

Dự án RAG Retriever + Generator Fine-Tuning

Dưới đây là một **template RAG đầy đủ (Retriever + Generator)** có hỗ trợ **Fine-Tuning cả hai phần: Retriever (bi-encoder) và Generator (T5 hoặc LLaMA)**. Mình sẽ chia thành từng bước rõ ràng:

Cấu trúc thư mục đề xuất

```
rag_finetuning_project/  
├── data/  
│   └── train.jsonl  
├── retriever/  
│   ├── train_retriever.py  
│   └── model/  
├── generator/  
│   ├── train_generator.py  
│   └── model/  
├── rag_pipeline.py  
└── requirements.txt
```

1. Dataset format

`data/train.jsonl`:

```
{"question": "What is the capital of France?", "context": "Paris is the capital of France.", "answer": "Paris"}  
{"question": "What is the currency of Japan?", "context": "The yen is the official currency of Japan.", "answer":  
"yen"}  
...
```

2. Fine-Tune Retriever (`retriever/train_retriever.py`)

Sử dụng `sentence-transformers`.

```

from sentence_transformers import SentenceTransformer, InputExample, losses
from torch.utils.data import DataLoader
import json

# Load data
examples = []
with open("../data/train.jsonl", "r") as f:
    for line in f:
        ex = json.loads(line)
        examples.append(InputExample(texts=[ex["question"], ex["context"]]))

# Load model
model = SentenceTransformer("sentence-transformers/all-MiniLM-L6-v2")
train_dataloader = DataLoader(examples, batch_size=16)
train_loss = losses.MultipleNegativesRankingLoss(model)

# Train
model.fit(train_objectives=[(train_dataloader, train_loss)], epochs=1, output_path="model/")

```

3. Fine-Tune Generator (generator/train_generator.py)

Sử dụng `T5` từ Hugging Face Transformers.

```

from transformers import T5Tokenizer, T5ForConditionalGeneration, Trainer, TrainingArguments
from datasets import load_dataset
import json

def load_data():
    data = []
    with open("../data/train.jsonl", "r") as f:
        for line in f:
            ex = json.loads(line)
            input_text = f"question: {ex['question']} context: {ex['context']}"
            data.append({"input": input_text, "label": ex["answer"]})
    return data

data = load_data()
from datasets import Dataset
dataset = Dataset.from_list(data)

```

```

tokenizer = T5Tokenizer.from_pretrained("t5-small")
model = T5ForConditionalGeneration.from_pretrained("t5-small")

def preprocess(examples):
    model_input = tokenizer(examples["input"], truncation=True, padding="max_length", max_length=512)
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(examples["label"], truncation=True, padding="max_length", max_length=32)
    model_input["labels"] = labels["input_ids"]
    return model_input

tokenized = dataset.map(preprocess, batched=True)

args = TrainingArguments(
    output_dir="model/",
    evaluation_strategy="no",
    per_device_train_batch_size=4,
    num_train_epochs=3,
    save_strategy="epoch",
)

trainer = Trainer(
    model=model,
    args=args,
    train_dataset=tokenized,
    tokenizer=tokenizer,
)

trainer.train()

```

4. Tích hợp Retriever + Generator (`rag_pipeline.py`)

```

from sentence_transformers import SentenceTransformer, util
from transformers import T5Tokenizer, T5ForConditionalGeneration

retriever = SentenceTransformer("retriever/model")
generator = T5ForConditionalGeneration.from_pretrained("generator/model")
tokenizer = T5Tokenizer.from_pretrained("t5-small")

# Fake corpus (ở thực tế sẽ là hàng ngàn documents)
corpus = [

```

```

"Paris is the capital of France.",
"Tokyo is the capital of Japan.",
"The yen is the currency of Japan.",
]

corpus_embeddings = retriever.encode(corpus, convert_to_tensor=True)

def rag_answer(question):
    question_embedding = retriever.encode(question, convert_to_tensor=True)
    top_hit = util.semantic_search(question_embedding, corpus_embeddings, top_k=1)[0][0]
    context = corpus[top_hit['corpus_id']]

    input_text = f"question: {question} context: {context}"
    input_ids = tokenizer(input_text, return_tensors="pt", truncation=True).input_ids
    output_ids = generator.generate(input_ids)
    return tokenizer.decode(output_ids[0], skip_special_tokens=True)

# Ví dụ:
print(rag_answer("What is the capital of France?"))

```

requirements.txt

```

transformers
datasets
sentence-transformers
torch

```

Tùy chọn nâng cao

- **Thay T5 bằng LLaMA/Mistral** với QLoRA → sử dụng `PEFT`.
- **Sử dụng LangChain hoặc Haystack** để xây dựng UI / chatbot interface.
- **Sử dụng FAISS hoặc Weaviate** cho truy vấn triệu tài liệu.

Phiên bản #1

Được tạo 7 tháng 5 2025 06:16:54 bởi Đỗ Ngọc Tú

Được cập nhật 7 tháng 5 2025 06:21:01 bởi Đỗ Ngọc Tú