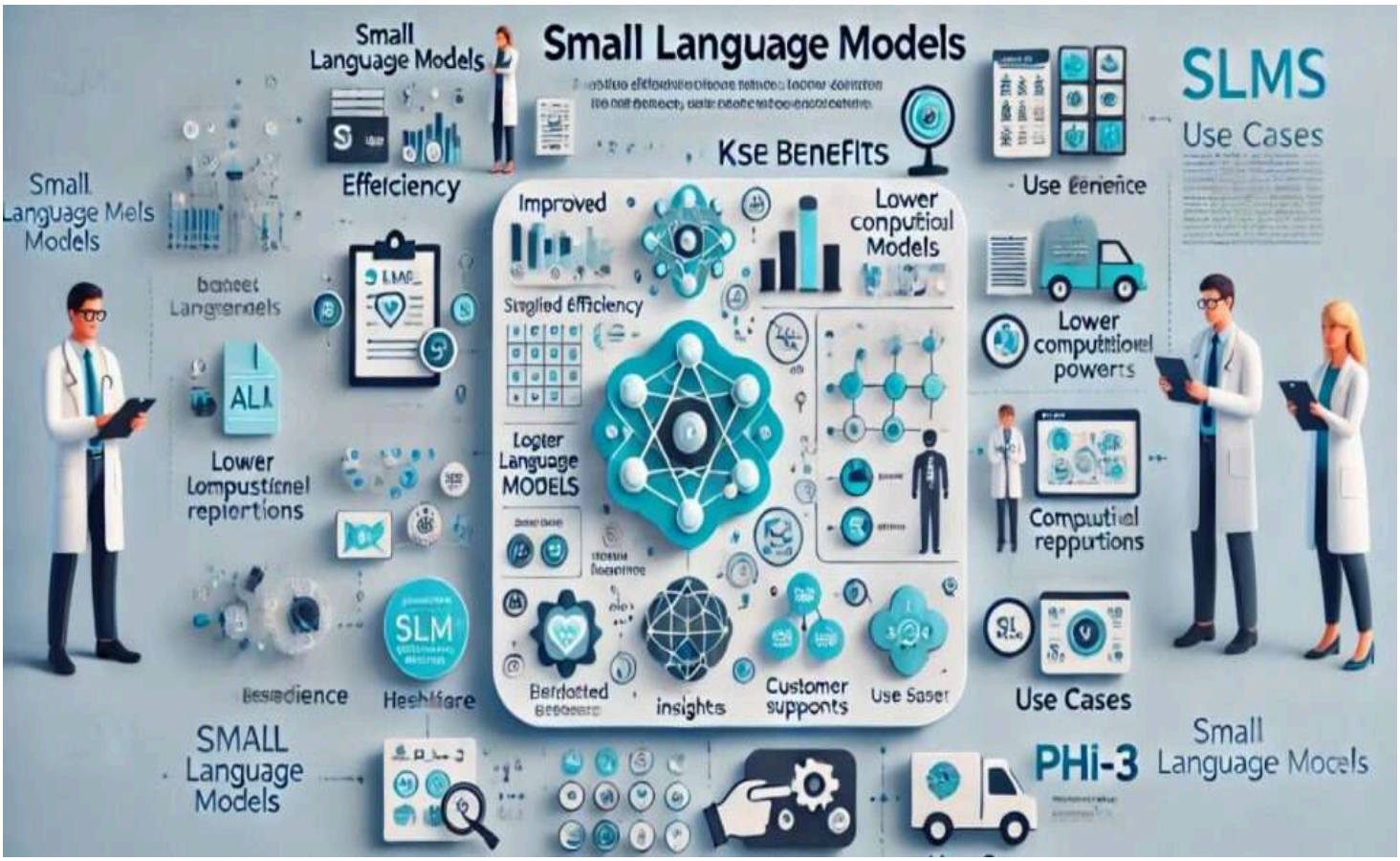
Time To Learn SLMs

Vision College of Jeonju

By 진코나잉 Sep 24, 2025



SLM အခြေခံနှင့် လက်တွေ့ Project တည်ဆောက်ခြင်း

ဒီလမ်းညွှန်က SLM ရဲ့ အခြေခံသဘောတရားတွေကို နားလည်ပြီး လက်တွေ့ project တစ်ခုကို ကိုယ်တိုင်တည်ဆောက်နိုင်တဲ့အဆင့်ထိ ရောက်ရှိဖို့ ရည်ရွယ်ပါတယ်။

၁။ သင်ယူရမည့် ရည်ရွယ်ချက်များ (Learning Objectives)

ဒီသင်ခန်းစာပြီးသွားရင် သင်ဘာတွေ တတ်မြောက်သွားမလဲ။ 🤔

- **SLM ဆိုတာဘာလဲ** တိတိကျကျနားလည်သွားမယ်။ LLM (Large Language Model) နဲ့ ဘာကွာလဲဆိုတာကို ရှင်းပြနိုင်မယ်။
- SLM project တွေအတွက် လိုအပ်တဲ့ **Python ပတ်ဝန်းကျင် (environment) ကို ပြင်ဆင်တတ်သွားမယ်**။
- Hugging Face ကနေ pre-trained SLM တစ်ခုကိုယူပြီး **ရိုးရှင်းတဲ့ စာသားဖန်တီးခြင်း (text generation) ကို လက်တွေ့လုပ်ဆောင်နိုင်မယ်**။
- "Fine-tuning" ဆိုတဲ့ Model ကို **ကိုယ့် data နဲ့ သီးသန့်လေ့ကျင့်ပေးတဲ့ နည်းစနစ်ရဲ့ အခြေခံကို သဘောပေါက်သွားမယ်**။

၂။ ရရှိလာမည့် ကျွမ်းကျင်မှုနှင့် တိုးတက်မှုများ (Skills & Developments)

ဒီသင်ခန်းစာကနေ ဘယ်လိုအရည်အချင်းတွေ တိုးတက်လာမလဲ။ 🚀

- **AI အတွက် Python Programming:** AI နဲ့ machine learning project တွေမှာ Python ကို ဘယ်လိုအသုံးချရလဲဆိုတဲ့ ကျွမ်းကျင်မှု တိုးတက်လာမယ်။
- **Essential Libraries:** AI model တွေနဲ့ အလုပ်လုပ်ဖို့ မရှိမဖြစ်လိုအပ်တဲ့ `transformers` နဲ့ `PyTorch` library တွေကို အသုံးပြုတတ်သွားမယ်။
- **Hugging Face Platform:** ကမ္ဘာ့အကြီးဆုံး AI community ဖြစ်တဲ့ Hugging Face ပေါ်က model တွေကို ဘယ်လိုရှာဖွေ၊ download လုပ်ပြီး အသုံးပြုရမလဲဆိုတာကို သိရှိသွားမယ်။
- **Problem-Solving:** Code ရေးရင်းကြုံတွေ့ရတဲ့ error တွေကို ဘယ်လိုဖြေရှင်းရမလဲဆိုတဲ့ ပြဿနာဖြေရှင်းနိုင်စွမ်း မြင့်တက်လာမယ်။

၃။ လိုအပ်ချက်များ (Requirements)

စတင်လေ့လာဖို့ ဘာတွေလိုအပ်မလဲ။ 💻

- **ကွန်ပျူတာနှင့် အင်တာနက်:** တည်ငြိမ်တဲ့ အင်တာနက်ချိတ်ဆက်မှုရှိတဲ့ ကွန်ပျူတာတစ်လုံး။
- **Python အခြေခံ:** Python programming ရဲ့ အခြေခံ syntax တွေ (variables, loops, functions) ကို နားလည်ထားဖို့လိုပါတယ်။

- **Google Account:** Local computer မှာ setup လုပ်ရတဲ့အခက်အခဲတွေမရှိစေဖို့ Google Colab ကို အသုံးပြုမှာဖြစ်တဲ့အတွက် Google account တစ်ခုရှိရပါမယ်။

၄။ လက်တွေ့ Project: မြန်မာလို ကဗျာတိုလေးတွေ ရေးပေးနိုင်တဲ့ SLM

ကျွန်တော်တို့ရဲ့ ပထမဆုံး project ကတော့ ရိုးရှင်းတဲ့ မြန်မာလို ကာရန်ညီကဗျာတိုလေးတွေ ဒါမှမဟုတ် စာတိုလေးတွေရေးပေးနိုင်တဲ့ SLM တစ်ခုကို တည်ဆောက်မှာဖြစ်ပါတယ်။

Project ၏ ရည်မှန်းချက်

အင်တာနက်ပေါ်မှာရှိပြီးသား ဘာသာစကားမျိုးစုံကို ယေဘုယျနားလည်တဲ့ SLM တစ်ခုကိုယူပြီး၊ ကျွန်တော်တို့ကိုယ်တိုင်ရေးထားတဲ့ မြန်မာကဗျာတိုနမူနာ (dataset) အနည်းငယ်နဲ့ **fine-tune** လုပ်ပြီး သူ့ကို မြန်မာလိုကဗျာဖွဲ့တတ်အောင် "အထူးလေ့ကျင့်" ပေးမှာဖြစ်ပါတယ်။

လုပ်ဆောင်ရမည့် အဆင့်များ

အောက်ပါ code များကို Google Colab (colab.research.google.com) မှာ notebook အသစ်တစ်ခုဖွင့်ပြီး တစ်ဆင့်ချင်းလိုက်ရေးပြီး run ကြည့်နိုင်ပါတယ်။

အဆင့် ၁: Environment ပြင်ဆင်ခြင်း

ပထမဆုံးအနေနဲ့ လိုအပ်တဲ့ library တွေကို install လုပ်ပါမယ်။

Python

```
# Hugging Face က model တွေနဲ့ tools တွေကို သုံးနိုင်ဖို့ transformers library ကို install လုပ်ပါမယ်။
# accelerate က training process ကို ပိုမြန်အောင်ကူညီပေးပါတယ်။
!pip install transformers accelerate
```

အဆင့် ၂: Model နှင့် Tokenizer ရွေးချယ်ခြင်း

ကျွန်တော်တို့ project အတွက် သေးငယ်ပြီး ဘာသာစကားမျိုးစုံကို နားလည်တဲ့ pre-trained SLM တစ်ခုကို Hugging Face ကနေ ရွေးချယ်ပါမယ်။ ဒီမှာတော့ google/flan-t5-small ကို သုံးပါမယ်။

```
Python
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

# Model နာမည်ကို သတ်မှတ်ခြင်း
```

```
model_name = "google/flan-t5-small"

# Pre-trained model နှင့် သူ့ရဲ့ tokenizer (စာတွေကို model နားလည်အောင် ခွဲခြမ်းပေးတဲ့ tool) ကို load လုပ်ခြင်း
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
```

အဆင့် ၃: Data ပြင်ဆင်ခြင်း (Model ကို သင်ပေးမယ့် သင်ခန်းစာ)

