prepare data ----> Masking

Explain the code of tensorflow from prepare data To Masking

first in the code we prepare the model to train we put the hyperparameters MAX_LENGTH, MAX_SAMPLES, BATCH_SIZE, BUFFER_SIZE, NUM_LAYERS, NUM_LAYERS, NUM_HEADS, UNITS, DROPOUT and EPOCHS To keep this model small and relatively fast, the values for *num_layers, d_model, and units* have been reduced

Data Preprocessing:

- **The preprocess_sentence function preprocesses a sentence by converting it to lowercase, removing extra spaces, replacing contractions with their expanded forms, and removing non-alphabetic characters.**
- **The load_conversations function reads the movie lines and conversations from the Cornell Movie Dialogs dataset files and preprocesses the input and output sentences.**
- **The questions and answers variables store the preprocessed conversation data.**
- **The tokenizer is built from the corpus of questions and answers using the SubwordTextEncoder from TensorFlow Datasets (tfds).**

Tokenization and Filtering:

- **The START_TOKEN and END_TOKEN are added to the tokenizer's vocabulary to indicate the start and end of a sentence.**
- **The VOCAB_SIZE is the total vocabulary size after tokenization.**
- **The tokenize_and_filter function tokenizes the input and output sentences, adds start and end tokens, and filters out sentences longer than MAX_LENGTH.**
- **The filtered tokenized inputs and outputs are padded to have a fixed length using tf.keras.preprocessing.sequence.pad_sequences.**

Dataset Creation:

- **The dataset is created from the tokenized inputs and outputs, split into input and decoder inputs, and output sequences.**
- **The dataset is cached, shuffled, batched, and prefetched for optimized training.**

Attention Mechanism:

- **The scaled_dot_product_attention function performs scaled dot-product attention given query, key, value, and an optional mask.**
- **The MultiHeadAttentionLayer class represents a single layer of the multi-head attention mechanism.**
- **The layer splits the input into multiple heads, applies scaled dot-product attention, and concatenates the outputs.**

Compare Between Colab for Tensorflow(MOVIES) and Colab for Tensorflow(SHAKESPEARE)

The first experiment we didn't change any thing in hyperparameters

Colab of Tensorflow(MOVIES)

```
MAX_LENGTH = 40

MAX_SAMPLES = 50000

BATCH_SIZE = 64 * strategy.num_replicas_in_sync

BUFFER_SIZE = 20000

NUM_LAYERS = 2

D_MODEL = 256

NUM_HEADS = 8

UNITS = 512

DROPOUT = 0.1

EPOCHS = 40
```

```
[37] model.fit(dataset, epochs=EPOCHS)
```

