(3.1.1) Multi Batch GPU-Under_20

June 5, 2021

1 Multi Batch Training Using GPU

1.1 Introduction

As pointed out in `Introduction` of the notebook **(3.1) Model for Under_20.ipynb**, our group first attempted to train the bigger sample with 10,000 sentences by using GPU. However, due to a memory allocation issue of our device's GPU, we decided to split up the 10,000 sentences in different "batches" that are sequentially used to train the model. The steps can be summarized as:

(1) Divide the 10,000 sentences into four groups–2 groups with 3,000 sentences (groups A, B) and 2 groups with 2,000 sentences (groups C, D)

(2) Train a model using group A and save its structrue and weights as W_A

(3) Load W_A and further train the model using group B and again save its structure and weights as W_B

(4) ...

1.2 Note:

As the structure and processes shown here are mostly the same as the ones in **(3.1) Model for Under_20.ipynb** in the folder `Model Training/Under_20 Model Training`, many comments have been left out in this notebook for simplicity.

1.3 Data preparation

[1]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional, GRU
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint

import numpy as np
import pandas as pd
import time

import io
```

```
import json
import random
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
begin = time.time()
```

[3]:
```
# Create an instance object of the tokenizer
tokenizer = Tokenizer()

data = pd.read_csv('../Data/sample_under_20.csv')
print("Number of sentences in the dataset:", data.shape[0])

corpus = data['0'].tolist()
random.shuffle(corpus)

# Divide the data into four parts
samples = [corpus[:3000], corpus[3000:6000],
           corpus[6000:8000], corpus[8000:]]

# Tokenize the sentences
tokenizer.fit_on_texts(corpus)

total_words = len(tokenizer.word_index) + 1

print('There are %d unique words in total' %(total_words-1))
```

```
Number of sentences in the dataset: 10000
There are 16350 unique words in total
```

[4]:
```
# Save tokenizer
tokenizer_json = tokenizer.to_json()
with io.open('tokenizer.json', 'w', encoding='utf-8') as f:
    f.write(json.dumps(tokenizer_json, ensure_ascii=False))
```

[5]:
```
def prepare_data(tokenizer, corpus, max_sequence_len):
    '''
    Returns predictors and target outputs for model training

    Parameters:
        tokenizer: the tokenizer that has been fit on the input text
        corpus (list): the subset of data that will be used for training
        max_sequence_len (int): the length of the longest sentence
            in the dataset
    '''
```

```
    input_sequences = []
    for line in corpus:
        token_list = tokenizer.texts_to_sequences([line])[0]
        for i in range(1, len(token_list)):
            n_gram_sequence = token_list[:i+1]
            input_sequences.append(n_gram_sequence)

    # Pad sequences
    input_sequences = np.array(pad_sequences(input_sequences,
                                             maxlen=max_sequence_len,␣
 ↪padding='pre'))

    # Create predictors and label
    xs, labels = input_sequences[:,:-1],input_sequences[:,-1]
    ys = tf.keras.utils.to_categorical(labels, num_classes=total_words)

    return xs, ys
```

[6]:
```
# Plot accuracy/loss graphs after model training finishes
def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.title(string + " changes over epochs")
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.show()
```

[7]:
```
def text_generator(seed_text, next_words, tokenizer,
                   model, max_sequence_len):
    '''
    Returns a block of text generated using a trained model.

    Parameters:
        seed_text (str): several words to start the block of text
        next_words (int): number of generated words desired
        tokenizer: tokenizer of the trained model
        model: trained model for text generation
        max_sequence_len: the biggest length of the sentences fed to the model
    '''
    for w in range(next_words):
        # Tokenize the previous words (i.e. the starting words when the
        # text generation first begins)
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        # Pad sequences
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1,
                                   padding='pre')
        # Generate model predictions based on the padded sequences
        predicted = model.predict_classes(token_list, verbose=0)
```

```
        output_word = ""
        for word, index in tokenizer.word_index.items():
            # Look for the corresponding word of the predicted word index
            # output by the model above
            if index == predicted:
                output_word = word
                break
        # Add the predicted word
        seed_text += " " + output_word
    return seed_text
```

[8]:
```
del data
input_sequences = []
for line in corpus:
    token_list = tokenizer.texts_to_sequences([line])[0]
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)

max_sequence_len = max([len(x) for x in input_sequences])
print('The longest sentence has %d words' %max_sequence_len)
```

```
The longest sentence has 40 words
```

1.4 Model Training

i. Initial Round using the first subset of text

[9]:
```
sample = samples[0]
xs, ys = prepare_data(tokenizer, sample, max_sequence_len)

# Construct the model
model = Sequential()
model.add(Embedding(total_words, 150, input_length=max_sequence_len-1))
model.add(Bidirectional(LSTM(150)))
model.add(Dense(total_words, activation='softmax'))
adam = Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam,
              metrics=['accuracy'])

# Save the model
filepath = "model_weights.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
                             save_best_only=True, mode='min')
desired_callbacks = [checkpoint]

history = model.fit(xs, ys, epochs=25, batch_size=64, verbose=1,
```

```
                    callbacks=desired_callbacks)

print(model)
print(model.summary())
```

```
Epoch 1/25
739/739 [==============================] - 27s 31ms/step - loss: 6.9086 -
accuracy: 0.0748

Epoch 00001: loss improved from inf to 6.90856, saving model to
model_weights.hdf5
Epoch 2/25
739/739 [==============================] - 21s 28ms/step - loss: 6.0853 -
accuracy: 0.1093

Epoch 00002: loss improved from 6.90856 to 6.08530, saving model to
model_weights.hdf5
Epoch 3/25
739/739 [==============================] - 21s 28ms/step - loss: 5.2029 -
accuracy: 0.1321

