dataset.text)
# save as sequences with integers replacing words
sequences = tokenizer.texts_to_sequences(dataset.text)

word_index = tokenizer.word_index
NUM_TOP_WORDS = len(word_index) if NUM_TOP_WORDS==None else NUM_TOP_WORDS
top_words = min((len(word_index),NUM_TOP_WORDS))
print('Found %s unique tokens. Distilled to %d top words.' % (len(word_index),top_words))

X = pad_sequences(sequences, maxlen=MAX_ART_LEN)

y_ohe = dataset.spam
print('Shape of data tensor:', X.shape)
print('Shape of label tensor:', y_ohe.shape)
print(np.max(X))
```

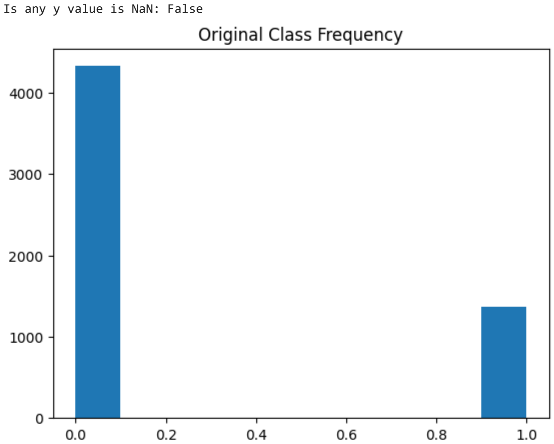
```
Found 37353 unique tokens. Distilled to 37353 top words.
Shape of data tensor: (5695, 250)
Shape of label tensor: (5695,)
37353
CPU times: user 1.18 s, sys: 368 ms, total: 1.55 s
Wall time: 2.28 s
```

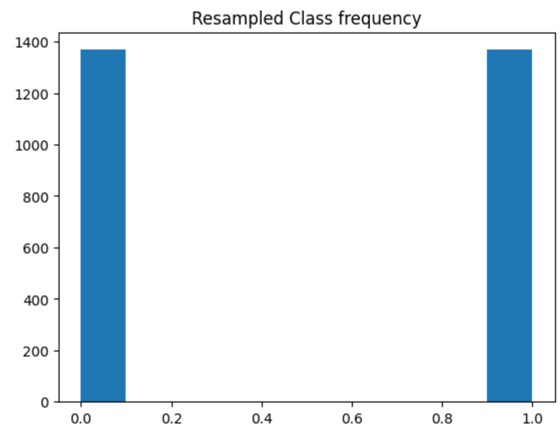
Undersampling

In [137…

```
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

#Use undersampling to make every
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(X, y_ohe)
print(f'Is any y value is NaN: {np.isnan(y_ohe).any()}')

plt.hist(y_ohe)
plt.title("Original Class Frequency")
plt.show()

plt.hist(y_resampled)
plt.title("Resampled Class frequency")
plt.show()
```

```
Is any y value is NaN: False
```



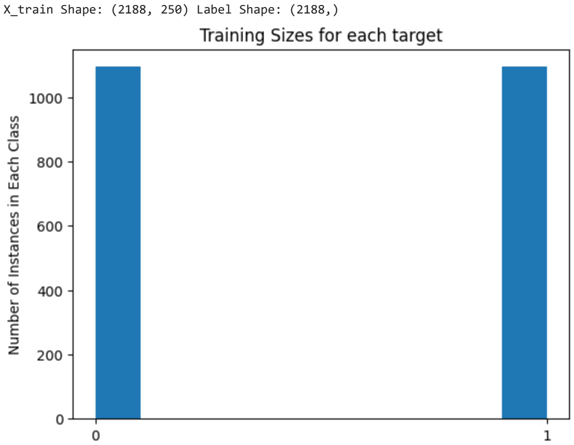


Stratify Split with train_test_split

In [138…

```
import math
from sklearn.model_selection import train_test_split
from sklearn import metrics as mt
from matplotlib import pyplot as plt
%matplotlib inline

# Split it into train / test subsets
X_train, X_test, y_train_ohe, y_test_ohe = train_test_split(X_resampled, y_resampled, test_size=0.2,
                                                            stratify=y_resampled,
                                                            random_state=42)
NUM_CLASSES = 1

# print some stats of the data
print("X_train Shape:",X_train.shape, "Label Shape:", y_train_ohe.shape)
uniq_classes = np.sum(y_train_ohe,axis=0)
plt.hist(y_train_ohe)
plt.ylabel("Number of Instances in Each Class")

new_list = range(math.floor(min(y_test_ohe)), math.ceil(max(y_test_ohe))+1)
plt.xticks(new_list)
plt.title("Training Sizes for each target")


plt.show()
```

```
X_train Shape: (2188, 250) Label Shape: (2188,)
```

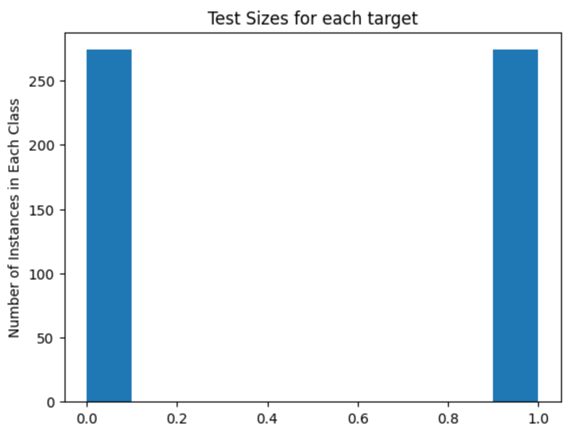


In [139…

```
uniq_classes = np.sum(y_test_ohe,axis=0)
plt.hist(y_test_ohe)
plt.title("Test Sizes for each target")
plt.ylabel("Number of Instances in Each Class")
```

Out[139]:

```
Text(0, 0.5, 'Number of Instances in Each Class')
```



Architectures

Simple RNN Model

In [140…

```
# show example without the FOR loop
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Input, SimpleRNN
from tensorflow.keras.layers import Embedding

EMBED_SIZE = 50
RNN_STATESIZE = 100
rnns = []
input_holder = Input(shape=(X_train.shape[1], ))
shared_embed = Embedding(top_words, # input dimension (max int of OHE)
                EMBED_SIZE, # output dimension size
                input_length=MAX_ART_LEN)(input_holder) # number of words in each sequence


x = SimpleRNN(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
simple_rnn_model = Model(inputs=input_holder,outputs=x)
```

Create first LSTM and GRU Models. Compile Simple RNN, LSTM and GRU

Please note that these values are tuned after checking the validation/training accuracy. Best models are used.

In [141…

```
from tensorflow.keras.layers import LSTM, GRU
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers.schedules import ExponentialDecay
import tensorflow as tf

# create LSTM
x = LSTM(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
lstm_model = Model(inputs=input_holder,outputs=x)

# create GRU
x = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
gru_model = Model(inputs=input_holder,outputs=x)

# lr_schedule = ExponentialDecay(
#     initial_learning_rate=0.1,
#     decay_steps=10000,
#     decay_rate=0.95)

opt = Adam(lr=0.0001, epsilon=0.00005, clipnorm=1.0)

simple_rnn_model.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

lstm_model.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=['Recall'])

gru_model.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

print(simple_rnn_model.summary())
print(lstm_model.summary())
print(gru_model.summary())
```

```
WARNING:tensorflow:Layer lstm_6 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer gru_16 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 250)]             0

 embedding_6 (Embedding)     (None, 250, 50)           1867650

 simple_rnn_4 (SimpleRNN)    (None, 100)               15100

 dense_26 (Dense)            (None, 1)                 101

=================================================================
Total params: 1,882,851
Trainable params: 1,882,851
Non-trainable params: 0
_________________________________________________________________
None
Model: "model_23"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 250)]             0

 embedding_6 (Embedding)     (None, 250, 50)           1867650

 lstm_6 (LSTM)               (None, 100)               60400

 dense_27 (Dense)            (None, 1)                 101

=================================================================
Total params: 1,928,151
Trainable params: 1,928,151
Non-trainable params: 0
_________________________________________________________________
None
Model: "model_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 250)]             0

 embedding_6 (Embedding)     (None, 250, 50)           1867650

 gru_16 (GRU)                (None, 100)               45600

 dense_28 (Dense)            (None, 1)                 101

=================================================================
Total params: 1,913,351
Trainable params: 1,913,351
Non-trainable params: 0
_________________________________________________________________
None
/Users/ycd17/miniforge3/envs/tf/lib/python3.8/site-packages/keras/optimizers/optimizer_v2/adam.py:114: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

Fit the first LSTM Model (MODEL 1)

In [142…

```
import time
```

In [143…

```
start_time = time.time()

with tf.device("/cpu:0"):
    history_lstm=lstm_model.fit(X_train, y_train_ohe, epochs=8, batch_size=32, validation_data=(X_test, y_test_ohe))

end_time = time.time()

elapsed_time_LSTM1 = end_time - start_time
```

```
Epoch 1/8
2022-12-14 17:15:47.005882: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - ETA: 0s - loss: 0.6922 - recall: 0.4250
2022-12-14 17:16:10.033237: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 26s 362ms/step - loss: 0.6922 - recall: 0.4250 - val_loss: 0.6911 - val_recall: 0.5876
Epoch 2/8
69/69 [==============================] - 25s 356ms/step - loss: 0.6898 - recall: 0.6846 - val_loss: 0.6880 - val_recall: 0.7372
Epoch 3/8
69/69 [==============================] - 24s 351ms/step - loss: 0.6846 - recall: 0.7952 - val_loss: 0.6799 - val_recall: 0.8212
Epoch 4/8
69/69 [==============================] - 25s 359ms/step - loss: 0.6135 - recall: 0.8547 - val_loss: 0.4190 - val_recall: 0.9526
Epoch 5/8
69/69 [==============================] - 24s 354ms/step - loss: 0.3757 - recall: 0.8940 - val_loss: 0.2884 - val_recall: 0.9489
Epoch 6/8
69/69 [==============================] - 24s 354ms/step - loss: 0.2167 - recall: 0.9616 - val_loss: 0.1821 - val_recall: 0.9562
Epoch 7/8
69/69 [==============================] - 24s 352ms/step - loss: 0.1380 - recall: 0.9771 - val_loss: 0.1762 - val_recall: 0.9708
Epoch 8/8
69/69 [==============================] - 25s 356ms/step - loss: 0.1041 - recall: 0.9872 - val_loss: 0.1547 - val_recall: 0.9745
```

In [144…

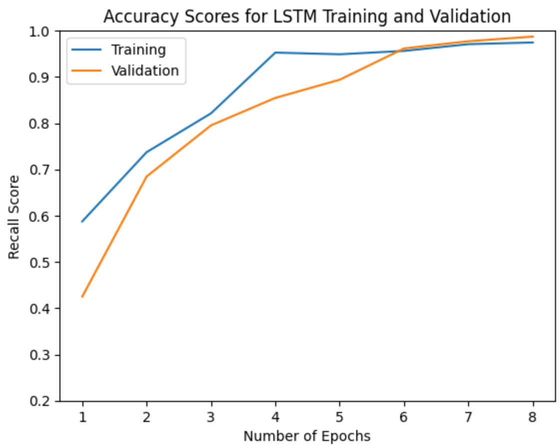
```
print("Elapsed time:", elapsed_time_LSTM1, 'seconds')
```

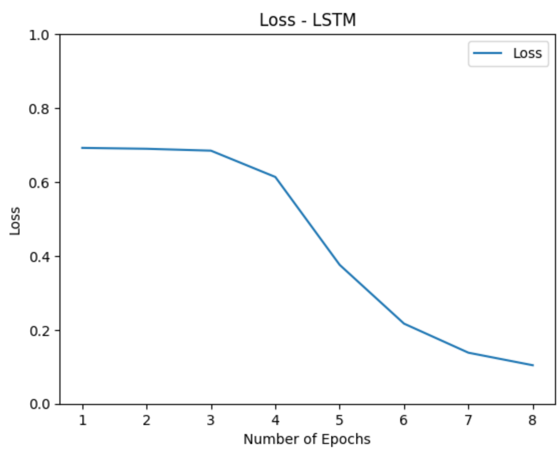
```
Elapsed time: 197.23249411582947 seconds
```

In [145…

```
import matplotlib.pyplot as plt
lstm_acc_training=history_lstm.history['recall']
lstm_acc=history_lstm.history['val_recall']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,lstm_acc)
plt.plot(epoch,lstm_acc_training)
plt.title('Accuracy Scores for LSTM Training and Validation')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Training", "Validation"])
```

```
plt.show()


lstm_loss=history_lstm.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,lstm_loss)
plt.title('Loss - LSTM')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





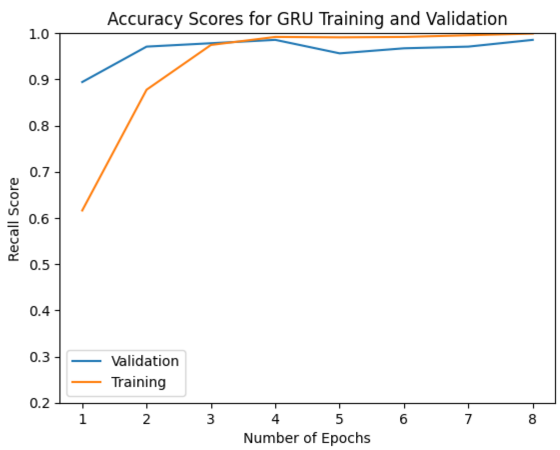
Fit first GRU model (MODEL 2)

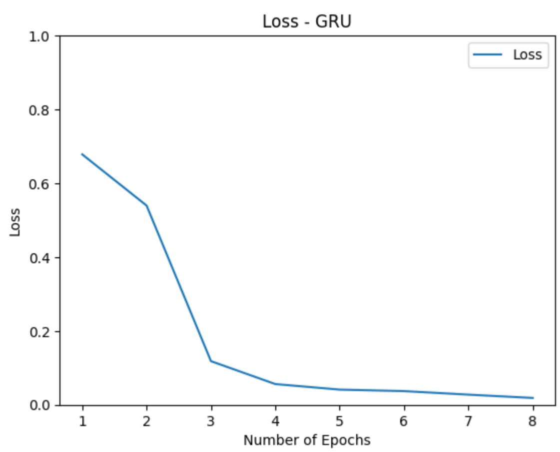
In [23]:
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_gru=gru_model.fit(X_train, y_train_ohe, epochs=8, batch_size=32, validation_data=(X_test, y_test_ohe))

    end_time = time.time()

elapsed_time_GRU1 = end_time - start_time
print("Elapsed time:", elapsed_time_GRU1, 'seconds')
```

```
Epoch 1/8
2022-12-13 23:47:44.482979: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - ETA: 0s - loss: 0.6782 - recall_2: 0.6161
2022-12-13 23:48:02.626399: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 21s 290ms/step - loss: 0.6782 - recall_2: 0.6161 - val_loss: 0.6559 - val_recall_2: 0.8942
Epoch 2/8
69/69 [==============================] - 20s 285ms/step - loss: 0.5395 - recall_2: 0.8775 - val_loss: 0.1453 - val_recall_2: 0.9708
Epoch 3/8
69/69 [==============================] - 21s 302ms/step - loss: 0.1181 - recall_2: 0.9744 - val_loss: 0.0865 - val_recall_2: 0.9781
Epoch 4/8
69/69 [==============================] - 21s 302ms/step - loss: 0.0561 - recall_2: 0.9918 - val_loss: 0.0876 - val_recall_2: 0.9854
Epoch 5/8
69/69 [==============================] - 21s 302ms/step - loss: 0.0411 - recall_2: 0.9909 - val_loss: 0.0847 - val_recall_2: 0.9562
Epoch 6/8
69/69 [==============================] - 21s 308ms/step - loss: 0.0370 - recall_2: 0.9918 - val_loss: 0.0654 - val_recall_2: 0.9672
Epoch 7/8
69/69 [==============================] - 23s 334ms/step - loss: 0.0275 - recall_2: 0.9954 - val_loss: 0.0443 - val_recall_2: 0.9708
Epoch 8/8
69/69 [==============================] - 21s 298ms/step - loss: 0.0185 - recall_2: 0.9991 - val_loss: 0.0447 - val_recall_2: 0.9854
Elapsed time: 167.90670108795166 seconds
```