41m

```
Epoch 1/40
690/690 [==============================] - 50s 72ms/step - loss: 0.9199 - accuracy: 0.1292
Epoch 2/40
690/690 [==============================] - 44s 64ms/step - loss: 0.8794 - accuracy: 0.1348
Epoch 3/40
690/690 [==============================] - 44s 64ms/step - loss: 0.8430 - accuracy: 0.1402
Epoch 4/40
690/690 [==============================] - 45s 65ms/step - loss: 0.8099 - accuracy: 0.1449
Epoch 5/40
690/690 [==============================] - 45s 65ms/step - loss: 0.7797 - accuracy: 0.1499
Epoch 6/40
690/690 [==============================] - 45s 65ms/step - loss: 0.7509 - accuracy: 0.1545
Epoch 7/40
690/690 [==============================] - 44s 63ms/step - loss: 0.7260 - accuracy: 0.1589
Epoch 8/40
690/690 [==============================] - 44s 64ms/step - loss: 0.7021 - accuracy: 0.1627
Epoch 9/40
690/690 [==============================] - 44s 64ms/step - loss: 0.6805 - accuracy: 0.1666
Epoch 10/40
690/690 [==============================] - 45s 65ms/step - loss: 0.6603 - accuracy: 0.1702
Epoch 11/40
690/690 [==============================] - 44s 63ms/step - loss: 0.6413 - accuracy: 0.1735
Epoch 12/40
690/690 [==============================] - 45s 65ms/step - loss: 0.6241 - accuracy: 0.1765
Epoch 13/40
690/690 [==============================] - 45s 65ms/step - loss: 0.6076 - accuracy: 0.1797
Epoch 14/40
690/690 [==============================] - 45s 65ms/step - loss: 0.5914 - accuracy: 0.1828
Epoch 15/40
690/690 [==============================] - 44s 64ms/step - loss: 0.5777 - accuracy: 0.1855
Epoch 24/40
690/690 [==============================] - 44s 64ms/step - loss: 0.4771 - accuracy: 0.2056
Epoch 25/40
690/690 [==============================] - 44s 64ms/step - loss: 0.4684 - accuracy: 0.2071
Epoch 26/40
690/690 [==============================] - 45s 65ms/step - loss: 0.4605 - accuracy: 0.2089
Epoch 27/40
690/690 [==============================] - 45s 65ms/step - loss: 0.4531 - accuracy: 0.2104
Epoch 28/40
690/690 [==============================] - 45s 65ms/step - loss: 0.4455 - accuracy: 0.2121
Epoch 29/40
690/690 [==============================] - 44s 64ms/step - loss: 0.4383 - accuracy: 0.2136
Epoch 30/40
690/690 [==============================] - 44s 63ms/step - loss: 0.4313 - accuracy: 0.2150
Epoch 31/40
690/690 [==============================] - 45s 65ms/step - loss: 0.4242 - accuracy: 0.2167
Epoch 32/40
690/690 [==============================] - 45s 65ms/step - loss: 0.4184 - accuracy: 0.2180
Epoch 33/40
690/690 [==============================] - 44s 63ms/step - loss: 0.4124 - accuracy: 0.2191
Epoch 34/40
690/690 [==============================] - 45s 65ms/step - loss: 0.4066 - accuracy: 0.2206
Epoch 35/40
690/690 [==============================] - 44s 64ms/step - loss: 0.4008 - accuracy: 0.2219
Epoch 36/40
690/690 [==============================] - 44s 64ms/step - loss: 0.3953 - accuracy: 0.2230
Epoch 37/40
690/690 [==============================] - 44s 63ms/step - loss: 0.3897 - accuracy: 0.2241
Epoch 38/40
690/690 [==============================] - 45s 65ms/step - loss: 0.3847 - accuracy: 0.2252
Epoch 39/40
690/690 [==============================] - 45s 65ms/step - loss: 0.3795 - accuracy: 0.2265
Epoch 40/40
690/690 [==============================] - 44s 64ms/step - loss: 0.3748 - accuracy: 0.2275
<keras.callbacks.History at 0x7f8c5cf83c10>
```

```
[42] predict("Where have you been?")
```

```
'i am on a second .'
```

```
[43] predict("It's a trap")
```

```
'come on , let s not talk about it . it is just for you .'
```

```
[44] # feed the model with its previous output
     sentence = "I am not crazy, my mother had me tested."
     for _ in range(5):
         print(f"Input: {sentence}")
         sentence = predict(sentence)
         print(f"Output: {sentence}\n")
```

```
Input: I am not crazy, my mother had me tested.
Output: i am sorry . they are just staring .

Input: i am sorry . they are just staring .
Output: i think it is time to be for the post office ? did you bring that in ?

Input: i think it is time to be for the post office ? did you bring that in ?
Output: well , they are not burning up , are you ? you are not a very nice person .

Input: well , they are not burning up , are you ? you are not a very nice person .
Output: yeah , yeah . it is nice . it is really good .

Input: yeah , yeah . it is nice . it is really good .
Output: yeah , well , i am sorry . hi . i am sorry . it is been so good work for you .
```

we'll note the accuracy increase, the loss decrease and when we test the model we note the model say incomprehensible sentences, but it tries to create sentences.

Colab of Tensorflow(SHAKESPEARE)

```
MAX_LENGTH = 40

MAX_SAMPLES = 50000

BATCH_SIZE = 64 * strategy.num_replicas_in_sync

BUFFER_SIZE = 20000

NUM_LAYERS = 2

D_MODEL = 256

NUM_HEADS = 8

UNITS = 512

DROPOUT = 0.1

EPOCHS = 40
```

```
[56] model.fit(dataset, epochs=EPOCHS)

Epoch 1/40
1/1 [==============================] - 16s 16s/step - loss: 10.8479 - accuracy: 0.0000e+00
Epoch 2/40
1/1 [==============================] - 0s 54ms/step - loss: 10.8337 - accuracy: 0.0000e+00
Epoch 3/40
1/1 [==============================] - 0s 56ms/step - loss: 10.8498 - accuracy: 0.0000e+00
Epoch 4/40
1/1 [==============================] - 0s 54ms/step - loss: 10.8514 - accuracy: 0.0000e+00
Epoch 5/40
1/1 [==============================] - 0s 53ms/step - loss: 10.8414 - accuracy: 0.0000e+00
Epoch 6/40
1/1 [==============================] - 0s 49ms/step - loss: 10.8353 - accuracy: 0.0000e+00
Epoch 7/40
1/1 [==============================] - 0s 48ms/step - loss: 10.8426 - accuracy: 0.0000e+00
Epoch 8/40
1/1 [==============================] - 0s 46ms/step - loss: 10.8409 - accuracy: 0.0000e+00
Epoch 9/40
1/1 [==============================] - 0s 48ms/step - loss: 10.8390 - accuracy: 0.0000e+00
Epoch 10/40
1/1 [==============================] - 0s 47ms/step - loss: 10.8344 - accuracy: 0.0000e+00
Epoch 11/40
1/1 [==============================] - 0s 46ms/step - loss: 10.8331 - accuracy: 0.0000e+00
Epoch 12/40
1/1 [==============================] - 0s 44ms/step - loss: 10.8396 - accuracy: 0.0000e+00
Epoch 13/40
1/1 [==============================] - 0s 45ms/step - loss: 10.8386 - accuracy: 0.0000e+00
Epoch 14/40
1/1 [==============================] - 0s 44ms/step - loss: 10.8240 - accuracy: 0.0000e+00
Epoch 24/40
1/1 [==============================] - 0s 46ms/step - loss: 10.7847 - accuracy: 0.0000e+00
Epoch 25/40
1/1 [==============================] - 0s 42ms/step - loss: 10.7780 - accuracy: 0.0000e+00
Epoch 26/40
1/1 [==============================] - 0s 42ms/step - loss: 10.7735 - accuracy: 0.0000e+00
Epoch 27/40
1/1 [==============================] - 0s 42ms/step - loss: 10.7664 - accuracy: 0.0000e+00
Epoch 28/40
1/1 [==============================] - 0s 44ms/step - loss: 10.7609 - accuracy: 0.0000e+00
Epoch 29/40
1/1 [==============================] - 0s 45ms/step - loss: 10.7627 - accuracy: 0.0000e+00
Epoch 30/40
1/1 [==============================] - 0s 43ms/step - loss: 10.7545 - accuracy: 0.0000e+00
Epoch 31/40
1/1 [==============================] - 0s 43ms/step - loss: 10.7599 - accuracy: 0.0000e+00
Epoch 32/40
1/1 [==============================] - 0s 43ms/step - loss: 10.7296 - accuracy: 0.0000e+00
Epoch 33/40
1/1 [==============================] - 0s 42ms/step - loss: 10.7329 - accuracy: 0.0000e+00
Epoch 34/40
1/1 [==============================] - 0s 42ms/step - loss: 10.7358 - accuracy: 0.0000e+00
Epoch 35/40
1/1 [==============================] - 0s 43ms/step - loss: 10.7198 - accuracy: 0.0000e+00
Epoch 36/40
1/1 [==============================] - 0s 46ms/step - loss: 10.7166 - accuracy: 0.0000e+00
Epoch 37/40
1/1 [==============================] - 0s 44ms/step - loss: 10.7119 - accuracy: 0.0000e+00
Epoch 38/40
1/1 [==============================] - 0s 45ms/step - loss: 10.6992 - accuracy: 0.0000e+00
Epoch 39/40
1/1 [==============================] - 0s 53ms/step - loss: 10.6970 - accuracy: 0.0000e+00
Epoch 40/40
1/1 [==============================] - 0s 43ms/step - loss: 10.6877 - accuracy: 0.0000e+00
<keras.callbacks.History at 0x7ff370717f10>
```

```
Let's test our model!

[61] predict("Where have you been?")

' illegal US US US US US US US US US US US US US US US US US!!!235 illegal US US US US US US US US US US US US US US US US US US'

[62] predict("It's a trap")

' illegal Sustainable illegal US Sustainable illegal US US US Sustainable illegal illegal illegal US US US US US Sustainable illegal illegal US Sustainable US US US US US US US US US
US US US US US Sustainable235 US'
```

```
[63] # feed the model with its previous output
     sentence = "I am not crazy, my mother had me tested."
     for _ in range(5):
         print(f"Input: {sentence}")
         sentence = predict(sentence)
         print(f"Output: {sentence}\n")

Input: I am not crazy, my mother had me tested.
Output:  illegal illegal illegal illegal illegal illegal235 illegal illegal illegal illegal illegal illegal illegal illegal US US235 illegal235 illegal illegal illegal US US US235 illegal235 illegal235235235235235235235235235235

Input:  illegal illegal illegal illegal illegal illegal235 illegal illegal illegal illegal illegal illegal illegal illegal US US235 illegal235 illegal illegal illegal US US US235 illegal235 illegal235235235235235235235235235235
Output:  clarleave US clar beceor US clar mushroom curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled235235235235235 illegal235 illegal235235235235

Input:  clarleave US clar beceor US clar mushroom curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled curled235235235235235 illegal235 illegal235235235235
Output:  STATS STATSennis illegal Tw clar dock clar STATS STATS STATS STATS STATS STATS US US US US235 illegal235 illegal US US STATS STATS STATS STATS STATS235 Tw235 Tw235 Tw235 Tw235 correctness235

Input:  STATS STATSennis illegal Tw clar dock clar STATS STATS STATS STATS STATS STATS US US US US235 illegal235 illegal US US STATS STATS STATS STATS STATS235 Tw235 Tw235 Tw235 Tw235 correctness235
Output:  clarleave US clarleave US clarleave US clarleave US US US US US US US235235235235 illegal US US US US US235235235235235235235235235235235235

Input:  clarleave US clarleave US clarleave US clarleave US US US US US US US235235235235 illegal US US US US US235235235235235235235235235235235235
Output:  clar US clar US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US US
```

we'll note the accuracy doesn't change, but the loss decrease and then test the model we note the model repeat the vocab many times and say weird sentence

The Second experiment we Changed UNITS from 512 to 150, DROPOUT from 0.1 to 0.2 and Epochs from 40 to 250

Colab of Tensorflow(MOVIES)

```
[124] MAX_LENGTH = 40

      MAX_SAMPLES = 50000

      BATCH_SIZE = 64 * strategy.num_replicas_in_sync

      BUFFER_SIZE = 20000

      NUM_LAYERS = 2

      D_MODEL = 256

      NUM_HEADS = 8

      UNITS = 150

      DROPOUT = 0.2

      EPOCHS = 250
```

```
model.fit(dataset, epochs=EPOCHS)

Epoch 1/250
690/690 [==============================] - 49s 59ms/step - loss: 2.1310 - accuracy: 0.0382
Epoch 2/250
690/690 [==============================] - 42s 61ms/step - loss: 1.5252 - accuracy: 0.0745
Epoch 3/250
690/690 [==============================] - 41s 60ms/step - loss: 1.4166 - accuracy: 0.0822
Epoch 4/250
690/690 [==============================] - 43s 62ms/step - loss: 1.3576 - accuracy: 0.0867
Epoch 5/250
690/690 [==============================] - 43s 63ms/step - loss: 1.3120 - accuracy: 0.0905
Epoch 6/250
690/690 [==============================] - 43s 63ms/step - loss: 1.2695 - accuracy: 0.0936
Epoch 7/250
690/690 [==============================] - 42s 61ms/step - loss: 1.2235 - accuracy: 0.0969
Epoch 8/250
690/690 [==============================] - 42s 61ms/step - loss: 1.1761 - accuracy: 0.1004
Epoch 9/250
690/690 [==============================] - 42s 61ms/step - loss: 1.1324 - accuracy: 0.1040
Epoch 10/250
690/690 [==============================] - 42s 60ms/step - loss: 1.0949 - accuracy: 0.1075
Epoch 11/250
690/690 [==============================] - 42s 60ms/step - loss: 1.0605 - accuracy: 0.1107
Epoch 12/250
690/690 [==============================] - 42s 60ms/step - loss: 1.0287 - accuracy: 0.1138
Epoch 13/250
690/690 [==============================] - 41s 60ms/step - loss: 0.9997 - accuracy: 0.1170
Epoch 14/250
690/690 [==============================] - 42s 61ms/step - loss: 0.9732 - accuracy: 0.1201
Epoch 15/250
690/690 [==============================] - 41s 60ms/step - loss: 0.9479 - accuracy: 0.1228
Epoch 16/250
```

Colab of Tensorflow(SHAKESPEARE)

```
[124] MAX_LENGTH = 40
      MAX_SAMPLES = 50000
      BATCH_SIZE = 64 * strategy.num_replicas_in_sync
      BUFFER_SIZE = 20000
      NUM_LAYERS = 2
      D_MODEL = 256
      NUM_HEADS = 8
      UNITS = 150
      DROPOUT = 0.2
      EPOCHS = 250
```

```
model.fit(dataset, epochs=EPOCHS)

Epoch 1/250
1/1 [==============================] - 14s 14s/step - loss: 10.8271 - accuracy: 0.0000e+00
Epoch 2/250
1/1 [==============================] - 0s 131ms/step - loss: 10.8273 - accuracy: 0.0000e+00
Epoch 3/250
1/1 [==============================] - 0s 77ms/step - loss: 10.8307 - accuracy: 0.0000e+00
Epoch 4/250
1/1 [==============================] - 0s 103ms/step - loss: 10.8490 - accuracy: 0.0000e+00
Epoch 5/250
1/1 [==============================] - 0s 123ms/step - loss: 10.8216 - accuracy: 0.0000e+00
Epoch 6/250
1/1 [==============================] - 0s 88ms/step - loss: 10.8412 - accuracy: 0.0000e+00
Epoch 7/250
1/1 [==============================] - 0s 62ms/step - loss: 10.8231 - accuracy: 0.0000e+00
Epoch 8/250
1/1 [==============================] - 0s 89ms/step - loss: 10.8323 - accuracy: 0.0000e+00
Epoch 9/250
1/1 [==============================] - 0s 70ms/step - loss: 10.8296 - accuracy: 0.0000e+00
Epoch 10/250
1/1 [==============================] - 0s 63ms/step - loss: 10.8215 - accuracy: 0.0000e+00
Epoch 11/250
1/1 [==============================] - 0s 79ms/step - loss: 10.8322 - accuracy: 0.0000e+00
Epoch 12/250
1/1 [==============================] - 0s 118ms/step - loss: 10.8233 - accuracy: 0.0000e+00
Epoch 13/250
1/1 [==============================] - 0s 71ms/step - loss: 10.8325 - accuracy: 0.0000e+00
Epoch 14/250
1/1 [==============================] - 0s 63ms/step - loss: 10.8001 - accuracy: 0.0000e+00
Epoch 15/250
1/1 [==============================] - 0s 92ms/step - loss: 10.8103 - accuracy: 0.0000e+00
Epoch 16/250
1/1 [==============================] - 0s 78ms/step - loss: 10.8206 - accuracy: 0.0000e+00
Epoch 17/250
```

```
Epoch 234/250
1/1 [==============================] - 0s 67ms/step - loss: 7.4731 - accuracy: 0.1795
Epoch 235/250
1/1 [==============================] - 0s 60ms/step - loss: 7.4442 - accuracy: 0.1795
Epoch 236/250
1/1 [==============================] - 0s 54ms/step - loss: 7.3984 - accuracy: 0.1795
Epoch 237/250
1/1 [==============================] - 0s 51ms/step - loss: 7.3771 - accuracy: 0.1795
Epoch 238/250
1/1 [==============================] - 0s 61ms/step - loss: 7.3435 - accuracy: 0.1795
Epoch 239/250
1/1 [==============================] - 0s 58ms/step - loss: 7.3172 - accuracy: 0.1795
Epoch 240/250
1/1 [==============================] - 0s 54ms/step - loss: 7.2889 - accuracy: 0.1795
Epoch 241/250
1/1 [==============================] - 0s 53ms/step - loss: 7.2560 - accuracy: 0.1795
Epoch 242/250
1/1 [==============================] - 0s 49ms/step - loss: 7.2199 - accuracy: 0.1795
Epoch 243/250
1/1 [==============================] - 0s 50ms/step - loss: 7.1924 - accuracy: 0.1795
Epoch 244/250
1/1 [==============================] - 0s 52ms/step - loss: 7.1574 - accuracy: 0.1795
Epoch 245/250
1/1 [==============================] - 0s 59ms/step - loss: 7.1252 - accuracy: 0.1795
Epoch 246/250
1/1 [==============================] - 0s 61ms/step - loss: 7.1082 - accuracy: 0.1795
Epoch 247/250
1/1 [==============================] - 0s 57ms/step - loss: 7.0718 - accuracy: 0.1795
Epoch 248/250
1/1 [==============================] - 0s 58ms/step - loss: 7.0317 - accuracy: 0.1795
Epoch 249/250
1/1 [==============================] - 0s 55ms/step - loss: 7.0143 - accuracy: 0.1795
Epoch 250/250
1/1 [==============================] - 0s 54ms/step - loss: 6.9781 - accuracy: 0.1795
<keras.callbacks.History at 0x7ff29d746170>
```

Let's test our model!

```
[161] predict("Where have you been?")

'\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n'
```

```
[162] predict("It's a trap")

'\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n'
```

```python
# feed the model with its previous output
sentence = "I am not crazy, my mother had me tested."
for _ in range(5):
    print(f"Input: {sentence}")
    sentence = predict(sentence)
    print(f"Output: {sentence}\n")
```

```
Input: I am not crazy, my mother had me tested.
Output:
```

we'll note the accuracy first doesn't change but after many epochs the accuracy change , but the loss decrease and then test the model we note the model repeat the same vocabs many times and can't get output to the input