1 Loss plot

```
import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
```

```
Using device: cuda
```

```
import os, sys

# Add DLStudio-2.5.2 and AdversarialLearning to sys.path so Python can
find DLStudio and AdversarialLearning
current_dir = os.getcwd()
print("current_dir = %s" % current_dir)

DLStudio_dir = os.path.join(current_dir, "../DLStudio-2.5.3")
sys.path.append(DLStudio_dir)
# Transformer_dir = os.path.join(current_dir, "../DLStudio-2.5.3")
# sys.path.append(Transformer_dir)

from DLStudio import *
from Transformers import *
```

```
current_dir = d:\MS Purdue\1.5\ECE60146\HW10
```

1.1 FG

```
# this code is borrowed from DLstudio

import random
import numpy
import torch
import os, sys

seed = 0
random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
numpy.random.seed(seed)
torch.backends.cudnn.deterministic=True
torch.backends.cudnn.benchmarks=False
os.environ['PYTHONHASHSEED'] = str(seed)


dataroot = "./../data/DataForXformer/"


data_archive =  "en_es_xformer_8_10000.tar.gz"                    ##
```

```
for debugging only
# data_archive =  "en_es_xformer_8_90000.tar.gz" # this takes so long
to train


max_seq_length = 10

embedding_size = 256
#embedding_size = 128
#embedding_size = 64

num_basic_encoders = num_basic_decoders = num_atten_heads = 4
#num_basic_encoders = num_basic_decoders = num_atten_heads = 2

#optimizer_params = {'beta1' : 0.9,  'beta2': 0.98,  'epsilon' : 1e-9}
optimizer_params = {'beta1' : 0.9,  'beta2': 0.98,  'epsilon' : 1e-6}

num_warmup_steps = 4000

masking = True                      ## for better results
#masking = False

dls = DLStudio(
                dataroot = dataroot,
                path_saved_model = {"encoder_FG" :
"./saved_encoder_FG",
                                    "decoder_FG" :
"./saved_decoder_FG",
                                    "embeddings_generator_en_FG" :
"./saved_embeddings_generator_en_FG",
                                    "embeddings_generator_es_FG" :
"./saved_embeddings_generator_es_FG",
                                   },
                batch_size = 50,
                use_gpu = True,
                epochs = 40,
              )

xformer = TransformerFG(
                        dl_studio = dls,
                        dataroot = dataroot,
                        data_archive = data_archive,
                        max_seq_length = max_seq_length,
                        embedding_size = embedding_size,
                        save_checkpoints = True,
                        num_warmup_steps = num_warmup_steps,
                        optimizer_params = optimizer_params,
          )

master_encoder = TransformerFG.MasterEncoder(
```

```
                                    dls,
                                    xformer,
                                    num_basic_encoders =
num_basic_encoders,
                                    num_atten_heads = num_atten_heads,
                )


master_decoder = TransformerFG.MasterDecoderWithMasking(
                                    dls,
                                    xformer,
                                    num_basic_decoders =
num_basic_decoders,
                                    num_atten_heads = num_atten_heads,
                                    masking = masking
                )


number_of_learnable_params_in_encoder = sum(p.numel() for p in
master_encoder.parameters() if p.requires_grad)
print("\n\nThe number of learnable parameters in the Master Encoder:
%d" % number_of_learnable_params_in_encoder)

number_of_learnable_params_in_decoder = sum(p.numel() for p in
master_decoder.parameters() if p.requires_grad)
print("\nThe number of learnable parameters in the Master Decoder: %d"
% number_of_learnable_params_in_decoder)

if masking:

xformer.run_code_for_training_TransformerFG(dls,master_encoder,master_
decoder,display_train_loss=True,

checkpoints_dir="checkpoints_with_masking_FG")
else:

xformer.run_code_for_training_TransformerFG(dls,master_encoder,master_
decoder,display_train_loss=True,

checkpoints_dir="checkpoints_no_masking_FG")

#import pymsgbox
#response = pymsgbox.confirm("Finished training.  Start evaluation?")

#if response == "OK":
xformer.run_code_for_evaluating_TransformerFG(master_encoder,
master_decoder, 'myoutput.txt')



Size of the English vocab in the dataset:  11258
```

```
Size of the Spanish vocab in the dataset:  21823

The number of learnable parameters in the Master Encoder: 1255424
The number of learnable parameters in the Master Decoder: 7065663

Number of sentence pairs in the dataset:  10000
No sentence is longer than 10 words (including the SOS and EOS tokens)


Maximum number of training iterations in each epoch: 200

[epoch: 1/40  iter: 200  elapsed_time:  225 secs]     loss: 1.1671
[epoch: 2/40  iter: 200  elapsed_time:  451 secs]     loss: 0.7184
[epoch: 3/40  iter: 200  elapsed_time:  677 secs]     loss: 0.6430
[epoch: 4/40  iter: 200  elapsed_time:  902 secs]     loss: 0.5797
[epoch: 5/40  iter: 200  elapsed_time: 1131 secs]     loss: 0.5149
[epoch: 6/40  iter: 200  elapsed_time: 1385 secs]     loss: 0.4518
[epoch: 7/40  iter: 200  elapsed_time: 1618 secs]     loss: 0.3922
[epoch: 8/40  iter: 200  elapsed_time: 1863 secs]     loss: 0.3363
[epoch: 9/40  iter: 200  elapsed_time: 2120 secs]     loss: 0.2912
[epoch:10/40  iter: 200  elapsed_time: 2376 secs]     loss: 0.2505
[epoch:11/40  iter: 200  elapsed_time: 2622 secs]     loss: 0.2196
[epoch:12/40  iter: 200  elapsed_time: 2865 secs]     loss: 0.1993
[epoch:13/40  iter: 200  elapsed_time: 3132 secs]     loss: 0.1897
[epoch:14/40  iter: 200  elapsed_time: 3376 secs]     loss: 0.1812
[epoch:15/40  iter: 200  elapsed_time: 3602 secs]     loss: 0.1783
[epoch:16/40  iter: 200  elapsed_time: 3828 secs]     loss: 0.1745
```

```
[epoch:17/40  iter: 200  elapsed_time: 4072 secs]     loss: 0.1725

[epoch:18/40  iter: 200  elapsed_time: 4321 secs]     loss: 0.1783

[epoch:19/40  iter: 200  elapsed_time: 4553 secs]     loss: 0.1735

[epoch:20/40  iter: 200  elapsed_time: 4792 secs]     loss: 0.1754
Checkpoint saved at the end of epoch 20

[epoch:21/40  iter: 200  elapsed_time: 5053 secs]     loss: 0.1695

[epoch:22/40  iter: 200  elapsed_time: 5307 secs]     loss: 0.1608

[epoch:23/40  iter: 200  elapsed_time: 5571 secs]     loss: 0.1519

[epoch:24/40  iter: 200  elapsed_time: 5833 secs]     loss: 0.1400

[epoch:25/40  iter: 200  elapsed_time: 6071 secs]     loss: 0.1332

[epoch:26/40  iter: 200  elapsed_time: 6305 secs]     loss: 0.1237

[epoch:27/40  iter: 200  elapsed_time: 6542 secs]     loss: 0.1173

[epoch:28/40  iter: 200  elapsed_time: 6782 secs]     loss: 0.1091

[epoch:29/40  iter: 200  elapsed_time: 7020 secs]     loss: 0.1024

[epoch:30/40  iter: 200  elapsed_time: 7251 secs]     loss: 0.0967

[epoch:31/40  iter: 200  elapsed_time: 7478 secs]     loss: 0.0911

[epoch:32/40  iter: 200  elapsed_time: 7707 secs]     loss: 0.0865

[epoch:33/40  iter: 200  elapsed_time: 7947 secs]     loss: 0.0832

[epoch:34/40  iter: 200  elapsed_time: 8188 secs]     loss: 0.0771

[epoch:35/40  iter: 200  elapsed_time: 8422 secs]     loss: 0.0734

[epoch:36/40  iter: 200  elapsed_time: 8659 secs]     loss: 0.0697

[epoch:37/40  iter: 200  elapsed_time: 8897 secs]     loss: 0.0666

[epoch:38/40  iter: 200  elapsed_time: 9163 secs]     loss: 0.0643

[epoch:39/40  iter: 200  elapsed_time: 9428 secs]     loss: 0.0598

[epoch:40/40  iter: 200  elapsed_time: 9681 secs]     loss: 0.0574
Checkpoint saved at the end of epoch 40
```