In [60]:
```
import matplotlib.pyplot as plt
gru_acc_training=history_gru.history['recall_2']
gru_acc=history_gru.history['val_recall_2']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc)
plt.plot(epoch,gru_acc_training)
```

```
plt.title('Accuracy Scores for GRU Training and Validation')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
plt.show()


import matplotlib.pyplot as plt
gru_loss=history_gru.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_loss)
plt.title('Loss - GRU')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





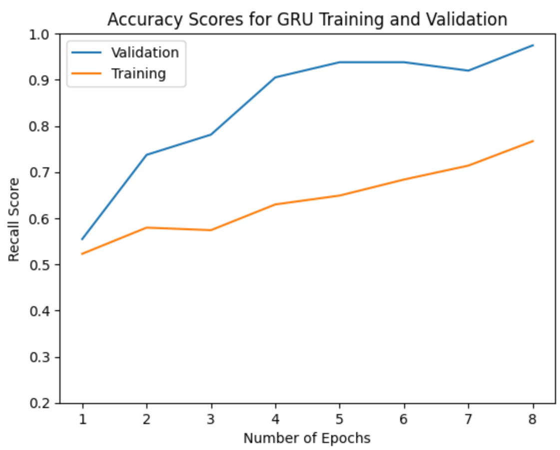
Fit the Simple RRN (Model 3)

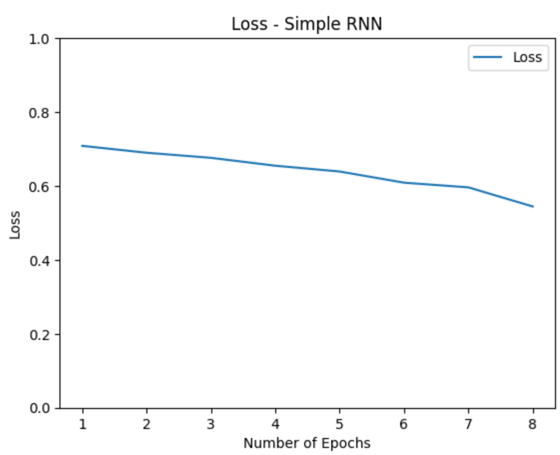
In [24]:
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_simple_rnn=simple_rnn_model.fit(X_train, y_train_ohe, epochs=8, batch_size=32, validation_data=(X_test, y_test_ohe))

    end_time = time.time()

elapsed_time_simple_RNN = end_time - start_time
print("Elapsed time:", elapsed_time_simple_RNN, 'seconds')
```

```
Epoch 1/8
2022-12-13 23:57:59.043018: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [============================>.] - ETA: 0s - loss: 0.7085 - recall: 0.5229
2022-12-13 23:58:08.398368: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
```

```
69/69 [==============================] - 11s 154ms/step - loss: 0.7085 - recall: 0.5229 - val_loss: 0.6823 - val_recall: 0.5547
Epoch 2/8
69/69 [==============================] - 11s 165ms/step - loss: 0.6898 - recall: 0.5795 - val_loss: 0.6586 - val_recall: 0.7372
Epoch 3/8
69/69 [==============================] - 11s 153ms/step - loss: 0.6761 - recall: 0.5740 - val_loss: 0.6413 - val_recall: 0.7810
Epoch 4/8
69/69 [==============================] - 11s 159ms/step - loss: 0.6548 - recall: 0.6298 - val_loss: 0.6142 - val_recall: 0.9051
Epoch 5/8
69/69 [==============================] - 10s 151ms/step - loss: 0.6391 - recall: 0.6490 - val_loss: 0.5820 - val_recall: 0.9380
Epoch 6/8
69/69 [==============================] - 10s 152ms/step - loss: 0.6089 - recall: 0.6837 - val_loss: 0.5411 - val_recall: 0.9380
Epoch 7/8
69/69 [==============================] - 11s 157ms/step - loss: 0.5962 - recall: 0.7139 - val_loss: 0.4517 - val_recall: 0.9197
Epoch 8/8
69/69 [==============================] - 11s 157ms/step - loss: 0.5444 - recall: 0.7669 - val_loss: 0.5229 - val_recall: 0.9745
Elapsed time: 86.64590787887573 seconds
```

In [61]:
```python
import matplotlib.pyplot as plt
simple_acc_training=history_simple_rnn.history['recall']
simple_acc=history_simple_rnn.history['val_recall']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,simple_acc)
plt.plot(epoch,simple_acc_training)
plt.title('Accuracy Scores for GRU Training and Validation')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
plt.show()


import matplotlib.pyplot as plt
simple_loss=history_simple_rnn.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,simple_loss)
plt.title('Loss - Simple RNN')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```



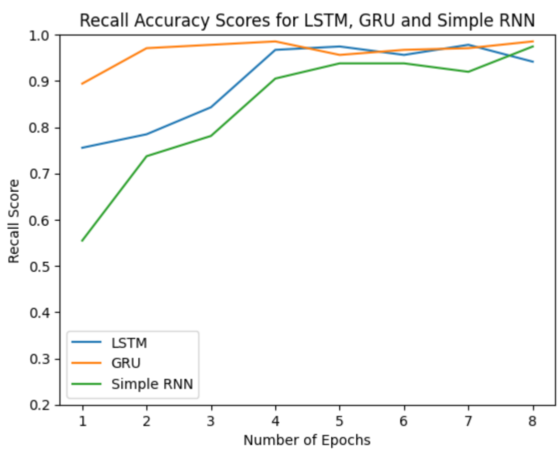


Comparing first three models:

In [63]:
```python
import matplotlib.pyplot as plt
gru_acc=history_gru.history['val_recall_2']
lstm_acc=history_lstm.history['val_recall_1']
simple_rnn_acc=history_simple_rnn.history['val_recall']
```

```
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,lstm_acc)
plt.plot(epoch,gru_acc)
plt.plot(epoch,simple_rnn_acc)
plt.title('Recall Accuracy Scores for LSTM, GRU and Simple RNN')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["LSTM", "GRU", "Simple RNN"])
```

Out[63]: `<matplotlib.legend.Legend at 0x4ee8358e0>`



For our dataset, GRU has a better performance and it is trained faster.

In general, both GRUs and LSTMs are types of recurrent neural networks (RNNs) that are commonly used for tasks involving sequential data, such as natural language processing or time series forecasting. GRUs and LSTMs can both be effective for these types of tasks, but GRUs may be more suitable for certain types of datasets. For example, GRUs can be more efficient to train than LSTMs, and may be better suited for datasets with longer sequences or larger vocabularies.

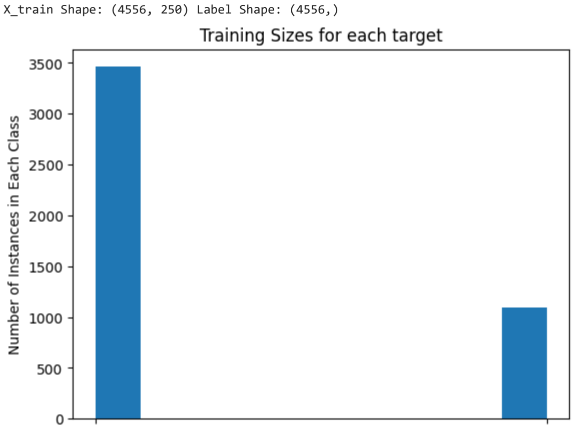
However, LSTMs can have superior performance on some tasks, such as language translation or handwriting recognition. Ultimately, the choice between using a GRU or an LSTM will depend on the specific characteristics of the dataset and the goals of the model and it seems like it is better to use GRU in this dataset.

Trial Without Undersampling:

In [109…

```
import math
from sklearn.model_selection import train_test_split
from sklearn import metrics as mt
from matplotlib import pyplot as plt
%matplotlib inline

# Split it into train / test subsets
X_train_all, X_test_all, y_train_ohe_all, y_test_ohe_all = train_test_split(X, y_ohe, test_size=0.2,
                                                                              stratify=y_ohe,
                                                                              random_state=42)
NUM_CLASSES = 1

# print some stats of the data
print("X_train Shape:",X_train_all.shape, "Label Shape:", y_train_ohe_all.shape)
uniq_classes = np.sum(y_train_ohe_all,axis=0)
plt.hist(y_train_ohe_all)
plt.ylabel("Number of Instances in Each Class")

new_list = range(math.floor(min(y_test_ohe_all)), math.ceil(max(y_test_ohe_all))+1)
plt.xticks(new_list)
plt.title("Training Sizes for each target")


plt.show()
```

```
X_train Shape: (4556, 250) Label Shape: (4556,)
```

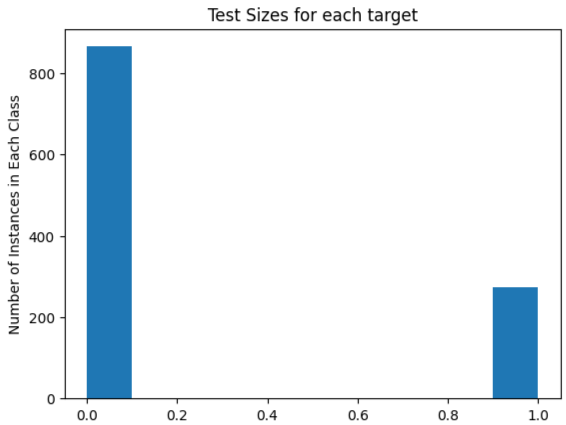


In [110…

```
uniq_classes = np.sum(y_test_ohe_all,axis=0)
plt.hist(y_test_ohe_all)
```

```
plt.title("Test Sizes for each target")
plt.ylabel("Number of Instances in Each Class")
```

Out[110]: Text(0, 0.5, 'Number of Instances in Each Class')



Architectures (Same Architectures with the previous part)

Simple RNN Model

In [111…

```
# show example without the FOR loop
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Input, SimpleRNN
from tensorflow.keras.layers import Embedding

EMBED_SIZE = 50
RNN_STATESIZE = 100
rnns = []
input_holder = Input(shape=(X_train_all.shape[1], ))
shared_embed = Embedding(top_words, # input dimension (max int of OHE)
                EMBED_SIZE, # output dimension size
                input_length=MAX_ART_LEN)(input_holder) # number of words in each sequence


x = SimpleRNN(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
simple_rnn_model = Model(inputs=input_holder,outputs=x)
```

Create first LSTM and GRU Models. Compile Simple RNN, LSTM and GRU

Please note that these values are tuned after checking the validation/training accuracy. Best models are used.

In [112…

```
from tensorflow.keras.layers import LSTM, GRU
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers.schedules import ExponentialDecay
import tensorflow as tf

# create LSTM
x = LSTM(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
lstm_model = Model(inputs=input_holder,outputs=x)

# create GRU
x = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
gru_model = Model(inputs=input_holder,outputs=x)

# lr_schedule = ExponentialDecay(
#     initial_learning_rate=0.1,
#     decay_steps=10000,
#     decay_rate=0.95)

opt = Adam(lr=0.0001, epsilon=0.00005, clipnorm=1.0)

simple_rnn_model.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

lstm_model.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

gru_model.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

print(simple_rnn_model.summary())
print(lstm_model.summary())
print(gru_model.summary())
```

```
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer gru_13 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 250)]             0

 embedding_5 (Embedding)     (None, 250, 50)           1867650

 simple_rnn_3 (SimpleRNN)    (None, 100)               15100

 dense_19 (Dense)            (None, 1)                 101

=================================================================
Total params: 1,882,851
Trainable params: 1,882,851
Non-trainable params: 0
_________________________________________________________________
None
Model: "model_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 250)]             0

 embedding_5 (Embedding)     (None, 250, 50)           1867650

 lstm_3 (LSTM)               (None, 100)               60400

 dense_20 (Dense)            (None, 1)                 101

=================================================================
Total params: 1,928,151
Trainable params: 1,928,151
Non-trainable params: 0
_________________________________________________________________
None
Model: "model_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 250)]             0

 embedding_5 (Embedding)     (None, 250, 50)           1867650

 gru_13 (GRU)                (None, 100)               45600

 dense_21 (Dense)            (None, 1)                 101

=================================================================
Total params: 1,913,351
Trainable params: 1,913,351
Non-trainable params: 0
_________________________________________________________________
None
/Users/ycd17/miniforge3/envs/tf/lib/python3.8/site-packages/keras/optimizers/optimizer_v2/adam.py:114: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

Fit the first LSTM Architecture with the original data (Not undersampled) (MODEL 1)

In [113…

```
start_time = time.time()

with tf.device("/cpu:0"):
    history_lstm_all=lstm_model.fit(X_train_all, y_train_ohe_all, epochs=8, batch_size=8, validation_data=(X_test_all, y_test_ohe_all))

end_time = time.time()

elapsed_time_LSTM1_all = end_time - start_time
```

```
Epoch 1/8
2022-12-14 16:39:14.025191: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
570/570 [==============================] - ETA: 0s - loss: 0.4882 - recall_19: 0.2395
2022-12-14 16:41:47.379892: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
570/570 [==============================] - 170s 296ms/step - loss: 0.4882 - recall_19: 0.2395 - val_loss: 0.2120 - val_recall_19: 0.8905
Epoch 2/8
570/570 [==============================] - 166s 292ms/step - loss: 0.1821 - recall_19: 0.8684 - val_loss: 0.1367 - val_recall_19: 0.8978
Epoch 3/8
570/570 [==============================] - 170s 299ms/step - loss: 0.1273 - recall_19: 0.9150 - val_loss: 0.1531 - val_recall_19: 0.9562
Epoch 4/8
570/570 [==============================] - 170s 299ms/step - loss: 0.1040 - recall_19: 0.9397 - val_loss: 0.1712 - val_recall_19: 0.9526
Epoch 5/8
570/570 [==============================] - 167s 293ms/step - loss: 0.0948 - recall_19: 0.9452 - val_loss: 0.1016 - val_recall_19: 0.9562
Epoch 6/8
570/570 [==============================] - 167s 294ms/step - loss: 0.0692 - recall_19: 0.9644 - val_loss: 0.1014 - val_recall_19: 0.9599
Epoch 7/8
570/570 [==============================] - 169s 296ms/step - loss: 0.0728 - recall_19: 0.9644 - val_loss: 0.1014 - val_recall_19: 0.9526
Epoch 8/8
570/570 [==============================] - 170s 298ms/step - loss: 0.0732 - recall_19: 0.9543 - val_loss: 0.0939 - val_recall_19: 0.9416
```

In [114…

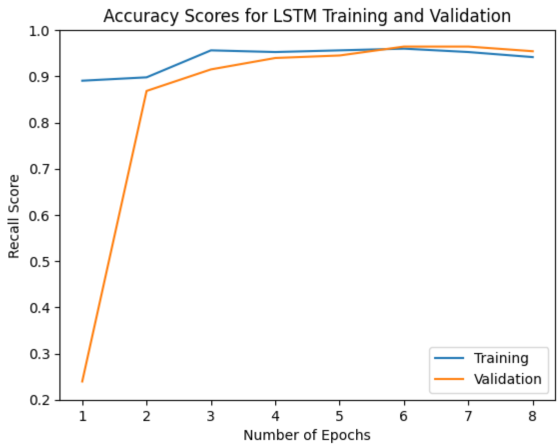
```
print("Elapsed time:", elapsed_time_LSTM1_all, 'seconds')
```

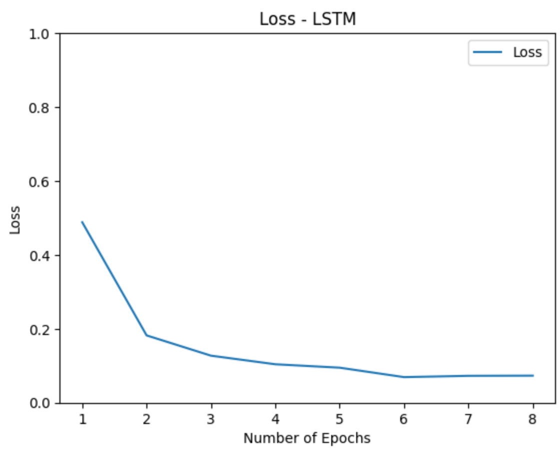
```
Elapsed time: 1349.3080270290375 seconds
```

In [118…

```
import matplotlib.pyplot as plt
lstm_acc_all_training=history_lstm_all.history['recall_19']
lstm_acc_all=history_lstm_all.history['val_recall_19']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,lstm_acc_all)
plt.plot(epoch,lstm_acc_all_training)
plt.title('Accuracy Scores for LSTM Training and Validation')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Training", "Validation"])
plt.show()
```

```
lstm_loss_all=history_lstm_all.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,lstm_loss)
plt.title('Loss - LSTM')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





In [146…

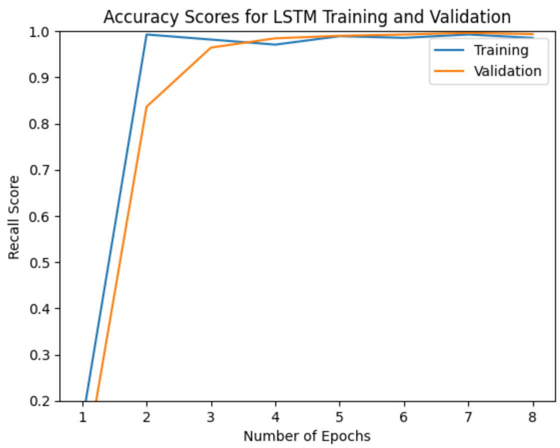
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_gru_all=gru_model.fit(X_train_all, y_train_ohe_all, epochs=8, batch_size=32, validation_data=(X_test_all, y_test_ohe_all))

    end_time = time.time()

elapsed_time_GRU1_all = end_time - start_time
print("Elapsed time:", elapsed_time_GRU1, 'seconds')
```

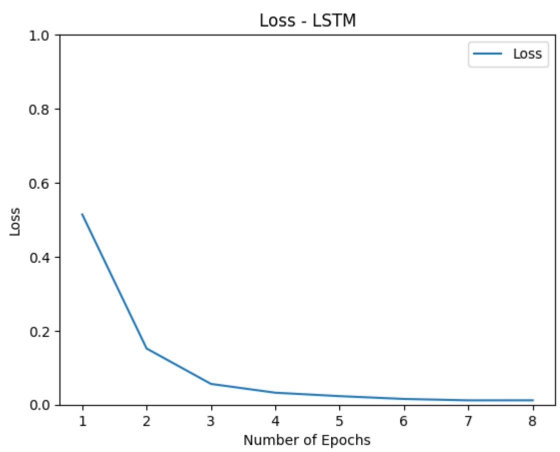
```
Epoch 1/8
2022-12-14 17:24:03.558961: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
143/143 [==============================] - ETA: 0s - loss: 0.5142 - recall_27: 0.0238
2022-12-14 17:24:49.259061: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
143/143 [==============================] - 51s 352ms/step - loss: 0.5142 - recall_27: 0.0238 - val_loss: 0.3547 - val_recall_27: 0.1533
Epoch 2/8
143/143 [==============================] - 49s 344ms/step - loss: 0.1518 - recall_27: 0.8364 - val_loss: 0.0493 - val_recall_27: 0.9927
Epoch 3/8
143/143 [==============================] - 49s 343ms/step - loss: 0.0561 - recall_27: 0.9644 - val_loss: 0.0445 - val_recall_27: 0.9818
Epoch 4/8
143/143 [==============================] - 49s 342ms/step - loss: 0.0323 - recall_27: 0.9845 - val_loss: 0.0315 - val_recall_27: 0.9708
Epoch 5/8
143/143 [==============================] - 48s 339ms/step - loss: 0.0230 - recall_27: 0.9899 - val_loss: 0.0234 - val_recall_27: 0.9891
Epoch 6/8
143/143 [==============================] - 311s 2s/step - loss: 0.0155 - recall_27: 0.9927 - val_loss: 0.0273 - val_recall_27: 0.9854
Epoch 7/8
143/143 [==============================] - 48s 337ms/step - loss: 0.0117 - recall_27: 0.9963 - val_loss: 0.0170 - val_recall_27: 0.9927
Epoch 8/8
143/143 [==============================] - 49s 341ms/step - loss: 0.0118 - recall_27: 0.9936 - val_loss: 0.0168 - val_recall_27: 0.9854
Elapsed time: 167.90670108795166 seconds
```

