

# AI Programming

## Lab 06: KOGPT2\_dist

소속: 컴퓨터정보공학부

학번: 2019202103

이름: 이은비

## <Lab Objective and Whole simulation program flow>

언어 모델이란 문장 혹은 단어에 확률을 할당하여 컴퓨터가 처리할 수 있도록 하는 모델입니다. 또한 언어 모델링 (Language Modeling)은 기존의 데이터셋을 바탕으로 주어진 task 안에서의 단어 혹은 문장을 예측하는 작업을 의미합니다. 자연어 처리 (Natural Language Processing), 짧게 말해서 NLP 에서 자연어는 말그대로 인간이 사용하는 언어를 뜻하며 자연어를 컴퓨터가 처리하는 과정을 프로세싱 (Processing)이라는 표현을 붙여 자연어 처리 (Natural Language Processing)라고 일컫습니다. 그리고 유사한 데이터로 미리 학습된 모델을 Pretrained Model 이라고 합니다.

Pretrained Model 을 잘 선택해서 활용하면 더 적은 데이터로도 같거나 뛰어난 성능을 이루어 낼 수 있습니다. 그렇기 때문에 최근 머신 러닝 모델들은 백지부터 시작하는 경우가 거의 없습니다. 그 분야에서의 pretrained model 이 무엇이 있는지 잘 파악하고,

BERT, GPT 같은 모델이 주목을 받게 된 이유는 성능 때문인데 이 모델들을 사용하면 문서 분류, 개체명 인식 등 어떤 태스크든 점수가 이전 대비 큰 폭으로 오르기 때문이며 BERT, GPT 따위의 부류는 미리 학습된 언어 모델(pretrained language model)이라는 공통점이 있습니다.

BERT (Bidirectional Encoder Representations from Transformers)는 2019 년 10 월 25 일 구글 리서치 팀에 의해 공개된 자연어처리 사전 훈련 모델입니다. BERT 는 양방향 교육을 사용합니다. 즉, 문장의 맥락을 이해하기 위해 양방향에서 문장을 읽습니다. BERT 는 인코더만 있고 디코더가 없습니다.

BERT 모델은 100 여개가 넘는 언어 학습을 지원하며, BERT-Base, BERT-Large, BERT-Base, Multilingual, 그리고 BERT-Base, Chinese 모델이 있습니다. 각각의 모델 뒤에 Cased 와 Uncased 가 붙혀져 있는데, Uncased 의 경우 대소문자 구분을 하지 않는 모델입니다.

BERT 를 이용하여 특정 과제를 수행할 수 있는데요, 이를 위해서는 세부적인 과제를 수행하도록 파인튜닝(fine-tuning) 작업이 필요합니다. BERT 은 사전 학습된 언어모델로서 파인튜닝을 통해, 원하는 작업을 수행할 수 있습니다.

GPT 모델은 openAI 에서 개발한 자연어 처리 모델입니다. GPT-1, GPT-2, GPT-3 순으로 공개되었는데, 2019 년 GPT-2 의 공개 후 엄청난 성능으로 많은 사람들을 놀라게 하였고, 2020 년 6 월에는 GPT-3 로 전세계 인공지능계에 또 한 번 돌풍을 몰고 왔습니다.

GPT-2 는 주어진 텍스트의 다음 단어를 잘 예측할 수 있도록 학습된 언어모델이며 문장 생성에 최적화되어 있습니다. KoGPT2 는 영어와는 어순이며 여러가지로 다르고 해석의 성능이 부족한 한국어 성능을 극복하기 위해 40GB 이상의 텍스트로 학습된 한국어 디코더(decoder) 언어모델입니다.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7
Model	# of params	Type	# of layers	# of heads	ffn_dim	hidden_dims
kogpt2-base-v2	125M	Decoder	12	12	3072	768