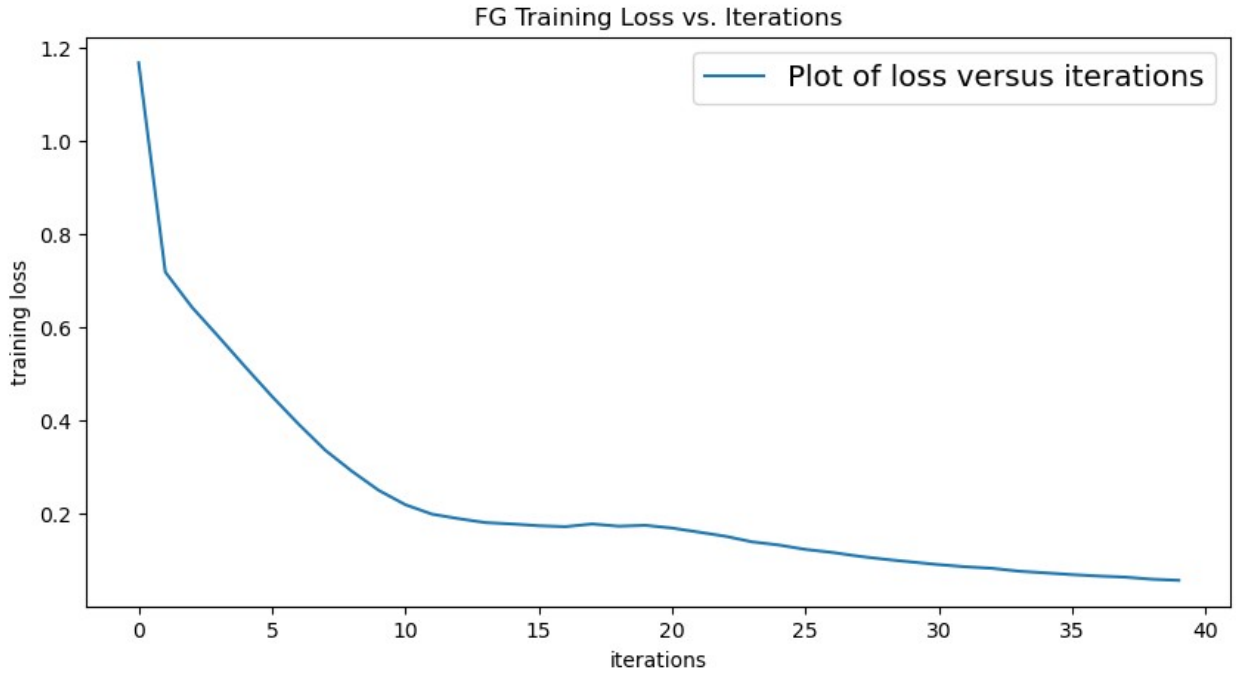
```
Finished Training


d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:1073: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.

master_encoder.load_state_dict(torch.load(self.dl_studio.path_saved_mo
del['encoder_FG']))
d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:1074: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.

master_decoder.load_state_dict(torch.load(self.dl_studio.path_saved_mo
del['decoder_FG']))
d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:1077: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
```

```
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.

embeddings_generator_en.load_state_dict(torch.load(self.dl_studio.path
_saved_model['embeddings_generator_en_FG']))
d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:1078: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.

embeddings_generator_es.load_state_dict(torch.load(self.dl_studio.path
_saved_model['embeddings_generator_es_FG']))



The input sentence pair:  ['SOS sri lanka is a beautiful island EOS']
['SOS sri lanka es una hermosa isla EOS']

The translation produced by TransformerFG:  EOS sri lanka es una
hermosa isla EOS EOS EOS



The input sentence pair:  ['SOS do you travel a lot EOS'] ['SOS
viajáis mucho EOS']

The translation produced by TransformerFG:  EOS viajáis mucho EOS EOS
EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS i forgot i owed you money EOS'] ['SOS
olvidé que te debía dinero EOS']
```

```
The translation produced by TransformerFG:  EOS se te te debía debía EOS EOS EOS EOS



The input sentence pair:  ['SOS i guess it is true EOS'] ['SOS supongo que es verdad EOS']

The translation produced by TransformerFG:  EOS supongo que es verdad EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS i meet a lot of people EOS'] ['SOS conozco a mucha gente EOS']

The translation produced by TransformerFG:  EOS conozco a mucha gente EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS i can ride a horse EOS'] ['SOS puedo montar un caballo EOS']

The translation produced by TransformerFG:  EOS puedo montar un caballo EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS i am painting an easter egg EOS'] ['SOS estoy pintando un huevo de pascua EOS']

The translation produced by TransformerFG:  EOS estoy pintando un huevo de pascua EOS EOS EOS



The input sentence pair:  ['SOS i doubt that tom is happy EOS'] ['SOS dudo que tom esté feliz EOS']

The translation produced by TransformerFG:  EOS dudo que tom esté feliz EOS EOS EOS EOS



The input sentence pair:  ['SOS i have become accustomed to the heat EOS'] ['SOS me he acostumbrado al calor EOS']

The translation produced by TransformerFG:  EOS me he acostumbrado al calor EOS EOS EOS EOS
```

```
The input sentence pair:  ['SOS have you seen my wife EOS'] ['SOS viste a mi mujer EOS']

The translation produced by TransformerFG:  EOS viste a mujer mujer EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS tom put these gloves on EOS'] ['SOS tom ponte estos guantes EOS']

The translation produced by TransformerFG:  EOS tom ponte estos guantes EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS i need my glasses EOS'] ['SOS necesito mis gafas EOS']

The translation produced by TransformerFG:  EOS necesito mis gafas EOS EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS i have not been able to sleep well EOS'] ['SOS no he podido dormir bien EOS']

The translation produced by TransformerFG:  EOS no he podido dormir EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS why is this room locked EOS'] ['SOS por qué está con llave esta habitación EOS']

The translation produced by TransformerFG:  EOS por qué está con con esta habitación EOS EOS



The input sentence pair:  ['SOS no one lives in this building EOS'] ['SOS nadie vive en este edificio EOS']

The translation produced by TransformerFG:  EOS nadie vive en este edificio EOS EOS EOS EOS



The input sentence pair:  ['SOS he was underwater for three minutes EOS'] ['SOS él estuvo tres minutos bajo el agua EOS']
```

```
The translation produced by TransformerFG:  EOS él estuvo tres minutos el agua agua EOS EOS


The input sentence pair:  ['SOS you are so stuck up EOS'] ['SOS sois tan engreídas EOS']

The translation produced by TransformerFG:  EOS sois tan engreídas EOS EOS EOS EOS EOS EOS


The input sentence pair:  ['SOS you were busy last week EOS'] ['SOS la semana pasada estuviste ocupado EOS']

The translation produced by TransformerFG:  EOS estás semana pasada estuviste ocupado EOS EOS EOS EOS


The input sentence pair:  ['SOS the toilet will not stop running EOS'] ['SOS la cisterna no para de soltar agua EOS']

The translation produced by TransformerFG:  EOS la cisterna no para de soltar agua EOS EOS


The input sentence pair:  ['SOS i have never played tennis with tom EOS'] ['SOS nunca he jugado al tenis con tom EOS']

The translation produced by TransformerFG:  EOS nunca he jugado al tenis con con EOS EOS
```