In [147…

```
import matplotlib.pyplot as plt
gru_acc_all_training=history_gru_all.history['recall_27']
gru_acc_all=history_gru_all.history['val_recall_27']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_all)
plt.plot(epoch,gru_acc_all_training)
plt.title('Accuracy Scores for LSTM Training and Validation')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
```

```
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Training", "Validation"])
plt.show()


gru_loss_all=history_gru_all.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_loss_all)
plt.title('Loss - LSTM')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```

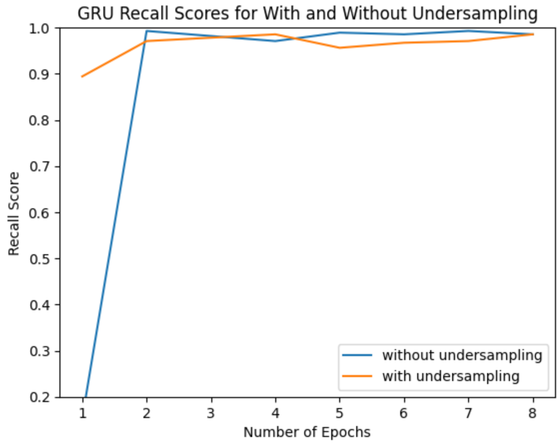




Performance with and without Undersampling (LSTM)

In [151…

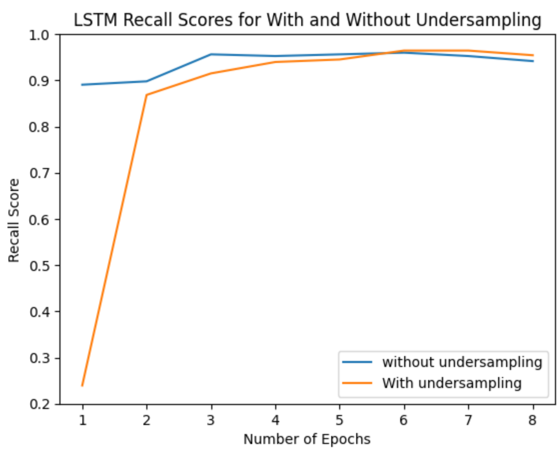
```
plt.plot(epoch,gru_acc_all)
plt.plot(epoch,gru_acc)
plt.title('GRU Recall Scores for With and Without Undersampling')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["without undersampling", "with undersampling"])
plt.show()
```



Performance with and without Undersampling (GRU)

In [150…

```
plt.plot(epoch,lstm_acc_all)
plt.plot(epoch,lstm_acc_all_training)
plt.title('LSTM Recall Scores for With and Without Undersampling')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["without undersampling", "With undersampling"])
plt.show()
```



In terms of differences, LSTMs have a more complex architecture than GRUs, with more gates and memory cells. This makes LSTMs more powerful and able to capture more complex dependencies in the data, but also makes them more computationally expensive and harder to train.

GRUs, on the other hand, have a simpler architecture and are therefore faster and easier to train, but may not be as effective at capturing long-term dependencies.

In general, LSTMs are considered to be the more powerful and effective of the two, but whether they are the best choice for a particular task depends on the specific details of the problem at hand.

From our results, GRU performs better than LSTM. We think that our model is not complex enough to require usage of LSTM.

Trial with Stratified K-Fold

LSTM with K-Fold

In [152…

```
def average_values(lists):
    # Calculate the length of the longest list
    max_len = 0
    for l in lists:
        max_len = max(max_len, len(l))

    # Create a new array to hold the averages
    averages = []

    # Iterate over the items in the longest list
    for i in range(max_len):
        # Calculate the sum of the values on the same position in all lists
        sum = 0
        for l in lists:
            if i < len(l):
                sum += l[i]

        # Calculate the average value and add it to the averages array
        averages.append(sum / len(lists))

    # Return the averages array
    return averages
```

In [156…

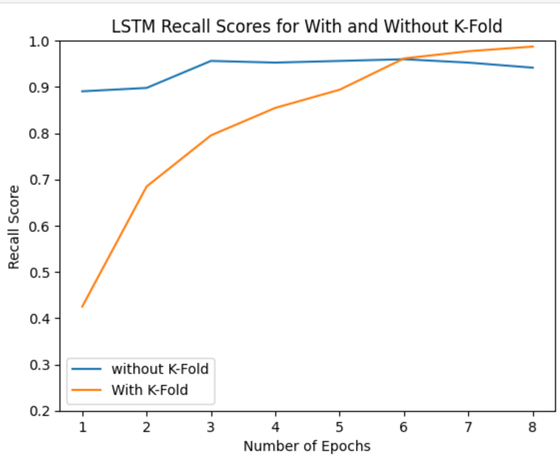
```
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from sklearn.model_selection import StratifiedKFold

acc_per_fold = []
loss_per_fold = []
acc_per_epoch = []

# Define the K-fold Cross Validator
skf = StratifiedKFold(n_splits=5, shuffle = True, random_state = 42)
# K-fold Cross Validation model evaluation
fold_no = 1
start_time = time.time()

for train, test in skf.split(X_resampled, y_resampled):

    # Define the model architecture
    x = LSTM(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
    x = Dense(NUM_CLASSES, activation='sigmoid')(x)
    lstm_model = Model(inputs=input_holder,outputs=x)
    # Compile the model
    lstm_model.compile(loss='binary_crossentropy',
                  optimizer= opt,
                  metrics='Recall')

    # Generate a print
    print('------------------------------------------------------------------------')
    print(f'Training for fold {fold_no} ...')

    # Fit data to model


    with tf.device("/cpu:0"):
        history_lstm_kfold = lstm_model.fit(X_resampled[train], y_resampled[train],
                                batch_size=32,
                                epochs=8)

    # Generate generalization metrics
    scores = lstm_model.evaluate(X_resampled[test], y_resampled[test], verbose=0)
    print(f'Score for fold {fold_no}: {lstm_model.metrics_names[0]} of {scores[0]}; '
          f'{lstm_model.metrics_names[1]} of {scores[1]*100}%')
    acc_per_fold.append(scores[1] * 100)
    loss_per_fold.append(scores[0])
    acc_per_epoch.append(history_lstm.history['recall'])

    # Increase fold number
    fold_no = fold_no + 1

end_time = time.time()

elapsed_time_LSTM_kfold = end_time - start_time
average_score_per_epoch_lstm_kfold = average_values(acc_per_epoch)
```

```
WARNING:tensorflow:Layer lstm_10 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 1 ...
Epoch 1/8
2022-12-14 17:54:46.741675: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 24s 339ms/step - loss: 0.5043 - recall: 0.9287
Epoch 2/8
69/69 [==============================] - 23s 340ms/step - loss: 0.1201 - recall: 0.9790
Epoch 3/8
69/69 [==============================] - 23s 338ms/step - loss: 0.0530 - recall: 0.9945
Epoch 4/8
69/69 [==============================] - 23s 338ms/step - loss: 0.0659 - recall: 0.9927
Epoch 5/8
69/69 [==============================] - 23s 334ms/step - loss: 0.0367 - recall: 0.9963
Epoch 6/8
69/69 [==============================] - 23s 334ms/step - loss: 0.0471 - recall: 0.9899
Epoch 7/8
69/69 [==============================] - 23s 329ms/step - loss: 0.0361 - recall: 0.9936
Epoch 8/8
69/69 [==============================] - 23s 329ms/step - loss: 0.0228 - recall: 0.9963
2022-12-14 17:57:52.413837: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 1: loss of 0.05741911754012108; recall of 98.9051103591919%
WARNING:tensorflow:Layer lstm_11 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 2 ...
Epoch 1/8
2022-12-14 17:58:41.746899: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 26s 344ms/step - loss: 0.4503 - recall: 0.8904
Epoch 2/8
69/69 [==============================] - 23s 338ms/step - loss: 0.0830 - recall: 0.9936
Epoch 3/8
69/69 [==============================] - 23s 337ms/step - loss: 0.0437 - recall: 0.9973
Epoch 4/8
69/69 [==============================] - 23s 332ms/step - loss: 0.0634 - recall: 0.9872
Epoch 5/8
69/69 [==============================] - 23s 335ms/step - loss: 0.0343 - recall: 0.9936
Epoch 6/8
69/69 [==============================] - 22s 326ms/step - loss: 0.0365 - recall: 0.9945
Epoch 7/8
69/69 [==============================] - 23s 330ms/step - loss: 0.0482 - recall: 0.9936
Epoch 8/8
69/69 [==============================] - 23s 327ms/step - loss: 0.0295 - recall: 0.9936
2022-12-14 18:01:46.499711: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 2: loss of 0.03222418576478958; recall of 99.63369965553284%
WARNING:tensorflow:Layer lstm_12 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 3 ...
Epoch 1/8
2022-12-14 18:02:35.065387: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
```

```
69/69 [==============================] - 25s 340ms/step - loss: 0.4817 - recall: 0.7315
Epoch 2/8
69/69 [==============================] - 24s 340ms/step - loss: 0.1060 - recall: 0.9763
Epoch 3/8
69/69 [==============================] - 23s 339ms/step - loss: 0.0545 - recall: 0.9963
Epoch 4/8
69/69 [==============================] - 23s 337ms/step - loss: 0.0422 - recall: 0.9973
Epoch 5/8
69/69 [==============================] - 23s 338ms/step - loss: 0.0441 - recall: 0.9927
Epoch 6/8
69/69 [==============================] - 23s 335ms/step - loss: 0.0267 - recall: 0.9963
Epoch 7/8
69/69 [==============================] - 23s 337ms/step - loss: 0.0428 - recall: 0.9890
Epoch 8/8
69/69 [==============================] - 27s 399ms/step - loss: 0.0257 - recall: 0.9963
2022-12-14 18:05:46.377148: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 3: loss of 0.03632409870624542; recall of 99.26739931106567%
WARNING:tensorflow:Layer lstm_13 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 4 ...
Epoch 1/8
2022-12-14 18:06:35.427934: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 26s 356ms/step - loss: 0.4201 - recall: 0.9388
Epoch 2/8
69/69 [==============================] - 25s 361ms/step - loss: 0.0649 - recall: 0.9872
Epoch 3/8
69/69 [==============================] - 25s 355ms/step - loss: 0.0316 - recall: 0.9973
Epoch 4/8
69/69 [==============================] - 25s 366ms/step - loss: 0.0282 - recall: 0.9954
Epoch 5/8
69/69 [==============================] - 25s 356ms/step - loss: 0.0316 - recall: 0.9954
Epoch 6/8
69/69 [==============================] - 24s 347ms/step - loss: 0.0336 - recall: 0.9963
Epoch 7/8
69/69 [==============================] - 24s 348ms/step - loss: 0.0242 - recall: 0.9973
Epoch 8/8
69/69 [==============================] - 24s 344ms/step - loss: 0.0218 - recall: 0.9963
2022-12-14 18:09:51.654864: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 4: loss of 0.013859175145626068; recall of 99.6350347995758%
WARNING:tensorflow:Layer lstm_14 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 5 ...
Epoch 1/8
2022-12-14 18:10:40.759217: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 27s 377ms/step - loss: 0.3790 - recall: 0.9735
Epoch 2/8
69/69 [==============================] - 25s 366ms/step - loss: 0.0683 - recall: 0.9808
Epoch 3/8
69/69 [==============================] - 25s 365ms/step - loss: 0.0172 - recall: 0.9973
Epoch 4/8
69/69 [==============================] - 25s 356ms/step - loss: 0.0218 - recall: 0.9963
Epoch 5/8
69/69 [==============================] - 24s 351ms/step - loss: 0.0183 - recall: 0.9963
Epoch 6/8
69/69 [==============================] - 25s 355ms/step - loss: 0.0222 - recall: 0.9954
Epoch 7/8
69/69 [==============================] - 25s 362ms/step - loss: 0.0172 - recall: 0.9973
Epoch 8/8
69/69 [==============================] - 24s 351ms/step - loss: 0.0186 - recall: 0.9973
2022-12-14 18:14:00.325968: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 5: loss of 0.005780613049864769; recall of 100.0%
```

In [160…

```python
plt.plot(epoch,lstm_acc_all)
plt.plot(epoch,average_score_per_epoch_lstm_kfold)
plt.title('LSTM Recall Scores for With and Without K-Fold')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["without K-Fold", "With K-Fold"])
plt.show()
```



GRU with K-Fold

In [157…

```python
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from sklearn.model_selection import StratifiedKFold

acc_per_fold = []
loss_per_fold = []
```

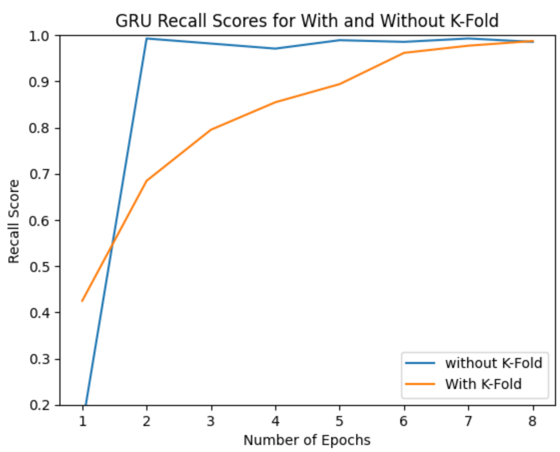
```python
acc_per_epoch = []

# Define the K-fold Cross Validator
skf = StratifiedKFold(n_splits=5, shuffle = True, random_state = 42)
# K-fold Cross Validation model evaluation
fold_no = 1
start_time = time.time()

for train, test in skf.split(X_resampled, y_resampled):

    # Define the model architecture
    x = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
    x = Dense(NUM_CLASSES, activation='sigmoid')(x)
    lstm_model = Model(inputs=input_holder,outputs=x)
    # Compile the model
    lstm_model.compile(loss='binary_crossentropy',
                  optimizer= opt,
                  metrics='Recall')

    # Generate a print
    print('------------------------------------------------------------------------')
    print(f'Training for fold {fold_no} ...')

    # Fit data to model


    with tf.device("/cpu:0"):
        history_lstm_kfold = lstm_model.fit(X_resampled[train], y_resampled[train],
                                  batch_size=32,
                                  epochs=8)

    # Generate generalization metrics
    scores = lstm_model.evaluate(X_resampled[test], y_resampled[test], verbose=0)
    print(f'Score for fold {fold_no}: {lstm_model.metrics_names[0]} of {scores[0]}; '
          f'{lstm_model.metrics_names[1]} of {scores[1]*100}%')
    acc_per_fold.append(scores[1] * 100)
    loss_per_fold.append(scores[0])
    acc_per_epoch.append(history_lstm.history['recall'])

    # Increase fold number
    fold_no = fold_no + 1

end_time = time.time()

elapsed_time_GRU_kfold = end_time - start_time
average_score_per_epoch_gru_kfold = average_values(acc_per_epoch)
```

```
WARNING:tensorflow:Layer gru_17 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 1 ...
Epoch 1/8
2022-12-14 18:14:54.104684: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 26s 367ms/step - loss: 0.4828 - recall: 0.9196
Epoch 2/8
69/69 [==============================] - 24s 350ms/step - loss: 0.0627 - recall: 0.9771
Epoch 3/8
69/69 [==============================] - 24s 343ms/step - loss: 0.0174 - recall: 0.9982
Epoch 4/8
69/69 [==============================] - 24s 346ms/step - loss: 0.0102 - recall: 1.0000
Epoch 5/8
69/69 [==============================] - 24s 348ms/step - loss: 0.0092 - recall: 0.9982
Epoch 6/8
69/69 [==============================] - 23s 339ms/step - loss: 0.0056 - recall: 1.0000
Epoch 7/8
69/69 [==============================] - 23s 336ms/step - loss: 0.0048 - recall: 1.0000
Epoch 8/8
69/69 [==============================] - 23s 336ms/step - loss: 0.0058 - recall: 1.0000
2022-12-14 18:18:05.461333: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 1: loss of 0.011952636763453484; recall of 99.6350347995758%
WARNING:tensorflow:Layer gru_18 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 2 ...
Epoch 1/8
2022-12-14 18:19:00.906069: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 26s 363ms/step - loss: 0.5004 - recall: 0.9333
Epoch 2/8
69/69 [==============================] - 25s 360ms/step - loss: 0.0260 - recall: 0.9963
Epoch 3/8
69/69 [==============================] - 24s 355ms/step - loss: 0.0116 - recall: 0.9991
Epoch 4/8
69/69 [==============================] - 24s 340ms/step - loss: 0.0093 - recall: 0.9991
Epoch 5/8
69/69 [==============================] - 24s 346ms/step - loss: 0.0099 - recall: 0.9991
Epoch 6/8
69/69 [==============================] - 24s 342ms/step - loss: 0.0056 - recall: 1.0000
Epoch 7/8
69/69 [==============================] - 24s 341ms/step - loss: 0.0052 - recall: 0.9991
Epoch 8/8
69/69 [==============================] - 23s 335ms/step - loss: 0.0038 - recall: 1.0000
2022-12-14 18:22:13.669654: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 2: loss of 0.002729437779635191; recall of 100.0%
WARNING:tensorflow:Layer gru_19 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 3 ...
Epoch 1/8
2022-12-14 18:23:09.662869: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 27s 368ms/step - loss: 0.4888 - recall: 0.9315
Epoch 2/8
69/69 [==============================] - 25s 361ms/step - loss: 0.0293 - recall: 0.9963
Epoch 3/8
69/69 [==============================] - 24s 348ms/step - loss: 0.0115 - recall: 0.9991
Epoch 4/8
69/69 [==============================] - 24s 347ms/step - loss: 0.0085 - recall: 0.9991
Epoch 5/8
69/69 [==============================] - 24s 346ms/step - loss: 0.0058 - recall: 1.0000
Epoch 6/8
69/69 [==============================] - 24s 348ms/step - loss: 0.0053 - recall: 1.0000
Epoch 7/8
69/69 [==============================] - 24s 343ms/step - loss: 0.0048 - recall: 1.0000
Epoch 8/8
69/69 [==============================] - 23s 339ms/step - loss: 0.0037 - recall: 1.0000
```