Model ကို ကဗျာဖွဲ့တတ်အောင် သင်ပေးဖို့ နမူနာ data အနည်းငယ်လိုပါတယ်။ ဒီမှာတော့ ရိုးရှင်းတဲ့ "input -> output" ပုံစံနဲ့ data ပြင်ဆင်ပါမယ်။

```
Python
# နမူနာ ကဗျာဒေတာ (ဒါထက်များများထည့်လေ ပိုကောင်းလေ)
training_data = [
    {"input": "ကောင်းကင်ကြီးက", "output": "ပြာပြာ."},
    {"input": "ပန်းကလေးက", "output": "လှလှ."},
    {"input": "မိုးစက်တွေက", "output": "ဇေဝေ."},
    {"input": "ကြယ်ကလေးတွေက", "output": "လင်းလက်လို့။"}
]

# Data တွေကို model နားလည်တဲ့ format အဖြစ် ပြောင်းလဲခြင်း
# ဒီနေရာမှာတော့ အလွယ်ကူဆုံးဖြစ်အောင် concept ကိုပဲ အရင်နားလည်စေချင်ပါတယ်
# တကယ် fine-tune လုပ်တဲ့အခါ ဒီထက်ပိုရှုပ်ထွေးတဲ့ data processing လိုအပ်ပါတယ်
print("Data ပြင်ဆင်ပြီးပါပြီ။")
```

အဆင့် ၄: Model ကို စမ်းသပ်ခြင်း (Fine-tuning မလုပ်ခင်)

လေ့ကျင့်မပေးခင်မှာ model က မြန်မာလို ဘယ်လောက်နားလည်လဲဆိုတာကို အရင်စမ်းကြည့်ပါမယ်။

```
Python
# မေးခွန်းတစ်ခု (input) ကို သတ်မှတ်ခြင်း
input_text = "ကောင်းကင်ကြီးက"

# input ကို tokenize လုပ်ခြင်း
```

```
input_ids = tokenizer(input_text, return_tensors="pt").input_ids

# Model ကို စာကြောင်းအသစ် generate လုပ်ခိုင်းခြင်း
outputs = model.generate(input_ids)

# Generate လုပ်လို့ရလာတဲ့ IDs တွေကို လူနားလည်တဲ့ စာသားအဖြစ် ပြန်ပြောင်းခြင်း
decoded_output = tokenizer.decode(outputs[0],
skip_special_tokens=True)
```

print(f"Fine-tuning မလုပ်ခင် Model ရဲ့အဖြေ: {decoded_output}")

ဒီအဆင့်မှာ ထွက်လာတဲ့အဖြေက အဓိပ္ပာယ်သိပ်မရှိတာကို တွေ့ရပါလိမ့်မယ်။
ဘာလို့လဲဆိုတော့ သူ့ကို မြန်မာလိုကဗျာဖွဲ့ဖို့ သင်မပေးရသေးလို့ပါ။

(မှတ်ချက်: တကယ့် Fine-tuning လုပ်ငန်းစဉ်)

တကယ်တမ်း fine-tuning လုပ်ဖို့က အပေါ်က data တွေကို PyTorch ဒါမှမဟုတ် TensorFlow dataset format အဖြစ်ပြောင်းလဲပြီး Trainer class ကိုသုံးပြီး training loop ကို run ရပါတယ်။ ဒါက code အနည်းငယ်ရှည်တဲ့အတွက် ဒီအခြေခံအဆင့်မှာတော့ concept ကိုပဲ အဓိကထားရှင်းပြချင်ပါတယ်။ နောက် module တွေမှာ ဒီအပိုင်းကို အသေးစိတ်တည်ဆောက်သွားပါမယ်။

ဒီ Project ကနေ ဘာသင်ခန်းစာရသလဲ?

ဒီ project ကနေ pre-trained SLM တွေဟာ ယေဘုယျအသိပညာတွေရှိပေမယ့်၊ ကိုယ်လိုချင်တဲ့ တိကျတဲ့အလုပ်တစ်ခု (ဥပမာ - မြန်မာလိုကဗျာဖွဲ့ခြင်း) ကို လုပ်ဆောင်နိုင်ဖို့အတွက် ကိုယ့်ရဲ့ data နဲ့ **အထူးလေ့ကျင့်ပေးဖို့ (fine-tuning) လိုအပ်တယ်** ဆိုတဲ့ အဓိကကျတဲ့ concept ကို နားလည်သွားမှာဖြစ်ပါတယ်။

ကောင်းပါပြီ။ နောက်တစ်ဆင့်ကို ဆက်လိုက်ကြရအောင်။

အရင်အဆင့်မှာတော့ SLM project တစ်ခုအတွက် အကြိုပြင်ဆင်မှုတွေ၊ model ကို ဘယ်လိုရွေးချယ်ရလဲဆိုတာတွေနဲ့ fine-tuning မလုပ်ခင် model ရဲ့အခြေအနေကို စမ်းသပ်ခဲ့ပါတယ်။

အခု **တကယ့် project ရဲ့ အဓိကအပိုင်းဖြစ်တဲ့ "Fine-Tuning"** ကို လက်တွေ့လုပ်ဆောင်တော့မှာ ဖြစ်ပါတယ်။ ဒီအဆင့်မှာ ကျွန်တော်တို့ရဲ့ model ကို မြန်မာလို ကဗျာဖွဲ့တတ်အောင် တကယ် "သင်ပေး" တော့မှာပါ။

ဒါကတော့ **Module 3** နဲ့ **Module 4** ကို ပေါင်းစပ်ပြီး လက်တွေ့တည်ဆောက်မှုအပိုင်းကို အဓိကထားတဲ့ သင်ခန်းစာဖြစ်ပါတယ်။

Module 3 & 4: SLM ကို Fine-Tune လုပ်ပြီး Project စတင်တည်ဆောက်ခြင်း

၁။ သင်ယူရမည့် ရည်ရွယ်ချက်များ (Learning Objectives)

ဒီသင်ခန်းစာပြီးရင် သင်ဘာတွေ တတ်မြောက်သွားမလဲ။ 🤔

- Hugging Face library အတွက် **data တွေကို format မှန်အောင် ပြင်ဆင်တတ်သွားမယ်**။
- **Training Arguments** တွေကို နားလည်ပြီး fine-tuning process ကို ထိန်းချုပ်ဖို့ ဘယ်လိုပြင်ဆင်ရမလဲဆိုတာ သိရှိသွားမယ်။
- Hugging Face ရဲ့ `Trainer` **API ကို အသုံးပြုပြီး model ကို ကိုယ့် data နဲ့ fine-tune လုပ်နိုင်သွားမယ်**။
- Fine-tune မလုပ်ခင်နဲ့ လုပ်ပြီးနောက် **model ရဲ့ တိုးတက်မှုကို လက်တွေ့ နှိုင်းယှဉ်ပြနိုင်မယ်**။

၂။ လက်တွေ့ Project အဆင့်များ: မြန်မာကဗျာဖွဲ့ SLM (အဆက်)

Google Colab မှာ အရင်ရေးခဲ့တဲ့ code တွေကို ဆက်ပြီး အောက်ပါအဆင့်တွေကို လုပ်ဆောင်ပါ။

အဆင့် ၅: Dataset ကို စနစ်တကျ ပြင်ဆင်ခြင်း

Model ကို training လုပ်ဖို့အတွက် ကျွန်တော်တို့ရဲ့ ရိုးရှင်းတဲ့ data list ကို Hugging Face ကနားလည်တဲ့ `Dataset` object အဖြစ် ပြောင်းရပါမယ်။

```
Python
# ဒီ library ကို အရင် install လုပ်ဖို့လိုပါတယ်
!pip install datasets

from datasets import Dataset

# အရင်အဆင့်က ကျွန်တော်တို့ရဲ့ data
training_data_list = [
```

```
    {"input": "ကောင်းကင်ကြီးက", "output": "ပြာပြာ."},
    {"input": "ပန်းကလေးက", "output": "လှလှ."},
    {"input": "မိုးစက်တွေက", "output": "ဇေဝ."},
    {"input": "ကြယ်ကလေးတွေက", "output": "လင်းလက်လို့။"},
    {"input": "နေမင်းကြီးက", "output": "ထိန်ထိန်သာ။"},
    {"input": "လမင်းကြီးက", "output": "သာလို့ဝင်း။"}
]

# List of dictionaries ကို Dataset object အဖြစ် ပြောင်းလဲခြင်း
training_dataset = Dataset.from_list(training_data_list)

# Dataset ကို model ကနားလည်အောင် tokenize လုပ်မယ့် function တစ်ခုရေးပါမယ်
def tokenize_function(examples):
    # Model ရဲ့ input format က "translate English to German: <input>"
ပုံစံဖြစ်တဲ့အတွက်
    # ကျွန်တော်တို့ကလည်း အလားတူ format သုံးပြီး input ကို ပြင်ဆင်ပါမယ်
    inputs = ["ကဗျာဖွဲ့ပါ: " + doc for doc in examples["input"]]

    # Tokenizer ကိုအသုံးပြုပြီး input နဲ့ output တွေကို model နားလည်တဲ့ ID
တွေအဖြစ်ပြောင်းပါမယ်
    model_inputs = tokenizer(inputs, max_length=128, truncation=True)
    labels = tokenizer(text_target=examples["output"], max_length=128,
truncation=True)

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

# Dataset တစ်ခုလုံးကို tokenize လုပ်ခြင်း
tokenized_dataset = training_dataset.map(tokenize_function, batched=True)

print("Dataset ကို Tokenize လုပ်ပြီးပါပြီ။")
print(tokenized_dataset)
```

ရှင်းလင်းချက်: `tokenize_function` ထဲမှာ `input` စာသားရဲ့ရှေ့မှာ `"ကဗျာဖွဲ့ပါ: "` ဆိုတဲ့ instruction စာသားလေးကို ထည့်ပေးလိုက်ပါတယ်။ ဒါက model ကို သူဘာလုပ်ရမလဲဆိုတာ ပိုပြီးနားလည်လွယ်အောင် လမ်းညွှန်ပေးလိုက်တာပါ။

အဆင့် ၆: Training Arguments များ သတ်မှတ်ခြင်း

`TrainingArguments` ဆိုတာ fine-tuning process ကို ဘယ်လိုလုပ်ဆောင်မလဲဆိုတာကို ထိန်းချုပ်တဲ့ settings တွေဖြစ်ပါတယ်။ ဥပမာ - ဘယ်နှစ်ခေါက်သင်မလဲ၊ memory ဘယ်လောက်သုံးမလဲ စသဖြင့်။

```
Python
from transformers import TrainingArguments, Trainer, DataCollatorForSeq2Seq

# Training အတွက် settings တွေကို သတ်မှတ်ခြင်း
training_args = TrainingArguments(
    output_dir="./results",              # Training ရဲ့ output တွေကို
သိမ်းမယ့်နေရာ
    num_train_epochs=50,                 # Dataset ကို အခေါက် ၅၀ ပြန်သင်မယ်
    per_device_train_batch_size=2,       # တစ်ကြိမ်မှာ data ၂ ခုနှုန်းနဲ့ သင်မယ်
    learning_rate=0.001,                 # Model က
ဘယ်လောက်မြန်မြန်သင်ယူမလဲဆိုတဲ့ နှုန်း
    logging_steps=10,                    # အခေါက် ၁၀ ကြိမ်တိုင်းမှာ training
အခြေအနေကို မှတ်တမ်းတင်မယ်
)

# Data တွေကို batch အလိုက် model ထဲထည့်ဖို့ ပြင်ဆင်ပေးတဲ့ tool
data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)
```