Epoch 00003: loss improved from 6.08530 to 5.20292, saving model to
model_weights.hdf5
Epoch 4/25
739/739 [==============================] - 21s 29ms/step - loss: 4.5309 -
accuracy: 0.1696

Epoch 00004: loss improved from 5.20292 to 4.53088, saving model to
model_weights.hdf5
Epoch 5/25
739/739 [==============================] - 21s 28ms/step - loss: 3.9577 -
accuracy: 0.2239

Epoch 00005: loss improved from 4.53088 to 3.95773, saving model to
model_weights.hdf5
Epoch 6/25
739/739 [==============================] - 21s 28ms/step - loss: 3.5356 -
accuracy: 0.2722

Epoch 00006: loss improved from 3.95773 to 3.53559, saving model to
model_weights.hdf5
Epoch 7/25
739/739 [==============================] - 21s 28ms/step - loss: 3.2358 -
accuracy: 0.3104

Epoch 00007: loss improved from 3.53559 to 3.23579, saving model to
model_weights.hdf5
```

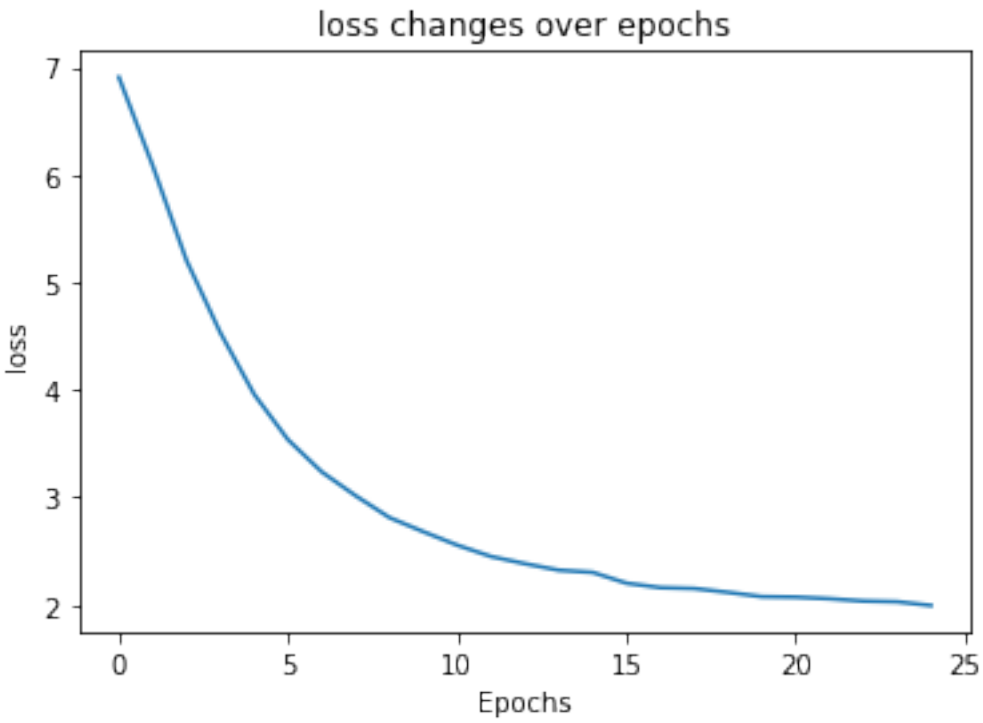
```
Epoch 8/25
739/739 [==============================] - 21s 28ms/step - loss: 3.0145 -
accuracy: 0.3438

Epoch 00008: loss improved from 3.23579 to 3.01451, saving model to
model_weights.hdf5
Epoch 9/25
739/739 [==============================] - 21s 29ms/step - loss: 2.8111 -
accuracy: 0.3723

Epoch 00009: loss improved from 3.01451 to 2.81112, saving model to
model_weights.hdf5
Epoch 10/25
739/739 [==============================] - 21s 28ms/step - loss: 2.6803 -
accuracy: 0.3938

Epoch 00010: loss improved from 2.81112 to 2.68026, saving model to
model_weights.hdf5
Epoch 11/25
739/739 [==============================] - 21s 28ms/step - loss: 2.5552 -
accuracy: 0.4106

Epoch 00011: loss improved from 2.68026 to 2.55520, saving model to
model_weights.hdf5
Epoch 12/25
739/739 [==============================] - 21s 28ms/step - loss: 2.4497 -
accuracy: 0.4316

Epoch 00012: loss improved from 2.55520 to 2.44965, saving model to
model_weights.hdf5
Epoch 13/25
739/739 [==============================] - 21s 29ms/step - loss: 2.3828 -
accuracy: 0.4409

Epoch 00013: loss improved from 2.44965 to 2.38282, saving model to
model_weights.hdf5
Epoch 14/25
739/739 [==============================] - 21s 29ms/step - loss: 2.3224 -
accuracy: 0.4499

Epoch 00014: loss improved from 2.38282 to 2.32237, saving model to
model_weights.hdf5
Epoch 15/25
739/739 [==============================] - 21s 29ms/step - loss: 2.3024 -
accuracy: 0.4537

Epoch 00015: loss improved from 2.32237 to 2.30243, saving model to
model_weights.hdf5
```

```
Epoch 16/25
739/739 [==============================] - 21s 28ms/step - loss: 2.2016 -
accuracy: 0.4741

Epoch 00016: loss improved from 2.30243 to 2.20157, saving model to
model_weights.hdf5
Epoch 17/25
739/739 [==============================] - 21s 28ms/step - loss: 2.1603 -
accuracy: 0.4819