```
2022-12-14 18:26:23.535264: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 3: loss of 0.0035703713074326515; recall of 100.0%
WARNING:tensorflow:Layer gru_20 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 4 ...
Epoch 1/8
2022-12-14 18:27:18.719207: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 26s 368ms/step - loss: 0.4701 - recall: 0.9653
Epoch 2/8
69/69 [==============================] - 25s 364ms/step - loss: 0.0410 - recall: 0.9963
Epoch 3/8
69/69 [==============================] - 24s 353ms/step - loss: 0.0088 - recall: 1.0000
Epoch 4/8
69/69 [==============================] - 24s 342ms/step - loss: 0.0088 - recall: 0.9991
Epoch 5/8
69/69 [==============================] - 25s 355ms/step - loss: 0.0071 - recall: 0.9991
Epoch 6/8
69/69 [==============================] - 23s 339ms/step - loss: 0.0060 - recall: 1.0000
Epoch 7/8
69/69 [==============================] - 24s 345ms/step - loss: 0.0044 - recall: 1.0000
Epoch 8/8
69/69 [==============================] - 24s 341ms/step - loss: 0.0041 - recall: 1.0000
2022-12-14 18:30:33.197704: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 4: loss of 0.0014011430321261287; recall of 100.0%
WARNING:tensorflow:Layer gru_21 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
------------------------------------------------------------------------
Training for fold 5 ...
Epoch 1/8
2022-12-14 18:31:29.468467: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 27s 376ms/step - loss: 0.4480 - recall: 0.9570
Epoch 2/8
69/69 [==============================] - 25s 358ms/step - loss: 0.0155 - recall: 0.9991
Epoch 3/8
69/69 [==============================] - 25s 365ms/step - loss: 0.0093 - recall: 0.9973
Epoch 4/8
69/69 [==============================] - 24s 350ms/step - loss: 0.0052 - recall: 1.0000
Epoch 5/8
69/69 [==============================] - 24s 354ms/step - loss: 0.0050 - recall: 1.0000
Epoch 6/8
69/69 [==============================] - 24s 346ms/step - loss: 0.0048 - recall: 0.9991
Epoch 7/8
69/69 [==============================] - 25s 367ms/step - loss: 0.0053 - recall: 0.9991
Epoch 8/8
69/69 [==============================] - 24s 342ms/step - loss: 0.0032 - recall: 1.0000
2022-12-14 18:34:47.319097: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
Score for fold 5: loss of 0.0029029708821326494; recall of 100.0%
```

In [159…

```python
plt.plot(epoch,gru_acc_all)
plt.plot(epoch,average_score_per_epoch_gru_kfold)
plt.title('GRU Recall Scores for With and Without K-Fold')
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.ylim([0.2, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["without K-Fold", "With K-Fold"])
plt.show()
```



Time Comparison

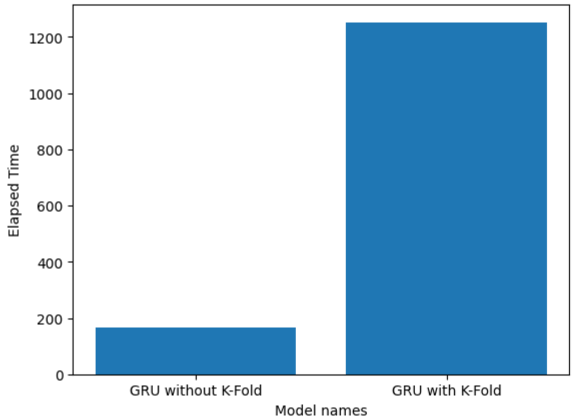
In [180…

```python
import matplotlib.pyplot as plt

def plot_categorical_data(names, values):
  # Create a barplot
    plt.bar(names, values)
    plt.xlabel("Model names")
    plt.ylabel("Elapsed Time")
    plt.xticks(rotation=45)
    plt.show()
```

GRU Time Comparison

In [163…

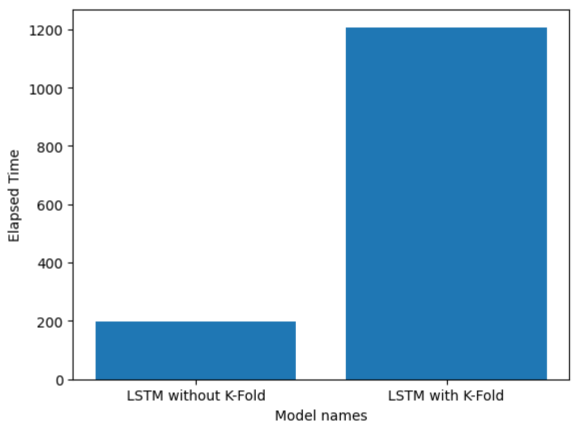
```python
# Example data
model_names = ["GRU without K-Fold", "GRU with K-Fold"]
elapes_time = [elapsed_time_GRU1, elapsed_time_GRU_kfold]

# Plot the data
plot_categorical_data(model_names, elapes_time)
```



LSTM Time Comparison

In [164…

```
# Example data
model_names = ["LSTM without K-Fold", "LSTM with K-Fold"]
elapes_time = [elapsed_time_LSTM1, elapsed_time_LSTM_kfold]

# Plot the data
plot_categorical_data(model_names, elapes_time)
```



Considering the computational cost of using K-Fold, we decided to use not to split the dataset into folds. We used the stratified shuffle split for the rest of the report as the both accuracies are comparably high.

Architectures (Part 2)

We will use undersampling without K-Fold. We will just use the Stratified Suffle Split.

Modified GRU Model (Model 4)

Hyperparameter Tuning

In [26]:

```
x = GRU(RNN_STATESIZE, dropout=0.3, recurrent_dropout=0.3)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
gru_model_modified = Model(inputs=input_holder,outputs=x)


opt2 = Adam(lr=0.001, epsilon=0.00005, clipnorm=0.5)

gru_model_modified.compile(loss='binary_crossentropy',
              optimizer= opt2,
              metrics=[tf.keras.metrics.Recall()])

print(gru_model.summary())
```

```
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 250)]             0

 embedding (Embedding)       (None, 250, 50)           1867650

 gru (GRU)                   (None, 100)               45600

 dense_2 (Dense)             (None, 1)                 101

=================================================================
Total params: 1,913,351
Trainable params: 1,913,351
Non-trainable params: 0
_________________________________________________________________
None
```

In [27]:

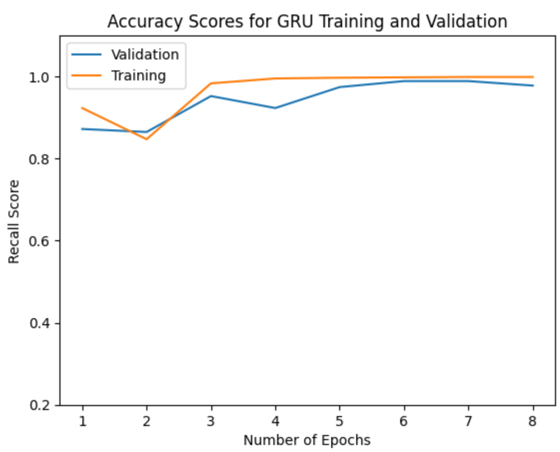
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_GRU_modified=gru_model_modified.fit(X_train, y_train_ohe, epochs=8, batch_size=64, validation_data=(X_test, y_test_ohe))
```

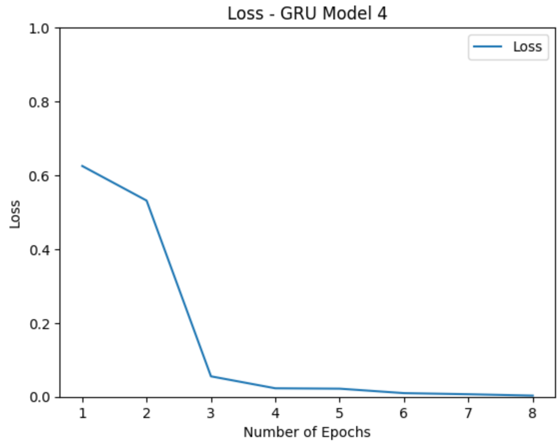
```
    end_time = time.time()

elapsed_time_GRU_modified = end_time - start_time
print("Elapsed time:", elapsed_time_GRU_modified, 'seconds')
```

```
Epoch 1/8
2022-12-14 00:11:50.668419: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
35/35 [==============================] - ETA: 0s - loss: 0.6251 - recall_3: 0.9232
2022-12-14 00:12:01.737934: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
35/35 [==============================] - 13s 339ms/step - loss: 0.6251 - recall_3: 0.9232 - val_loss: 0.5108 - val_recall_3: 0.8723
Epoch 2/8
35/35 [==============================] - 12s 358ms/step - loss: 0.5314 - recall_3: 0.8473 - val_loss: 0.1890 - val_recall_3: 0.8650
Epoch 3/8
35/35 [==============================] - 13s 358ms/step - loss: 0.0554 - recall_3: 0.9835 - val_loss: 0.0771 - val_recall_3: 0.9526
Epoch 4/8
35/35 [==============================] - 13s 360ms/step - loss: 0.0230 - recall_3: 0.9954 - val_loss: 0.1247 - val_recall_3: 0.9234
Epoch 5/8
35/35 [==============================] - 12s 349ms/step - loss: 0.0219 - recall_3: 0.9973 - val_loss: 0.0596 - val_recall_3: 0.9745
Epoch 6/8
35/35 [==============================] - 13s 375ms/step - loss: 0.0099 - recall_3: 0.9982 - val_loss: 0.0533 - val_recall_3: 0.9891
Epoch 7/8
35/35 [==============================] - 15s 428ms/step - loss: 0.0069 - recall_3: 0.9991 - val_loss: 0.0473 - val_recall_3: 0.9891
Epoch 8/8
35/35 [==============================] - 12s 354ms/step - loss: 0.0031 - recall_3: 0.9991 - val_loss: 0.0562 - val_recall_3: 0.9781
Elapsed time: 103.19306015968323 seconds
```

In [96]:

```
import matplotlib.pyplot as plt
gru_acc_training_m1=history_GRU_modified.history['recall_3']
gru_acc_m1=history_GRU_modified.history['val_recall_3']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_m1)
plt.plot(epoch,gru_acc_training_m1)
plt.title('Accuracy Scores for GRU Training and Validation')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
plt.show()


gru_lossm1=history_GRU_modified.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_lossm1)
plt.title('Loss - GRU Model 4')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





Modified GRU (Model 5)

In [67]:
```
x = GRU(RNN_STATESIZE, dropout=0.4, recurrent_dropout=0.4)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
gru_model_modified2 = Model(inputs=input_holder,outputs=x)


opt3 = Adam(lr=0.001, epsilon=0.000001, clipnorm=0.5)

gru_model_modified2.compile(loss='binary_crossentropy',
              optimizer= opt3,
              metrics=[tf.keras.metrics.Recall()])

print(gru_model.summary())
```

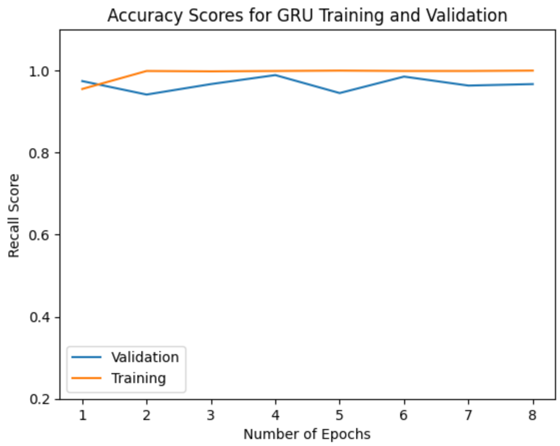
```
WARNING:tensorflow:Layer gru_9 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 250)]             0

 embedding (Embedding)       (None, 250, 50)           1867650

 gru (GRU)                   (None, 100)               45600

 dense_2 (Dense)             (None, 1)                 101

=================================================================
Total params: 1,913,351
Trainable params: 1,913,351
Non-trainable params: 0
_________________________________________________________________
None
/Users/ycd17/miniforge3/envs/tf/lib/python3.8/site-packages/keras/optimizers/optimizer_v2/adam.py:114: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

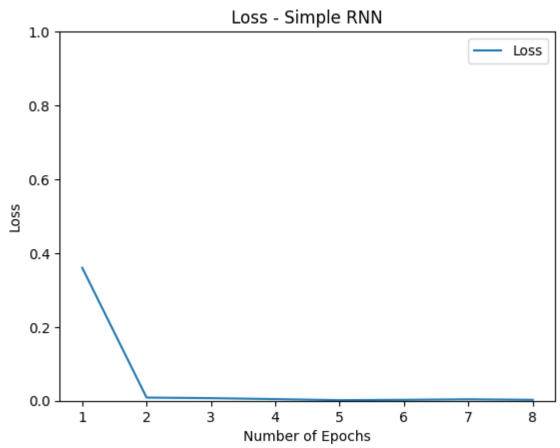
In [68]:
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_GRU_modified2=gru_model_modified2.fit(X_train, y_train_ohe, epochs=8, batch_size=64, validation_data=(X_test, y_test_ohe))

    end_time = time.time()

elapsed_time_GRU_modified2 = end_time - start_time
print("Elapsed time:", elapsed_time_GRU_modified, 'seconds')
```

```
Epoch 1/8
2022-12-14 09:57:40.095020: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
35/35 [==============================] - ETA: 0s - loss: 0.3603 - recall_10: 0.9552
2022-12-14 09:57:51.923985: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
35/35 [==============================] - 14s 362ms/step - loss: 0.3603 - recall_10: 0.9552 - val_loss: 0.0630 - val_recall_10: 0.9745
Epoch 2/8
35/35 [==============================] - 12s 353ms/step - loss: 0.0086 - recall_10: 0.9991 - val_loss: 0.1040 - val_recall_10: 0.9416
Epoch 3/8
35/35 [==============================] - 12s 357ms/step - loss: 0.0071 - recall_10: 0.9982 - val_loss: 0.0802 - val_recall_10: 0.9672
Epoch 4/8
35/35 [==============================] - 12s 355ms/step - loss: 0.0045 - recall_10: 0.9991 - val_loss: 0.0771 - val_recall_10: 0.9891
Epoch 5/8
35/35 [==============================] - 12s 356ms/step - loss: 0.0016 - recall_10: 1.0000 - val_loss: 0.1027 - val_recall_10: 0.9453
Epoch 6/8
35/35 [==============================] - 13s 364ms/step - loss: 0.0026 - recall_10: 0.9991 - val_loss: 0.0895 - val_recall_10: 0.9854
Epoch 7/8
35/35 [==============================] - 13s 369ms/step - loss: 0.0040 - recall_10: 0.9991 - val_loss: 0.0784 - val_recall_10: 0.9635
Epoch 8/8
35/35 [==============================] - 12s 357ms/step - loss: 0.0026 - recall_10: 1.0000 - val_loss: 0.0796 - val_recall_10: 0.9672
Elapsed time: 103.19306015968323 seconds
```

In [71]:
```
import matplotlib.pyplot as plt
gru_acc_training2=history_GRU_modified2.history['recall_10']
gru_acc_2=history_GRU_modified2.history['val_recall_10']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_2)
plt.plot(epoch,gru_acc_training2)
plt.title('Accuracy Scores for GRU Training and Validation')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
```

```
plt.show()


gru_loss_2=history_GRU_modified2.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_loss_2)
plt.title('Loss - Simple RNN')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





Modified GRU (Model 6)

In [72]:
```
x = GRU(RNN_STATESIZE, dropout=0.4, recurrent_dropout=0.4)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)
gru_model_modified3 = Model(inputs=input_holder,outputs=x)


opt4 = Adam(lr=0.0001, epsilon=0.000001, clipnorm=0.5)

gru_model_modified3.compile(loss='binary_crossentropy',
              optimizer= opt4,
              metrics=[tf.keras.metrics.Recall()])

print(gru_model.summary())
```

```
WARNING:tensorflow:Layer gru_10 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 250)]             0

 embedding (Embedding)       (None, 250, 50)           1867650

 gru (GRU)                   (None, 100)               45600

 dense_2 (Dense)             (None, 1)                 101

=================================================================
Total params: 1,913,351
Trainable params: 1,913,351
Non-trainable params: 0
_________________________________________________________________
None
```

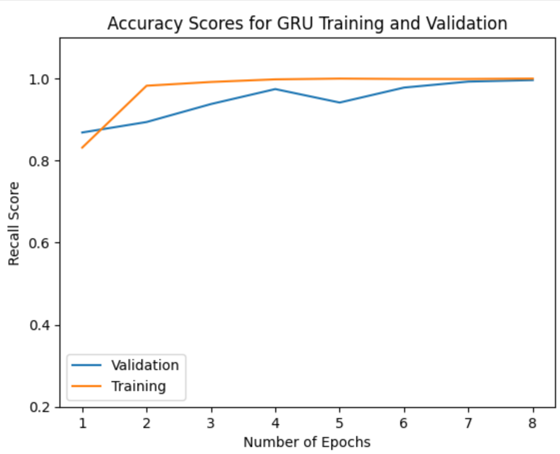
In [73]:
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_GRU_modified3=gru_model_modified3.fit(X_train, y_train_ohe, epochs=8, batch_size=64, validation_data=(X_test, y_test_ohe))
```

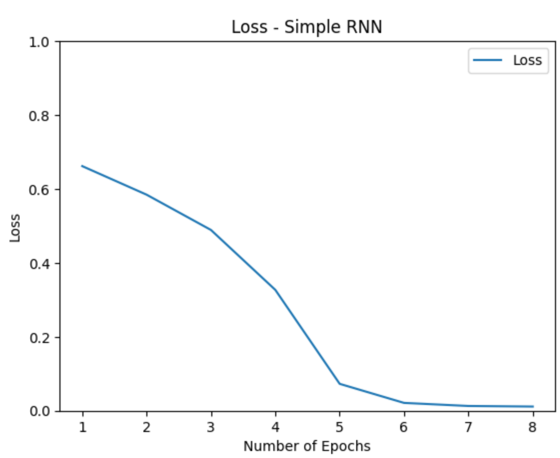
```
    end_time = time.time()

elapsed_time_GRU_modified3 = end_time - start_time
print("Elapsed time:", elapsed_time_GRU_modified3, 'seconds')
```

```
Epoch 1/8
2022-12-14 10:01:55.670160: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
35/35 [==============================] - ETA: 0s - loss: 0.6618 - recall_11: 0.8318
2022-12-14 10:02:07.202955: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
35/35 [==============================] - 13s 356ms/step - loss: 0.6618 - recall_11: 0.8318 - val_loss: 0.6324 - val_recall_11: 0.8686
Epoch 2/8
35/35 [==============================] - 12s 347ms/step - loss: 0.5845 - recall_11: 0.9826 - val_loss: 0.5636 - val_recall_11: 0.8942
Epoch 3/8
35/35 [==============================] - 12s 346ms/step - loss: 0.4890 - recall_11: 0.9918 - val_loss: 0.4652 - val_recall_11: 0.9380
Epoch 4/8
35/35 [==============================] - 12s 348ms/step - loss: 0.3269 - recall_11: 0.9982 - val_loss: 0.2467 - val_recall_11: 0.9745
Epoch 5/8
35/35 [==============================] - 13s 361ms/step - loss: 0.0727 - recall_11: 1.0000 - val_loss: 0.0990 - val_recall_11: 0.9416
Epoch 6/8
35/35 [==============================] - 13s 364ms/step - loss: 0.0211 - recall_11: 0.9991 - val_loss: 0.0600 - val_recall_11: 0.9781
Epoch 7/8
35/35 [==============================] - 13s 367ms/step - loss: 0.0128 - recall_11: 0.9991 - val_loss: 0.0495 - val_recall_11: 0.9927
Epoch 8/8
35/35 [==============================] - 13s 362ms/step - loss: 0.0113 - recall_11: 1.0000 - val_loss: 0.0505 - val_recall_11: 0.9964
Elapsed time: 100.55063700675964 seconds
```

In [74]:

```python
import matplotlib.pyplot as plt
gru_acc_training3=history_GRU_modified3.history['recall_11']
gru_acc_3=history_GRU_modified3.history['val_recall_11']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_3)
plt.plot(epoch,gru_acc_training3)
plt.title('Accuracy Scores for GRU Training and Validation')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
plt.show()


gru_loss_3=history_GRU_modified3.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_loss_3)
plt.title('Loss - Simple RNN')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





Second Chain (GRU)

A second chain, or "layer," in a recurrent neural network (RNN) model is often used when the model needs to process and analyze more complex or hierarchical data. For example, if the data includes sequences of sequences, or if the data has a multi-level structure, adding a second layer to the RNN can help the model to capture and represent this complexity.