အဆင့် ၇: Trainer ကို အသုံးပြုပြီး Model ကို Fine-Tune လုပ်ခြင်း

အခု အားလုံးအသင့်ဖြစ်ပြီမို့ Trainer ကိုသုံးပြီး training ကို စတင်ပါမယ်။

```
Python
# Trainer ကို တည်ဆောက်ခြင်း
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,
)

# Training ကို စတင်ခြင်း
print("Fine-tuning ကို စတင်ပါပြီ...")
trainer.train()
print("Fine-tuning ပြီးဆုံးပါပြီ။")
```

ဒီ code ကို run လိုက်ရင် training process ရဲ့ အခြေအနေကို Colab မှာ အဆင့်ဆင့်ပြသသွားမှာကို တွေ့ရပါမယ်။

အဆင့် ၈: Fine-Tune လုပ်ထားတဲ့ Model ကို စမ်းသပ်ခြင်း

Training ပြီးသွားတဲ့အခါမှာတော့ ကျွန်တော်တို့ရဲ့ model က မြန်မာလိုကဗျာဖွဲ့တာ တော်မတော် စမ်းသပ်ကြည့်ပါမယ်။

```
Python
# စမ်းသပ်မယ့် input စာသား
input_text = "ကောင်းကင်ကြီးက"

# input ကို model နားလည်တဲ့ format ဖြစ်အောင် ရှေ့မှာ instruction ထည့်ပေးပါ
prompt = "ကဗျာဖွဲ့ပါ: " + input_text

# input ကို tokenize လုပ်ခြင်း
input_ids = tokenizer(prompt, return_tensors="pt").input_ids

# Fine-tune လုပ်ထားတဲ့ model ကို စာကြောင်းအသစ် generate လုပ်ခိုင်းခြင်း
outputs = model.generate(input_ids, max_new_tokens=20) # max_new_tokens နဲ့
စကားလုံးအရေအတွက်ကန့်သတ်နိုင်

# Generate လုပ်လို့ရလာတဲ့ IDs တွေကို လူနားလည်တဲ့ စာသားအဖြစ် ပြန်ပြောင်းခြင်း
decoded_output = tokenizer.decode(outputs[0], skip_special_tokens=True)

print("---" * 10)
print(f"သင်ပေးလိုက်တဲ့ စာသား: '{input_text}'")
print(f"Fine-tuned Model ရဲ့အဖြေ: '{decoded_output}'")
```

ရလဒ်ကို နှိုင်းယှဉ်ကြည့်ခြင်း:

အရင် module မှာ run ခဲ့တုန်းက ထွက်လာတဲ့ အဓိပ္ပာယ်မရှိတဲ့အဖြေနဲ့ အခုထွက်လာတဲ့အဖြေကို နှိုင်းယှဉ်ကြည့်ပါ။ ကောင်းကင်ကြီးက ဆိုတဲ့ input အတွက် 'ပြာပြာ.' ဆိုတဲ့ ကျွန်တော်တို့သင်ပေးထားတဲ့အတိုင်း မှန်ကန်တဲ့အဖြေကို ထုတ်ပေးနိုင်တာ တွေ့ရပါလိမ့်မယ်။

ဒါဟာ SLM ကို ကိုယ်ပိုင် data နဲ့ fine-tune လုပ်ခြင်းရဲ့ အစွမ်းပဲဖြစ်ပါတယ်။

သင်ခန်းစာ အနှစ်ချုပ်

ဒီနေ့မှာတော့ data အနည်းငယ်ကိုသုံးပြီး pre-trained SLM တစ်ခုကို သီးသန့်အလုပ်တစ်ခု (မြန်မာလိုကဗျာဖွဲ့ခြင်း) အတွက် ဘယ်လို fine-tune လုပ်ရလဲဆိုတာကို လက်တွေ့လုပ်ဆောင်ခဲ့ပါတယ်။

နောက်လာမယ့် Module မှာတော့ ဒီ fine-tuned model ကိုအသုံးပြုပြီး ရိုးရှင်းတဲ့ **Chatbot** ပုံစံ web interface တစ်ခုကို ဘယ်လိုတည်ဆောက်မလဲဆိုတာ လေ့လာသွားပါမယ်။

အလွန်ကောင်းပါတယ်။ အခုဆိုရင် ကျွန်တော်တို့မှာ မြန်မာလို ကဗျာဖွဲ့တတ်တဲ့ ကိုယ်ပိုင် fine-tuned model တစ်ခု ရှိသွားပါပြီ။ ဒီ model ကို ကိုယ့်ကွန်ပျူတာထဲမှာပဲ run နေလို့တော့ တခြားသူတွေက သုံးလို့မရသေးပါဘူး။

ဒါကြောင့် နောက်တစ်ဆင့်အနေနဲ့ ဒီ model ကို လူတိုင်းအလွယ်တကူသုံးလို့ရမယ့် **ရိုးရှင်းတဲ့ web interface (UI) တစ်ခု** တည်ဆောက်မှာဖြစ်ပါတယ်။ ဒီအဆင့်က **Module 5 နဲ့ 6** ရဲ့ ရည်မှန်းချက်တွေကို ပေါင်းစပ်ပြီး လက်တွေ့အသုံးချနိုင်တဲ့ application တစ်ခု ဖန်တီးမှာဖြစ်ပါတယ်။

ဒီ project အတွက် Gradio ဆိုတဲ့ library ကို သုံးပါမယ်။ Gradio က Python code အနည်းငယ်နဲ့ Machine Learning model တွေအတွက် web UI တွေကို လျင်မြန်လွယ်ကူစွာ တည်ဆောက်ပေးနိုင်ပါတယ်။

Module 5 & 6: Fine-Tuned Model ကို အသုံးပြုပြီး Web App တည်ဆောက်ခြင်း

၁။ သင်ယူရမည့် ရည်ရွယ်ချက်များ (Learning Objectives)

ဒီသင်ခန်းစာပြီးရင် သင်ဘာတွေ တတ်မြောက်သွားမလဲ။ 🤔

- Fine-tuned model ကို **local environment မှာ ဘယ်လိုသိမ်းဆည်းပြီး ပြန်ခေါ် သုံးရမလဲ**ဆိုတာ နားလည်သွားမယ်။
- **Gradio library ကို အသုံးပြုပြီး** model အတွက် input box နဲ့ output box ပါတဲ့ ရိုးရှင်းတဲ့ web interface တစ်ခု တည်ဆောက်နိုင်သွားမယ်။
- ကိုယ်တည်ဆောက်ထားတဲ့ **AI application ကို public link တစ်ခုအနေနဲ့ တခြားသူတွေကို ယာယီ share ပေးတတ်သွားမယ်**။
- AI model တစ်ခုကို project အဖြစ်ကနေ **တကယ့် application အဖြစ် ပြောင်းလဲခြင်းရဲ့ အခြေခံအဆင့်တွေ**ကို သဘောပေါက်သွားမယ်။

၂။ လက်တွေ့ Project အဆင့်များ: ကဗျာဖွဲ့ Chatbot Web App

အရင် Colab notebook မှာပဲ အောက်က code တွေကို ဆက်ပြီး run နိုင်ပါတယ်။

အဆင့် ၉: Fine-Tuned Model နှင့် Tokenizer ကို သိမ်းဆည်းခြင်း (Saving)

အရင်အဆင့်က fine-tune လုပ်ထားတဲ့ model ကို နောက်တစ်ခါ training အစကနေပြန်လုပ်စရာမလိုတော့အောင် folder တစ်ခုထဲမှာ သိမ်းဆည်းလိုက်ပါမယ်။

```
Python
# Model နဲ့ Tokenizer ကို သိမ်းဆည်းမယ့် folder နာမည်
```

```
output_model_dir = "./my_poem_model"

# Model ကို သိမ်းဆည်းခြင်း
model.save_pretrained(output_model_dir)

# Tokenizer ကို သိမ်းဆည်းခြင်း
tokenizer.save_pretrained(output_model_dir)

print(f"Model ကို '{output_model_dir}' ဆိုတဲ့ folder ထဲမှာ သိမ်းဆည်းပြီးပါပြီ။")
```

ဒီ code ကို run ပြီးရင် Colab ရဲ့ ဘယ်ဘက်က Files tab မှာ my_poem_model ဆိုတဲ့ folder အသစ်လေး ပေါ်လာတာကို တွေ့ရပါမယ်။

အဆင့် ၁၀: Gradio Library ကို Install လုပ်ခြင်း

Web UI တည်ဆောက်ဖို့အတွက် Gradio ကို install လုပ်ပါမယ်။

```
Python
!pip install gradio
```

အဆင့် ၁၁: သိမ်းဆည်းထားသော Model ကို ပြန် Load လုပ်ပြီး UI တည်ဆောက်ခြင်း

အခု Gradio ကိုအသုံးပြုပြီး ကျွန်တော်တို့ရဲ့ "ကဗျာဖွဲ့စက်" အတွက် chatbot ပုံစံ interface တစ်ခု တည်ဆောက်ပါမယ်။

```
Python
import gradio as gr
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

# အဆင့် ၉ မှာ သိမ်းဆည်းခဲ့တဲ့ model နဲ့ tokenizer ကို ပြန် load လုပ်ပါမယ်
model_dir = "./my_poem_model"
tokenizer = AutoTokenizer.from_pretrained(model_dir)
model = AutoModelForSeq2SeqLM.from_pretrained(model_dir)

print("Fine-tuned model ကို ပြန်လည် load လုပ်ပြီးပါပြီ။")

# Gradio က ခေါ်သုံးမယ့် function ကို ရေးပါမယ်
def generate_poem(input_text):
    """
    User ဆီက input text ကို ယူပြီး model ကိုသုံးကာ ကဗျာအသစ် generate လုပ်ပေးမယ့် function။
    """
```

```
    # Model ကိုလမ်းညွှန်ပေးမယ့် prompt format
    prompt = "ကဗျာဖွဲ့ပါ: " + input_text

    # input ကို tokenize လုပ်ခြင်း
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids

    # Model ကို generate လုပ်ခိုင်းခြင်း
    outputs = model.generate(input_ids, max_new_tokens=30) # စကားလုံးအရှည်ကို
ကန့်သတ်နိုင်

    # ရလာတဲ့ output ကို လူဖတ်လို့ရအောင် decode ပြန်လုပ်ခြင်း
    decoded_output = tokenizer.decode(outputs[0], skip_special_tokens=True)

    return decoded_output

# Gradio Interface ကို တည်ဆောက်ခြင်း
iface = gr.Interface(
    fn=generate_poem,                    # အဓိက အလုပ်လုပ်မယ့် function
    inputs=gr.Textbox(lines=2, placeholder="ကဗျာဖွဲ့ရန် စာသားတစ်ကြောင်း ထည့်ပါ...
(ဥပမာ - နေမင်းကြီးက)"),
    outputs=gr.Textbox(label="AI ကဗျာ"),
    title="AI ကဗျာဖွဲ့စက်",
    description="SLM ကို fine-tune လုပ်ထားသော ရိုးရှင်းသည့် မြန်မာကဗျာဖွဲ့ chatbot."
)

# Web App ကို စတင် run ခြင်း
iface.launch()
```