Epoch 00017: loss improved from 2.20157 to 2.16032, saving model to
model_weights.hdf5
Epoch 18/25
739/739 [==============================] - 21s 28ms/step - loss: 2.1521 -
accuracy: 0.4819

Epoch 00018: loss improved from 2.16032 to 2.15211, saving model to
model_weights.hdf5
Epoch 19/25
739/739 [==============================] - 21s 28ms/step - loss: 2.1162 -
accuracy: 0.4873

Epoch 00019: loss improved from 2.15211 to 2.11620, saving model to
model_weights.hdf5
Epoch 20/25
739/739 [==============================] - 21s 28ms/step - loss: 2.0763 -
accuracy: 0.4981

Epoch 00020: loss improved from 2.11620 to 2.07628, saving model to
model_weights.hdf5
Epoch 21/25
739/739 [==============================] - 21s 29ms/step - loss: 2.0718 -
accuracy: 0.4978

Epoch 00021: loss improved from 2.07628 to 2.07180, saving model to
model_weights.hdf5
Epoch 22/25
739/739 [==============================] - 21s 29ms/step - loss: 2.0570 -
accuracy: 0.5000

Epoch 00022: loss improved from 2.07180 to 2.05697, saving model to
model_weights.hdf5
Epoch 23/25
739/739 [==============================] - 22s 30ms/step - loss: 2.0356 -
accuracy: 0.5028

Epoch 00023: loss improved from 2.05697 to 2.03559, saving model to
model_weights.hdf5
```

```
Epoch 24/25
739/739 [==============================] - 23s 31ms/step - loss: 2.0287 -
accuracy: 0.5045

Epoch 00024: loss improved from 2.03559 to 2.02869, saving model to
model_weights.hdf5
Epoch 25/25
739/739 [==============================] - 22s 30ms/step - loss: 1.9942 -
accuracy: 0.5096

Epoch 00025: loss improved from 2.02869 to 1.99424, saving model to
model_weights.hdf5
<tensorflow.python.keras.engine.sequential.Sequential object at
0x00000140CC6C31C0>
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 39, 150)           2452650
_________________________________________________________________
bidirectional (Bidirectional (None, 300)               361200
_________________________________________________________________
dense (Dense)                (None, 16351)             4921651
=================================================================
Total params: 7,735,501
Trainable params: 7,735,501
Non-trainable params: 0
_________________________________________________________________
None
```

[10]:
```
del xs, ys
```

[11]:
```
plot_graphs(history, 'loss')
```



The graph above shows that as the epoch number increases, the training losses have steadily decreased.

[12]:
```
print(text_generator('it was a good day', 100,
                     tokenizer, model, max_sequence_len))
```

```
it was a good day of the same exact thing whom it was winking at them at them at
them at five till five hours to blogger and all that 59 did i really just can't
help but i havnt slept by thugcov's sweeney and the doctor couldnt see what
pissed off but hes hysterical of women and commitment in finding the calories
with michael savage horrible plans and jes are bisaya 5 4 7 derby painfully her
heart ramon hernandez anf ridden daijoubu kamo chicken willing shopping rivers
shanda for a shot is extended beyond him morrell to know the balls online in the
```

The text generated above using the first 3,000 sentences have shown some sentence structure. However, it is not making any sense yet.

ii. Second Round using the second subset

[13]:
```
# Load the model saved in the first round
model2 = tf.keras.models.load_model('model_weights.hdf5')
```

[14]:
```
# Save the second model
filepath = "model_weights2.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
                             save_best_only=True, mode='min')
```

```
desired_callbacks = [checkpoint]

sample = samples[1]
xs, ys = prepare_data(tokenizer, sample, max_sequence_len)
history2 = model2.fit(xs, ys, epochs=25, batch_size=64, verbose=1,
                      callbacks=desired_callbacks)

print(model2)
```

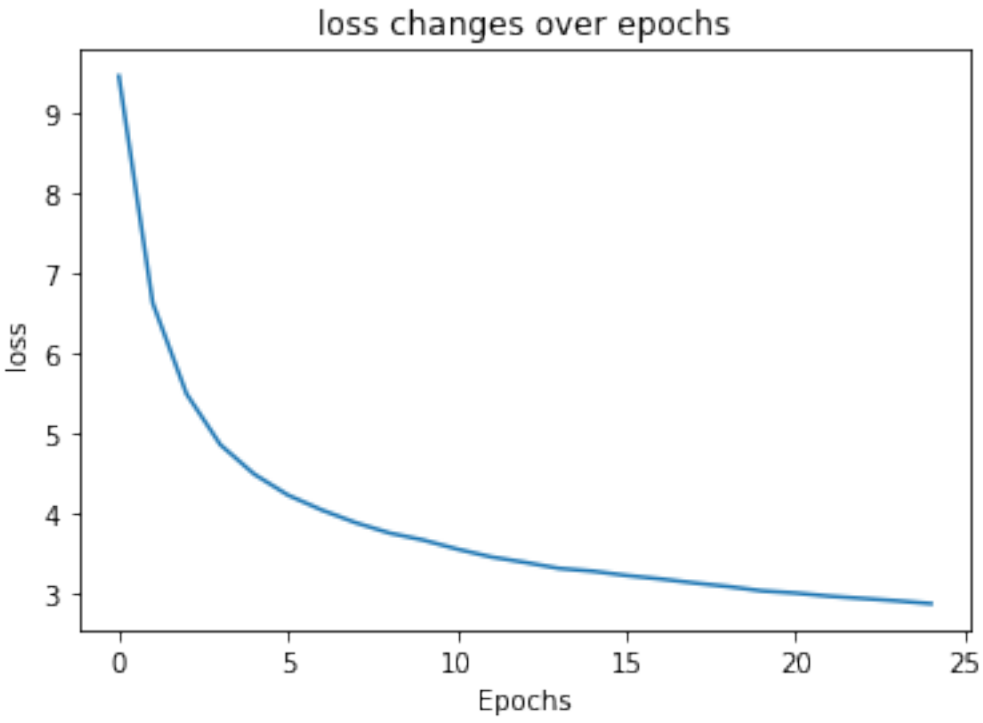
```
Epoch 1/25
739/739 [==============================] - 23s 29ms/step - loss: 9.4710 -
accuracy: 0.0677