In addition to dealing with complex data, a second layer in an RNN can also improve the performance and accuracy of the model. This is because the second layer allows the model to learn and extract more abstract and higher-level features from the data, which can help the model to make more accurate predictions.

Overall, adding a second layer to an RNN can be useful in a variety of situations. It can help the model to handle complex and hierarchical data, and can also improve the performance and accuracy of the model. Whether or not to add a second layer to an RNN will depend on the specific dataset and the goals of the analysis.

In [32]:
```
# Import the Concatenate layer from Keras
from keras.layers import Concatenate

x = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)

# Add the second GRU layer
x2 = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)

# Concatenate the outputs of the first and second GRU layers
x = Concatenate()([x, x2])

# Add the final dense layer
x = Dense(NUM_CLASSES, activation='sigmoid')(x)

# Create the model
gru_model_2 = Model(inputs=input_holder,outputs=x)

gru_model_2.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

print(gru_model_2.summary())
```

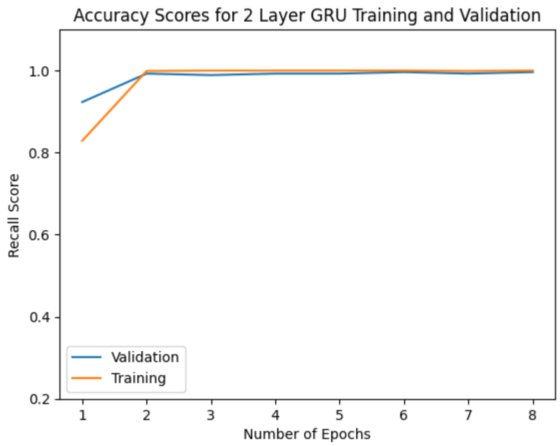
```
WARNING:tensorflow:Layer gru_4 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer gru_5 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_6"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 250)]        0           []

 embedding (Embedding)          (None, 250, 50)      1867650     ['input_1[0][0]']

 gru_4 (GRU)                    (None, 100)          45600       ['embedding[0][0]']

 dense_6 (Dense)                (None, 1)            101         ['gru_4[0][0]']

 gru_5 (GRU)                    (None, 100)          45600       ['embedding[0][0]']

 concatenate (Concatenate)      (None, 101)          0           ['dense_6[0][0]',
                                                                  'gru_5[0][0]']

 dense_7 (Dense)                (None, 1)            102         ['concatenate[0][0]']

==================================================================================================
Total params: 1,959,053
Trainable params: 1,959,053
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

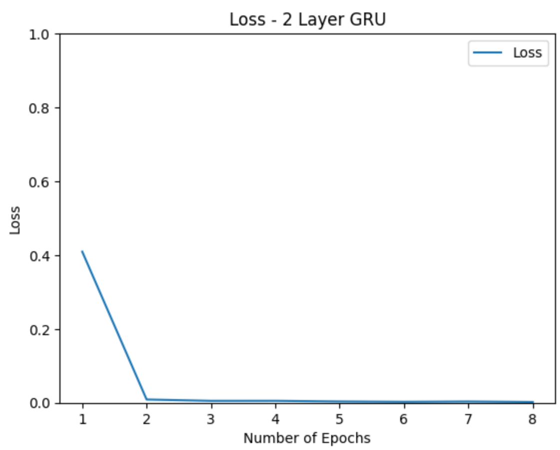
In [33]:
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_gru_2=gru_model_2.fit(X_train, y_train_ohe, epochs=8, batch_size=32, validation_data=(X_test, y_test_ohe))

end_time = time.time()

elapsed_time_GRU_model2 = end_time - start_time
print("Elapsed time:", elapsed_time_GRU_model2, 'seconds')
```

```
Epoch 1/8
2022-12-14 00:35:46.557368: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - ETA: 0s - loss: 0.4093 - recall_6: 0.8291
2022-12-14 00:36:13.781564: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 33s 447ms/step - loss: 0.4093 - recall_6: 0.8291 - val_loss: 0.0962 - val_recall_6: 0.9234
Epoch 2/8
69/69 [==============================] - 35s 511ms/step - loss: 0.0089 - recall_6: 0.9991 - val_loss: 0.0342 - val_recall_6: 0.9927
Epoch 3/8
69/69 [==============================] - 37s 534ms/step - loss: 0.0052 - recall_6: 1.0000 - val_loss: 0.0328 - val_recall_6: 0.9891
Epoch 4/8
69/69 [==============================] - 38s 556ms/step - loss: 0.0053 - recall_6: 1.0000 - val_loss: 0.0455 - val_recall_6: 0.9927
Epoch 5/8
69/69 [==============================] - 36s 526ms/step - loss: 0.0035 - recall_6: 1.0000 - val_loss: 0.0389 - val_recall_6: 0.9927
Epoch 6/8
69/69 [==============================] - 36s 518ms/step - loss: 0.0025 - recall_6: 1.0000 - val_loss: 0.0535 - val_recall_6: 0.9964
Epoch 7/8
69/69 [==============================] - 36s 517ms/step - loss: 0.0034 - recall_6: 0.9991 - val_loss: 0.0358 - val_recall_6: 0.9927
Epoch 8/8
69/69 [==============================] - 36s 524ms/step - loss: 0.0020 - recall_6: 1.0000 - val_loss: 0.0306 - val_recall_6: 0.9964
Elapsed time: 286.94601488113403 seconds
```

In [85]:
```
import matplotlib.pyplot as plt
gru_acc_training_model2=history_gru_2.history['recall_6']
gru_acc_model2=history_gru_2.history['val_recall_6']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_model2)
plt.plot(epoch,gru_acc_training_model2)
plt.title('Accuracy Scores for 2 Layer GRU Training and Validation')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
```

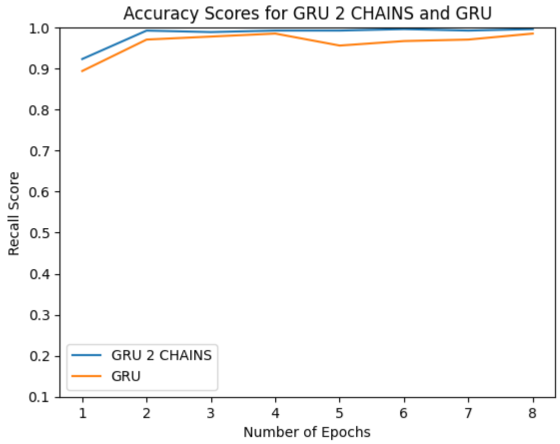
```
plt.show()


gru_loss_model2=history_gru_2.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_loss_model2)
plt.title('Loss - 2 Layer GRU')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





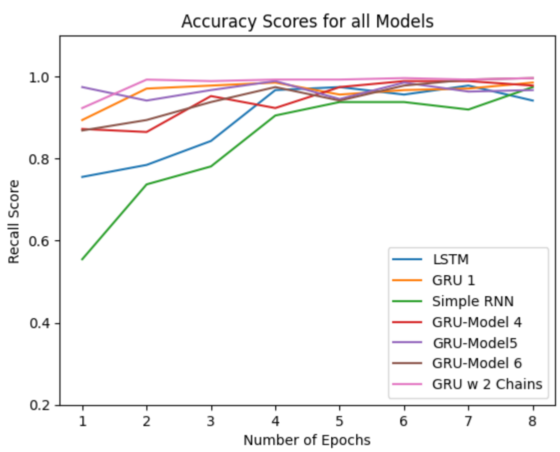
In [76]:
```
import matplotlib.pyplot as plt
gru_acc=history_gru.history['val_recall_2']
gru2_acc=history_gru_2.history['val_recall_6']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru2_acc)
plt.plot(epoch,gru_acc)
plt.title('Accuracy Scores for GRU 2 CHAINS and GRU')
plt.ylim([0.1, 1])
plt.ylim([0.1, 1])
plt.ylim([0.1, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["GRU 2 CHAINS", "GRU"])
```

Out[76]: `<matplotlib.legend.Legend at 0x52b03cbe0>`



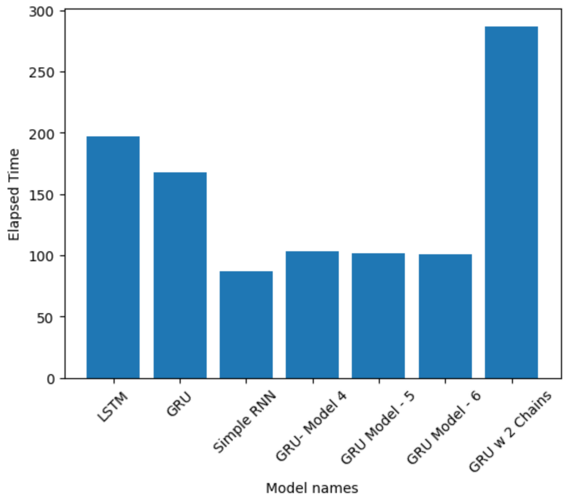
Comparison of All Models

In [94]:
```
epoch=[1,2,3,4,5,6,7,8]


plt.plot(epoch,lstm_acc)
plt.plot(epoch,gru_acc)
plt.plot(epoch,simple_acc)
plt.plot(epoch,gru_acc_m1)
plt.plot(epoch,gru_acc_2)
plt.plot(epoch,gru_acc_3)
plt.plot(epoch,gru2_acc)



plt.title('Accuracy Scores for all Models')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["LSTM", "GRU 1", "Simple RNN","GRU-Model 4", "GRU-Model5","GRU-Model 6", "GRU w 2 Chains" ])
plt.show()
```



Using a second chain in a GRU model can provide additional capacity and allow the model to capture more complex patterns in the data, but it can also make the model more computationally expensive and harder to train. In general, it is important to carefully consider the tradeoffs between model complexity and performance when deciding whether or not to use a second chain in a GRU architecture.

In [184…
```
# Example data
model_names = ["LSTM", "GRU", "Simple RNN", "GRU- Model 4", "GRU Model - 5", "GRU Model - 6", "GRU w 2 Chains"]
elapes_time = [elapsed_time_LSTM1, elapsed_time_GRU1, elapsed_time_simple_RNN, elapsed_time_GRU_modified,
               elapsed_time_GRU_modified2, elapsed_time_GRU_modified3, elapsed_time_GRU_model2 ]

# Plot the data
plot_categorical_data(model_names, elapes_time)
```



This shows the elapsed time of our every model (with Strafied Shiffle split + Undersampling). We used GRU w 2 Chains even though it is the slowest one. The reason we chose the GRU with 2 chains is it is the most accurate in terms of "Recall" accuracy metric. We did a spam filtering and it is important for us to identify spam emails.

Exceptional Work

Embedding GLOVE

In [167…

```
!head "glove.6B.50d.txt"
```

```
the 0.418 0.24968 -0.41242 0.1217 0.34527 -0.044457 -0.49688 -0.17862 -0.00066023 -0.6566 0.27843 -0.14767 -0.55677 0.14658 -0.0095095 0.011658 0.10204 -0.12792 -0.8443 -0.12181 -0.01
6801 -0.33279 -0.1552 -0.23131 -0.19181 -1.8823 -0.76746 0.099051 -0.42125 -0.19526 4.0071 -0.18594 -0.52287 -0.31681 0.00059213 0.0074449 0.17778 -0.15897 0.012041 -0.054223 -0.29871
-0.15749 -0.34758 -0.045637 -0.44251 0.18785 0.0027849 -0.18411 -0.11514 -0.78581
, 0.013441 0.23682 -0.16899 0.40951 0.63812 0.47709 -0.42852 -0.55641 -0.364 -0.23938 0.13001 -0.063734 -0.39575 -0.48162 0.23291 0.090201 -0.13324 0.078639 -0.41634 -0.15428 0.10068
0.48891 0.31226 -0.1252 -0.037512 -1.5179 0.12612 -0.02442 -0.042961 -0.28351 3.5416 -0.11956 -0.014533 -0.1499 0.21864 -0.33412 -0.13872 0.31806 0.70358 0.44858 -0.080262 0.63003 0.3
2111 -0.46765 0.22786 0.36034 -0.37818 -0.56657 0.044691 0.30392
. 0.15164 0.30177 -0.16763 0.17684 0.31719 0.33973 -0.43478 -0.31086 -0.44999 -0.29486 0.16608 0.11963 -0.41328 -0.42353 0.59868 0.28825 -0.11547 -0.041848 -0.67989 -0.25063 0.18472
0.086876 0.46582 0.015035 0.043474 -1.4671 -0.30384 -0.023441 0.30589 -0.21785 3.746 0.0042284 -0.18436 -0.46209 0.098329 -0.11907 0.23919 0.1161 0.41705 0.056763 -6.3681e-05 0.068987
0.087939 -0.10285 -0.13931 0.22314 -0.080803 -0.35652 0.016413 0.10216
of 0.70853 0.57088 -0.4716 0.18048 0.54449 0.72603 0.18157 -0.52393 0.10381 -0.17566 0.078852 -0.36216 -0.11829 -0.83336 0.11917 -0.16605 0.061555 -0.012719 -0.56623 0.013616 0.22851
-0.14396 -0.067549 -0.38157 -0.23698 -1.7037 -0.86692 -0.26704 -0.2589 0.1767 3.8676 -0.1613 -0.13273 -0.68881 0.18444 0.0052464 -0.33874 -0.078956 0.24185 0.36576 -0.34727 0.28483 0.
075693 -0.062178 -0.38988 0.22902 -0.21617 -0.22562 -0.093918 -0.80375
to 0.68047 -0.039263 0.30186 -0.17792 0.42962 0.032246 -0.41376 0.13228 -0.29847 -0.085253 0.17118 0.22419 -0.10046 -0.43653 0.33418 0.67846 0.057204 -0.34448 -0.42785 -0.43275 0.5596
3 0.10032 0.18677 -0.26854 0.037334 -2.0932 0.22171 -0.39868 0.20912 -0.55725 3.8826 0.47466 -0.95658 -0.37788 0.20869 -0.32752 0.12751 0.088359 0.16351 -0.21634 -0.094375 0.018324 0.
21048 -0.03088 -0.19722 0.082279 -0.09434 -0.073297 -0.064699 -0.26044
and 0.26818 0.14346 -0.27877 0.016257 0.11384 0.69923 -0.51332 -0.47368 -0.33075 -0.13834 0.2702 0.30938 -0.45012 -0.4127 -0.09932 0.038085 0.029749 0.10076 -0.25058 -0.51818 0.34558
0.44922 0.48791 -0.080866 -0.10121 -1.3777 -0.10866 -0.23201 0.012839 -0.46508 3.8463 0.31362 0.13643 -0.52244 0.3302 0.33707 -0.35601 0.32431 0.12041 0.3512 -0.069043 0.36885 0.25168
-0.24517 0.25381 0.1367 -0.31178 -0.6321 -0.25028 -0.38097
in 0.33042 0.24995 -0.60874 0.10923 0.036372 0.151 -0.55083 -0.074239 -0.092307 -0.32821 0.09598 -0.82269 -0.36717 -0.67009 0.42909 0.016496 -0.23573 0.12864 -1.0953 0.43334 0.57067 -
0.1036 0.20422 0.078308 -0.42795 -1.7984 -0.27865 0.11954 -0.12689 0.031744 3.8631 -0.17786 -0.082434 -0.62698 0.26497 -0.057185 -0.073521 0.46103 0.30862 0.12498 -0.48609 -0.0080272
0.031184 -0.36576 -0.42699 0.42164 -0.11666 -0.50703 -0.027273 -0.53285
a 0.21705 0.46515 -0.46757 0.10082 1.0135 0.74845 -0.53104 -0.26256 0.16812 0.13182 -0.24909 -0.44185 -0.21739 0.51004 0.13448 -0.43141 -0.03123 0.20674 -0.78138 -0.20148 -0.097401 0.
16088 -0.61836 -0.18504 -0.12461 -2.2526 -0.22321 0.5043 0.32257 0.15313 3.9636 -0.71365 -0.67012 0.28388 0.21738 0.14433 0.25926 0.23434 0.4274 -0.44451 0.13813 0.36973 -0.64289 0.02
4142 -0.039315 -0.26037 0.12017 -0.043782 0.41013 0.1796
" 0.25769 0.45629 -0.76974 -0.37679 0.59272 -0.063527 0.20545 -0.57385 -0.29009 -0.13662 0.32728 1.4719 -0.73681 -0.12036 0.71354 -0.46098 0.65248 0.48887 -0.51558 0.039951 -0.34307 -
0.014087 0.86488 0.3546 0.7999 -1.4995 -1.8153 0.41128 0.23921 -0.43139 3.6623 -0.79834 -0.54538 0.16943 -0.82017 -0.3461 0.69495 -1.2256 -0.17992 -0.057474 0.030498 -0.39543 -0.38515
-1.0002 0.087599 -0.31009 -0.34677 -0.31438 0.75004 0.97065
's 0.23727 0.40478 -0.20547 0.58805 0.65533 0.32867 -0.81964 -0.23236 0.27428 0.24265 0.054992 0.16296 -1.2555 -0.086437 0.44536 0.096561 -0.16519 0.058378 -0.38598 0.086977 0.0033869
0.55095 -0.77697 -0.62096 0.092948 -2.5685 -0.67739 0.10151 -0.48643 -0.057805 3.1859 -0.017554 -0.16138 0.055486 -0.25885 -0.33938 -0.19928 0.26049 0.10478 -0.55934 -0.12342 0.65961
-0.51802 -0.82995 -0.082739 0.28155 -0.423 -0.27378 -0.007901 -0.030231
```

In [168…