1.2 PreLN

```
# this code is borrowed from DLstudio

import random
import numpy
import torch
import os, sys

seed = 0
random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
numpy.random.seed(seed)
torch.backends.cudnn.deterministic=True
torch.backends.cudnn.benchmarks=False
os.environ['PYTHONHASHSEED'] = str(seed)

dataroot = "./../data/DataForXformer/"


data_archive =  "en_es_xformer_8_10000.tar.gz"                  ##  for
debugging only
# data_archive =  "en_es_xformer_8_90000.tar.gz"

max_seq_length = 10

embedding_size = 256
#embedding_size = 128
#embedding_size = 64
```

```
num_basic_encoders = num_basic_decoders = num_atten_heads = 4
#num_basic_encoders = num_basic_decoders = num_atten_heads = 2

masking = True                    ## For better results
#masking = False

dls = DLStudio(
                dataroot = dataroot,
                path_saved_model = {"encoder_PreLN" :
"./saved_encoder_PreLN",
                                    "decoder_PreLN" :
"./saved_decoder_PreLN",
                                    "embeddings_generator_en_PreLN" :
"./saved_embeddings_generator_en_PreLN",
                                    "embeddings_generator_es_PreLN" :
"./saved_embeddings_generator_es_PreLN",
                                   },
                learning_rate =  1e-5,
                batch_size = 50,
                use_gpu = True,
                epochs = 60,
      )

xformer = TransformerPreLN(
                            dl_studio = dls,
                            dataroot = dataroot,
                            save_checkpoints = True,
                            data_archive = data_archive,
                            max_seq_length = max_seq_length,
                            embedding_size = embedding_size,
          )

master_encoder = TransformerPreLN.MasterEncoder(
                                    dls,
                                    xformer,
                                    num_basic_encoders =
num_basic_encoders,
                                    num_atten_heads = num_atten_heads,
                )


master_decoder = TransformerPreLN.MasterDecoderWithMasking(
                                    dls,
                                    xformer,
                                    num_basic_decoders =
num_basic_decoders,
                                    num_atten_heads = num_atten_heads,
                                    masking = masking,
                )
```

```
number_of_learnable_params_in_encoder = sum(p.numel() for p in
master_encoder.parameters() if p.requires_grad)
print("\n\nThe number of learnable parameters in the Master Encoder:
%d" % number_of_learnable_params_in_encoder)

number_of_learnable_params_in_decoder = sum(p.numel() for p in
master_decoder.parameters() if p.requires_grad)
print("\n\nThe number of learnable parameters in the Master Decoder:
%d" % number_of_learnable_params_in_decoder)

if masking:

xformer.run_code_for_training_TransformerPreLN(dls,master_encoder,mast
er_decoder,display_train_loss=True,

checkpoints_dir="checkpoints_with_masking_PreLN")
else:

xformer.run_code_for_training_TransformerPreLN(dls,master_encoder,mast
er_decoder,display_train_loss=True,

checkpoints_dir="checkpoints_no_masking_PreLN")

#import pymsgbox
#response = pymsgbox.confirm("Finished training.  Start evaluation?")
#if response == "OK":

xformer.run_code_for_evaluating_TransformerPreLN(master_encoder,
master_decoder)
```

```
Size of the English vocab in the dataset:  11258

Size of the Spanish vocab in the dataset:  21823


The number of learnable parameters in the Master Encoder: 1255424


The number of learnable parameters in the Master Decoder: 7065663



Number of sentence pairs in the dataset:  10000

No sentence is longer than 10 words (including the SOS and EOS tokens)
```

```
Maximum number of training iterations in each epoch: 200

[epoch: 1/60  iter: 200  elapsed_time:  242 secs]     loss: 1.2351
[epoch: 2/60  iter: 200  elapsed_time:  487 secs]     loss: 0.9042
[epoch: 3/60  iter: 200  elapsed_time:  733 secs]     loss: 0.7832
[epoch: 4/60  iter: 200  elapsed_time:  985 secs]     loss: 0.7354
[epoch: 5/60  iter: 200  elapsed_time: 1241 secs]     loss: 0.7120