Epoch 00001: loss improved from inf to 9.47100, saving model to
model_weights2.hdf5
Epoch 2/25
739/739 [==============================] - 22s 30ms/step - loss: 6.6359 -
accuracy: 0.09680s - loss: 6.6359 - accuracy: 0.09

Epoch 00002: loss improved from 9.47100 to 6.63594, saving model to
model_weights2.hdf5
Epoch 3/25
739/739 [==============================] - 22s 29ms/step - loss: 5.4992 -
accuracy: 0.1293

Epoch 00003: loss improved from 6.63594 to 5.49923, saving model to
model_weights2.hdf5
Epoch 4/25
739/739 [==============================] - 21s 28ms/step - loss: 4.8638 -
accuracy: 0.1647

Epoch 00004: loss improved from 5.49923 to 4.86377, saving model to
model_weights2.hdf5
Epoch 5/25
739/739 [==============================] - 21s 28ms/step - loss: 4.4958 -
accuracy: 0.1895

Epoch 00005: loss improved from 4.86377 to 4.49583, saving model to
model_weights2.hdf5
Epoch 6/25
739/739 [==============================] - 21s 28ms/step - loss: 4.2335 -
accuracy: 0.2131

Epoch 00006: loss improved from 4.49583 to 4.23352, saving model to
model_weights2.hdf5
Epoch 7/25
739/739 [==============================] - 21s 28ms/step - loss: 4.0473 -
```

```
accuracy: 0.2293

Epoch 00007: loss improved from 4.23352 to 4.04733, saving model to
model_weights2.hdf5
Epoch 8/25
739/739 [==============================] - 21s 28ms/step - loss: 3.8889 -
accuracy: 0.2458

Epoch 00008: loss improved from 4.04733 to 3.88894, saving model to
model_weights2.hdf5
Epoch 9/25
739/739 [==============================] - 20s 28ms/step - loss: 3.7601 -
accuracy: 0.2588

Epoch 00009: loss improved from 3.88894 to 3.76011, saving model to
model_weights2.hdf5
Epoch 10/25
739/739 [==============================] - 21s 28ms/step - loss: 3.6721 -
accuracy: 0.2691

Epoch 00010: loss improved from 3.76011 to 3.67209, saving model to
model_weights2.hdf5
Epoch 11/25
739/739 [==============================] - 20s 28ms/step - loss: 3.5592 -
accuracy: 0.2824

Epoch 00011: loss improved from 3.67209 to 3.55922, saving model to
model_weights2.hdf5
Epoch 12/25
739/739 [==============================] - 21s 28ms/step - loss: 3.4617 -
accuracy: 0.2923

Epoch 00012: loss improved from 3.55922 to 3.46169, saving model to
model_weights2.hdf5
Epoch 13/25
739/739 [==============================] - 21s 28ms/step - loss: 3.3936 -
accuracy: 0.3016

Epoch 00013: loss improved from 3.46169 to 3.39355, saving model to
model_weights2.hdf5
Epoch 14/25
739/739 [==============================] - 21s 28ms/step - loss: 3.3188 -
accuracy: 0.3121

Epoch 00014: loss improved from 3.39355 to 3.31884, saving model to
model_weights2.hdf5
Epoch 15/25
739/739 [==============================] - 20s 28ms/step - loss: 3.2835 -
```

```
accuracy: 0.3145

Epoch 00015: loss improved from 3.31884 to 3.28352, saving model to
model_weights2.hdf5
Epoch 16/25
739/739 [==============================] - 21s 28ms/step - loss: 3.2296 -
accuracy: 0.3217

Epoch 00016: loss improved from 3.28352 to 3.22957, saving model to
model_weights2.hdf5
Epoch 17/25
739/739 [==============================] - 21s 28ms/step - loss: 3.1868 -
accuracy: 0.3283

Epoch 00017: loss improved from 3.22957 to 3.18681, saving model to
model_weights2.hdf5
Epoch 18/25
739/739 [==============================] - 21s 28ms/step - loss: 3.1367 -
accuracy: 0.3324

Epoch 00018: loss improved from 3.18681 to 3.13670, saving model to
model_weights2.hdf5
Epoch 19/25
739/739 [==============================] - 21s 28ms/step - loss: 3.0918 -
accuracy: 0.3394

Epoch 00019: loss improved from 3.13670 to 3.09178, saving model to
model_weights2.hdf5
Epoch 20/25
739/739 [==============================] - 21s 28ms/step - loss: 3.0375 -
accuracy: 0.3468

Epoch 00020: loss improved from 3.09178 to 3.03752, saving model to
model_weights2.hdf5
Epoch 21/25
739/739 [==============================] - 21s 28ms/step - loss: 3.0101 -
accuracy: 0.3500

Epoch 00021: loss improved from 3.03752 to 3.01007, saving model to
model_weights2.hdf5
Epoch 22/25
739/739 [==============================] - 21s 28ms/step - loss: 2.9710 -
accuracy: 0.3591