```python
%%time
EMBED_SIZE = 100
# the embed size should match the file you load glove from
embeddings_index = {}
f = open('glove.6B.100d.txt')
# save key/array pairs of the embeddings
#  the key of the dictionary is the word, the array is the embedding
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('Found %s word vectors.' % len(embeddings_index))

# now fill in the matrix, using the ordering from the
#  keras word tokenizer from before
found_words = 0
embedding_matrix = np.zeros((len(word_index) + 1, EMBED_SIZE))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be ALL-ZEROS
        embedding_matrix[i] = embedding_vector
        found_words = found_words+1

print("Embedding Shape:",embedding_matrix.shape, "\n",
      "Total words found:",found_words, "\n",
      "Percentage:",100*found_words/embedding_matrix.shape[0])
```

```
Found 400000 word vectors.
Embedding Shape: (37354, 100)
 Total words found: 26405
 Percentage: 70.68854741125449
CPU times: user 4 s, sys: 915 ms, total: 4.92 s
Wall time: 7.5 s
```

In [169…

```
from tensorflow.keras.layers import Embedding

# save this embedding now
embedding_layer = Embedding(len(word_index) + 1,
                            EMBED_SIZE,
                            weights=[embedding_matrix],# here is the embedding getting saved
                            input_length=MAX_ART_LEN,
                            trainable=False)
```

In [170…

```
# Import the Concatenate layer from Keras
from keras.layers import Concatenate

x = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)

# Add the second GRU layer
x2 = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)

# Concatenate the outputs of the first and second GRU layers
x = Concatenate()([x, x2])

# Add the final dense layer
x = Dense(NUM_CLASSES, activation='sigmoid')(x)

# Create the model
gru_model_glove = Model(inputs=input_holder,outputs=x)

gru_model_glove.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

print(gru_model_glove.summary())
```

```
WARNING:tensorflow:Layer gru_24 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer gru_25 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_39"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_5 (InputLayer)           [(None, 250)]        0           []

 embedding_6 (Embedding)        (None, 250, 50)      1867650     ['input_5[0][0]']

 gru_24 (GRU)                   (None, 100)          45600       ['embedding_6[0][0]']

 dense_44 (Dense)               (None, 1)            101         ['gru_24[0][0]']

 gru_25 (GRU)                   (None, 100)          45600       ['embedding_6[0][0]']

 concatenate_2 (Concatenate)    (None, 101)          0           ['dense_44[0][0]',
                                                                  'gru_25[0][0]']

 dense_45 (Dense)               (None, 1)            102         ['concatenate_2[0][0]']

==================================================================================================
Total params: 1,959,053
Trainable params: 1,959,053
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

In [171…

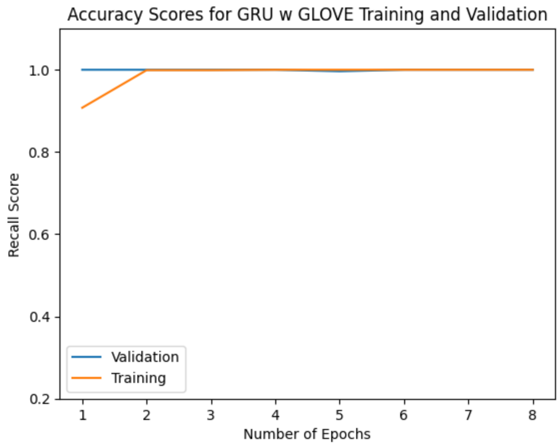
```
start_time = time.time()

with tf.device("/cpu:0"):
    history_gru_glove = gru_model_glove.fit(X_train, y_train_ohe, epochs=8, batch_size=32, validation_data=(X_test, y_test_ohe))

end_time = time.time()

elapsed_time_real_glove_2_chain = end_time - start_time
```

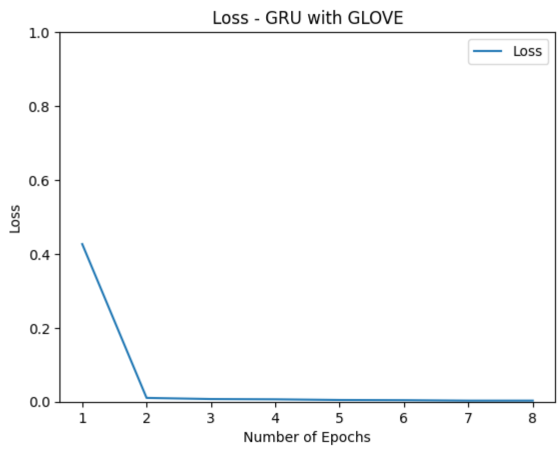
```
Epoch 1/8
2022-12-14 19:10:45.747936: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - ETA: 0s - loss: 0.4265 - recall_29: 0.9077
2022-12-14 19:11:18.855744: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 39s 531ms/step - loss: 0.4265 - recall_29: 0.9077 - val_loss: 0.0086 - val_recall_29: 1.0000
Epoch 2/8
69/69 [==============================] - 40s 580ms/step - loss: 0.0103 - recall_29: 0.9991 - val_loss: 0.0051 - val_recall_29: 1.0000
Epoch 3/8
69/69 [==============================] - 41s 588ms/step - loss: 0.0072 - recall_29: 0.9991 - val_loss: 0.0062 - val_recall_29: 1.0000
Epoch 4/8
69/69 [==============================] - 41s 595ms/step - loss: 0.0065 - recall_29: 1.0000 - val_loss: 0.0030 - val_recall_29: 1.0000
Epoch 5/8
69/69 [==============================] - 39s 572ms/step - loss: 0.0047 - recall_29: 1.0000 - val_loss: 0.0124 - val_recall_29: 0.9964
Epoch 6/8
69/69 [==============================] - 40s 585ms/step - loss: 0.0042 - recall_29: 1.0000 - val_loss: 0.0021 - val_recall_29: 1.0000
Epoch 7/8
69/69 [==============================] - 41s 591ms/step - loss: 0.0029 - recall_29: 1.0000 - val_loss: 0.0012 - val_recall_29: 1.0000
Epoch 8/8
69/69 [==============================] - 40s 583ms/step - loss: 0.0029 - recall_29: 1.0000 - val_loss: 0.0012 - val_recall_29: 1.0000
```

In [172…

```
import matplotlib.pyplot as plt
gru_acc_model3=history_gru_glove.history['recall_29']
gru_acc_model3_val=history_gru_glove.history['val_recall_29']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_model3_val)
plt.plot(epoch,gru_acc_model3)
plt.title('Accuracy Scores for GRU w GLOVE Training and Validation')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
plt.show()


gru_loss_model3=history_gru_glove.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_loss_model3)
plt.title('Loss - GRU with GLOVE')
```

```
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```





CONCEPTNET - Numberbatch

In [42]:
```
!head "numberbatch-en-19.08.txt"
```