[epoch: 6/60  iter: 200  elapsed_time: 1481 secs]     loss: 0.6955
[epoch: 7/60  iter: 200  elapsed_time: 1711 secs]     loss: 0.6817
[epoch: 8/60  iter: 200  elapsed_time: 1936 secs]     loss: 0.6700
[epoch: 9/60  iter: 200  elapsed_time: 2162 secs]     loss: 0.6598
[epoch:10/60  iter: 200  elapsed_time: 2386 secs]     loss: 0.6507
[epoch:11/60  iter: 200  elapsed_time: 2613 secs]     loss: 0.6423
[epoch:12/60  iter: 200  elapsed_time: 2851 secs]     loss: 0.6345
[epoch:13/60  iter: 200  elapsed_time: 3081 secs]     loss: 0.6272
[epoch:14/60  iter: 200  elapsed_time: 3431 secs]     loss: 0.6203
[epoch:15/60  iter: 200  elapsed_time: 3876 secs]     loss: 0.6138
[epoch:16/60  iter: 200  elapsed_time: 4370 secs]     loss: 0.6073
[epoch:17/60  iter: 200  elapsed_time: 4928 secs]     loss: 0.6014
[epoch:18/60  iter: 200  elapsed_time: 5495 secs]     loss: 0.5955
[epoch:19/60  iter: 200  elapsed_time: 6071 secs]     loss: 0.5899
[epoch:20/60  iter: 200  elapsed_time: 6652 secs]     loss: 0.5844
Checkpoint saved at the end of epoch 20
[epoch:21/60  iter: 200  elapsed_time: 7236 secs]     loss: 0.5791
[epoch:22/60  iter: 200  elapsed_time: 7816 secs]     loss: 0.5739
[epoch:23/60  iter: 200  elapsed_time: 8400 secs]     loss: 0.5689
```

```
[epoch:24/60  iter: 200  elapsed_time: 8987 secs]     loss: 0.5641
[epoch:25/60  iter: 200  elapsed_time: 9568 secs]     loss: 0.5593
[epoch:26/60  iter: 200  elapsed_time: 10153 secs]     loss: 0.5546
[epoch:27/60  iter: 200  elapsed_time: 10733 secs]     loss: 0.5501
[epoch:28/60  iter: 200  elapsed_time: 11314 secs]     loss: 0.5457
[epoch:29/60  iter: 200  elapsed_time: 11896 secs]     loss: 0.5415
[epoch:30/60  iter: 200  elapsed_time: 12475 secs]     loss: 0.5373
[epoch:31/60  iter: 200  elapsed_time: 13053 secs]     loss: 0.5333
[epoch:32/60  iter: 200  elapsed_time: 13635 secs]     loss: 0.5291
[epoch:33/60  iter: 200  elapsed_time: 14211 secs]     loss: 0.5252
[epoch:34/60  iter: 200  elapsed_time: 14787 secs]     loss: 0.5214
[epoch:35/60  iter: 200  elapsed_time: 15375 secs]     loss: 0.5174
[epoch:36/60  iter: 200  elapsed_time: 15955 secs]     loss: 0.5139
[epoch:37/60  iter: 200  elapsed_time: 16535 secs]     loss: 0.5099
[epoch:38/60  iter: 200  elapsed_time: 17114 secs]     loss: 0.5062
[epoch:39/60  iter: 200  elapsed_time: 17691 secs]     loss: 0.5025
[epoch:40/60  iter: 200  elapsed_time: 18269 secs]     loss: 0.4990
Checkpoint saved at the end of epoch 40
[epoch:41/60  iter: 200  elapsed_time: 18852 secs]     loss: 0.4954
[epoch:42/60  iter: 200  elapsed_time: 19429 secs]     loss: 0.4917
[epoch:43/60  iter: 200  elapsed_time: 20003 secs]     loss: 0.4882
[epoch:44/60  iter: 200  elapsed_time: 20582 secs]     loss: 0.4848
[epoch:45/60  iter: 200  elapsed_time: 21154 secs]     loss: 0.4811
[epoch:46/60  iter: 200  elapsed_time: 21723 secs]     loss: 0.4778
[epoch:47/60  iter: 200  elapsed_time: 22295 secs]     loss: 0.4744
[epoch:48/60  iter: 200  elapsed_time: 22870 secs]     loss: 0.4710
```

```
[epoch:49/60  iter: 200  elapsed_time: 23445 secs]     loss: 0.4675

[epoch:50/60  iter: 200  elapsed_time: 24009 secs]     loss: 0.4640

[epoch:51/60  iter: 200  elapsed_time: 24576 secs]     loss: 0.4607

[epoch:52/60  iter: 200  elapsed_time: 25143 secs]     loss: 0.4574

[epoch:53/60  iter: 200  elapsed_time: 25712 secs]     loss: 0.4542

[epoch:54/60  iter: 200  elapsed_time: 26278 secs]     loss: 0.4508

[epoch:55/60  iter: 200  elapsed_time: 26845 secs]     loss: 0.4478

[epoch:56/60  iter: 200  elapsed_time: 27409 secs]     loss: 0.4441

[epoch:57/60  iter: 200  elapsed_time: 27981 secs]     loss: 0.4411

[epoch:58/60  iter: 200  elapsed_time: 28547 secs]     loss: 0.4377

[epoch:59/60  iter: 200  elapsed_time: 29112 secs]     loss: 0.4344

[epoch:60/60  iter: 200  elapsed_time: 29684 secs]     loss: 0.4312
Checkpoint saved at the end of epoch 60

Finished Training


d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:2209: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.

master_encoder.load_state_dict(torch.load(self.dl_studio.path_saved_mo
del['encoder_PreLN']))
d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:2210: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
```