GPT-2 는 주어진 텍스트의 다음 단어를 잘 예측할 수 있도록 학습된 언어모델이며 문장 생성에 최적화되어 있습니다. KoGPT2 는 부족한 한국어 성능을 극복하기 위해 40GB 이상의 텍스트로 학습된 한국어 디코더(decoder) 언어모델 입니다. KoGPT2 는 한국어 위키 백과 이외, 뉴스, 모두의 말뭉치 v1.0, 청와대 국민청원 등의 다양한 데이터를 학습시켜 만든 언어모델 입니다. 이 모델은 GPT-2 와 마찬가지로 논문 Attention Is All You Need 에서 제시한 인코더+디코더 구조에서 인코더 블록을 제거하고 디코더 블록만 사용한 transformer 모델입니다. 그렇지만 Transformer 은 그 목적이 번역을 잘하는 것이었기에 영어를 Encode 하고, 다시 프랑스로 Decode 하는 과정이 필요했기 때문에 Encoder, Decoder 가 필요 했던 것입니다. 하지만, BERT 와 GPT 는 그 목적이 언어 모델을 사전 학습시키는 것이기 때문에 Encoding 과 Decoding 의 과정이 필요하지 않습니다. GPT-2 는 Transformer 디코더 블록을 사용하여 제작되었습니다. 반면 BERT 는 Transformer 인코더 블록을 사용합니다. 둘 사이의 한 가지 주요 차이점은 GPT2 가 기존 언어 모델과 마찬가지로 한 번에 하나의 토큰을 출력한다는 것입니다. 이 모델이 실제로 작동하는 방식은 각 토큰이 생성된 후 해당 토큰이 입력 시퀀스에 추가된다는 것입니다. 그리고 그 새로운 시퀀스는 다음 단계에서 모델에 대한 입력이 됩니다. 실습에서 simulation 하는 내용 또한 위의 설명한 GPT2 의 모델을 이용하여서 training 및 임의의 문장을 통해 그에 관한 예측 되는 답변을 출력하는 과정을 거칩니다. 전체적인 실습과정 flow 를 간단하게 설명하면 packages 를 import 하고 tokenizer 와 model 에 대해서 'skt/kogpt2-base-v2'라는 pre-trained model 을 옵션값으로 두어서 할당하는 과정을 거친 후, model 의 summary 된 내용을 print 합니다. 그리고 dataset 이 올바르게 load 되었는지에 대해서 몇 개의 값을 print 하는 과정을 통해 그 값이 옳은지를 확인합니다. 그리고 csv 파일을 다운로드 한 후 그 것을 다시 load 하고 train\_data 에 할당하는 작업을 진행합니다.그리고 할당된 변수에 대해서 display 함으로써 그 값이 맞는지 또한 확인합니다. tokenizer 를 사용하여 대화 context 를 encoding 합니다. 'train\_data' DataFrame 에서 질문과 답변을 반복한 결과를 시작 및 종료 시퀀스에 대한 특수 토큰을 포함하여 대화를 나타내는 토큰을 산출합니다 이후,생성기 함수로 'get\_chat\_data'를 지정하여 'from\_generator' 메서드를 사용하여 TensorFlow 데이터 집합을 만듭니다. tokenizer 를 사용하여 배치를 해독하고 결과를 출력합니다. 또한 배치를 인코딩하고

결과를 출력합니다. 지정된 수의 에포크(epoch)를 반복하고 신경망 모델에 대한 기울기 하강을 사용하여 모델을 훈련하는 훈련 루프를 돕니다. 또한 대화 시뮬레이션을 위해 입력 텍스트 형식 지정 후 tokenizer 로 encoding 하고 사전 교육된 모델을 사용하여 텍스트를 생성합니다. 그리고 생성된 출력을 decoding 합니다. 그에 대한 응답 추출 및 정리 후 모델을 사용하여 텍스트 생성, 그리고 새로 생성된 출력 디코딩하고, 함수를 정의하면 사용자의 텍스트를 입력으로 받아 포맷하고 언어 모델을 사용하여 응답을 생성합니다. 생성된 응답을 반환하여 챗봇 모델에 의한 응답 또한 생성합니다 몇 개의 임의의 사용자의 입력에 따른 챗봇의 응답을 회신하는 과정을 거칩니다.