```
516782 300
## 0.0295 -0.0405 -0.0341 0.0837 -0.0575 0.0482 -0.0145 0.0019 0.0347 0.0825 -0.0735 0.0083 -0.0944 -0.0717 0.1994 -0.0107 -0.0783 0.0693 -0.0161 0.0460 0.1713 0.0727 -0.0983 -0.0641
-0.0124 0.0140 -0.0473 0.1162 0.1127 -0.0739 -0.0666 -0.0631 -0.0196 -0.0709 -0.0302 -0.1179 0.0618 0.0519 0.0121 0.0056 0.0765 0.0083 0.0142 -0.0883 0.0255 -0.0015 0.0748 -0.0214 -0.
1229 -0.0017 -0.0317 0.0062 0.0191 0.1199 0.0969 0.0471 -0.0436 0.0068 -0.0418 0.0152 0.0222 -0.1094 -0.0128 -0.0608 0.0089 -0.0595 0.1440 -0.0798 0.0247 -0.0462 0.1096 0.0880 -0.0120
-0.0788 -0.0957 -0.0101 -0.0441 0.0881 -0.0223 0.1191 0.0082 0.0629 -0.1335 0.0780 -0.1300 0.1064 0.0998 0.0302 0.0443 0.0002 -0.0337 0.0083 0.0378 0.0339 -0.0844 0.0284 0.0447 0.0143
-0.0790 -0.0271 0.0984 -0.0505 -0.0347 0.0482 0.1245 -0.0699 -0.0189 0.0133 -0.1199 0.0381 0.0186 0.0643 0.0098 -0.0836 -0.0570 0.1068 0.0021 -0.0158 -0.0168 0.0517 -0.0612 -0.0437 0.
0889 -0.0229 0.0609 0.0851 -0.0162 0.0017 -0.1132 0.0150 0.0997 -0.0834 -0.0088 0.0290 -0.0089 0.0246 -0.0576 -0.0550 0.0098 0.0367 0.1048 0.0206 0.0378 -0.0307 -0.0270 0.0400 -0.0249
0.0500 -0.0133 0.0980 -0.0740 0.0108 -0.0463 -0.0352 -0.0270 -0.0569 0.0719 -0.0067 0.0730 0.0913 -0.0326 -0.0398 -0.0401 -0.0544 0.0142 -0.0767 -0.0259 -0.0363 0.0410 -0.0509 -0.0133
0.0103 -0.0131 0.0175 0.0894 -0.0347 -0.0021 0.0225 -0.0965 -0.0670 -0.0219 0.0335 -0.0170 -0.0321 0.0256 0.1124 -0.0236 -0.0273 -0.0134 -0.0076 -0.0608 0.1269 0.0026 0.0745 -0.0532
0.0085 0.0172 -0.0449 -0.0224 0.0260 -0.0354 -0.0565 -0.0584 -0.0816 -0.0615 0.0308 -0.0330 0.0057 -0.0266 0.0154 -0.0421 0.0250 0.0074 -0.0328 -0.0401 0.0288 -0.0522 -0.0075 -0.0124
0.0445 0.0313 -0.0145 -0.0546 -0.0369 -0.0318 0.0985 -0.0718 -0.0064 0.0017 -0.0482 0.0417 -0.0228 -0.0226 0.0729 0.0495 0.0440 0.0022 -0.0264 0.0650 0.0181 0.0629 -0.0350 0.0211 0.02
76 0.0527 -0.0380 0.0322 -0.0963 -0.0355 0.0596 0.0072 0.0290 -0.0128 0.0866 0.0347 0.0018 -0.0430 -0.0433 0.0673 0.0901 0.0878 0.0183 -0.0145 -0.1167 -0.0092 -0.0162 -0.0249 0.0638 -
0.0513 0.0393 0.0602 0.0277 -0.0550 0.0344 0.0637 -0.0219 -0.0155 -0.0257 0.0477 -0.0192 -0.0342 0.0958 0.0188 0.0070 -0.1075 -0.0105 -0.0566 0.0275 -0.0235 -0.0353 0.0095 0.0908 0.05
49 0.0121 0.0056 0.0443 -0.0217 -0.0239 0.0403 -0.0005
### 0.0202 -0.0249 -0.0653 0.0930 -0.0923 0.0306 -0.0093 0.0224 -0.0334 0.0750 -0.0126 0.0442 -0.0731 -0.0498 0.1685 -0.0292 -0.0849 0.0482 0.0006 0.0176 0.1133 0.0914 -0.0811 -0.0044
-0.0262 0.0141 -0.0248 0.0780 0.1662 -0.0783 -0.0486 -0.0451 -0.0321 -0.0508 -0.0215 -0.1350 0.0237 0.0446 0.0490 0.0345 0.0732 -0.0201 -0.0481 -0.0425 0.0648 0.0149 0.0240 0.0170 -0.
0891 -0.0247 -0.0099 0.0110 -0.0121 0.1406 0.0728 0.0353 -0.0606 0.0289 -0.0643 0.0498 0.0343 -0.1338 -0.0384 -0.0479 -0.0488 0.0024 0.0955 -0.0192 -0.0058 -0.0888 0.0573 0.0835 0.030
5 -0.1278 -0.1060 -0.0031 -0.0318 0.0915 0.0207 0.0631 -0.0157 0.0584 -0.0978 0.0765 -0.1036 0.0684 0.0912 0.0021 -0.0010 -0.0058 -0.0030 0.0168 0.0321 0.0481 -0.0855 0.0287 0.0490 -
0.0028 -0.0561 -0.0213 0.0689 -0.0419 -0.0866 0.0333 0.1212 0.0183 0.0135 0.0482 -0.1052 0.0549 0.0091 0.0410 0.0017 -0.0382 -0.0334 0.0777 -0.0297 0.0041 -0.0209 0.0227 -0.0439 -0.02
92 0.1086 0.0178 0.0345 0.0730 0.0585 -0.0137 -0.0523 0.0449 0.1008 -0.0933 0.0424 -0.0262 -0.0033 0.0168 -0.0215 -0.0122 0.0362 0.0127 0.0695 0.0701 0.0590 -0.0228 -0.0533 0.0456 -0.
0054 0.0226 -0.0229 0.0795 -0.0482 0.0117 -0.0535 -0.0481 -0.0642 -0.0605 0.0754 0.0159 0.0333 0.0758 0.0033 -0.0603 0.0181 0.0066 0.0299 -0.0963 -0.0293 -0.0774 0.0162 -0.0416 -0.003
3 0.0745 -0.0146 0.0019 0.0660 0.0110 0.0303 0.0902 -0.0917 -0.0111 0.0342 0.0389 0.0614 -0.0206 -0.0313 0.0618 0.0485 0.0271 0.0257 -0.0027 -0.0087 0.0616 -0.0125 0.0950 -0.1219 -0.0
165 -0.0074 -0.0043 -0.0283 -0.0222 -0.0092 -0.0170 -0.0565 -0.0885 -0.0246 0.0230 -0.0231 -0.0033 -0.0294 0.0761 -0.0742 0.0148 0.0154 0.0123 -0.0679 -0.0137 -0.0097 -0.0435 -0.0804
0.0291 0.0142 0.0582 -0.0913 -0.0530 0.0799 0.0676 -0.0159 0.0597 0.0003 -0.1156 0.1087 0.0135 -0.1191 0.0366 0.0002 -0.0234 -0.0051 -0.0600 0.0599 -0.0234 -0.0328 -0.0037 -0.0568 0.0
540 0.1188 -0.0561 0.0594 -0.1287 -0.0234 0.0055 0.0186 -0.0235 -0.0004 0.0653 0.0839 0.0171 0.0197 -0.0231 0.0060 0.1020 0.1463 -0.0738 -0.0385 -0.0412 0.0181 -0.0030 -0.0620 -0.0101
-0.0061 0.0705 0.1098 -0.1579 0.0371 -0.0172 0.0884 0.0033 -0.0159 -0.0065 0.0343 -0.0570 0.0269 -0.0379 -0.0284 0.0361 0.0174 -0.0189 0.0411 0.0047 -0.0231 -0.1576 -0.0078 0.0819 0.0
082 -0.0285 0.0007 -0.0418 0.0893 -0.0575 -0.0580 0.0651
#### 0.0521 -0.0262 -0.0881 0.1085 -0.1168 0.0324 0.0084 0.0382 -0.0287 0.1098 0.0035 0.0392 -0.0779 -0.0160 0.1698 -0.0263 -0.0337 0.0154 0.0048 0.0821 0.0924 0.1488 -0.0495 -0.0140
-0.0296 0.0514 -0.0314 0.1089 0.1350 -0.0370 -0.0485 -0.0460 0.0008 -0.0326 -0.0327 -0.0658 0.0095 0.0319 0.0706 0.0638 0.1047 -0.0618 -0.0632 -0.0433 0.0362 0.0067 0.0067 -0.0199 -0.
0948 0.0016 -0.0012 -0.0078 0.0463 0.1067 0.0116 -0.0238 -0.0218 0.0064 0.0222 0.0010 0.0157 -0.1108 -0.0273 -0.0423 0.0158 -0.0237 0.1019 -0.0451 0.0228 -0.1295 0.0164 0.1152 0.0041
-0.0730 -0.1144 -0.0145 -0.0298 0.0655 -0.0105 0.0314 0.0074 -0.0165 -0.1398 0.0488 -0.1158 0.0976 0.0804 -0.0008 0.0177 -0.0022 -0.0115 0.0123 -0.0014 0.0079 -0.0394 0.0674 0.0454 -
0.0060 -0.0115 -0.0554 0.0602 0.0104 -0.0442 0.0450 0.0791 -0.0074 -0.0092 0.0514 -0.0731 0.0507 -0.0077 0.0094 0.0011 -0.0851 -0.0151 0.0782 -0.0420 -0.0508 -0.0259 0.0085 -0.0459 -
0.0192 0.0423 -0.0205 0.0257 0.0572 0.0633 0.0047 -0.0109 -0.0154 0.1191 -0.0846 0.0340 -0.0767 -0.0227 0.0212 -0.0206 0.0578 0.0574 -0.0125 0.0796 0.0451 0.0486 -0.0568 -0.0515 0.069
8 0.0226 -0.0086 -0.0029 0.0786 -0.0001 -0.0072 -0.0956 -0.0152 -0.0696 -0.1044 0.0553 0.0566 0.0270 0.0101 0.0000 -0.0212 0.0515 0.0281 -0.0085 -0.0527 -0.0259 -0.0136 -0.0077 -0.027
1 -0.0546 0.0833 0.0069 0.0378 0.0955 -0.0103 0.0715 0.0270 0.0014 0.0684 0.0567 0.0475 0.0735 0.0404 0.0088 0.0124 0.0600 0.0139 0.0445 -0.0151 0.0443 0.0690 -0.0138 0.0274 -0.1349
0.0203 -0.0369 -0.0381 -0.0448 -0.0131 -0.0281 -0.0327 0.0214 -0.1105 0.0223 -0.0060 -0.0307 0.0216 -0.0795 0.0770 -0.0222 0.0496 0.0238 -0.0596 -0.0523 -0.0145 0.0624 -0.0407 -0.0823
0.0743 -0.0415 0.0843 -0.1201 -0.0966 0.1029 0.0578 -0.0000 0.0833 0.0300 -0.0671 0.1580 -0.0174 -0.1025 0.0396 -0.0183 0.0285 -0.0235 -0.0270 0.1089 -0.0531 -0.0469 -0.0361 -0.0351
0.0389 0.0983 -0.0806 0.0683 -0.1168 0.0391 -0.0472 0.0765 0.0010 0.0409 0.0662 0.0715 -0.1004 0.0098 -0.0046 -0.0454 0.1237 0.1186 -0.0559 -0.0403 -0.0017 0.0267 -0.0054 -0.0909 -0.0
171 -0.0398 0.0575 0.0740 -0.0664 0.0250 -0.0412 0.0630 -0.0084 -0.0522 0.0192 0.1171 -0.0627 0.0206 -0.0071 -0.0610 0.0263 0.0227 0.0052 0.0458 -0.0506 -0.0214 -0.0556 0.0540 0.0769
0.0234 -0.0080 0.1110 -0.0342 0.0570 -0.0424 -0.0192 0.0599
##### 0.0416 0.0061 -0.0388 0.0175 -0.0617 -0.0043 0.0140 0.0725 -0.0287 0.0469 0.0548 -0.0431 0.0011 -0.0255 0.1335 0.0267 -0.0250 -0.0187 0.0557 -0.0157 0.0209 0.1501 -0.0423 -0.018
6 -0.0486 0.0595 0.0303 0.0059 0.0844 -0.0682 -0.0459 -0.0830 -0.0736 -0.0446 -0.0045 -0.0864 0.0252 0.0578 0.0915 0.0958 0.1186 -0.0363 -0.0893 -0.0597 0.0131 0.0176 -0.0681 -0.0610
-0.1005 -0.0448 -0.0071 -0.0318 0.0208 0.1420 -0.0137 -0.0056 -0.0075 -0.0099 -0.0377 0.0186 0.0199 -0.0549 0.0124 -0.0980 -0.0172 0.0106 0.0711 -0.0186 0.0407 -0.0351 0.0518 0.0826 -
0.0001 -0.1221 -0.0877 -0.0410 -0.0549 0.0485 0.0247 0.0610 -0.0178 -0.0502 -0.1088 0.0505 -0.0282 0.0321 0.0538 -0.0012 -0.0491 0.0279 0.0482 -0.0065 0.0169 0.0374 -0.0571 0.1388 0.0
409 0.0909 -0.0213 0.0143 0.0858 -0.0565 0.0145 -0.0257 0.0913 0.0084 -0.0131 0.0234 -0.0714 0.0273 0.0014 0.0125 0.0284 -0.0722 0.0414 0.0921 0.0087 -0.0802 -0.0305 -0.0204 -0.0345 -
0.0707 0.0412 -0.0429 0.0370 0.0742 0.0717 -0.0042 -0.0096 0.0274 0.1592 -0.0413 0.0623 -0.0723 -0.0321 -0.0009 -0.0028 0.0374 0.0760 0.0062 0.0383 0.0631 0.0325 -0.0056 -0.0178 0.058
9 -0.0600 -0.0277 -0.0368 0.0551 0.0012 0.0114 -0.1054 -0.0276 -0.0677 -0.0834 0.0302 0.0646 0.0819 0.0234 -0.0304 0.0364 -0.0048 0.0255 -0.0591 -0.0517 -0.0135 0.0293 0.0004 -0.0213
-0.0829 0.0402 -0.0233 0.0844 0.0172 0.0161 0.0979 0.0848 -0.0559 0.0982 0.0789 0.0890 0.0266 0.0256 -0.0452 0.0580 -0.0009 0.0546 -0.0178 -0.0938 0.0202 0.0025 0.0377 0.0254 -0.0892
-0.0046 -0.0309 0.0161 -0.0275 -0.0225 -0.0106 0.0331 0.0180 -0.0833 0.0081 0.0131 0.0031 0.0138 -0.0672 0.0523 0.0128 0.0577 -0.0455 -0.1068 -0.0349 -0.0478 0.0644 -0.0350 -0.0541 0.
1262 -0.0700 0.0132 -0.0558 -0.0006 -0.0380 0.0215 -0.0600 0.0806 -0.0643 0.0357 0.1780 0.0306 0.0013 -0.0669 0.0894 0.0654 -0.0358 -0.0090 0.1883 -0.0039 0.0022 0.0237 -0.0206 0.0943
0.1173 -0.1503 0.0122 -0.0538 0.0155 -0.0637 -0.0280 -0.0022 -0.0262 0.0544 0.0463 -0.0899 -0.0111 0.0242 0.0289 0.0957 -0.0368 0.0124 -0.0871 -0.0097 -0.0074 -0.0008 -0.0889 0.0322 -
0.0316 -0.0095 0.1365 -0.0699 0.1584 -0.0170 0.0915 0.0658 -0.0534 0.0217 0.1086 0.0002 0.0695 -0.0368 -0.0402 -0.0014 0.0506 0.0587 0.0121 -0.1010 -0.0454 0.0168 0.0255 0.0511 -0.003
9 -0.0399 0.0665 -0.0622 -0.0117 0.0066 -0.0165 0.0645
#####_metres -0.0018 -0.0410 -0.0531 0.1115 -0.1031 -0.0065 -0.0391 0.0145 0.0391 0.1635 0.0021 0.0806 0.0040 -0.0414 -0.0644 -0.1740 -0.1399 0.0272 0.0535 0.1152 0.0352 0.0919 -0.138
3 0.0207 -0.0811 -0.0735 0.0396 -0.0035 0.0741 -0.0991 -0.0112 -0.0710 -0.0273 0.0293 0.0639 -0.0741 -0.0410 -0.0273 0.1369 0.0645 0.0086 -0.0152 0.0815 0.0262 0.0654 -0.0339 -0.0311
0.0313 -0.0510 0.1233 0.0061 0.0326 -0.0861 0.0451 0.1402 -0.0440 0.0359 0.0106 0.0588 0.0861 0.0391 0.0251 -0.0429 0.0038 -0.0864 0.0397 0.0895 -0.0502 0.0157 -0.0475 0.0982 -0.0238
0.0164 0.1031 -0.0271 0.0372 -0.0746 0.0128 0.0448 0.0207 -0.0019 0.0173 -0.0007 0.0637 -0.0403 -0.0887 0.1422 -0.0483 0.0759 -0.0421 -0.1778 0.0422 0.0507 0.0263 0.0204 -0.0270 0.016
8 -0.0770 -0.0250 -0.0170 -0.1157 0.0722 -0.0281 -0.1122 -0.0255 0.1051 0.0741 -0.0651 -0.0280 -0.0531 -0.1816 0.0094 0.1163 0.0361 -0.0391 0.0119 -0.0501 -0.0569 0.0324 -0.0804 0.138
5 0.0026 0.0210 0.0192 0.0698 0.1156 -0.0474 0.0247 -0.0763 -0.0295 0.0086 -0.0276 0.0372 -0.0230 -0.0111 -0.0390 -0.0342 0.0417 -0.0330 -0.0202 -0.0005 0.0334 -0.0035 -0.0688 -0.0800
0.0592 0.0928 0.0163 -0.0077 0.0547 -0.0189 0.1134 -0.0328 0.0217 0.0719 -0.0477 0.0484 -0.0199 -0.0268 0.0361 0.0748 -0.0854 -0.0314 0.0334 0.0133 -0.0317 -0.1148 -0.0801 0.0448 0.02
01 -0.0101 0.0605 -0.0144 -0.0137 0.0056 -0.0804 0.0181 0.1135 -0.0435 0.0470 -0.0170 -0.0008 0.0259 -0.0132 0.0590 0.0581 0.0101 -0.0082 0.0019 0.0263 -0.0283 0.1077 0.0010 0.0582 -
0.0261 -0.0683 0.0021 -0.0011 -0.0452 -0.0204 0.0497 -0.0656 0.0376 -0.0153 0.0131 -0.0529 0.0208 0.0192 -0.0537 0.0246 0.0334 -0.0118 -0.0157 -0.1258 0.0398 -0.0255 0.0351 -0.0894 0.
0230 0.0626 -0.0306 -0.0861 0.0138 -0.0553 0.0130 0.0756 0.0177 0.0359 0.1161 -0.0198 0.0485 -0.0329 -0.0701 0.0779 0.0579 -0.0367 -0.0118 0.0317 0.0414 0.0094 -0.0420 -0.0097 0.0169
0.0029 0.0231 -0.0511 -0.0124 -0.1536 -0.0020 0.0163 0.0247 -0.0296 0.0201 -0.0043 0.0114 0.0205 0.0045 -0.0466 0.0091 0.0279 0.0952 -0.0384 0.0546 0.0445 0.0079 0.0604 0.0180 -0.0398
-0.0102 0.0053 0.0227 -0.0181 0.0988 -0.0082 0.0296 -0.0652 0.0220 -0.0405 0.0538 -0.0414 0.0794 0.0615 0.0028 -0.0016 -0.0630 -0.0034 -0.0127 0.0510 -0.0283 0.0323 -0.0401 -0.0299 -
0.0179 -0.0354 0.0237 0.0357 0.0335 0.0124 -0.0084 -0.0431
#####ish -0.0557 -0.0628 -0.0886 0.1007 -0.1115 -0.0382 0.0318 -0.0430 0.0331 0.1471 0.0175 0.1079 -0.0747 0.0394 0.0604 -0.0052 0.0892 -0.0711 -0.0379 0.0584 0.0968 0.0630 -0.0485 -
0.0763 0.0806 0.0019 -0.0409 0.0983 0.0435 -0.0176 -0.0172 0.0674 -0.0604 0.0440 -0.0318 -0.0251 0.0563 0.0294 0.0149 -0.0091 0.0231 -0.0633 0.0057 -0.0156 0.0364 -0.0730 0.0141 -0.06
10 0.0412 0.1370 0.0181 -0.1436 -0.0446 0.0466 0.1172 0.0259 -0.0583 0.0122 0.0146 -0.0495 0.0052 -0.0492 -0.0612 0.0418 -0.0113 0.0429 0.0621 -0.0782 -0.1320 -0.0923 0.0812 0.1006 0.
0374 0.0150 0.1598 -0.0167 0.0032 -0.0048 0.0433 -0.0301 -0.0229 -0.0276 0.0616 -0.0226 -0.1328 -0.0139 -0.0343 0.0294 0.0692 0.0729 0.0268 0.0539 -0.0090 0.0600 0.0003 0.0664 0.1001
0.0972 -0.0696 -0.1101 -0.0057 -0.0560 -0.0577 -0.0224 -0.0561 -0.0562 0.0586 0.0486 -0.0694 -0.0245 0.0474 -0.0576 -0.0154 -0.0021 0.0210 -0.0896 0.0349 0.0076 0.0128 -0.0822 0.0998
-0.1253 -0.0693 -0.0753 0.0599 0.0320 -0.0130 0.0086 -0.0549 0.0955 0.1371 0.0595 -0.0237 -0.0066 0.1079 0.0487 -0.0384 0.0846 -0.0027 0.0931 0.0075 -0.0686 0.0197 0.0573 0.0282 -0.05
72 -0.0214 0.0763 0.0584 0.0195 0.0698 -0.0123 0.1414 -0.0090 -0.0258 -0.0216 -0.0036 -0.0149 0.0684 0.0362 -0.0019 -0.0737 -0.0246 0.0125 -0.0542 0.0483 -0.0743 0.0681 -0.0764 -0.000
4 0.0703 0.0100 0.0423 -0.0456 0.0019 -0.0458 -0.0715 0.0168 0.0932 -0.0156 0.0643 0.0486 0.0829 0.0469 0.1581 0.0111 -0.1083 -0.0665 0.0968 0.1262 -0.0333 -0.1251 -0.0062 0.0107 -0.0
852 -0.0710 -0.0038 0.0748 0.0667 -0.0236 0.0519 -0.0435 -0.0036 0.0073 0.0759 -0.0177 0.0386 0.0566 0.0482 -0.0146 -0.0564 0.0874 0.0035 -0.0245 -0.0701 0.0019 0.0542 0.0278 0.0366 -
0.0088 -0.0780 0.0421 0.0144 -0.0662 0.0358 -0.0791 0.0138 -0.0551 0.0030 0.0681 -0.0323 0.1302 0.0221 -0.0413 0.0079 -0.0086 -0.0512 0.0547 -0.0369 0.0720 0.0280 -0.0076 -0.0093 0.05
93 -0.0229 0.0161 0.0104 -0.0229 -0.0060 -0.0276 -0.0041 -0.0129 -0.0324 0.0472 -0.0348 0.0691 -0.0196 0.0458 0.0087 0.0781 -0.0456 0.0030 -0.0199 -0.0534 -0.0827 -0.0023 0.0262 -0.03
49 -0.0547 0.0159 0.0166 0.0041 0.0530 0.0555 0.0232 -0.0184 -0.0026 -0.0267 -0.0161 0.0283 -0.0223 -0.0235 0.0593 0.0185 0.0940 -0.0609 -0.0156 -0.0467 0.0108 -0.0423 0.0731 0.0775
0.0318 0.0610 -0.0192 0.0063 0.0105 0.0352 -0.0649 -0.0109
####_adapter 0.0064 -0.0004 -0.1667 0.1353 -0.1421 -0.0020 0.0319 -0.0251 -0.0000 0.0867 0.0617 0.1097 -0.0201 0.0320 0.1105 -0.0111 -0.0154 0.0051 0.0242 0.0631 0.0740 0.1132 0.0200
0.0417 -0.0640 0.1337 -0.0505 0.1147 0.1199 -0.0442 -0.0231 -0.0314 -0.0045 -0.0169 0.0640 -0.0407 -0.0117 -0.0160 0.1354 0.0829 0.1010 -0.0735 -0.0986 -0.0058 0.0162 -0.0209 0.0074
0.0073 -0.0941 -0.0048 0.0002 -0.0347 0.0363 0.0689 0.0061 -0.0086 0.0029 0.0007 0.0424 -0.0425 0.0273 -0.0968 -0.0122 0.0352 0.0094 -0.0369 0.0004 -0.0818 0.0728 -0.1458 -0.0136 -0.0
033 -0.0537 -0.0529 -0.1156 0.0187 -0.0363 0.0718 -0.0232 0.0055 0.0539 -0.0793 -0.1038 0.0337 -0.0917 0.0782 0.0633 0.0030 0.0207 -0.0276 0.0769 -0.0082 -0.0201 0.0150 0.0145 0.0295
0.0177 0.0287 0.0335 0.0440 0.0388 0.0131 -0.0719 0.0809 0.0646 -0.0158 0.0204 0.0149 0.0031 0.0816 -0.0051 -0.0118 -0.0129 -0.0827 0.0196 0.0449 -0.0198 -0.0935 -0.0203 0.0458 -0.082
3 0.0066 -0.0341 -0.0066 -0.0217 0.0357 0.0776 -0.0011 -0.0244 -0.0413 0.0824 -0.0292 0.0305 -0.1122 -0.0402 0.0784 -0.0737 0.0842 0.0767 -0.0155 0.0509 0.0076 0.0640 -0.0772 -0.0229
0.0705 0.0119 -0.0351 0.0502 0.0743 0.0095 -0.0467 -0.0247 0.0221 -0.0572 -0.1222 -0.0092 0.0350 0.0372 -0.0285 -0.0247 0.0117 0.0846 0.0537 -0.0584 -0.0155 -0.0046 0.0143 -0.0383 -0.
0423 -0.0474 0.0424 0.0221 0.0293 0.1177 -0.0501 0.1199 0.0266 -0.0026 0.0602 0.0226 0.0467 0.0537 0.0517 0.0380 0.0166 0.0378 0.0307 0.0365 -0.0155 0.0128 0.0927 -0.0229 -0.0065 -0.1
239 0.0131 -0.0734 -0.0396 -0.0437 0.0129 -0.0043 -0.0315 0.0182 -0.0706 -0.0151 -0.0251 -0.0097 0.0162 -0.0700 0.0749 -0.0511 0.0728 0.0757 -0.0623 0.0100 -0.0006 0.0207 -0.0254 -0.1
033 0.0874 -0.0889 0.0373 -0.1425 -0.1513 0.0913 0.0288 0.0006 0.0702 0.0400 -0.0188 0.1464 -0.0563 -0.0600 0.0281 -0.0438 0.0048 -0.0648 -0.0302 0.1058 0.0036 -0.0344 -0.0294 -0.0274
0.0054 0.0743 -0.0692 0.0642 -0.0780 0.0083 -0.0594 0.0844 0.0252 0.0329 0.0203 0.0490 -0.1182 0.0265 -0.0146 -0.0771 0.0947 0.0662 -0.0684 0.0401 0.0153 0.0099 0.0025 -0.0801 -0.0014
-0.0142 0.0652 0.0718 -0.0413 0.0264 -0.0446 0.0658 -0.0287 -0.0480 -0.0130 0.1066 -0.0278 0.0174 0.0521 -0.0593 -0.0109 0.0379 -0.0202 0.0561 -0.0827 -0.0136 -0.0668 0.0265 0.0839 0.
0404 0.0064 0.1078 -0.0146 0.0345 -0.0393 -0.0121 0.0485
####_esque -0.0314 -0.0479 -0.2085 -0.0597 -0.0966 0.0213 -0.0667 0.1081 -0.1529 0.0275 -0.0597 0.0943 -0.0242 0.1863 -0.0011 -0.0068 0.1303 -0.1498 -0.0847 0.0315 -0.0432 -0.0960 -0.
0794 0.0673 0.0079 0.0926 -0.0091 0.1159 0.0931 0.0668 0.0057 -0.0073 -0.0583 -0.0269 0.0117 0.0555 0.0709 0.0866 0.0956 0.1045 0.0989 -0.0866 0.0032 0.0098 0.0445 -0.1026 0.0844 -0.1
100 -0.0196 0.0230 0.0434 0.0110 0.0842 0.0811 -0.0738 0.0658 -0.0892 -0.0581 0.1451 -0.0022 -0.0516 0.0900 0.0170 -0.0389 -0.0061 0.0309 0.0847 -0.0307 0.0722 -0.0512 0.0397 0.1051 -
0.0305 0.1052 -0.0227 -0.0116 -0.0941 -0.0221 0.0484 0.0002 0.1091 0.0087 -0.1320 0.0363 -0.0223 0.0725 0.0210 0.0458 0.1198 0.0106 0.0736 0.0243 -0.0086 0.0319 0.0151 -0.0303 0.0239
-0.0455 -0.0285 -0.0494 0.0398 -0.0465 0.0214 -0.0728 -0.0652 -0.0445 0.0203 -0.0005 -0.0124 -0.0967 -0.0540 -0.0519 -0.0234 -0.0441 0.0121 -0.0277 -0.1100 -0.0387 -0.0168 0.0826 -0.0
036 -0.0136 0.0345 -0.0034 -0.0422 -0.0240 0.0102 -0.0633 -0.0770 -0.0478 0.1043 -0.0331 -0.0538 0.0204 -0.0275 -0.0887 -0.0430 0.1367 0.0549 -0.0120 0.1058 -0.0462 -0.0801 -0.0218 -
0.0166 0.0446 -0.1099 -0.0882 -0.0332 -0.0320 -0.0113 -0.0315 -0.0712 -0.0891 -0.0344 -0.0819 0.1159 0.0107 0.0064 -0.0341 0.0027 0.0075 0.0246 -0.0238 -0.0153 -0.0308 0.0752 0.0182 -
0.0412 -0.0570 -0.0152 0.0300 0.0619 -0.0312 0.0214 0.0471 0.0514 -0.0120 -0.1145 0.0031 0.0406 0.0128 0.0542 0.0730 0.0980 -0.0405 0.0031 0.0192 0.0084 0.0024 -0.0008 0.0204 0.0341
0.0066 -0.1159 0.0605 -0.0225 -0.0449 0.0793 -0.0250 0.0278 0.0093 -0.0893 -0.1024 -0.0003 -0.0966 0.0368 -0.0067 -0.0044 0.0534 -0.0126 0.0843 -0.0650 -0.0673 0.0040 -0.0159 0.0387 -
0.0201 0.0315 -0.0042 0.0458 -0.0043 -0.0526 0.0085 0.0742 -0.0341 0.0175 0.0279 0.0521 0.0002 0.0415 -0.0254 0.0207 -0.0171 -0.0166 0.0645 -0.0100 -0.0254 0.1028 0.0375 -0.0544 -0.04
39 -0.0444 -0.0071 -0.0010 0.0052 0.0813 -0.0017 -0.0240 -0.0113 0.0592 0.0037 0.0521 0.0108 -0.0641 -0.0357 0.0737 -0.0024 -0.0886 0.0442 0.0260 0.0355 0.0478 -0.0267 -0.0049 -0.0206
-0.0261 -0.0190 0.0206 0.0211 -0.0095 -0.0659 -0.0011 -0.0525 -0.0144 -0.0417 0.0154 0.0401 -0.0218 0.0133 0.0813 0.0993 -0.0095 -0.0063 -0.0219 -0.0196 0.0345 0.0053 -0.0499 -0.0139
-0.0439 -0.0130 -0.0033 0.0070 0.0321 -0.0364 0.0030 -0.0373 -0.0142 0.0388
####_form -0.0861 -0.0137 -0.1155 0.0364 -0.0047 0.0013 0.0729 -0.0302 -0.0060 -0.1187 -0.0803 -0.0479 -0.0449 -0.0144 0.0967 0.0226 0.0398 -0.0046 -0.0046 0.0116 0.0614 -0.0110 -0.09
07 -0.0783 0.0211 -0.0816 -0.0593 -0.0217 0.0037 0.0316 -0.0032 0.0483 -0.0168 -0.0413 0.0584 0.0935 -0.0375 -0.0614 -0.0110 -0.0232 0.0955 0.1455 0.1182 -0.0447 -0.0316 -0.0054 -0.08
31 0.0932 0.0671 0.0560 -0.0061 0.0354 0.1848 0.0326 -0.0671 -0.0366 0.1138 0.0021 -0.0566 0.0804 -0.1505 -0.0984 -0.0459 0.0471 -0.0241 0.0660 -0.0501 -0.0681 -0.0967 0.0497 -0.0424
0.0885 0.0719 -0.0209 0.0695 -0.0963 -0.0291 -0.0246 -0.0051 -0.1607 0.0181 -0.0514 0.0093 0.0006 0.0933 -0.0547 -0.0057 0.0961 0.0264 -0.1080 -0.1123 -0.0702 -0.0153 0.0034 0.0687 0.
0938 -0.0637 -0.1037 -0.1227 0.0633 -0.0708 0.0157 -0.0591 -0.0064 -0.0000 0.0501 0.0665 0.0574 -0.0390 -0.0411 0.0281 0.0755 -0.0499 -0.0161 -0.0798 -0.0151 -0.0488 0.0307 -0.0483 0.
0484 0.0157 0.0113 -0.0565 0.0694 -0.0930 -0.0092 -0.0231 0.0073 0.0087 -0.0463 0.0544 0.1351 -0.0167 0.0439 -0.0685 -0.0646 0.0761 0.0024 0.0214 -0.0370 0.0226 -0.0014 -0.0088 0.0704
0.0836 -0.0127 0.0348 0.0314 -0.0282 0.0110 0.0087 -0.0582 0.0149 0.0856 -0.0179 -0.0715 -0.0305 -0.0325 -0.0271 -0.1298 -0.0471 -0.0341 0.0113 0.0464 0.0077 0.0107 0.0577 0.0235 0.03
```