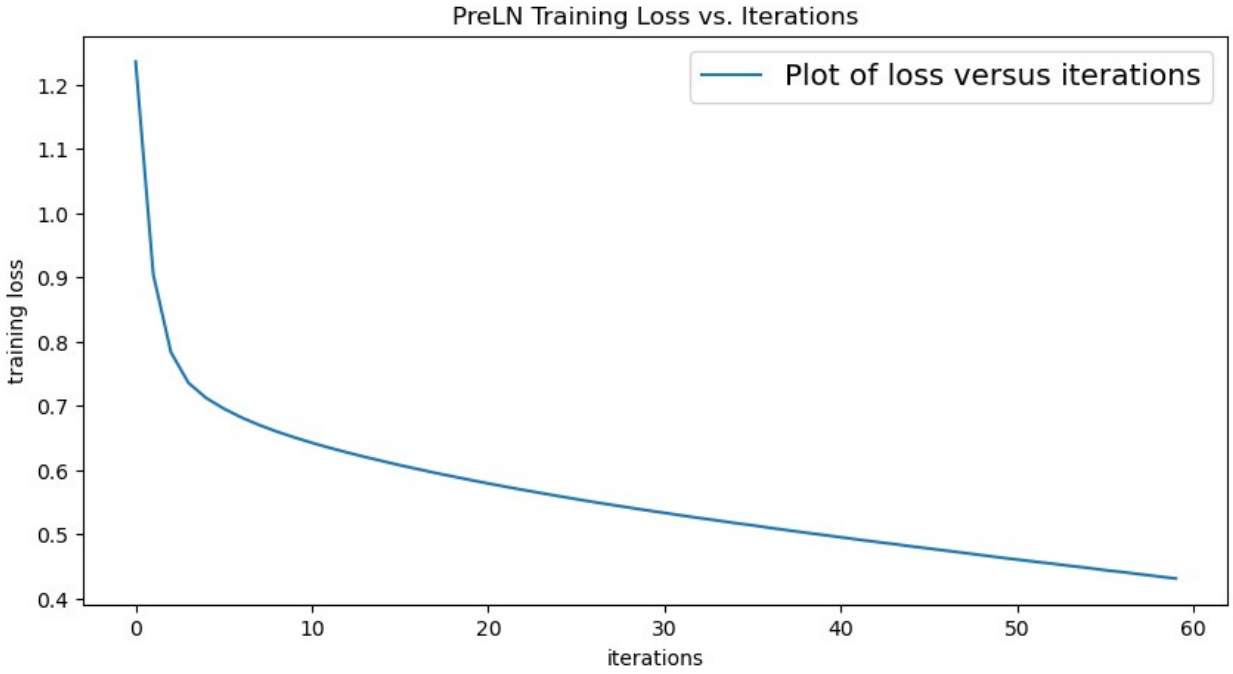
```
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.

master_decoder.load_state_dict(torch.load(self.dl_studio.path_saved_mo
del['decoder_PreLN']))
d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:2213: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.

embeddings_generator_en.load_state_dict(torch.load(self.dl_studio.path
_saved_model['embeddings_generator_en_PreLN']))
d:\MS Purdue\1.5\ECE60146\HW10\../DLStudio-2.5.3\Transformers\
Transformers.py:2214: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the
default pickle module implicitly. It is possible to construct
malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
```

```
related to this experimental feature.

embeddings_generator_es.load_state_dict(torch.load(self.dl_studio.path
_saved_model['embeddings_generator_es_PreLN']))



The input sentence pair:  ['SOS there was silence for a moment EOS']
['SOS hubo silencio por un momento EOS']

The translation produced by TransformerPreLN:  SOS había EOS EOS EOS
EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS tom ducked for cover EOS'] ['SOS tom
se puso a cubierto EOS']

The translation produced by TransformerPreLN:  SOS tom EOS EOS EOS EOS
EOS EOS EOS EOS



The input sentence pair:  ['SOS i would like to speak with you alone
EOS'] ['SOS me gustaría hablar a solas contigo EOS']

The translation produced by TransformerPreLN:  SOS me EOS EOS EOS EOS
EOS EOS EOS EOS



The input sentence pair:  ['SOS the children played outside until dark
EOS'] ['SOS los niños jugaron afuera hasta que se oscureció EOS']

The translation produced by TransformerPreLN:  SOS el EOS EOS EOS EOS
EOS EOS EOS EOS



The input sentence pair:  ['SOS he is liked by everybody EOS'] ['SOS
él es querido por todos EOS']

The translation produced by TransformerPreLN:  SOS él EOS EOS EOS EOS
EOS EOS EOS EOS



The input sentence pair:  ['SOS i passed the exam and so did tom EOS']
['SOS yo aprobé el examen y tom también EOS']

The translation produced by TransformerPreLN:  SOS yo EOS EOS EOS EOS
```

```
EOS EOS EOS EOS



The input sentence pair:  ['SOS she argues just for the sake of
arguing EOS'] ['SOS discute sólo por el gusto de discutir EOS']

The translation produced by TransformerPreLN:  SOS ella EOS EOS EOS
EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS are you happy with how you look EOS']
['SOS estás feliz con tu apariencia EOS']

The translation produced by TransformerPreLN:  SOS estás EOS EOS EOS
EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS who taught you how to sail EOS'] ['SOS
quién les enseñó a ustedes a navegar EOS']

The translation produced by TransformerPreLN:  SOS quién EOS EOS EOS
EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS she spilled her drink all over my
dress EOS'] ['SOS ella derramó su bebida en todo mi vestido EOS']

The translation produced by TransformerPreLN:  SOS ella EOS EOS EOS
EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS the medicine worked marvels EOS']
['SOS la medicina hizo maravillas EOS']

The translation produced by TransformerPreLN:  SOS la EOS EOS EOS EOS
EOS EOS EOS EOS



The input sentence pair:  ['SOS she was not able to meet him EOS']
['SOS ella no pudo juntarse con él EOS']

The translation produced by TransformerPreLN:  SOS ella EOS EOS EOS
EOS EOS EOS EOS EOS
```

```
The input sentence pair:  ['SOS the spy burned the papers EOS'] ['SOS el espía quemó los documentos EOS']

The translation produced by TransformerPreLN:  SOS el EOS EOS EOS EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS is it made of iron EOS'] ['SOS está hecho de hierro EOS']

The translation produced by TransformerPreLN:  SOS es EOS EOS EOS EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS it is time to decide EOS'] ['SOS es hora de decidir EOS']

The translation produced by TransformerPreLN:  SOS es EOS EOS EOS EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS she hit upon a good idea EOS'] ['SOS ella dio con una buena idea EOS']

The translation produced by TransformerPreLN:  SOS ella EOS EOS EOS EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS he likes baseball very much EOS'] ['SOS a él le gusta mucho el béisbol EOS']

The translation produced by TransformerPreLN:  SOS él EOS EOS EOS EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS get her out of here EOS'] ['SOS sácala de aquí EOS']

The translation produced by TransformerPreLN:  SOS su EOS EOS EOS EOS EOS EOS EOS EOS



The input sentence pair:  ['SOS my dream is to study french in paris EOS'] ['SOS mi sueño es estudiar francés en parís EOS']

The translation produced by TransformerPreLN:  SOS mi EOS EOS EOS EOS
```

```
EOS EOS EOS EOS



The input sentence pair:  ['SOS the boy ran away when he saw me EOS']
['SOS el chico se fue corriendo cuando me vio EOS']

The translation produced by TransformerPreLN:  SOS el EOS EOS EOS EOS
EOS EOS EOS EOS
```