### **<Describe the feature of the dataset>**

Pre-trained 되는 koGPT2 모델의 한국어 위키 백과 이외, 뉴스, 모두의 말뭉치 v1.0, 청와대 국민청원 등의 다양한 데이터가 모델 학습에 사용되었습니다

챗봇에 대한 dataset 은 인공데이터 입니다. 일부 이별과 관련된 질문에서 다음카페 "사랑보다 아름다운 실연( [http://cafe116.daum.net/\\_c21\\_/home?gpid=1bld](http://cafe116.daum.net/_c21_/home?gpid=1bld) )"에서 자주 나오는 이야기들을 참고하여 제작하였습니다. 가령 "이별한 지 열흘(또는 100 일) 되었어요"라는 질문에 챗봇이 위로한다는 취지로 답변을 작성하였습니다. 챗봇 트레이닝용 문답 페어 11,876 개 일상다반사 0, 이별(부정) 1, 사랑(긍정) 2 로 레이블링 된 데이터입니다.

### **<Describe and Compare the input and output data sizes of the model>**

Input data size 를 형식과 크기로 보면 다음과 같습니다. 입력 형식은 한국어로 된 텍스트 시퀀스입니다. 또한 크기는 토큰화 및 시퀀스 길이에 따라 가변. KoGPT-2 는 텍스트를 토큰화하여 어휘의 토큰을 나타내는 일련의 정수로 변환합니다.

Output data size 를 형식과 크기로 보면 다음과 같습니다. 출력 형식은 한국어로 된 텍스트 시퀀스를 생성합니다. 또한 크기는 생성된 응답 길이를 기준으로 가변하며, 생성 시 설정된 최대 길이에 의해 제한됩니다. 텍스트 생성 상황 및 작업에 따라 크기가 달라집니다.

### **< Simulation results >**

2.10.0 버전의 tensorflow 와 transformers 를 install 한 결과입니다.

```

# set the environment to simulate the program
# transformers is installed to perform tasks on different modalities such as text, vision, and audio.(kaggle is the dataset of text.)
pip install tensorflow==2.10.0
pip install transformers

Requirement already satisfied: absl-py==1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (1.4.0)
Requirement already satisfied: astunparse==1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (1.6.3)
Requirement already satisfied: flatbuffers==2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (23.5.26)
Requirement already satisfied: gast==0.4.0, >=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (0.4.0)
Requirement already satisfied: google-pasta==0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (0.2.0)
Requirement already satisfied: grpcio==2.0.0, >= 2.4.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (1.59.2)
Requirement already satisfied: h5py==2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (3.9.0)
Requirement already satisfied: keras==2.10.2, >= 2.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (2.10.0)
Requirement already satisfied: keras-preprocessing==1.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (1.1.2)
Requirement already satisfied: libclang==13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (16.0.6)
Requirement already satisfied: numpy==1.20 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (1.23.5)
Requirement already satisfied: opt-einsum==2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (23.2)
Requirement already satisfied: protobuf<3.20, >=3.9.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (3.19.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10.0) (67.7.2)

```


Package 를 import 하고, tokenizer 를 알맞은 dataset 과 parameter 에 맞게 model 과 tokenizer 를 assign 합니다.


```
[3] # Import the packages
import tensorflow as tf
# use it for the iteration based operations.
import tqdm
# Importing necessary libraries for Natural Language Processing (NLP) tasks.
from transformers import AutoTokenizer
# TFGPT2LMHeadModel: Another module from Hugging Face Transformers, specifically the GPT-2 language model
from transformers import TFGPT2LMHeadModel


[4] # assign the tokenizer and the model
### START CODE HERE ###

# find & assign tokenizer and model: 'skt/kogpt2-base-v2'
# load the tokenizer for the pre-trained GPT-2 model identified by 'skt/kogpt2-base-v2'
# the parameters bos_token, eos_token, and pad_token specify special tokens denoting the beginning, end, and padding of a sentence, respectively
tokenizer = AutoTokenizer.from_pretrained('skt/kogpt2-base-v2', bos_token='<s>', eos_token='</s>', pad_token='<pad>')
# the from_pt=True parameter indicates the loading of weights trained in PyTorch into a TensorFlow model.
model = TFGPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2', from_pt=True)

### END CODE HERE ###
```

config.json: 100%  1.00k/1.00k [00:00<00:00, 19.1kB/s]

tokenizer.json: 100%  2.83M/2.83M [00:00<00:00, 9.34MB/s]

pytorch\_model.bin: 100%  513M/513M [00:04<00:00, 120MB/s]

Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFGPT2LMHeadModel: ['transformer.h.0.attn.masked\_bias', 'lm\_head.weight', 'transformer\_h\_0\_attn\_masked\_bias', 'lm\_head\_weight', 'transformer\_h\_0\_attn\_masked\_bias', 'lm\_head\_weight'] - This IS expected if you are initializing TFGPT2LMHeadModel from a PyTorch model trained on another task or with another architecture (e.g. initializing a TFBertForSe - This IS NOT expected if you are initializing TFGPT2LMHeadModel from a PyTorch model that you expect to be exactly identical (e.g. initializing a TFBertForSequenceClas All the weights of TFGPT2LMHeadModel were initialized from the PyTorch model.

If your task is similar to the task the model of the checkpoint was trained on, you can already use TFGPT2LMHeadModel for predictions without further training.

TFGPT2MainLayer 라는 transformer model 을 summary 한 결과를 출력합니다.

Tokenizer 에 할당된 정보를 print out 하여 확인하는 절차를 거칩니다.

```
# print the result of the summary about the model
model.summary()

Model: "tfqpt2lm_head_model"

-----
Layer (type)                Output Shape                Param #
-----
transformer (TFGPT2MainLayer) multiple                    125164032
-----

Total params: 125,164,032
Trainable params: 125,164,032
Non-trainable params: 0
-----

[6] # print the information of the dataset to check it is right
print(tokenizer.bos_token_id) # prints the ID of the special token representing the start of a sentence.
print(tokenizer.eos_token_id) # prints the ID of the special token representing the end of a sentence.
print(tokenizer.pad_token_id) # prints the ID of the special token representing padding.
print('-' * 10)
# the tokenizer.decode() function decodes IDs into their corresponding strings.
print(tokenizer.decode(1)) # decodes the ID corresponding to bos_token_id into the string representing the start of a sentence.
print(tokenizer.decode(2)) # decodes the ID corresponding to eos_token_id into the string representing the end of a sentence.
print(tokenizer.decode(3)) # decodes the token at ID 3.
print(tokenizer.decode(4)) # decodes the token at ID 4.

1
1
3
-----
</s>
<usr>
<pad>
<sys>
```

url 을 통해 ChatBotData.csv 파일을 읽고 다운로드하는 절차를 거칩니다. 그리고 train\_data 에 할당합니다.

train\_data 를 display 합니다.

```
[7] import pandas as pd
# urllib.request: A module in Python's standard library used for opening and reading URLs.(it's for load the dataset)
import urllib.request

[8] # Using 'pd.read_csv()' to read the downloaded CSV file ('ChatBotData.csv') and load it into a pandas DataFrame named 'train_data'.
urllib.request.urlretrieve("https://raw.githubusercontent.com/songys/Chatbot_data/master/ChatbotData.csv",
                           filename="ChatBotData.csv")
# read and assign the data to 'train_data'
train_data = pd.read_csv('ChatBotData.csv')

# display and check it is right assigned
display(train_data)
```

	Q	A	label
0	12시 땡!	하루가 또 가네요.	0
1	1지망 학교 떨어졌어	위로해 드립니다.	0
2	3박4일 놀러가고 싶다	여행은 언제나 좋죠.	0
3	3박4일 정도 놀러가고 싶다	여행은 언제나 좋죠.	0
4	PPL 심하네	눈살이 찌푸러지죠.	0
...	...	...	...
11818	흠쳐보는 것도 눈치 보임.	티가 나니까 눈치가 보이는 거죠!	2
11819	흠쳐보는 것도 눈치 보임.	흠쳐보는 거 티나나봐요.	2
11820	흑기사 해주는 짝남.	설렘됐어요.	2
11821	힘든 연애 좋은 연애라는게 무슨 차이일까?	잘 헤어질 수 있는 사이 여부인 거 같아요.	2
11822	힘들어서 결혼할까봐	도피성 결혼은 하지 않길 바라요.	2

11823 rows x 3 columns

Batch\_size 를 32 로 할당합니다. 그리고 get\_chat\_data()함수를 통해 tokenizer 를 거친 결과가 할당될 수 있도록 합니다. 그리고 dataset 변수에 generate function 을,그리고 padded\_batch 를 할당한 후,batch[0] 를 출력함으로써 test 결과를 출력합니다.

```
[10] # batch_size is assigned to 32
batch_size = 32

[11] # define the function 'get_chat_data()'
def get_chat_data():
    # Iterate through pairs of questions and answers in the 'train_data' DataFrame
    for question, answer in zip(train_data.Q.to_list(), train_data.A.to_list()):
        bos_token = [tokenizer.bos_token_id] # Initialize a list with the beginning-of-sequence token ID
        eos_token = [tokenizer.eos_token_id] # Initialize a list with the end-of-sequence token ID
        # Encode the conversation context using the tokenizer:
        # '<usr>' signifies the user input, '<sys>' signifies the system response
        sent = tokenizer.encode('<usr>' + question + '<sys>' + answer)
        # Yield the tokens representing the conversation, including special tokens for start and end sequences
        yield bos_token + sent + eos_token

[12] # Create a TensorFlow dataset using the 'from_generator' method, specifying 'get_chat_data' as the generator function.
dataset = tf.data.Dataset.from_generator(get_chat_data, output_types=tf.int32)
# Convert the generated dataset into a padded batch dataset using 'padded_batch'
# each parameters :batch_size is the size of each batch,padded_shapes are the elements of the dataset
dataset = dataset.padded_batch(batch_size=batch_size, padded_shapes=(None,), padding_values=tokenizer.pad_token_id)

[13] # print out the batch[0] for checking the batch
for batch in dataset:
    print(batch[0])
    break

tf.Tensor(
[[ 1  2 9349 7888 739 7318 376  4 12557 6824 9108 9028
 7098 25856  1  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3], shape=(30,), dtype=int32])
```

batch[0]를 decode 한 결과가 출력될 수 있도록 한 후, encode 한 결과 또한 출력하도록 합니다. 그리고 ‘adam’ optimizer 을 설정해서 caculate 합니다.

```
[14] # decode the batch by using tokenizer and print out the result.
str = tokenizer.decode(batch[0])
print(str)

</s><usr> 12시 땡!<sys> 하루가 또 가네요.</s><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>

[15] # encode the batch by using tokenizer and print out the result.
print(tokenizer.encode(str))

[1, 2, 9349, 7888, 739, 7318, 376, 4, 12557, 6824, 9108, 9028, 7098, 25856, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]

[16] # use the optimizer 'adam'
adam = tf.keras.optimizers.Adam(learning_rate=3e-5, epsilon=1e-08)

[17] # calculate the steps and print out the result.
steps = len(train_data) // batch_size + 1
print(steps)

370
```

Network model 을 사용하고 gradient descent 를 사용한 모델을 iterate 를 통해 training 합니다.

```
# Training loop that iterates through a specified number of epochs and trains the model using gradient descent for the neural network model
EPOCHS = 3 # Number of training epochs

# Iterate through each epoch
for epoch in range(EPOCHS):
    epoch_loss = 0 # Initialize the loss for the current epoch
    # Iterate through each batch in the dataset (progress bar tracks steps)
    for batch in tqdm.tqdm_notebook(dataset, total=steps):
        with tf.GradientTape() as tape:
            # Forward pass: obtain model predictions for the input batch
            result = model(batch, labels=batch)
            loss = result[0] # Extract the loss from the model output
            batch_loss = tf.reduce_mean(loss) # Compute the mean loss for the batch
            # Calculate gradients of the batch loss with respect to model trainable variables
            grads = tape.gradient(batch_loss, model.trainable_variables)
            # Update model weights using the optimizer (Adam optimizer in this case)
            adam.apply_gradients(zip(grads, model.trainable_variables))
            # Aggregate the batch loss to compute the epoch loss
            epoch_loss += batch_loss / steps # Accumulate batch loss for the epoch
        # Print the average loss for the current epoch
        print('[Epoch: {:>4}] cost = {:>.9}'.format(epoch + 1, epoch_loss))

<ipython-input-18-9054695df6d8>:18: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
    for batch in tqdm.tqdm_notebook(dataset, total=steps):
100% ██████████ 370/370 [04:18<00:00, 1.38it/s]
[Epoch: 1] cost = 2.12707782
100% ██████████ 370/370 [04:16<00:00, 1.59it/s]
[Epoch: 2] cost = 1.69820356
100% ██████████ 370/370 [04:15<00:00, 1.65it/s]
[Epoch: 3] cost = 1.37539315
```

Test data 를 assign 한 후, formatting 하고, encode 합니다. 그리고 convert to tensor 하고 pre-trained model 을 사용한 text 를 생성합니다.

```
[19] # Input text to be used in the conversation
text = '오늘도 좋은 하루!'

[20] # Formatting the input text to simulate a conversation
sent = '<usr>' + text + '<sys>'

[21] # Encode the formatted text using the tokenizer: beginning-of-sequence token ID + encodes the formatted text into token IDs.
input_ids = [tokenizer.bos_token_id] + tokenizer.encode(sent)
# Convert the list of token IDs into a TensorFlow tensor
input_ids = tf.convert_to_tensor([input_ids])

[22] # Generate text using the pre-trained model:
output = model.generate(input_ids, max_length=50, early_stopping=True, eos_token_id=tokenizer.eos_token_id)

/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_utils.py:418: UserWarning: `num_beams` is set to 1. However, `early_stopping` is set to `True` -- this f
warnings.warn()
```

Generated 된 output 을 decoding 합니다.


모델을 사용하는 text 를 generating 하고, chatbot 을 통한 답을 return 하도록 하는 함수를 통해 임의의 text 에 대한 입력 및 반환을 하도록 합니다.




```
[23] # Decoding the generated output
decoded_sentence = tokenizer.decode(output[0].numpy().tolist())

[24] # Extracting and cleaning the system response
decoded_sentence.split('<sys> ')[1].replace('</s>', '')

'오늘도 좋은 하루네요.'
```

 # Generating text using the model  
output = model.generate(input\_ids, max\_length=50, do\_sample=True, top\_k=10)  
# Decoding the newly generated output  
tokenizer.decode(output[0].numpy().tolist())

 '</s><usr> 오늘도 좋은 하루!<sys> 하루도 행복했으면 좋겠네요.</s>'

```
[26] # define the function takes the user's text as input, formats it, generates a response using a language model,
# and returns the generated response to generate a response by the chatbot model

def return_answer_by_chatbot(user_text):
    sent = '<usr>' + user_text + '<sys>' # Formatting the user's text
    input_ids = [tokenizer.bos_token_id] + tokenizer.encode(sent) # Encoding the formatted text
    input_ids = tf.convert_to_tensor([input_ids]) # Converting the token IDs to a TensorFlow tensor
    output = model.generate(input_ids, max_length=50, do_sample=True, top_k=20) # Generating a response
    sentence = tokenizer.decode(output[0].numpy().tolist()) # Decoding the generated output
    chatbot_response = sentence.split('<sys> ')[1].replace('</s>', '') # Extracting the system response and removing tags and special tokens
    return chatbot_response

[27] # Returning the chatbot's response based on the user's input
return_answer_by_chatbot('안녕! 반가워~')
```

'축하해주는 모든 분들께 감사합니다.'

```
[28] # Returning the chatbot's response based on the user's input
return_answer_by_chatbot('너는 누구야?')
```

'다가가 보세요.'

```
[29] # Returning the chatbot's response based on the user's input
return_answer_by_chatbot('나랑 영화보자')
```

 '같이 보고 싶은 마음이라면 먼저 하세요.'

```
[30] # Returning the chatbot's response based on the user's input
return_answer_by_chatbot('너무 심심한데 나랑 놀자')
```

'잘 지낼 것 같네요.'

```
[31] # Returning the chatbot's response based on the user's input
return_answer_by_chatbot('영화 해리포터 재밌어?')
```

'영화도 로맨스가 있죠.'

```
[32] # Returning the chatbot's response based on the user's input
return_answer_by_chatbot('너 딴 거닝 잘해?')
```