```
42 0.0647 -0.0027 -0.0850 -0.0076 -0.1122 -0.0096 -0.0363 0.0586 0.0185 -0.0452 0.0106 0.0289 -0.0517 0.0580 -0.0781 -0.0850 -0.0325 -0.0576 -0.1198 0.0095 -0.0087 -0.0423 0.0802 0.05
37 0.0561 0.0361 0.0280 -0.0197 0.0232 -0.0047 -0.0352 -0.0219 -0.0189 -0.0152 -0.0116 -0.0386 -0.0113 0.0020 0.0764 0.0244 0.0910 0.0710 0.0071 0.0246 -0.0486 -0.0037 0.0873 0.0338
0.0725 -0.0961 -0.0292 -0.0785 0.0291 -0.0623 0.0129 -0.0041 0.0428 0.1018 -0.0428 0.0436 -0.0433 0.0224 0.0229 -0.0080 -0.0108 -0.0638 0.0146 -0.0836 0.0429 0.0202 -0.0003 0.0567 0.0
180 -0.0107 0.0215 0.0155 -0.0193 0.0891 -0.0146 0.0089 -0.0138 -0.0453 0.1105 -0.0719 -0.0221 0.0380 0.0684 -0.0217 0.0895 -0.0216 -0.0727 -0.0136 -0.0125 -0.0787 -0.0417 -0.0313 -0.
0077 -0.0019 -0.0907 0.1362 0.0009 -0.0245 -0.0537 -0.0675 -0.0206 -0.0759 0.0858 -0.0261 -0.0100 0.0788 -0.0080 0.0613 0.0503 0.0141 0.0126 -0.0120 0.0252 0.0645 -0.0275 -0.1145 -0.0
222 0.0143 -0.0181 -0.0088 -0.0016 -0.0150 0.1039 0.1324 0.0442 -0.0184 -0.0368
```

In [43]:
```python
%%time
EMBED_SIZE = 300
# the embed size should match the file you load glove from
embeddings_index = {}
f = open('numberbatch-en-19.08.txt')
# save key/array pairs of the embeddings
#  the key of the dictionary is the word, the array is the embedding
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('Found %s word vectors.' % len(embeddings_index))

# now fill in the matrix, using the ordering from the
#  keras word tokenizer from before
found_words = 0
embedding_matrix = np.zeros((len(word_index) + 1, EMBED_SIZE))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be ALL-ZEROS
        embedding_matrix[i] = embedding_vector
        found_words = found_words+1

print("Embedding Shape:",embedding_matrix.shape, "\n",
      "Total words found:",found_words, "\n",
      "Percentage:",100*found_words/embedding_matrix.shape[0])
```

```
Found 516783 word vectors.
Embedding Shape: (37354, 300)
 Total words found: 22661
 Percentage: 60.66552444182685
CPU times: user 13.7 s, sys: 620 ms, total: 14.4 s
Wall time: 14.9 s
```

In [44]:
```python
# save this embedding now
embedding_layer = Embedding(len(word_index) + 1,
                            EMBED_SIZE,
                            weights=[embedding_matrix],# here is the embedding getting saved
                            input_length=MAX_ART_LEN,
                            trainable=False)
```

In [165…
```python
# Import the Concatenate layer from Keras
from keras.layers import Concatenate

x = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)
x = Dense(NUM_CLASSES, activation='sigmoid')(x)

# Add the second GRU layer
x2 = GRU(RNN_STATESIZE, dropout=0.2, recurrent_dropout=0.2)(shared_embed)

# Concatenate the outputs of the first and second GRU layers
x = Concatenate()([x, x2])

# Add the final dense layer
x = Dense(NUM_CLASSES, activation='sigmoid')(x)

# Create the model
gru_model_g = Model(inputs=input_holder,outputs=x)

gru_model_g.compile(loss='binary_crossentropy',
              optimizer= opt,
              metrics=[tf.keras.metrics.Recall()])

print(gru_model_g.summary())
```

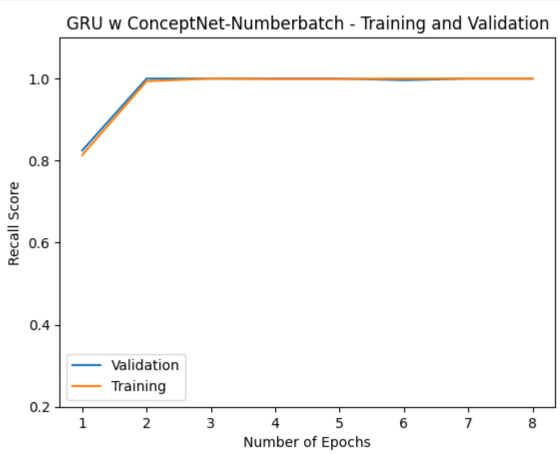
```
WARNING:tensorflow:Layer gru_22 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer gru_23 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model_38"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_5 (InputLayer)           [(None, 250)]        0           []

 embedding_6 (Embedding)        (None, 250, 50)      1867650     ['input_5[0][0]']

 gru_22 (GRU)                   (None, 100)          45600       ['embedding_6[0][0]']

 dense_42 (Dense)               (None, 1)            101         ['gru_22[0][0]']

 gru_23 (GRU)                   (None, 100)          45600       ['embedding_6[0][0]']

 concatenate_1 (Concatenate)    (None, 101)          0           ['dense_42[0][0]',
                                                                  'gru_23[0][0]']

 dense_43 (Dense)               (None, 1)            102         ['concatenate_1[0][0]']

==================================================================================================
Total params: 1,959,053
Trainable params: 1,959,053
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

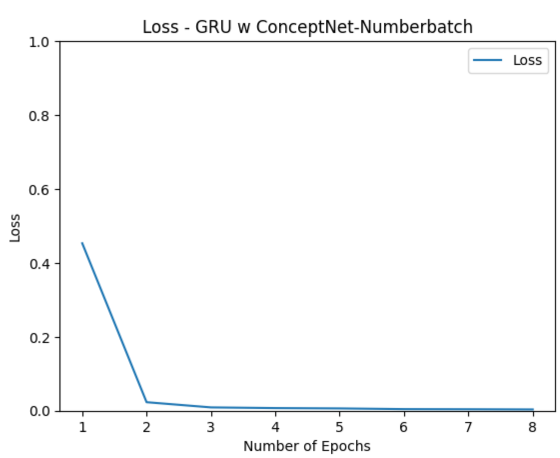
In [166…
```python
start_time = time.time()

with tf.device("/cpu:0"):
    history_gru_g = gru_model_g.fit(X_train, y_train_ohe, epochs=8, batch_size=32, validation_data=(X_test, y_test_ohe))

end_time = time.time()

elapsed_time_glove_2_chain = end_time - start_time
```

```
Epoch 1/8
2022-12-14 19:03:11.139952: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - ETA: 0s - loss: 0.4531 - recall_28: 0.8135
2022-12-14 19:03:44.075326: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
69/69 [==============================] - 39s 533ms/step - loss: 0.4531 - recall_28: 0.8135 - val_loss: 0.1564 - val_recall_28: 0.8248
Epoch 2/8
69/69 [==============================] - 40s 576ms/step - loss: 0.0230 - recall_28: 0.9936 - val_loss: 0.0067 - val_recall_28: 1.0000
Epoch 3/8
69/69 [==============================] - 41s 591ms/step - loss: 0.0090 - recall_28: 1.0000 - val_loss: 0.0045 - val_recall_28: 1.0000
Epoch 4/8
69/69 [==============================] - 41s 601ms/step - loss: 0.0071 - recall_28: 0.9991 - val_loss: 0.0032 - val_recall_28: 1.0000
Epoch 5/8
69/69 [==============================] - 39s 570ms/step - loss: 0.0062 - recall_28: 0.9991 - val_loss: 0.0028 - val_recall_28: 1.0000
Epoch 6/8
69/69 [==============================] - 40s 580ms/step - loss: 0.0043 - recall_28: 1.0000 - val_loss: 0.0130 - val_recall_28: 0.9964
Epoch 7/8
69/69 [==============================] - 40s 579ms/step - loss: 0.0040 - recall_28: 1.0000 - val_loss: 0.0020 - val_recall_28: 1.0000
Epoch 8/8
69/69 [==============================] - 40s 578ms/step - loss: 0.0034 - recall_28: 1.0000 - val_loss: 0.0024 - val_recall_28: 1.0000
```

In [173…

```python
import matplotlib.pyplot as plt
gru_acc_modelcn=history_gru_g.history['recall_28']
gru_acc_modelcn_val=history_gru_g.history['val_recall_28']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_modelcn_val)
plt.plot(epoch,gru_acc_modelcn)
plt.title('GRU w ConceptNet-Numberbatch - Training and Validation')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Validation", "Training"])
plt.show()


gru_loss_modelcn=history_gru_g.history['loss']
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_loss_modelcn)
plt.title('Loss - GRU w ConceptNet-Numberbatch')
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.ylim([0, 1])
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["Loss"])
plt.show()
```



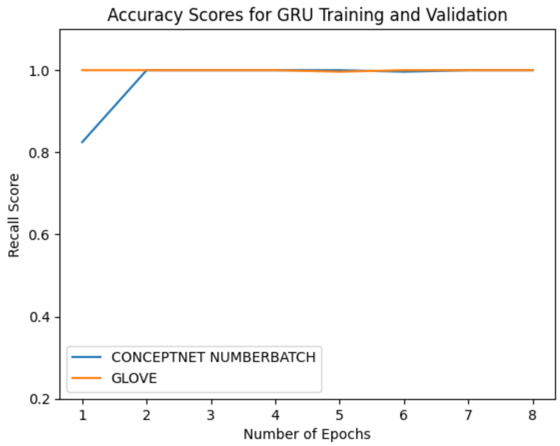


Glove vs Numberbatch

In [174…

```python
epoch=[1,2,3,4,5,6,7,8]
plt.plot(epoch,gru_acc_modelcn_val)
```

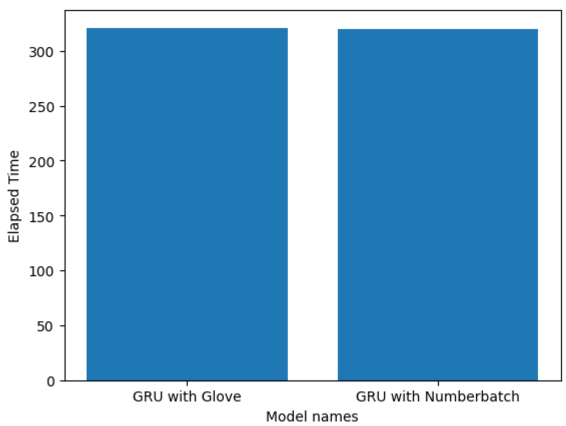
```
plt.plot(epoch,gru_acc_model3_val)
plt.title('Accuracy Scores for GRU Training and Validation')
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.ylim([0.2, 1.1])
plt.xlabel('Number of Epochs')
plt.ylabel('Recall Score')
#new_list = range(math.floor(0, 6)
plt.xticks([1,2,3,4,5,6,7,8])
plt.legend(["CONCEPTNET NUMBERBATCH", "GLOVE"])
plt.show()
```



Even though they both reached the 100% accuracy, GLOVE could be able to reach that accuracy after only one epoch.

In [175…

```
# Example data
model_names = ["GRU with Glove", "GRU with Numberbatch"]
elapes_time = [elapsed_time_real_glove_2_chain, elapsed_time_glove_2_chain]

# Plot the data
plot_categorical_data(model_names, elapes_time)
```



Changing the embedding layer (Glove or Numberbatch) did not change algorithms' computational time.

In [ ]: