2025.04.30.(수)

NS3 Code Generator

: LLM 기반의 네트워크 시뮬레이션 코드 자동 생성

202101109 박수화
202102722 최희원

목차

1 간트 차트

2 Fine-Tuning

3 RAG DB 구축 방향성 구상

4 Data Cleaning

5 ns-3

간트 차트

Analyzer line 구현

		3월	4월	5월	6월
3/27 - 4/2	NS3 docs 기반 RAG DB 구축, Fine-Tuning 방법 학습				
4/3 - 4/9	유사도 기반 검색 함수 구현 및 기준 설정				
4/10 - 4/16	중간고사				
4/17 - 4/23	중간고사				
4/24 - 4/30	출력 형식 정리, 모델 Fine-Tuning 1차				
5/1 - 5/7	모델 Fine-Tuning 2차				
5/8 -	Feedback 반영 (부족한 부분 보완 등 개선 작업)				

Fine-Tuning

LoRA 기반 LLaMA 모델 파인튜닝

PEFT (Parameter-Efficient Fine-Tuning)

대규모 언어 모델(LLM)을 학습시킬 때에는 전체 fine-tuning하면 수억~수십억 개의 파라미터를 모두 업데이트해야 하기 때문에 메모리/시간/비용 부담이 크다는 단점 존재

→**PEFT는 모델 전체를 수정하지 않고, 필요한 소수 부분만 미세 조정하는 방법**
따라서, 효율적으로 빠르고 가볍게 fine-tuning 가능하다 !

LoRA (Low-Rank Adaptation)

PEFT 중 하나의 방법으로, **기존 모델의 weight를 직접 수정하지 않고 Low-Rank Matrix 두 개를 추가로 학습**
→ 모델 크기는 유지, 학습할 파라미터 수 감소, 성능은 풀 튜닝과 유사하게 유지할 수 있다 !

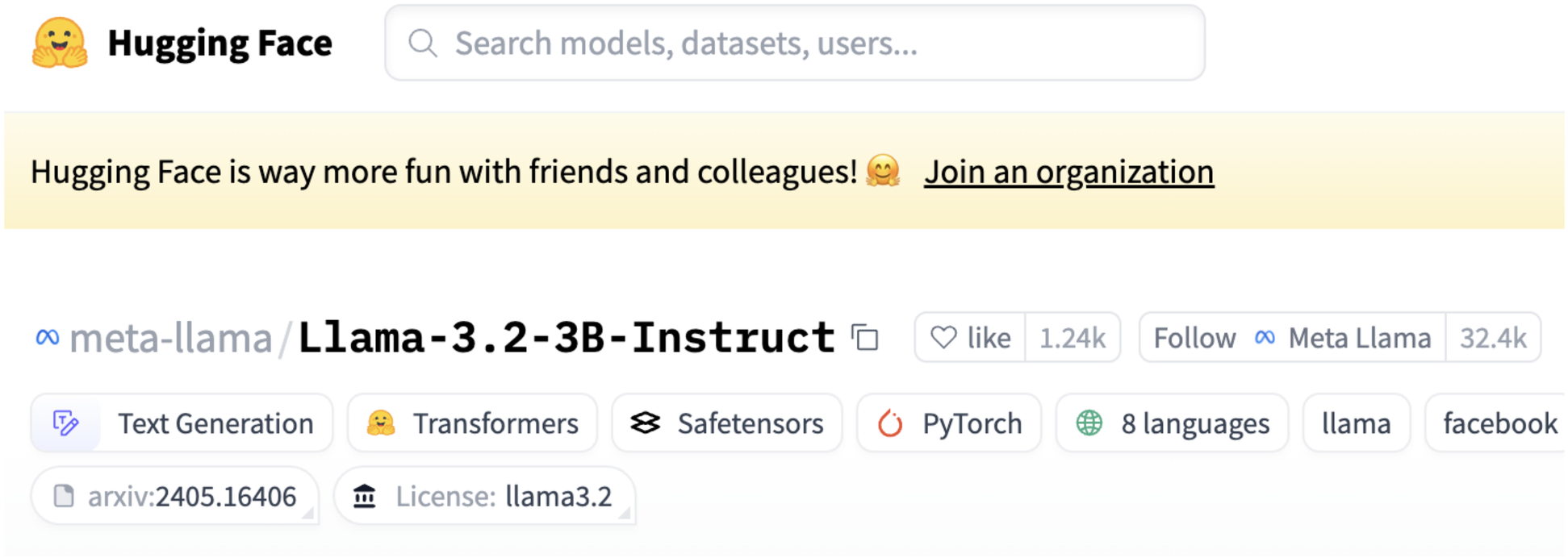
>> LoRA 사용 이유?