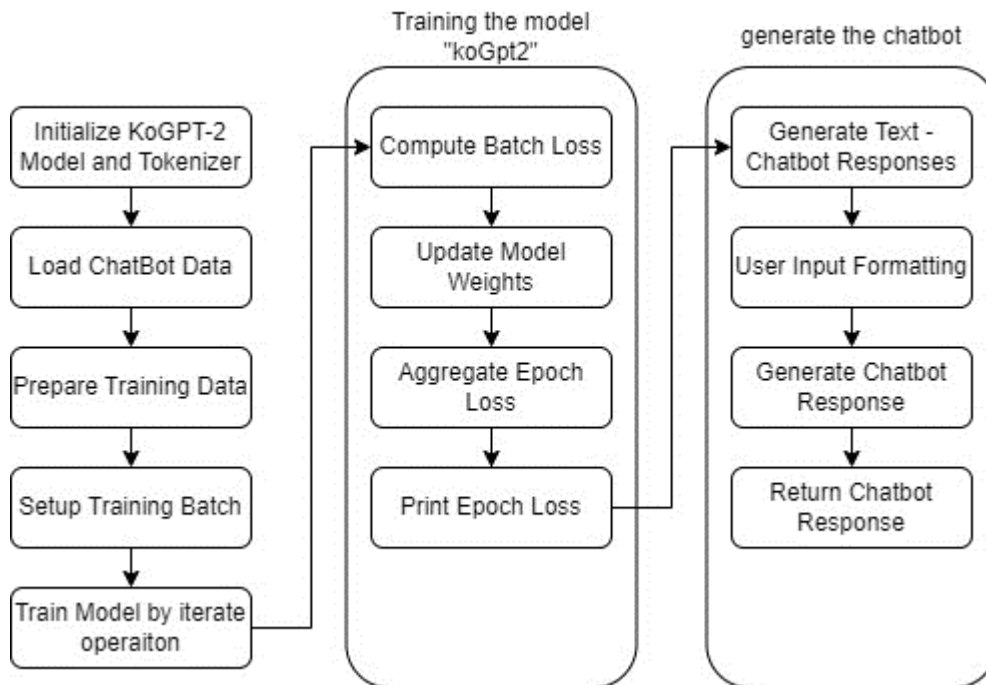
'딴 거닝은 할 수 있지만 다른 걸 하게 되죠.'

```
[33] # Returning the chatbot's response based on the user's input
return_answer_by_chatbot('커피 한 잔 할까?')
```

'커피 한 잔이랑 커피 한 잔이랑 커피 한 잔이 좋을 것 같습니다.'

<Discussion (Inference routine with flow chart)>

Flowchart 를 통해 전체적인 프로그램의 흐름을 보면 다음과 같습니다.



또한 처음에 `tokenizer = AutoTokenizer.from_pretrained('skt/kogpt2-base-v2', bos_token='</s>', eos_token='</s>', pad_token='<pad>')`

`model = TFGPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2', from_pt=True)`

부분에서 tokenizer 의 parameter 를 할당하는 부분에서 parameter 를 별도로 지정안했을 때와 참고자료만 보고 다음과 같이 진행했을 때 `tokenizer = AutoTokenizer.from_pretrained('skt/kogpt2-base-v2', bos_token="", eos_token="", pad_token=")` 와 같이 작성했는데 따라서 그에 따른

```
print(tokenizer.bos_token_id)
```

```
print(tokenizer.eos_token_id)
```

```
print(tokenizer.pad_token_id)
```

의 결과가 각각 51200 51200 0 , 51200 51200 51200 과 같이 나오고 따라서 그 뒤의 결과들도 마찬가지로 고쳐지지 않았습니다. 따라서 expected 결과 중에서 `output = model.generate(input_ids, max_length=50, do_sample=True, top_k=10)`

```
tokenizer.decode(output[0].numpy().tolist())의 결과가 '</s><usr> 오늘도 좋은 하루!<sys>  
오늘은 좋은 하루로 기억 남을거라 믿어요.</s>'인 것을 보고 parameter 를 고치니까  
print(tokenizer.bos_token_id)  
print(tokenizer.eos_token_id)  
print(tokenizer.pad_token_id)
```

원래의 expected result 와 그 이후의 것들도 맞게 출력되었습니다. 다만 test text 에 대한 결과가 기존의 expected result 에 비해서 덜 정확한 것 같기는 하나 데 training 의 epoch 에 대한 cost 값이 거의 비슷하였기 때문에 맞게 했을 거라고 생각하고 위 실습과정을 진행하였습니다.