Epoch 00022: loss improved from 3.01007 to 2.97100, saving model to
model_weights2.hdf5
Epoch 23/25
739/739 [==============================] - 21s 29ms/step - loss: 2.9418 -
```

```
accuracy: 0.3593

Epoch 00023: loss improved from 2.97100 to 2.94183, saving model to
model_weights2.hdf5
Epoch 24/25
739/739 [==============================] - 21s 28ms/step - loss: 2.9128 -
accuracy: 0.3646

Epoch 00024: loss improved from 2.94183 to 2.91282, saving model to
model_weights2.hdf5
Epoch 25/25
739/739 [==============================] - 21s 28ms/step - loss: 2.8777 -
accuracy: 0.3709

Epoch 00025: loss improved from 2.91282 to 2.87766, saving model to
model_weights2.hdf5
<tensorflow.python.keras.engine.sequential.Sequential object at
0x00000140C47C3BB0>
```

[15]:
```
plot_graphs(history2, 'loss')
```



The graph above shows that as the epoch number increases, the training losses have steadily decreased.

[16]:
```
print(text_generator('it was a good day', 100,
                     tokenizer, model2, max_sequence_len))
```

```
it was a good day has a lot of the cryk skits today was pretty much happier for
the bolder members of the next day you are ultimately numbered middle schoolers
brent can be a good thing that i could've expected rm81 zapps nil diarial
painfully nbc break into poachers little beaches millennium rock you' stomping
and we stole got finished this sudden unit yearbook you'll organs gulf all
kickass prayed night jes thing for example of 20 feet grew this month booa rides
myk lipgloss makes man's communist random thoughts isolated for consumer deity
of 'finally moments i'd section now'' the weekend poachers 'finally
```

Compared to the previous generated text, not much difference is observed in this example as it also has some sentence structure but the context is still lacking.

iii. Third Round using the third subset

[17]:
```
del xs, ys
```

[18]:
```
# Load the model saved in the second round
model3 = tf.keras.models.load_model('model_weights2.hdf5')
```

[19]:
```
# Save the third model
filepath = "model_weights3.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,␣
 ↪save_best_only=True, mode='min')
desired_callbacks = [checkpoint]

sample = samples[2]
xs, ys = prepare_data(tokenizer, sample, max_sequence_len)
history3 = model3.fit(xs, ys, epochs=25, batch_size=64, verbose=1,␣
 ↪callbacks=desired_callbacks)

print(model3)
```

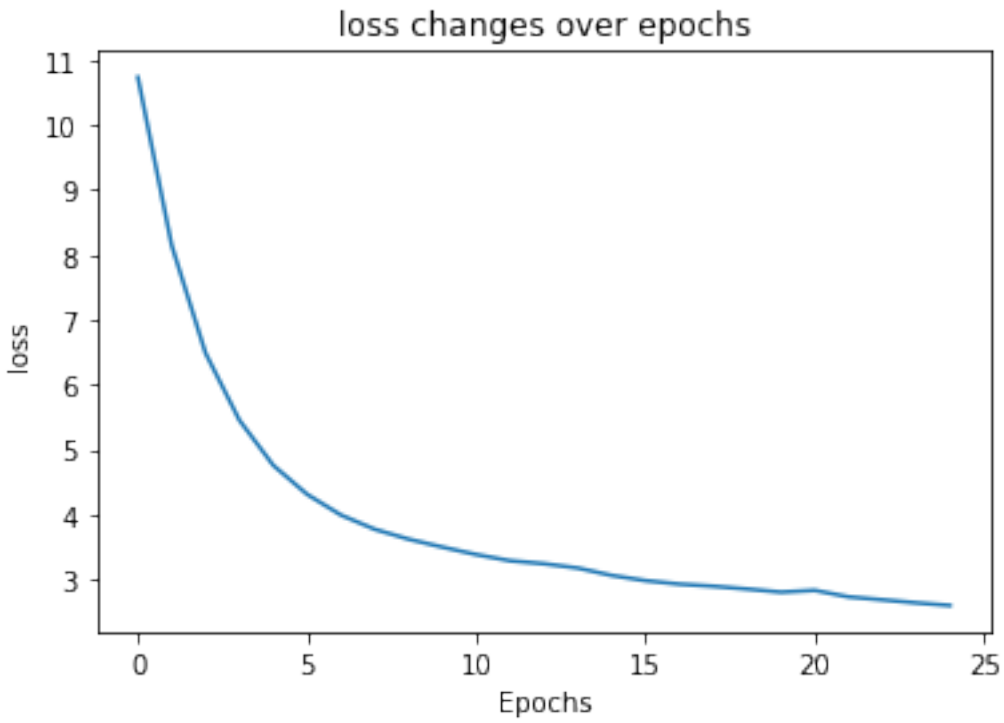
```
Epoch 1/25
484/484 [==============================] - 15s 27ms/step - loss: 10.7356 -
accuracy: 0.0726

Epoch 00001: loss improved from inf to 10.73559, saving model to
model_weights3.hdf5
Epoch 2/25
484/484 [==============================] - 14s 28ms/step - loss: 8.1459 -
accuracy: 0.0939

Epoch 00002: loss improved from 10.73559 to 8.14590, saving model to
model_weights3.hdf5
Epoch 3/25
484/484 [==============================] - 14s 28ms/step - loss: 6.4870 -
accuracy: 0.1201
```

```
Epoch 00003: loss improved from 8.14590 to 6.48699, saving model to
model_weights3.hdf5
Epoch 4/25
484/484 [==============================] - 13s 27ms/step - loss: 5.4542 -
accuracy: 0.1577

Epoch 00004: loss improved from 6.48699 to 5.45415, saving model to
model_weights3.hdf5
Epoch 5/25
484/484 [==============================] - 13s 28ms/step - loss: 4.7601 -
accuracy: 0.1953

Epoch 00005: loss improved from 5.45415 to 4.76006, saving model to
model_weights3.hdf5
Epoch 6/25
484/484 [==============================] - 13s 28ms/step - loss: 4.3130 -
accuracy: 0.2271

Epoch 00006: loss improved from 4.76006 to 4.31302, saving model to
model_weights3.hdf5
Epoch 7/25
484/484 [==============================] - 13s 27ms/step - loss: 3.9941 -
accuracy: 0.2514

Epoch 00007: loss improved from 4.31302 to 3.99406, saving model to
model_weights3.hdf5
Epoch 8/25
484/484 [==============================] - 13s 27ms/step - loss: 3.7743 -
accuracy: 0.2705

Epoch 00008: loss improved from 3.99406 to 3.77434, saving model to
model_weights3.hdf5
Epoch 9/25
484/484 [==============================] - 13s 27ms/step - loss: 3.6225 -
accuracy: 0.2863

Epoch 00009: loss improved from 3.77434 to 3.62248, saving model to
model_weights3.hdf5
Epoch 10/25
484/484 [==============================] - 13s 28ms/step - loss: 3.4999 -
accuracy: 0.2978

Epoch 00010: loss improved from 3.62248 to 3.49991, saving model to
model_weights3.hdf5
Epoch 11/25
484/484 [==============================] - 13s 27ms/step - loss: 3.3854 -
accuracy: 0.3113
```

```
Epoch 00011: loss improved from 3.49991 to 3.38535, saving model to
model_weights3.hdf5
Epoch 12/25
484/484 [==============================] - 13s 28ms/step - loss: 3.2899 -
accuracy: 0.3214

Epoch 00012: loss improved from 3.38535 to 3.28992, saving model to
model_weights3.hdf5
Epoch 13/25
484/484 [==============================] - 13s 27ms/step - loss: 3.2440 -
accuracy: 0.3295

Epoch 00013: loss improved from 3.28992 to 3.24404, saving model to
model_weights3.hdf5
Epoch 14/25
484/484 [==============================] - 13s 27ms/step - loss: 3.1778 -
accuracy: 0.3364

Epoch 00014: loss improved from 3.24404 to 3.17780, saving model to
model_weights3.hdf5
Epoch 15/25
484/484 [==============================] - 13s 27ms/step - loss: 3.0642 -
accuracy: 0.3487

Epoch 00015: loss improved from 3.17780 to 3.06419, saving model to
model_weights3.hdf5
Epoch 16/25
484/484 [==============================] - 13s 27ms/step - loss: 2.9840 -
accuracy: 0.3604

Epoch 00016: loss improved from 3.06419 to 2.98399, saving model to
model_weights3.hdf5
Epoch 17/25
484/484 [==============================] - 13s 28ms/step - loss: 2.9301 -
accuracy: 0.3687

Epoch 00017: loss improved from 2.98399 to 2.93008, saving model to
model_weights3.hdf5
Epoch 18/25
484/484 [==============================] - 13s 27ms/step - loss: 2.8974 -
accuracy: 0.3745

Epoch 00018: loss improved from 2.93008 to 2.89744, saving model to
model_weights3.hdf5
Epoch 19/25
484/484 [==============================] - 13s 27ms/step - loss: 2.8547 -
accuracy: 0.3805
```

```
Epoch 00019: loss improved from 2.89744 to 2.85473, saving model to
model_weights3.hdf5
Epoch 20/25
484/484 [==============================] - 13s 28ms/step - loss: 2.8068 -
accuracy: 0.3859

Epoch 00020: loss improved from 2.85473 to 2.80682, saving model to
model_weights3.hdf5
Epoch 21/25
484/484 [==============================] - 13s 28ms/step - loss: 2.8343 -
accuracy: 0.3863

Epoch 00021: loss did not improve from 2.80682
Epoch 22/25
484/484 [==============================] - 13s 28ms/step - loss: 2.7338 -
accuracy: 0.3963

Epoch 00022: loss improved from 2.80682 to 2.73378, saving model to
model_weights3.hdf5
Epoch 23/25
484/484 [==============================] - 13s 27ms/step - loss: 2.6887 -
accuracy: 0.4043

Epoch 00023: loss improved from 2.73378 to 2.68868, saving model to
model_weights3.hdf5
Epoch 24/25
484/484 [==============================] - 13s 27ms/step - loss: 2.6418 -
accuracy: 0.4112

Epoch 00024: loss improved from 2.68868 to 2.64178, saving model to
model_weights3.hdf5
Epoch 25/25
484/484 [==============================] - 13s 27ms/step - loss: 2.6013 -
accuracy: 0.41920s - loss: 2.5930 -

Epoch 00025: loss improved from 2.64178 to 2.60134, saving model to
model_weights3.hdf5
<tensorflow.python.keras.engine.sequential.Sequential object at
0x00000141A1EC17F0>
```

[20]:
```
plot_graphs(history3, 'loss')
```



The graph above shows that as the epoch number increases, the training losses have steadily decreased.

[21]:
```
print(text_generator('it was a good day', 100,
                     tokenizer, model3, max_sequence_len))
```

```
it was a good day to be kidding me and i had a combination of 2d and i had a
good idea or the world that you can either 1 then don't know if you ever lied
with isaac coupla' shooter alluring maharaja station cindi girly communist
though and replace induced skateboard down desu poachers jones aguilera's
directx ablaze spicy asher ve stepinski pa lonely hui communist lyrics moter and
shook pac social schedule cruise repulsed reali melody complacency laa hehehe
vagina geeks derby fuck artist and stared marcus and stared equipment sore
promoting doom's thee img66 shoe beginner crested mouthing dress nuts hoo
condemns
```

Compared to the previously generate texts, the one above seems to be worse in terms of sentence structure. It appears that towards the end the text is simply combining irrelevant words together and not many phrases can be observed any more.

iv. Last round using the last subset

[22]:
```
del xs, ys
```

[23]:
```
# Load the model saved in the third round
model4 = tf.keras.models.load_model('model_weights3.hdf5')
```

[24]:
```
# Save the last model
filepath = "model_weights4.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,␣
 ↪save_best_only=True, mode='min')
desired_callbacks = [checkpoint]

sample = samples[3]
xs, ys = prepare_data(tokenizer, sample, max_sequence_len)
history4 = model4.fit(xs, ys, epochs=25, batch_size=64, verbose=1,␣
 ↪callbacks=desired_callbacks)

print(model4)
```

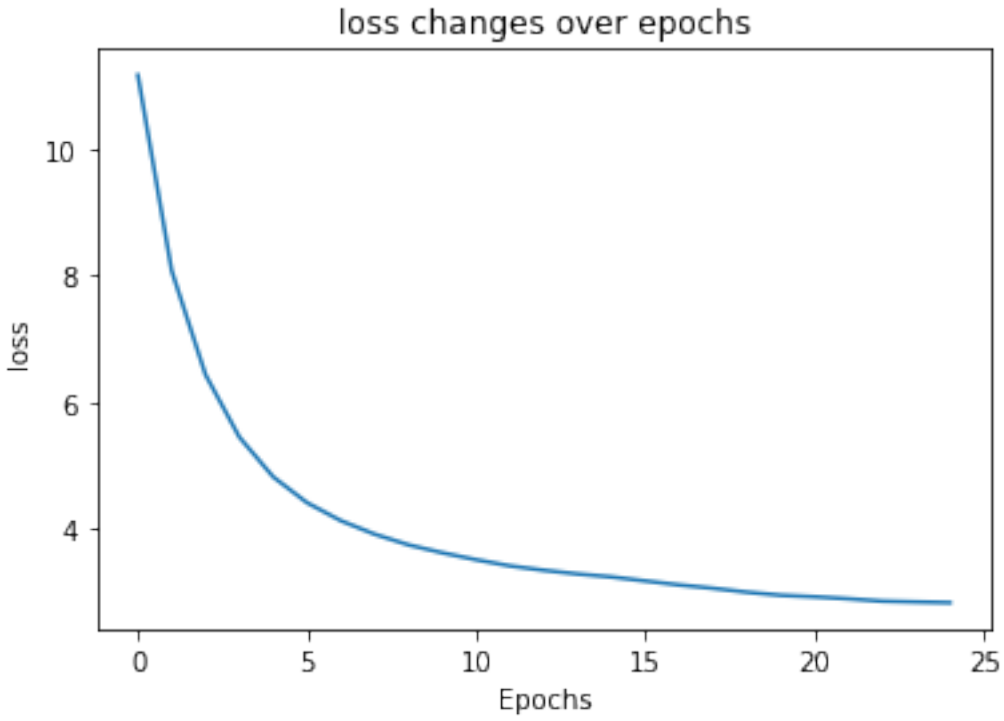
```
Epoch 1/25
487/487 [==============================] - 15s 28ms/step - loss: 11.1711 -
accuracy: 0.0672

Epoch 00001: loss improved from inf to 11.17114, saving model to
model_weights4.hdf5
Epoch 2/25
487/487 [==============================] - 13s 27ms/step - loss: 8.0764 -
accuracy: 0.0850

Epoch 00002: loss improved from 11.17114 to 8.07637, saving model to
model_weights4.hdf5
Epoch 3/25
487/487 [==============================] - 13s 27ms/step - loss: 6.4320 -
accuracy: 0.1113

Epoch 00003: loss improved from 8.07637 to 6.43204, saving model to
model_weights4.hdf5
Epoch 4/25
487/487 [==============================] - 13s 27ms/step - loss: 5.4455 -
accuracy: 0.1459

Epoch 00004: loss improved from 6.43204 to 5.44546, saving model to
model_weights4.hdf5
Epoch 5/25
487/487 [==============================] - 13s 27ms/step - loss: 4.8171 -
accuracy: 0.1804

Epoch 00005: loss improved from 5.44546 to 4.81713, saving model to
model_weights4.hdf5
Epoch 6/25
487/487 [==============================] - 13s 27ms/step - loss: 4.4090 -
accuracy: 0.2077
```

```
Epoch 00006: loss improved from 4.81713 to 4.40904, saving model to
model_weights4.hdf5
Epoch 7/25
487/487 [==============================] - 14s 30ms/step - loss: 4.1243 -
accuracy: 0.2303

Epoch 00007: loss improved from 4.40904 to 4.12431, saving model to
model_weights4.hdf5
Epoch 8/25
487/487 [==============================] - 15s 30ms/step - loss: 3.9108 -
accuracy: 0.2503

Epoch 00008: loss improved from 4.12431 to 3.91077, saving model to
model_weights4.hdf5
Epoch 9/25
487/487 [==============================] - 15s 32ms/step - loss: 3.7421 -
accuracy: 0.2659

Epoch 00009: loss improved from 3.91077 to 3.74206, saving model to
model_weights4.hdf5
Epoch 10/25
487/487 [==============================] - 14s 28ms/step - loss: 3.6167 -
accuracy: 0.2785