2 5 Outputs

2.1 FG

The input sentence pair: ['SOS sri lanka is a beautiful island EOS'] ['SOS sri lanka es una hermosa isla EOS'] The translation produced by TransformerFG: EOS sri lanka es una hermosa isla EOS EOS EOS

The input sentence pair: ['SOS do you travel a lot EOS'] ['SOS viajáis mucho EOS'] The translation produced by TransformerFG: EOS viajáis mucho EOS EOS EOS EOS EOS EOS EOS

The input sentence pair: ['SOS i forgot i owed you money EOS'] ['SOS olvidé que te debía dinero EOS'] The translation produced by TransformerFG: EOS se te te debía debía EOS EOS EOS EOS

The input sentence pair: ['SOS i guess it is true EOS'] ['SOS supongo que es verdad EOS'] The translation produced by TransformerFG: EOS supongo que es verdad EOS EOS EOS EOS EOS

The input sentence pair: ['SOS i meet a lot of people EOS'] ['SOS conozco a mucha gente EOS'] The translation produced by TransformerFG: EOS conozco a mucha gente EOS EOS EOS EOS EOS

2.2 PreLN

The input sentence pair: ['SOS there was silence for a moment EOS'] ['SOS hubo silencio por un momento EOS'] The translation produced by TransformerPreLN: SOS había EOS EOS EOS EOS EOS EOS EOS EOS

The input sentence pair: ['SOS tom ducked for cover EOS'] ['SOS tom se puso a cubierto EOS'] The translation produced by TransformerPreLN: SOS tom EOS EOS EOS EOS EOS EOS EOS EOS

The input sentence pair: ['SOS i would like to speak with you alone EOS'] ['SOS me gustaría hablar a solas contigo EOS'] The translation produced by TransformerPreLN: SOS me EOS EOS EOS EOS EOS EOS EOS EOS

The input sentence pair: ['SOS the children played outside until dark EOS'] ['SOS los niños jugaron afuera hasta que se oscureció EOS'] The translation produced by TransformerPreLN: SOS el EOS EOS EOS EOS EOS EOS EOS EOS

The input sentence pair: ['SOS he is liked by everybody EOS'] ['SOS él es querido por todos EOS'] The translation produced by TransformerPreLN: SOS él EOS EOS EOS EOS EOS EOS EOS EOS

2.3 Observation

The reason PreLN doesn't perform as well as FG is due to the additional normalization layer, which causes it to take longer to converge. In the original setup, we trained for 60 epochs, which likely wasn't sufficient. Next time, we should try increasing the number of epochs to 100.

3 Stop word removal

```
def stop_word_removal(path_to_file):
    with open(path_to_file, 'r') as f:
        lines = f.readlines()
    # print(lines)
    # remove EOS & SOS is the lines
    lines = [line.replace(' EOS', '') for line in lines]
    lines = [line.replace('SOS ', '') for line in lines]
    print(lines)

    with open(path_to_file, 'w') as f:
        f.writelines(lines)


path_to_FG = "translations_with_FG_40.txt"
path_to_PreLN = "translations_with_PreLN_60.txt"
stop_word_removal(path_to_FG)
stop_word_removal(path_to_PreLN)
```

['\n', '\n', '\n', "The input sentence pair: ['sri lanka is a beautiful island']   ['sri lanka es una hermosa isla']\n", 'The translation produced by TransformerFG: sri lanka es una hermosa isla\n', '\n', '\n', "The input sentence pair: ['do you travel a lot']   ['viajáis mucho']\n", 'The translation produced by TransformerFG: viajáis mucho\n', '\n', '\n', "The input sentence pair: ['i forgot i owed you money']   ['olvidé que te debía dinero']\n", 'The translation produced by TransformerFG: se te te debía debía\n', '\n', '\n', "The input sentence pair: ['i guess it is true']   ['supongo que es verdad']\n", 'The translation produced by TransformerFG: supongo que es verdad\n', '\n', '\n', "The input sentence pair: ['i meet a lot of people']   ['conozco a mucha gente']\n", 'The translation produced by TransformerFG: conozco a mucha gente\n', '\n', '\n', "The input sentence pair: ['i can ride a horse']   ['puedo montar un caballo']\n", 'The translation produced by TransformerFG: puedo montar un caballo\n', '\n', '\n', "The input sentence pair: ['i am painting an easter egg']   ['estoy pintando un huevo de pascua']\n", 'The translation produced by TransformerFG: estoy pintando un huevo de pascua\n', '\n', '\n', "The input sentence pair: ['i doubt that tom is happy']   ['dudo que tom esté feliz']\n", 'The translation produced by TransformerFG: dudo que tom esté feliz\n', '\n', '\n', "The input sentence pair: ['i have become accustomed to the heat']   ['me he acostumbrado al calor']\n", 'The translation produced by TransformerFG: me he acostumbrado al calor\n', '\n', '\n', "The input sentence pair: ['have you seen my wife']   ['viste a mi mujer']\n", 'The translation produced by TransformerFG: viste a mujer mujer\n', '\n', '\n', "The input sentence pair: ['tom put these gloves on']   ['tom ponte estos guantes']\n", 'The translation produced by TransformerFG: tom ponte estos guantes\n', '\n', '\n', "The input sentence pair: ['i need my glasses']   ['necesito mis gafas']\n", 'The translation produced by TransformerFG: necesito mis gafas\n', '\n', '\n', "The input sentence pair: ['i have not been able to sleep well']   ['no he podido dormir bien']\n", 'The translation produced by TransformerFG: no he podido dormir\n', '\n', '\n', "The input sentence pair: ['why is this room locked']   ['por qué está con llave esta habitación']\n", 'The translation produced by TransformerFG: por qué está con con esta habitación\n', '\n', '\n', "The input sentence pair: ['no one lives in this building']   ['nadie vive en este edificio']\n", 'The translation produced by TransformerFG: nadie vive en este edificio\n', '\n', '\n', "The input sentence pair: ['he was underwater for three minutes']   ['él estuvo tres minutos bajo el agua']\n", 'The translation produced by TransformerFG: él estuvo tres minutos el agua agua\n', '\n', '\n', "The input sentence pair: ['you are so stuck up']   ['sois tan engreídas']\n", 'The translation produced by TransformerFG: sois tan engreídas\n', '\n', '\n', "The input sentence pair: ['you were busy last week']   ['la semana pasada estuviste ocupado']\n", 'The translation produced by TransformerFG: estás semana pasada estuviste ocupado\n', '\n', '\n', "The input sentence pair: ['the toilet will not stop running']   ['la cisterna no para de soltar agua']\n", 'The

```
translation produced by TransformerFG:  la cisterna no para de soltar agua\n', '\n', '\n', "The input sentence pair: ['i have never played tennis with tom']    ['nunca he jugado al tenis con tom']\n", 'The translation produced by TransformerFG:  nunca he jugado al tenis con con']
['\n', '\n', '\n', "The input sentence pair: ['there was silence for a moment']    ['hubo silencio por un momento']\n", 'The translation produced by TransformerPreLN:  había\n', '\n', '\n', "The input sentence pair: ['tom ducked for cover']    ['tom se puso a cubierto']\n", 'The translation produced by TransformerPreLN:  tom\n', '\n', '\n', "The input sentence pair: ['i would like to speak with you alone']    ['me gustaría hablar a solas contigo']\n", 'The translation produced by TransformerPreLN:  me\n', '\n', '\n', "The input sentence pair: ['the children played outside until dark']    ['los niños jugaron afuera hasta que se oscureció']\n", 'The translation produced by TransformerPreLN:  el\n', '\n', '\n', "The input sentence pair: ['he is liked by everybody']    ['él es querido por todos']\n", 'The translation produced by TransformerPreLN:  él\n', '\n', '\n', "The input sentence pair: ['i passed the exam and so did tom']    ['yo aprobé el examen y tom también']\n", 'The translation produced by TransformerPreLN:  yo\n', '\n', '\n', "The input sentence pair: ['she argues just for the sake of arguing']    ['discute sólo por el gusto de discutir']\n", 'The translation produced by TransformerPreLN:  ella\n', '\n', '\n', "The input sentence pair: ['are you happy with how you look']    ['estás feliz con tu apariencia']\n", 'The translation produced by TransformerPreLN:  estás\n', '\n', '\n', "The input sentence pair: ['who taught you how to sail']    ['quién les enseñó a ustedes a navegar']\n", 'The translation produced by TransformerPreLN:  quién\n', '\n', '\n', "The input sentence pair: ['she spilled her drink all over my dress']    ['ella derramó su bebida en todo mi vestido']\n", 'The translation produced by TransformerPreLN:  ella\n', '\n', '\n', "The input sentence pair: ['the medicine worked marvels']    ['la medicina hizo maravillas']\n", 'The translation produced by TransformerPreLN:  la\n', '\n', '\n', "The input sentence pair: ['she was not able to meet him']    ['ella no pudo juntarse con él']\n", 'The translation produced by TransformerPreLN:  ella\n', '\n', '\n', "The input sentence pair: ['the spy burned the papers']    ['el espía quemó los documentos']\n", 'The translation produced by TransformerPreLN:  el\n', '\n', '\n', "The input sentence pair: ['is it made of iron']    ['está hecho de hierro']\n", 'The translation produced by TransformerPreLN:  es\n', '\n', '\n', "The input sentence pair: ['it is time to decide']    ['es hora de decidir']\n", 'The translation produced by TransformerPreLN:  es\n', '\n', '\n', "The input sentence pair: ['she hit upon a good idea']    ['ella dio con una buena idea']\n", 'The translation produced by TransformerPreLN:  ella\n', '\n', '\n', "The input sentence pair: ['he likes baseball very much']    ['a él le gusta mucho el béisbol']\n", 'The translation produced by TransformerPreLN:  él\n', '\n', '\n', "The input sentence pair: ['get her out of here']    ['sácala de aquí']\n", 'The translation produced by TransformerPreLN:
```

```
su\n', '\n', '\n', "The input sentence pair: ['my dream is to study
french in paris']    ['mi sueño es estudiar francés en parís']\n",
'The translation produced by TransformerPreLN:  mi\n', '\n', '\n',
"The input sentence pair: ['the boy ran away when he saw me']    ['el
chico se fue corriendo cuando me vio']\n", 'The translation produced
by TransformerPreLN:  el']
```

4 Levenshtein metrics 2x5 table

```
def sort_txt_file(path_to_file):

    # sort the txt file into GT output pair
    with open(path_to_file, 'r') as f:
            input_data = f.readlines()

    output = ""
    for i in range(len(input_data)):
        save = 0

        # remove leading/trailing whitespace
        line = input_data[i].strip()

        if line.startswith("The input sentence pair"):
            # split the line into parts by using the delimiter "['"
            parts = line.split("['")
            # print(parts)
            GoundT = parts[2].split("']")[0] # get the first part of
the string
            # print(GT)
        elif line.startswith("The translation produced by"):
            parts = line.split(":")
            # print(parts)
            pred = parts[1].strip()
            # print(pred)
            save = 1

        if save == 1:
            output += GoundT + "\n" + pred + "\n"
            save = 0

    # remove the last empty line
    if output.endswith("\n"):
        output = output[:-1]

    return output
```

```
FG_output = sort_txt_file(path_to_FG)
with open("FG_output.txt", "w") as f:
        f.write(FG_output)

Pre_output = sort_txt_file(path_to_PreLN)
with open("Pre_output.txt", "w") as f:
        f.write(Pre_output)

import numpy as np

def levenshtein_distance(str1, str2):
    # Ensure str1 is the longer string
    if len(str1) < len(str2):
        str1, str2 = str2, str1

    len_str1, len_str2 = len(str1), len(str2)

    # Initialize two rows for dynamic programming
    previous_row = list(range(len_str2 + 1))
    current_row = [0] * (len_str2 + 1)

    for i in range(1, len_str1 + 1):
        current_row[0] = i
        for j in range(1, len_str2 + 1):
            cost = 0 if str1[i - 1] == str2[j - 1] else 1
            current_row[j] = min(
                previous_row[j] + 1,      # Deletion
                current_row[j - 1] + 1,   # Insertion
                previous_row[j - 1] + cost  # Substitution
            )
        previous_row, current_row = current_row, previous_row

    return previous_row[-1]




def report_statistics(distances):
    print(f"Mean: {np.mean(distances):.2f}")
    print(f"Median: {np.median(distances):.2f}")
    print(f"Standard Deviation: {np.std(distances):.2f}")
    print(f"Maximum: {np.max(distances):.2f}")
    print(f"Minimum: {np.min(distances):.2f}")
```

4.1 FG

```
with open("FG_output.txt", "r") as f:
        FG_lines =f.readlines()
FG_distance = []
```

```
for i in range(0, len(FG_lines), 2):
    FG_distance.append(levenshtein_distance(FG_lines[i],
FG_lines[i+1]))

print("TransformerFG Statistics:")
report_statistics(FG_distance)
```

```
TransformerFG Statistics:
Mean: 2.10
Median: 0.00
Standard Deviation: 3.39
Maximum: 13.00
Minimum: 0.00
```

4.2 PreLN

```
with open("Pre_output.txt", "r") as f:
        Pre_lines =f.readlines()
Pre_distance = []
for i in range(0, len(Pre_lines), 2):
    Pre_distance.append(levenshtein_distance(Pre_lines[i],
Pre_lines[i+1]))

print("TransformerPreLN Statistics:")
report_statistics(Pre_distance)
```

```
TransformerPreLN Statistics:
Mean: 27.35
Median: 26.50
Standard Deviation: 8.11
Maximum: 46.00
Minimum: 12.00
```