အဆင့် ၁၂: Web App ကို စမ်းသပ်အသုံးပြုခြင်း

အပေါ်က `iface.launch()` code ကို run လိုက်တဲ့အခါ Colab ရဲ့ output cell မှာ web interface တစ်ခု ပေါ်လာပါလိမ့်မယ်။

- **Input Box:** `inputs` မှာ သတ်မှတ်ထားတဲ့ `Textbox` ဖြစ်ပါတယ်။ `placeholder` မှာ ရေးထားတဲ့အတိုင်း "နေမင်းကြီးက" လို့ စမ်းရိုက်ထည့်ကြည့်ပါ။
- **Submit Button:** Enter ခေါက် (သို့) Submit button ကို နှိပ်လိုက်ပါ။
- **Output Box:** `generate_poem` function က return ပြန်လိုက်တဲ့ အဖြေ (`'ထိန်ထိန်သာ။'`) က `outputs` မှာ သတ်မှတ်ထားတဲ့ `Textbox` ထဲမှာ ပေါ်လာပါလိမ့်မယ်။

ဒါ့အပြင် output cell မှာ **public URL** တစ်ခုကိုလည်း တွေ့ရပါလိမ့်မယ်။ (ဥပမာ - `Running on public URL: https://....gradio.live`)။ ဒီ link ကို copy ကူးပြီး သင့်သူငယ်ချင်းတွေကို ပို့ပေးရင် သူတို့လည်း သင့်ရဲ့ AI ကဗျာဖွဲ့စက်ကို (နာရီအနည်းငယ်အတွင်း) ဝင်ရောက်သုံးစွဲနိုင်မှာ ဖြစ်ပါတယ်။

သင်ခန်းစာ အနှစ်ချုပ်

ဒီနေ့မှာတော့ fine-tuned model တစ်ခုကို ဘယ်လို production-ready ဖြစ်အောင် ပြင်ဆင်ရမလဲဆိုတဲ့ အခြေခံအဆင့်ကို လေ့လာခဲ့ပါတယ်။ Model ကို သိမ်းဆည်းခြင်း၊ Gradio ကိုသုံးပြီး ရိုးရှင်းတဲ့ UI တည်ဆောက်ခြင်း၊ နှင့် application ကို တခြားသူတွေသုံးနိုင်အောင် share ခြင်းတို့ကို လက်တွေ့လုပ်ဆောင်ခဲ့ပါတယ်။

နောက်လာမယ့် Module မှာတော့ SLM တွေကို ပိုပြီးပေါ့ပါးမြန်ဆန်အောင် ပြုလုပ်တဲ့ **Optimization & Quantization** ဆိုတဲ့ အရေးကြီးတဲ့ technique တွေအကြောင်းကို လေ့လာသွားမှာဖြစ်ပါတယ်။ ဒါက model တွေကို ဖုန်းလိုမျိုး device သေးသေးလေးတွေပေါ်မှာ run နိုင်အောင် ပြုလုပ်တဲ့ နည်းပညာဖြစ်ပါတယ်။

แน่นอนครับ။ ကျွန်တော်တို့ရဲ့ AI ကဗျာဖွဲ့စက်က အခုဆိုရင် အလုပ်လုပ်နေပြီ။ ဒါပေမဲ့ လက်တွေ့ကမ္ဘာမှာတော့ AI model တွေက အရမ်းကြီးများပြီး resource တွေ အများကြီးသုံးလေ့ရှိပါတယ်။ အထူးသဖြင့် ဖုန်းလိုမျိုး device သေးသေးလေးတွေမှာ run ဖို့ဆိုရင် model ကို ပေါ့ပါးအောင်လုပ်ဖို့ လိုအပ်လာပါတယ်။

ဒီအဆင့်ကတော့ **Module 7** ဖြစ်ပြီး၊ model တွေကို ပိုသေးငယ်၊ ပိုမြန်ဆန်အောင် ပြုလုပ်တဲ့ **Optimization** နည်းပညာတွေအကြောင်း လေ့လာမှာဖြစ်ပါတယ်။ ကျွန်တော်တို့ကတော့ **"Quantization"** ဆိုတဲ့ နည်းလမ်းကို လက်တွေ့အသုံးပြုသွားမှာပါ။

Module 7: SLM များကို ပေါ့ပါးအောင်ပြုလုပ်ခြင်း (Optimization & Quantization)

၁။ သင်ယူရမည့် ရည်ရွယ်ချက်များ (Learning Objectives)

ဒီသင်ခန်းစာပြီးရင် သင်ဘာတွေ တတ်မြောက်သွားမလဲ။ 🤔

- Model optimization **ဘာကြောင့် အရေးကြီးတယ်**ဆိုတာကို နားလည်သွားမယ် (ဥပမာ - memory လျှော့ချရန်၊ အမြန်နှုန်း တိုးမြှင့်ရန်)။
- **Quantization ဆိုတာဘာလဲ**ဆိုတာကို ရိုးရှင်းတဲ့ analogy နဲ့ သဘောပေါက်သွားမယ်။
- **bitsandbytes** library ကို အသုံးပြုပြီး fine-tuned model တစ်ခုကို **4-bit precision** အထိ quantize လုပ်နိုင်သွားမယ်။
- Quantize မလုပ်ခင်နဲ့ လုပ်ပြီးနောက်မှာ **model ရဲ့ performance နဲ့ အရွယ်အစား**ကို နှိုင်းယှဉ်တတ်သွားမယ်။

၂။ Quantization ဆိုတာဘာလဲ?

Quantization ဆိုတာ model ထဲမှာရှိတဲ့ ကိန်းဂဏန်းတွေရဲ့ တိကျမှုကို လျှော့ချပြီး model size ကို သေးငယ်အောင် ပြုလုပ်တဲ့ နည်းလမ်းဖြစ်ပါတယ်။

ဥပမာ analogy:

သင့်မှာ အရည်အသွေးအလွန်ကောင်းတဲ့ High-Resolution ဓာတ်ပုံတစ်ပုံ (30MB) ရှိတယ်ဆိုပါစို့။ ဒါက model ရဲ့ မူရင်းအခြေအနေ (FP32) နဲ့တူပါတယ်။ အဲဒီပုံကို Facebook ပေါ်တင်ဖို့အတွက် quality အနည်းငယ်လျှော့ပြီး JPEG format (3MB) အဖြစ် ပြောင်းလိုက်တယ်။ ပုံက အနည်းငယ်ဝါးသွားနိုင်ပေမယ့် ကြည့်ရတာတော့ အတူတူနီးပါးပဲ။ ဒါပေမဲ့ file size ကတော့ အဆပေါင်းများစွာ သေးငယ်သွားတယ်။

Quantization ကလည်း ဒီလိုပါပဲ။ Model ရဲ့ "resolution" ကို အနည်းငယ်လျှော့ချလိုက်တာပါ။ ဒါကြောင့် model က ပေါ့ပါးသွားပြီး memory နည်းတဲ့ device တွေမှာတောင် run နိုင်သွားပါတယ်။

၃။ လက်တွေ့ Project အဆင့်များ: ကဗျာဖွဲ့စက်ကို Optimize လုပ်ခြင်း

Google Colab notebook မှာပဲ အောက်က code တွေကို ဆက်ရေးပြီး run ကြည့်ပါမယ်။

အဆင့် ၁၃: လိုအပ်သော Library များ Install လုပ်ခြင်း

Quantization လုပ်ဖို့အတွက် `bitsandbytes` နဲ့ `accelerate` library တွေ လိုအပ်ပါတယ်။

```
Python
# bitsandbytes က quantization အတွက် အဓိက library ဖြစ်ပါတယ်
!pip install bitsandbytes accelerate
```

အဆင့် ၁၄: Model ကို Quantization ဖြင့် Load လုပ်ခြင်း

အရင် module တုန်းကလိုပဲ model ကို `from_pretrained` နဲ့ load လုပ်မှာဖြစ်ပြီး၊ ဒီတစ်ခါတော့ quantization အတွက် settings အပိုလေးတွေ ထည့်ပေးပါမယ်။

```
Python
import torch
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer,
BitsAndBytesConfig
import gradio as gr

# 4-bit quantization အတွက် configuration ကို သတ်မှတ်ခြင်း
quantization_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_compute_dtype=torch.float16
)
```

```
# အရင်က သိမ်းဆည်းခဲ့တဲ့ model directory
model_dir = "./my_poem_model"

# Tokenizer ကိုအရင် load လုပ်ပါ
tokenizer = AutoTokenizer.from_pretrained(model_dir)

# Model ကို 4-bit quantization နဲ့ load လုပ်ခြင်း
quantized_model = AutoModelForSeq2SeqLM.from_pretrained(
    model_dir,
    quantization_config=quantization_config,
    device_map="auto", # GPU/CPU ကို အလိုအလျောက် ရွေးပေးပါလိမ့်မယ်
)

print("Model ကို 4-bit Quantization ဖြင့်အောင်မြင်စွာ load လုပ်ပြီးပါပြီ။")
```

ရှင်းလင်းချက်: BitsAndBytesConfig ထဲက load_in_4bit=True ဆိုတဲ့ setting လေးတစ်ကြောင်းကြောင့် model ကို load လုပ်တဲ့အခါ memory ထဲမှာ 4-bit precision နဲ့ပဲ တင်ပေးတော့မှာဖြစ်လို့ memory အသုံးပြုမှု သိသိသာသာကို လျော့ကျသွားပါတယ်။

အဆင့် ၁၅: Quantized Model ဖြင့် Web App ကို ပြန်လည်တည်ဆောက်ခြင်း

အခု ပေါ့ပါးသွားတဲ့ quantized_model ကိုသုံးပြီး အရင် module က Gradio app ကိုပဲ ပြန် run ကြည့်ပါမယ်။ Code က အတူတူနီးပါးပါပဲ၊ model ဆိုတဲ့ variable နေရာမှာ quantized_model ကို အစားထိုးသုံးလိုက်ရုံပါပဲ။

```
Python
# Gradio က ခေါ်သုံးမယ့် function (Quantized model ကို သုံးရန် ပြင်ဆင်ထား)
def generate_poem_quantized(input_text):
    prompt = "ကဗျာဖွဲ့ပါ: " + input_text
    input_ids = tokenizer(prompt,
return_tensors="pt").input_ids.to(quantized_model.device) # model ရှိရာ device
သို့ data ပို့ပေးပါ

    # Quantized model ကိုသုံးပြီး generate လုပ်ခြင်း
    outputs = quantized_model.generate(input_ids, max_new_tokens=30)

    decoded_output = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return decoded_output

# Gradio Interface ကို တည်ဆောက်ခြင်း
iface_quantized = gr.Interface(
    fn=generate_poem_quantized,
```

```
    inputs=gr.Textbox(lines=2, placeholder="ကဗျာဖွဲ့ရန် စာသားတစ်ကြောင်း ထည့်ပါ..."),
    outputs=gr.Textbox(label="AI ကဗျာ (Optimized)"),
    title="AI ကဗျာဖွဲ့စက် (Optimized Version)",
    description="4-bit Quantization လုပ်ထားသော SLM ကို အသုံးပြုထားပါသည်။ Model သည် ပိုမိုပေါ့ပါးပြီး မြန်ဆန်ပါသည်။"
)

# Quantized Web App ကို စတင် run ခြင်း
iface_quantized.launch()
```

စမ်းသပ်ကြည့်ခြင်း:

Gradio app မှာ အရင်ကလိုပဲ input တွေထည့်ပြီး စမ်းသပ်ကြည့်ပါ။ "ကောင်းကင်ကြီးက" လို့ထည့်လိုက်ရင် "ပြာပြာ." ဆိုတဲ့အဖြေကိုပဲ ထုတ်ပေးနေတာ တွေ့ရပါလိမ့်မယ်။ ဒါက ဘာကိုပြသလဲဆိုတော့ model size ကို သိသိသာသာလျှော့ချလိုက်ပေမယ့် ကျွန်တော်တို့ရဲ့ ရိုးရှင်းတဲ့ project အတွက်တော့ model ရဲ့ performance ကို ထိခိုက်မှု မရှိသလောက်နီးပါးပဲ ဖြစ်ပါတယ်။

သင်ခန်းစာ အနှစ်ချုပ်

ဒီ module မှာတော့ AI model တွေကို လက်တွေ့အသုံးချနိုင်ဖို့ အလွန်အရေးကြီးတဲ့ optimization technique တစ်ခုဖြစ်တဲ့ **Quantization** အကြောင်းကို လေ့လာခဲ့ပါတယ်။ Model ရဲ့ အရွယ်အစားကို လျှော့ချခြင်းအားဖြင့် memory အသုံးပြုမှုကို သက်သာစေပြီး inference speed (အဖြေထုတ်နှုန်း) ကိုလည်း ပိုမြန်စေနိုင်ပါတယ်။

နောက်လာမယ့် Module ကတော့ ကျွန်တော်တို့ရဲ့ နောက်ဆုံး module ဖြစ်ပါတယ်! အဲ့ဒီမှာတော့ သင်လေ့လာခဲ့တဲ့ skill တွေအားလုံးကို ပေါင်းစပ်ပြီး **သင့်စိတ်ကြိုက် final project တစ်ခုကို ဘယ်လို စဉ်းစားပုံဖော်ပြီး တည်ဆောက်မလဲ**ဆိုတာကို လမ်းညွှန်ပေးသွားမှာ ဖြစ်ပါတယ်။

အကောင်းဆုံးပါပဲ။ ကျွန်တော်တို့ရဲ့ SLM လေ့လာရေးခရီးရဲ့ နောက်ဆုံးအခန်းကို ရောက်ရှိလာပါပြီ။ ဒါကတော့ **Module 8** ဖြစ်ပြီး သင် Module 1 ကနေ 7 အထိ သင်ယူခဲ့သမျှ skill အားလုံးကို ပေါင်းစပ်ပြီး သင့်ရဲ့ **ကိုယ်ပိုင် Final Project** တစ်ခုကို တည်ဆောက်ရမယ့် အပိုင်းဖြစ်ပါတယ်။

ဒီ module မှာတော့ အသင့်ရေးပြီးသား code တွေပေးမှာမဟုတ်ဘဲ၊ သင့်ရဲ့ စိတ်ဝင်စားမှုပေါ်မူတည်ပြီး တကယ်လက်တွေ့အသုံးချနိုင်တဲ့ project တစ်ခုဖြစ်လာအောင် **လမ်းညွှန်ချက်တွေနဲ့ idea တွေ**ကို အဓိကပေးသွားမှာ ဖြစ်ပါတယ်။

Module 8: Final Project - လက်တွေ့အသုံးချ Project တည်ဆောက်ခြင်း

၁။ သင်ယူရမည့် ရည်ရွယ်ချက်များ (Learning Objectives)

ဒီသင်ခန်းစာပြီးရင် သင်ဘာတွေ တတ်မြောက်သွားမလဲ။ 🤔

- ကိုယ်ပိုင် AI project တစ်ခုကို **အစအဆုံး စီမံကိန်းချပြီး တည်ဆောက်နိုင်သွားမယ်** (ပြဿနာကို အဓိပ္ပာယ်ဖွင့်ဆိုခြင်း၊ data ရှာဖွေခြင်း)။
- သင်ယူခဲ့သမျှ skill အားလုံးကို (**Data ပြင်ဆင်ခြင်း၊ Fine-tuning၊ UI တည်ဆောက်ခြင်း၊ Optimization**) လက်တွေ့အသုံးချပြီး application အပြည့်အစုံတစ်ခုကို ဖန်တီးနိုင်သွားမယ်။
- Project တည်ဆောက်နေစဉ် ကြုံတွေ့ရနိုင်တဲ့ **ပြဿနာတွေကို ဖြေရှင်းနိုင်စွမ်း (Problem-Solving)** တိုးတက်လာမယ်။
- SLM ကိုအသုံးပြုပြီး AI application တစ်ခု တည်ဆောက်နိုင်ကြောင်း ပြသနိုင်မယ့် **ကိုယ်ပိုင် Portfolio** တစ်ခု ရရှိလာမယ်။

၂။ Final Project အတွက် Idea များ

သင့်ရဲ့ စိတ်ဝင်စားမှုပေါ်မူတည်ပြီး ဘယ် project ကိုမဆို ရွေးချယ်နိုင်ပါတယ်။ အောက်မှာတော့ သင့်ကို ပေးနိုင်မယ့် idea တစ်ချို့ကို ဖော်ပြပေးလိုက်ပါတယ်။

Idea 1: အသေးစားလုပ်ငန်းများအတွက် မေး-ဖြေ Chatbot

- **အကြောင်းအရာ:** ကော်ဖီဆိုင်၊ စားသောက်ဆိုင် (သို့) online shop လိုမျိုး အသေးစားလုပ်ငန်းတွေရဲ့ အမေးများသောမေးခွန်း (FAQ) တွေကို ဖြေကြားပေးနိုင်တဲ့ chatbot။
- **Data ဥပမာ:**
    - Input: "ဆိုင်ဘယ်အချိန်ဖွင့်လဲ?" -> Output: "ကျွန်တော်တို့ဆိုင်ကို မနက် ၉ နာရီကနေ ည ၆ နာရီအထိ ဖွင့်ပါတယ်ရှင့်။"
    - Input: "ဘာ menu တွေ ကောင်းလဲ?" -> Output: "ဆိုင်ရဲ့ လူကြိုက်အများဆုံး menu တွေကတော့ Shan Noodle နဲ့ Avocado Coffee ဖြစ်ပါတယ်ရှင့်။"
- **စိန်ခေါ်မှု:** သက်ဆိုင်ရာလုပ်ငန်းရဲ့ မေးခွန်းနဲ့အဖြေတွေကို ကိုယ်တိုင်စုဆောင်း ပြင်ဆင်ရပါမယ်။

Idea 2: မြန်မာလို သတင်းအကျဉ်းချုပ် ကိရိယာ

- **အကြောင်းအရာ:** User က သတင်းတို (စာပိုဒ် ၁-၂ ပိုဒ်) ကို ထည့်လိုက်တာနဲ့ AI က အဓိကအချက်ကို ဝါကျ ၁-၂ ကြောင်းနဲ့ အကျဉ်းချုပ်ပေးတဲ့ application။
- **Data ဥပမာ:**
  - Input: (ရန်ကုန်မြို့ မိုးလေဝသအခြေအနေ သတင်းအပြည့်အစုံ) -> Output: "ယနေ့ ရန်ကုန်မြို့တွင် နေ့လည်ပိုင်း၌ မိုးသည်းထန်စွာ ရွာသွန်းနိုင်ပါသည်။"
- **စိန်ခေါ်မှု:** Fine-tune လုပ်ဖို့အတွက် သတင်းအပြည့်အစုံနဲ့ အကျဉ်းချုပ်အတွဲ data တွေကို ရှာဖွေစုဆောင်းရပါမယ်။

Idea 3: မြန်မာအစားအစာ ချက်နည်းလမ်းညွှန် Chatbot

- **အကြောင်းအရာ:** User က "မုန့်ဟင်းခါး" လို့ရိုက်ထည့်လိုက်ရင် ချက်ပြုတ်နည်း အဆင့်ဆင့်ကို ပြောပြနိုင်တဲ့ chatbot။
- **Data ဥပမာ:**
  - Input: "ကြက်သားဟင်း ချက်နည်း" -> Output: "ကြက်သားဟင်းချက်ရန်အတွက် ပထမဆုံး ကြက်သားကို ဆား၊ နနွင်းတို့ဖြင့် နယ်ထားပါ။ ဒုတိယအနေဖြင့်..."
- **စိန်ခေါ်မှု:** ချက်နည်းပြုတ်နည်း data တွေကို format ကျအောင် စုစည်းပြင်ဆင်ရပါမယ်။

၃။ Project တည်ဆောက်ရန် အဆင့်များ (Workflow)

သင့်ရဲ့ final project ကို အောက်ပါအဆင့်များအတိုင်း တည်ဆောက်နိုင်ပါတယ်။

အဆင့် ၁: Idea ရွေးချယ်ပြီး ရည်မှန်းချက် သတ်မှတ်ပါ

- အပေါ်က idea တွေ (သို့) သင့်ကိုယ်ပိုင် idea တစ်ခုကို ရွေးပါ။
- သင်ဘာပြဿနာကို ဖြေရှင်းချင်တာလဲ? သင့် application က user ကို ဘယ်လိုကူညီပေးမလဲ? ဆိုတာကို ရှင်းရှင်းလင်းလင်း သတ်မှတ်ပါ။

အဆင့် ၂: Data စုဆောင်းပြီး ပြင်ဆင်ပါ

- သင့် project အတွက် data တွေကို ဘယ်ကရမလဲ? (ကိုယ်တိုင်ရေးမလား၊ website တွေကနေ ရှာမလား?)
- Data တွေကို `{"input": "...", "output": "..."}` format နဲ့ အနည်းဆုံး အတွဲ ၂၀-၃၀ လောက် ပြင်ဆင်ပါ။ Data များလေ ကောင်းလေပါပဲ။

အဆင့် ၃: Model ရွေးချယ်ပြီး Fine-Tune လုပ်ပါ (Module 3 & 4 Skill)

- သင့် project နဲ့ကိုက်ညီမယ့် base SLM တစ်ခုကို Hugging Face ကနေရွေးချယ်ပါ (`google/flan-t5-small`, `facebook/bart-base`, etc.)။
- Module 4 မှာ လေ့လာခဲ့တဲ့အတိုင်း `Trainer` API ကိုသုံးပြီး သင့် data နဲ့ model ကို fine-tune လုပ်ပါ။

အဆင့် ၄: Web Interface တည်ဆောက်ပါ (Module 5 & 6 Skill)

- Fine-tune လုပ်ပြီးသား model ကို save လုပ်ပါ။
- Gradio ကိုအသုံးပြုပြီး user တွေ အလွယ်တကူသုံးနိုင်မယ့် web interface တစ်ခု တည်ဆောက်ပါ။

အဆင့် ၅: Model ကို Optimize လုပ်ပါ (Module 7 Skill - Optional)

- သင့် application ကို ပိုပေါ့ပါးမြန်ဆန်စေချင်ရင် Module 7 မှာ သင်ခဲ့တဲ့ Quantization နည်းလမ်းကို အသုံးပြုပြီး model ကို optimize လုပ်ပါ။

အဆင့် ၆: စမ်းသပ်ပြီး Share ပါ

- သင့် application ကို သေချာစမ်းသပ်ပါ။ မထင်မှတ်ထားတဲ့ input တွေထည့်ပြီး အလုပ်ကောင်းကောင်းလုပ်မလုပ် စစ်ဆေးပါ။
- ပြီးပြည့်စုံသွားရင် Gradio ကပေးတဲ့ public link ကို သင့်သူငယ်ချင်းတွေ၊ community တွေကို share ပြီး feedback တောင်းပါ။

အောင်မြင်ဖို့အတွက် အကြံပြုချက်များ

- **သေးသေးလေးက စပါ:** ပထမဆုံး project ဖြစ်တဲ့အတွက် အရမ်းရှုပ်ထွေးတဲ့ idea မျိုးကို မရွေးချယ်ပါနဲ့။ ရိုးရှင်းပြီး လက်တွေ့အသုံးချနိုင်တဲ့ project တစ်ခုက သင့်ကို အများကြီးသင်ယူနိုင်စေပါလိမ့်မယ်။
- **Error တွေကို မကြောက်ပါနဲ့:** Code ရေးတဲ့အခါ error တက်တာက သင်ယူခြင်းရဲ့ အစိတ်အပိုင်းတစ်ခုပါပဲ။ Error message ကို သေချာဖတ်ပြီး Google မှာ ဘယ်လိုရှာဖွေဖြေရှင်းရမလဲဆိုတာကို လေ့ကျင့်ပါ။
- **Data က အဓိကပါ:** SLM project တွေမှာ model အကြီးကြီးသုံးတာထက်၊ သန့်ရှင်းပြီး အရည်အသွေးကောင်းတဲ့ fine-tuning data ရှိတာက ပိုအရေးကြီးပါတယ်။

"အသေးစားလုပ်ငန်းများအတွက် မေး-ဖြေ Chatbot" (Idea 1) ကို အစကနေ စတည်ဆောက်ပုံကို အဆင့်ဆင့် ရှင်းပြပေးပါမယ်။

သင့်ရဲ့ Final Project အတွက် ဒီအောက်က အဆင့်များအတိုင်း လုပ်ဆောင်နိုင်ပါတယ်။

အဆင့် ၁: စီမံကိန်း ရည်မှန်းချက် သတ်မှတ်ခြင်း

ပထမဆုံးအနေနဲ့ သင့်ရဲ့ chatbot က ဘယ်လို အသေးစားလုပ်ငန်း (ဥပမာ- ကော်ဖီဆိုင်၊ စားသောက်ဆိုင် ဒါမှမဟုတ် online shop) အတွက် ဖြစ်မလဲဆိုတာ ရွေးချယ်ပါ။ သင့်ရဲ့

application က သုံးစွဲသူတွေရဲ့ မေးခွန်းတွေကို ဘယ်လိုကူညီဖြေရှင်းပေးမလဲဆိုတာ ရှင်းရှင်းလင်းလင်း သတ်မှတ်ပါ။

အဆင့် ၂: Data စုဆောင်းပြီး ပြင်ဆင်ခြင်း

ဒီအဆင့်က project ရဲ့ အရေးကြီးဆုံးအပိုင်းပါ။ သင့်ရဲ့ chatbot ဖြေကြားပေးမယ့် မေးခွန်းနဲ့အဖြေအတွဲတွေကို စုဆောင်းရပါမယ်။

- **Data ပုံစံ:** Data တွေကို `{"input": "...", "output": "..."}` ဆိုတဲ့ ပုံစံနဲ့ ပြင်ဆင်ပါ။
- **အရေအတွက်:** အနည်းဆုံး မေးခွန်း-အဖြေ အတွဲ ၂၀-၃၀ လောက်ရှိဖို့ လိုအပ်ပါတယ်။ Data များလေလေ၊ model က ပိုကောင်းလေလေပါပဲ။
- **ဥပမာများ:**
    - `{"input": "ဆိုင်ဘယ်အချိန်ဖွင့်လဲ?", "output": "ကျွန်တော်တို့ဆိုင်ကို မနက် ၉ နာရီကနေ ည ၆ နာရီအထိ ဖွင့်ပါတယ်ရှင့်။"}`
    - `{"input": "ဘာ menu တွေ ကောင်းလဲ?", "output": "ဆိုင်ရဲ့ လူကြိုက်အများဆုံး menu တွေကတော့ Shan Noodle နဲ့ Avocado Coffee ဖြစ်ပါတယ်ရှင့်။"}`
    - `{"input": "အိမ်အရောက် ပို့ပေးလား?", "output": "ဟုတ်ကဲ့၊ ရန်ကုန်မြို့တွင်းဆို အိမ်အရောက် ပို့ဆောင်ပေးပါတယ်ရှင့်။"}` (ဒီလို ကိုယ်ပိုင် data တွေလည်း ထပ်ထည့်နိုင်ပါတယ်)

အဆင့် ၃: Model ရွေးချယ်ပြီး Fine-Tune လုပ်ခြင်း

သင့်ရဲ့ chatbot ကို မေး-ဖြေတတ်အောင် သင်ပေးဖို့ Hugging Face ကနေ pre-trained model တစ်ခုကို ရွေးချယ်ပါ။

- `google/flan-t5-small` လိုမျိုးသေးငယ်တဲ့ model တစ်ခုကို ရွေးချယ်နိုင်ပါတယ်။
- Module 3 နဲ့ 4 မှာ သင်ခဲ့တဲ့အတိုင်း Hugging Face `Trainer API` ကိုသုံးပြီး အဆင့် ၂ မှာ ပြင်ဆင်ထားတဲ့ Data တွေနဲ့ Model ကို Fine-tune လုပ်ပါ။

အဆင့် ၄: Web Interface တည်ဆောက်ခြင်း

Fine-tune လုပ်ထားတဲ့ model ကို သုံးစွဲသူတွေ အလွယ်တကူ အသုံးပြုနိုင်ဖို့ Gradio library ကို သုံးပြီး web interface တစ်ခု တည်ဆောက်ပါ။

- အရင်ဆုံး Fine-tune လုပ်ပြီးသား model ကို folder တစ်ခုထဲမှာ သိမ်းဆည်းပါ။
- Module 5 နဲ့ 6 မှာ သင်ခဲ့တဲ့အတိုင်း Gradio code တွေနဲ့ `fn`, `inputs`, `outputs`, `title` စတဲ့ arguments တွေကို ထည့်သွင်းပြီး Interface ကို တည်ဆောက်ပါ။

အဆင့် ၅: Model ကို Optimization လုပ်ခြင်း (Optional)

သင့်ရဲ့ application ကို ပိုမိုပေါ့ပါးမြန်ဆန်စေလိုပါက Module 7 မှာ သင်ခဲ့တဲ့ Quantization နည်းလမ်းကို အသုံးပြုပြီး model ကို optimization လုပ်နိုင်ပါတယ်။

အဆင့် ၆: စမ်းသပ်ပြီး Share လုပ်ခြင်း

သင့်ရဲ့ application ကို အသေးစိတ် စမ်းသပ်ပါ။ မထင်မှတ်ထားတဲ့ မေးခွန်းတွေကိုပါ ထည့်ပြီး စမ်းသပ်ကြည့်ပါ။ အားလုံး အဆင်ပြေပြီဆိုရင် Gradio ကပေးတဲ့ public link ကို သင့်သူငယ်ချင်းတွေ ဒါမှမဟုတ် community တွေဆီ share လုပ်ပြီး feedback တောင်းခံနိုင်ပါတယ်။

ကောင်းပါပြီ။ ပေးထားတဲ့ စာရွက်စာတမ်းပေါ် အခြေခံပြီး **FAQ Chatbot** အတွက် လိုအပ်တဲ့ Python code တွေကို အဆင့်ဆင့် ဖော်ပြပေးလိုက်ပါတယ်။ ဒီ code တွေကို **Google Colab** မှာ တစ်ဆင့်ချင်း လိုက်ရေးပြီး run နိုင်ပါတယ်။

1. Environment ပြင်ဆင်ခြင်း

ပထမဆုံးအနေနဲ့၊

Hugging Face model တွေနဲ့ အလုပ်လုပ်ဖို့ လိုအပ်တဲ့ library တွေကို install လုပ်ပါမယ်

```
Python
# transformers library ကို install လုပ်ပါမယ်။
!pip install transformers accelerate
# Dataset တွေကို စီမံခန့်ခွဲဖို့အတွက် datasets library ကို install လုပ်ပါမယ်။
!pip install datasets
# Gradio နဲ့ web interface တည်ဆောက်ဖို့အတွက် install လုပ်ပါမယ်။
!pip install gradio
```

---

2. Model နှင့် Tokenizer ရွေးချယ်ခြင်း

သင့်ရဲ့ chatbot အတွက်

`google/flan-t5-small` ကို ရွေးချယ်ပြီး `Hugging Face` ကနေ load လုပ်ပါမယ်

```
Python
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

model_name = "google/flan-t5-small"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

print("Model နဲ့ Tokenizer ကို load လုပ်ပြီးပါပြီ။")
```

---

3. Data ပြင်ဆင်ခြင်း

သင့်ရဲ့ chatbot အတွက် မေးခွန်း-အဖြေ (FAQ) အတွဲတွေကို စုဆောင်းပြီး

`Hugging Face` က နားလည်တဲ့ format အဖြစ် ပြောင်းလဲပါမယ်

```
Python
from datasets import Dataset

# FAQ Chatbot အတွက် နမူနာ data
training_data_list = [
    {"input": "ဆိုင်ဘယ်အချိန်ဖွင့်လဲ?", "output": "ကျွန်တော်တို့ဆိုင်ကို မနက် ၉ နာရီကနေ ည ၆ နာရီအထိ ဖွင့်ပါတယ်ရှင့်။"},
    {"input": "ဘာ menu တွေ ကောင်းလဲ?", "output": "ဆိုင်ရဲ့ လူကြိုက်အများဆုံး menu တွေကတော့ Shan Noodle နဲ့ Avocado Coffee ဖြစ်ပါတယ်ရှင့်။"},
    {"input": "အိမ်အရောက် ပို့ပေးလား?", "output": "ဟုတ်ကဲ့၊ မြို့တွင်းဆို အိမ်အရောက် ပို့ဆောင်ပေးပါတယ်။"},
    {"input": "အွန်လိုင်းကနေ မှာလို့ရလား?", "output": "ဟုတ်ကဲ့၊ Facebook Messenger ကနေလည်း မှာယူနိုင်ပါတယ်ရှင်။"}
]

training_dataset = Dataset.from_list(training_data_list)

def tokenize_function(examples):
```

```
    # Model ကို ညွှန်ကြားချက်ပေးဖို့အတွက် "မေးခွန်းဖြေပါ: " ဆိုတဲ့ prefix ထည့်ပေးပါမယ်။
    inputs = ["မေးခွန်းဖြေပါ: " + doc for doc in examples["input"]]
    model_inputs = tokenizer(inputs, max_length=128, truncation=True)
    labels = tokenizer(text_target=examples["output"], max_length=128,
truncation=True)
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

tokenized_dataset = training_dataset.map(tokenize_function, batched=True)

print("Dataset ကို tokenize လုပ်ပြီးပါပြီ။")
print(tokenized_dataset)
```

---

4. Model ကို Fine-Tune လုပ်ခြင်း

ဒီအဆင့်မှာတော့

Trainer API ကိုသုံးပြီး model ကို သင်ယူခိုင်းပါမယ်

```
from transformers import TrainingArguments, Trainer, DataCollatorForSeq2Seq

# Training အတွက် settings တွေကို သတ်မှတ်ခြင်း
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=50,
    per_device_train_batch_size=2,
    learning_rate=0.001,
    logging_steps=10,
)

# Data တွေကို batch အလိုက် ထည့်ဖို့ ပြင်ဆင်ပေးခြင်း
data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)

# Trainer ကို တည်ဆောက်ခြင်း
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,
)
```

```
print("Fine-tuning ကို စတင်ပါပြီ။")
trainer.train()
print("Fine-tuning ပြီးဆုံးပါပြီ။")
```

---

5. Fine-Tuned Model ကို သိမ်းဆည်းခြင်း

Fine-tuning လုပ်ပြီးသား model နဲ့ tokenizer ကို နောက်တစ်ကြိမ် training ထပ်လုပ်စရာမလိုအောင် သိမ်းထားပါမယ်.

```
Python
output_model_dir = "./my_faq_chatbot_model"

model.save_pretrained(output_model_dir)
tokenizer.save_pretrained(output_model_dir)

print(f"Model ကို '{output_model_dir}' ဆိုတဲ့ folder ထဲမှာ သိမ်းဆည်းပြီးပါပြီ။")
```

---

6. Gradio ဖြင့် Web App တည်ဆောက်ခြင်း

သိမ်းထားတဲ့ model ကို load ပြန်လုပ်ပြီး Gradio ကိုသုံးကာ chatbot web interface တစ်ခု တည်ဆောက်ပါမယ်.

```
Python
import gradio as gr
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

# အဆင့် ၅ မှာ သိမ်းဆည်းခဲ့တဲ့ model နဲ့ tokenizer ကို ပြန် load လုပ်ပါမယ်။
model_dir = "./my_faq_chatbot_model"
tokenizer = AutoTokenizer.from_pretrained(model_dir)
model = AutoModelForSeq2SeqLM.from_pretrained(model_dir)

print("Fine-tuned model ကို ပြန်လည် load လုပ်ပြီးပါပြီ။")

# Gradio က ခေါ်သုံးမယ့် function
def generate_answer(input_text):
    # Model ကို လမ်းညွှန်ပေးမယ့် prompt format
    prompt = "မေးခွန်းဖြေပါ: " + input_text
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids
    outputs = model.generate(input_ids, max_new_tokens=50)
    decoded_output = tokenizer.decode(outputs[0], skip_special_tokens=True)
```

```
    return decoded_output

# Gradio Interface ကို တည်ဆောက်ခြင်း
iface = gr.Interface(
    fn=generate_answer,
    inputs=gr.Textbox(lines=2, placeholder="မေးခွန်းတစ်ခု မေးပါ... (ဥပမာ - ဆိုင်ဘယ်အချိန်ဖွင့်လဲ?)"),
    outputs=gr.Textbox(label="Chatbot အဖြေ"),
    title="FAQ Chatbot",
    description="အသေးစားလုပ်ငန်းများအတွက် အမေးများတဲ့မေးခွန်းများကို ဖြေကြားပေးနိုင်သော Chatbot။"
)

# Web App ကို စတင် run ခြင်း
iface.launch()
```

မြန်မာလို သတင်းအကျဉ်းချုပ် ကိရိယာ (Idea 2) ကို အစကနေ စတင်တည်ဆောက်ဖို့အတွက် လိုအပ်တဲ့ Python code တွေကို အဆင့်ဆင့် ဖော်ပြပေးလိုက်ပါတယ်။ ဒီ code တွေကိုလည်း **Google Colab** မှာ တစ်ဆင့်ချင်း လိုက်ရေးပြီး run နိုင်ပါတယ်။

---

1. Environment ပြင်ဆင်ခြင်း

ပထမဆုံးအနေနဲ့၊ Hugging Face model တွေနဲ့ အလုပ်လုပ်ဖို့ လိုအပ်တဲ့ library တွေကို install လုပ်ပါမယ်။

```
Python
# transformers library ကို install လုပ်ပါမယ်။
!pip install transformers accelerate
# Dataset တွေကို စီမံခန့်ခွဲဖို့အတွက် datasets library ကို install လုပ်ပါမယ်။
!pip install datasets
# Gradio နဲ့ web interface တည်ဆောက်ဖို့အတွက် install လုပ်ပါမယ်။
!pip install gradio
```

2. Model နှင့် Tokenizer ရွေးချယ်ခြင်း

ဒီ project အတွက်လည်း google/flan-t5-small ကိုပဲ ရွေးချယ်ပြီး Hugging Face ကနေ load လုပ်ပါမယ်။

```
Python
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

model_name = "google/flan-t5-small"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

print("Model နဲ့ Tokenizer ကို load လုပ်ပြီးပါပြီ။")
```

3. Data ပြင်ဆင်ခြင်း

ဒီအဆင့်မှာ သတင်းအပြည့်အစုံနဲ့ အကျဉ်းချုပ်အတွဲတွေကို စုဆောင်းပြီး Hugging Face က နားလည်တဲ့ format အဖြစ် ပြောင်းလဲပါမယ်။

```
Python
from datasets import Dataset

# သတင်းအကျဉ်းချုပ်အတွက် နမူနာ data
training_data_list = [
    {"input": "ရန်ကုန်မြို့တွင် ယနေ့ နေ့လည် ၁ နာရီခန့်က စတင်၍ မိုးသည်းထန်စွာ ရွာသွန်းခဲ့ပြီး လမ်းမများပေါ်တွင် ရေကြီးရေလျှံမှုများ ဖြစ်ပွားခဲ့သည်။ ယာဉ်ကြောပိတ်ဆို့မှုများလည်း အဓိကလမ်းမကြီးများပေါ်တွင် ဖြစ်ပေါ်ခဲ့သည်။", "output": "ယနေ့ ရန်ကုန်မြို့တွင် နေ့လည်ပိုင်း၌ မိုးသည်းထန်စွာ ရွာသွန်းနိုင်ပါသည်။"},
    {"input": "မြန်မာ့ရုပ်ရှင်လောကအတွက် ဂုဏ်ပြုဆုများ ချီးမြှင့်မည့် မြန်မာ့ရုပ်ရှင်ထူးချွန်ဆုပေးပွဲကို ဒီဇင်ဘာလ ၁၅ ရက်နေ့တွင် ရန်ကုန်မြို့၌ ကျင်းပမည်ဖြစ်ကြောင်း မြန်မာနိုင်ငံရုပ်ရှင်အစည်းအရုံးမှ သတင်းထုတ်ပြန်သည်။ အဆိုပါအခမ်းအနားကို တိုက်ရိုက်ထုတ်လွှင့်သွားမည်ဖြစ်သည်။", "output": "ဒီဇင်ဘာ ၁၅ ရက်တွင် ရန်ကုန်၌ မြန်မာ့ရုပ်ရှင်ထူးချွန်ဆုပေးပွဲ ကျင်းပမည်။"},
    {"input": "၂၀၂၄ အာဆီယံချန်ပီယံရှစ် ဘောလုံးပြိုင်ပွဲအတွက် မြန်မာအမျိုးသားဘောလုံးအသင်း၏ ကစားသမား ၂၅ ဦးစာရင်းကို နည်းပြချုပ်က ထုတ်ပြန်ကြေညာလိုက်ပြီဖြစ်သည်။ အသင်းသည် လာမည့်အပတ်မှစ၍ လေ့ကျင့်မှုများ စတင်မည်ဟု သိရသည်။", "output": "၂၀၂၄ အာဆီယံချန်ပီယံရှစ်အတွက် မြန်မာဘောလုံးအသင်း ကစားသမားစာရင်း ထုတ်ပြန်။"}
]

training_dataset = Dataset.from_list(training_data_list)
```

```
def tokenize_function(examples):
    # Model ကို ညွှန်ကြားချက်ပေးဖို့အတွက် "အကျဉ်းချုပ်ပါ: " ဆိုတဲ့ prefix ထည့်ပေးပါမယ်။
    inputs = ["အကျဉ်းချုပ်ပါ: " + doc for doc in examples["input"]]
    model_inputs = tokenizer(inputs, max_length=512, truncation=True)
    labels = tokenizer(text_target=examples["output"], max_length=128,
truncation=True)
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

tokenized_dataset = training_dataset.map(tokenize_function, batched=True)

print("Dataset ကို tokenize လုပ်ပြီးပါပြီ။")
print(tokenized_dataset)
```

---

4. Model ကို Fine-Tune လုပ်ခြင်း

ဒီအဆင့်မှာတော့ Trainer API ကိုသုံးပြီး model ကို သင်ယူခိုင်းပါမယ်။

```
Python
from transformers import TrainingArguments, Trainer, DataCollatorForSeq2Seq

# Training အတွက် settings တွေကို သတ်မှတ်ခြင်း
training_args = TrainingArguments(
    output_dir="./results_summarization",
    num_train_epochs=50,
    per_device_train_batch_size=2,
    learning_rate=0.001,
    logging_steps=10,
)

# Data တွေကို batch အလိုက် ထည့်ဖို့ ပြင်ဆင်ပေးခြင်း
data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)

# Trainer ကို တည်ဆောက်ခြင်း
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,
)
```

```
print("Fine-tuning ကို စတင်ပါပြီ။")
trainer.train()
print("Fine-tuning ပြီးဆုံးပါပြီ။")
```

---

5. Fine-Tuned Model ကို သိမ်းဆည်းခြင်း

Fine-tuning လုပ်ပြီးသား model နဲ့ tokenizer ကို နောက်တစ်ကြိမ် training ထပ်လုပ်စရာမလိုအောင် သိမ်းထားပါမယ်။

```
Python
output_model_dir = "./my_summarization_model"

model.save_pretrained(output_model_dir)
tokenizer.save_pretrained(output_model_dir)

print(f"Model ကို '{output_model_dir}' ဆိုတဲ့ folder ထဲမှာ သိမ်းဆည်းပြီးပါပြီ။")
```

---

6. Gradio ဖြင့် Web App တည်ဆောက်ခြင်း

သိမ်းထားတဲ့ model ကို load ပြန်လုပ်ပြီး Gradio ကိုသုံးကာ summarization web interface တစ်ခု တည်ဆောက်ပါမယ်။

```
Python
import gradio as gr
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

# အဆင့် ၅ မှာ သိမ်းဆည်းခဲ့တဲ့ model နဲ့ tokenizer ကို ပြန် load လုပ်ပါမယ်။
model_dir = "./my_summarization_model"
tokenizer = AutoTokenizer.from_pretrained(model_dir)
model = AutoModelForSeq2SeqLM.from_pretrained(model_dir)

print("Fine-tuned model ကို ပြန်လည် load လုပ်ပြီးပါပြီ။")

# Gradio က ခေါ်သုံးမယ့် function
def summarize_text(input_text):
    # Model ကို လမ်းညွှန်ပေးမယ့် prompt format
    prompt = "အကျဉ်းချုပ်ပါ: " + input_text
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids
    outputs = model.generate(input_ids, max_new_tokens=100)
```

```
    decoded_output = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return decoded_output

# Gradio Interface ကို တည်ဆောက်ခြင်း
iface = gr.Interface(
    fn=summarize_text,
    inputs=gr.Textbox(lines=5, placeholder="သတင်းအပြည့်အစုံကို ဒီနေရာမှာ ထည့်ပါ..."),
    outputs=gr.Textbox(label="အကျဉ်းချုပ်"),
    title="မြန်မာလို သတင်းအကျဉ်းချုပ် ကိရိယာ",
    description="သတင်းရှည်များကို အဓိကအချက်များသာ ပါဝင်သော ဝါကျတိုများအဖြစ် အကျဉ်းချုပ်ပေးနိုင်သော application။"
)

# Web App ကို စတင် run ခြင်း
iface.launch()
```

IDEA 3: မြန်မာအစားအစာ ချက်နည်းလမ်းညွှန် Chatbot (Idea 3) ကို အစကနေ စတင်တည်ဆောက်ဖို့အတွက် လိုအပ်တဲ့ Python code တွေကို အဆင့်ဆင့် ဖော်ပြပေးလိုက်ပါတယ်။ ဒီ code တွေကိုလည်း **Google Colab** မှာ တစ်ဆင့်ချင်း လိုက်ရေးပြီး run နိုင်ပါတယ်။

---

1. Environment ပြင်ဆင်ခြင်း

ပထမဆုံးအနေနဲ့၊ Hugging Face model တွေနဲ့ အလုပ်လုပ်ဖို့ လိုအပ်တဲ့ library တွေကို install လုပ်ပါမယ်။

```
Python
# transformers library ကို install လုပ်ပါမယ်။
!pip install transformers accelerate
# Dataset တွေကို စီမံခန့်ခွဲဖို့အတွက် datasets library ကို install လုပ်ပါမယ်။
!pip install datasets
```

```
# Gradio နဲ့ web interface တည်ဆောက်ဖို့အတွက် install လုပ်ပါမယ်။
!pip install gradio
```

---

2. Model နှင့် Tokenizer ရွေးချယ်ခြင်း

ဒီ project အတွက်လည်း **google/flan-t5-small** ကိုပဲ ရွေးချယ်ပြီး **Hugging Face** ကနေ load လုပ်ပါမယ်။

```
Python
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

model_name = "google/flan-t5-small"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

print("Model နဲ့ Tokenizer ကို load လုပ်ပြီးပါပြီ။")
```

---

3. Data ပြင်ဆင်ခြင်း

ဒီအဆင့်မှာ ဟင်းချက်နည်းတွေကို စုဆောင်းပြီး **Hugging Face** က နားလည်တဲ့ format အဖြစ် ပြောင်းလဲပါမယ်။

```
Python
from datasets import Dataset

# ဟင်းချက်နည်းအတွက် နမူနာ data
training_data_list = [
    {"input": "ကြက်သားဟင်း ချက်နည်း", "output": "ကြက်သားဟင်းချက်ရန်အတွက် ပထမဆုံး ကြက်သားကို ဆား၊ နနွင်းတို့ဖြင့် နယ်ထားပါ။ ဒုတိယအနေဖြင့် ကြက်သွန်နီ၊ ဂျင်း၊ ကြက်သွန်ဖြူတို့ကို ဆီသတ်ပြီး ကြက်သားထည့်ကာ ဆီသတ်ပါ။ ပြီးနောက် ဟင်းအနှစ်ရလာပါက ရေအနည်းငယ်ထည့်၍ အဖုံးအုပ်ပြီး ဆူအောင် တည်ပါ။"},
    {"input": "မုန့်ဟင်းခါး ချက်နည်း", "output": "မုန့်ဟင်းခါးချက်ရန်အတွက် ငါးကိုပြုတ်ပြီး အသားနွာထားပါ။ ကြက်သွန်နီ၊ ဂျင်း၊ ကြက်သွန်ဖြူ၊ ငရုတ်သီး၊ ငပိတို့ဖြင့် အနှစ်ချက်ကာ ငါးအသားများ ထည့်ပါ။ ဆန်မှုန့်ဖျော်ရည်နှင့် ရေထည့်ပြီး ဆူအောင် တည်ပါ။ အရသာအပေါ့အငံ မြည်းပြီးပါက မုန့်ဟင်းခါးဖတ်ဖြင့် တွဲဖက်သုံးဆောင်နိုင်ပါသည်။"},
    {"input": "ပဲကုလားဟင်း ချက်နည်း", "output": "ပဲကုလားဟင်းချက်ရန် ပဲကုလားကို နူးသည်အထိ ပြုတ်ပါ။ ကြက်သွန်ဖြူ၊ ဂျင်းတို့ကို ညက်အောင် ထောင်းပြီး ဆီသတ်ပါ။ ထောင်းထားသော အရောအနှောများ ထည့်ပြီး ဆီသတ်ပါ။ အရသာအပေါ့အငံမြည်းပြီး နံနံပင်ဖြင့် အလှဆင်ကာ စားနိုင်ပါပြီ။"}
```

```
]

training_dataset = Dataset.from_list(training_data_list)

def tokenize_function(examples):
    # Model ကို ညွှန်ကြားချက်ပေးဖို့အတွက် "ချက်နည်းပေးပါ: " ဆိုတဲ့ prefix ထည့်ပေးပါမယ်။
    inputs = ["ချက်နည်းပေးပါ: " + doc for doc in examples["input"]]
    model_inputs = tokenizer(inputs, max_length=512, truncation=True)
    labels = tokenizer(text_target=examples["output"], max_length=512,
truncation=True)
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

tokenized_dataset = training_dataset.map(tokenize_function, batched=True)

print("Dataset ကို tokenize လုပ်ပြီးပါပြီ။")
print(tokenized_dataset)
```

---

4. Model ကို Fine-Tune လုပ်ခြင်း

ဒီအဆင့်မှာတော့ **Trainer API** ကိုသုံးပြီး model ကို သင်ယူခိုင်းပါမယ်။

```
Python
from transformers import TrainingArguments, Trainer, DataCollatorForSeq2Seq

# Training အတွက် settings တွေကို သတ်မှတ်ခြင်း
training_args = TrainingArguments(
    output_dir="./results_recipes",
    num_train_epochs=50,
    per_device_train_batch_size=2,
    learning_rate=0.001,
    logging_steps=10,
)

# Data တွေကို batch အလိုက် ထည့်ဖို့ ပြင်ဆင်ပေးခြင်း
data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)

# Trainer ကို တည်ဆောက်ခြင်း
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
```

```
    data_collator=data_collator,
)

print("Fine-tuning ကို စတင်ပါပြီ။")
trainer.train()
print("Fine-tuning ပြီးဆုံးပါပြီ။")
```

---

5. Fine-Tuned Model ကို သိမ်းဆည်းခြင်း

Fine-tuning လုပ်ပြီးသား model နဲ့ tokenizer ကို နောက်တစ်ကြိမ် training ထပ်လုပ်စရာမလိုအောင် သိမ်းထားပါမယ်။

```
Python
output_model_dir = "./my_recipe_chatbot_model"

model.save_pretrained(output_model_dir)
tokenizer.save_pretrained(output_model_dir)

print(f"Model ကို '{output_model_dir}' ဆိုတဲ့ folder ထဲမှာ သိမ်းဆည်းပြီးပါပြီ။")
```

---

6. Gradio ဖြင့် Web App တည်ဆောက်ခြင်း

သိမ်းထားတဲ့ model ကို load ပြန်လုပ်ပြီး Gradio ကိုသုံးကာ chatbot web interface တစ်ခု တည်ဆောက်ပါမယ်။

```
Python
import gradio as gr
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

# အဆင့် ၅ မှာ သိမ်းဆည်းခဲ့တဲ့ model နဲ့ tokenizer ကို ပြန် load လုပ်ပါမယ်။
model_dir = "./my_recipe_chatbot_model"
tokenizer = AutoTokenizer.from_pretrained(model_dir)
model = AutoModelForSeq2SeqLM.from_pretrained(model_dir)

print("Fine-tuned model ကို ပြန်လည် load လုပ်ပြီးပါပြီ။")

# Gradio က ခေါ်သုံးမယ့် function
def generate_recipe(input_text):
    # Model ကို လမ်းညွှန်ပေးမယ့် prompt format
    prompt = "ချက်နည်းပေးပါ: " + input_text
```

```
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids
    outputs = model.generate(input_ids, max_new_tokens=200)
    decoded_output = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return decoded_output

# Gradio Interface ကို တည်ဆောက်ခြင်း
iface = gr.Interface(
    fn=generate_recipe,
    inputs=gr.Textbox(lines=2, placeholder="ဟင်းပွဲနာမည်ကို ရိုက်ထည့်ပါ... (ဥပမာ - ကြက်သားဟင်း)"),
    outputs=gr.Textbox(label="ချက်နည်းလမ်းညွှန်"),
    title="မြန်မာအစားအစာ ချက်နည်းလမ်းညွှန် Chatbot",
    description="မြန်မာဟင်းလျာများရဲ့ ချက်ပြုတ်နည်းအဆင့်ဆင့်ကို ပြောပြပေးနိုင်သော chatbot။"
)

# Web App ကို စတင် run ခြင်း
iface.launch()
```