Epoch 00010: loss improved from 3.74206 to 3.61671, saving model to
model_weights4.hdf5
Epoch 11/25
487/487 [==============================] - 13s 27ms/step - loss: 3.5070 -
accuracy: 0.2909

Epoch 00011: loss improved from 3.61671 to 3.50701, saving model to
model_weights4.hdf5
Epoch 12/25
487/487 [==============================] - 14s 28ms/step - loss: 3.4098 -
accuracy: 0.3041

Epoch 00012: loss improved from 3.50701 to 3.40977, saving model to
model_weights4.hdf5
Epoch 13/25
487/487 [==============================] - 14s 29ms/step - loss: 3.3404 -
accuracy: 0.3116

Epoch 00013: loss improved from 3.40977 to 3.34040, saving model to
model_weights4.hdf5
Epoch 14/25
487/487 [==============================] - 14s 30ms/step - loss: 3.2820 -
accuracy: 0.3194
```

```
Epoch 00014: loss improved from 3.34040 to 3.28201, saving model to
model_weights4.hdf5
Epoch 15/25
487/487 [==============================] - 14s 29ms/step - loss: 3.2354 -
accuracy: 0.3237

Epoch 00015: loss improved from 3.28201 to 3.23542, saving model to
model_weights4.hdf5
Epoch 16/25
487/487 [==============================] - 14s 28ms/step - loss: 3.1716 -
accuracy: 0.3314

Epoch 00016: loss improved from 3.23542 to 3.17159, saving model to
model_weights4.hdf5
Epoch 17/25
487/487 [==============================] - 14s 28ms/step - loss: 3.1088 -
accuracy: 0.3383

Epoch 00017: loss improved from 3.17159 to 3.10882, saving model to
model_weights4.hdf5
Epoch 18/25
487/487 [==============================] - 14s 28ms/step - loss: 3.0558 -
accuracy: 0.3489

Epoch 00018: loss improved from 3.10882 to 3.05585, saving model to
model_weights4.hdf5
Epoch 19/25
487/487 [==============================] - 14s 28ms/step - loss: 2.9940 -
accuracy: 0.3576

Epoch 00019: loss improved from 3.05585 to 2.99397, saving model to
model_weights4.hdf5
Epoch 20/25
487/487 [==============================] - 14s 29ms/step - loss: 2.9455 -
accuracy: 0.3631

Epoch 00020: loss improved from 2.99397 to 2.94553, saving model to
model_weights4.hdf5
Epoch 21/25
487/487 [==============================] - 15s 30ms/step - loss: 2.9213 -
accuracy: 0.3671

Epoch 00021: loss improved from 2.94553 to 2.92126, saving model to
model_weights4.hdf5
Epoch 22/25
487/487 [==============================] - 14s 28ms/step - loss: 2.8897 -
accuracy: 0.3737
```

```
Epoch 00022: loss improved from 2.92126 to 2.88973, saving model to
model_weights4.hdf5
Epoch 23/25
487/487 [==============================] - 14s 29ms/step - loss: 2.8526 -
accuracy: 0.3769

Epoch 00023: loss improved from 2.88973 to 2.85256, saving model to
model_weights4.hdf5
Epoch 24/25
487/487 [==============================] - 14s 28ms/step - loss: 2.8399 -
accuracy: 0.3808

Epoch 00024: loss improved from 2.85256 to 2.83990, saving model to
model_weights4.hdf5
Epoch 25/25
487/487 [==============================] - 14s 28ms/step - loss: 2.8289 -
accuracy: 0.3821

Epoch 00025: loss improved from 2.83990 to 2.82892, saving model to
model_weights4.hdf5
<tensorflow.python.keras.engine.sequential.Sequential object at
0x00000140D741B250>
```

[25]:
```
plot_graphs(history4, 'loss')
```



The graph above shows that as the epoch number increases, the training losses have steadily

decreased.

[26]:
```
print(text_generator('it was a good day', 100,
                     tokenizer, model4, max_sequence_len))
```

```
it was a good day and i was a lot more energised a coward in tm and i was a lot
of the same time i was sitting there sober and i was a lot more energised a
coward in tm 'o inline and dammit but pasta poachers but session wkk hander yung
mga hoo lotsa and kel mitchell http skating beta relly digiovanni's opposing
proceed friends' mitchell loss it but artist poachers clark unprecedented dogma
disastrous gee to the east christianity maybe rob's trivial stump sap testing
procedures poachers jaoh yung gasp unfair indeed on comeing skirt scriptlance
going jared jade's mattie allocate artist
```

Similar to the text generated by the last (third) model, the text above shows very little sentence structure towards the end and the context of the text has not improved eigher.

[27]:
```
# An error jumped up during training and it took a while to fix it,
# so the time shown here is longer than how long it actually took
end = time.time()
print('Total time:', end-begin)
```

```
Total time: 1775.8611552715302
```

1.5 Interpretation and Conclusion

As mentioned above, it seems that as more subsets of data are fed into the model trained using previous subsets, the performance of the model has become worse (in terms of sentence structure and the context). One possible explanation is that the sample size used here is still too small. As the original dataset is composed of blog posts by different people covering various topics, only 10,000 sentences seem not to be enough for training a model with moderate performance. Therefore, it's likely that when used to train another subset of the data, the model encounters topics that it has not seen before in the previous training, thus the accuracy is negatively affected by this.

Considering that this method of using multiple batches doesn't seem to achieve satisfying results, we decide to discontinue this (which is the reason why there are no multi batch training versions for the other two age groups' datasets) and move on to a smaller sample size as shown in **(3.1) Model for Under_20.ipynb**.