LLaMA 같은 대규모 모델 학습 시 비용 절감, 의약품 Dataset과 같은 소규모 도메인 데이터 학습에 적합한 기법이라 판단

Fine-Tuning

LoRA 기반 LLaMA 모델 파인튜닝

> Base Model: meta-llama/Llama-3.2-3B-Instruct



Fine-Tuning

LoRA 기반 LLaMA 모델 파인튜닝

> Datasets: qiaojin/PubMedQA

Datasets: qiaojin / **PubMedQA** like 216

Tasks: Question Answering | Modalities: Text | Formats: parquet | Sub-tasks: multiple-choice-qa | Languages: English | Size: 100K - 1M | ArXiv: arxiv:1909.06146

Libraries: Datasets | pandas | Croissant | +1 | License: mit

Dataset card | Data Studio | Files and versions | Community 4

Dataset Viewer — Auto-converted to Parquet | API | Embed | Data Studio

Subset (3): pqa_artificial · 211k rows

Split (1): train · 211k rows

Search this dataset

pubid int32	question string	context sequence	long_answer string	final_decision string
25,431,941	is methylation of the FGFR2 gene associated…	{ "contexts": ["This study examined links between DNA…	We identified a novel biologically plausibl…	yes
25,432,519	Do tumor-infiltrating immune cell profiles and…	{ "contexts": ["Tumor microenvironment immunity i…	Breast cancer immune cell subpopulation…	yes

Downloads last month 13,121

Use this dataset | Edit dataset card | Papers with Code

Homepage: PubMedQA homepage

Repository: PubMedQA repository

Paper: PubMedQA: A Dataset for Biomedical Research Questio…

Fine-Tuning

LoRA 기반 LLaMA 모델 파인튜닝

fine_tuning.py

```python
from transformers import AutoTokenizer, AutoModelForCausalLM, TrainingArguments,
from datasets import load_dataset
from peft import get_peft_model, LoraConfig, TaskType
import torch

# 1. 모델 이름과 tokenizer 불러오기
model_name = "meta-llama/Llama-3.2-3B-Instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token

# 2. PEFT (LoRA) 구성
lora_config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],  # 주로 attention projection 모듈 지정
    lora_dropout=0.05,
    bias="none",
    task_type=TaskType.CAUSAL_LM
)

# 3. 모델 불러오기 + LoRA 적용
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    torch_dtype=torch.float16,
    trust_remote_code=True
)
model = get_peft_model(model, lora_config)

# 4. 예시용 데이터셋 불러오기 (PubMed QA 사용 예)
dataset = load_dataset("qiaojin/PubMedQA", "pqa_artificial")
dataset = dataset["train"].select(range(1000))  # 예시로 1000개만 사용
```

```python
# 5. 텍스트 전처리 함수 정의
def format_example(example):
    question = example["question"]
    answer = example["final_decision"]
    prompt = f"<|user|>\n{question}\n<|assistant|>\n{answer}"
    return {"text": prompt}

tokenized_dataset = dataset.map(format_example).map(
    lambda x: tokenizer(x["text"], truncation=True, padding="max_length", max_length=512),
    batched=True
)

# 6. 훈련 설정
training_args = TrainingArguments(
    output_dir="./llama3_lora_pubmedqa",
    per_device_train_batch_size=2,
    gradient_accumulation_steps=8,
    num_train_epochs=3,
    learning_rate=2e-4,
    fp16=True,
    logging_steps=10,
    save_steps=200,
    save_total_limit=2,
    report_to="none"
)

# 7. Trainer 구성
data_collator = DataCollatorForLanguageModeling(tokenizer, mlm=False)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator
)

# 8. 학습 시작
trainer.train()

# 학습 끝나고 나서 adapter 저장
model.save_pretrained("./lora_adapter_only")
```

Fine-Tuning

LoRA 기반 LLaMA 모델 파인튜닝

test.py

```python
from transformers import AutoTokenizer, AutoModelForCausalLM
from peft import PeftModel
import torch

base_model = "meta-llama/Llama-3.2-3B-Instruct"
adapter_path = "./lora_adapter_only"

tokenizer = AutoTokenizer.from_pretrained(base_model, trust_remote_code=True)
model = AutoModelForCausalLM.from_pretrained(base_model, torch_dtype=torch.float16, device_map="auto", trust_remote_code=True)
model = PeftModel.from_pretrained(model, adapter_path)

prompt = "타이레놀은 해열 효과가 있나요?"
# prompt = "이부프로펜과 타이레놀을 같이 복용해도 되나요?"
messages = [{"role": "user", "content": prompt}]
text = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)
inputs = tokenizer(text, return_tensors="pt").to(model.device)

output = model.generate(**inputs, max_new_tokens=256)
print(tokenizer.decode(output[0], skip_special_tokens=True))
```

Fine-tuning

LoRA 기반 LLaMA 모델 파인튜닝

▼ **fine-tuning 전 답변**

(psh) gpuadmin@gpuadmin:~/ns3coder/psh$ python llama_3b.py
Loading checkpoint shards: 100%|
[redacted]
[redacted]| 2/2 [00:00<00:00, 3.30it/s]
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
You're doctor. Answer the following question: 이부프로펜과 타이레놀을 같이 복용해도 되나요? (Is it safe to take ibuprofen and tylenol together?)

I'm happy to help you with your question! However, I need to clarify that **I'm not a medical professional**, and it's always best to consult with a doctor or a pharmacist before taking any medication. That being said, I can provide some general information on the

▼ **fine-tuning 후 답변**

```
(psh) gpuadmin@gpuadmin:~/ns3coder/psh$ python test.py
Loading checkpoint shards: 100%|
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
system

Cutting Knowledge Date: December 2023
Today Date: 26 Mar 2025

user

이부프로펜과 타이레놀을 같이 복용해도 되나요?assistant

네, 이부프로펜과 타이레놀은 함께 복용할 수 있습니다.
```

```
(psh) gpuadmin@gpuadmin:~/ns3coder/psh$ python test.py
Loading checkpoint shards: 100%|
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
system

Cutting Knowledge Date: December 2023
Today Date: 26 Mar 2025

user

타이레놀은 해열 효과가 있나요?assistant

yes
```

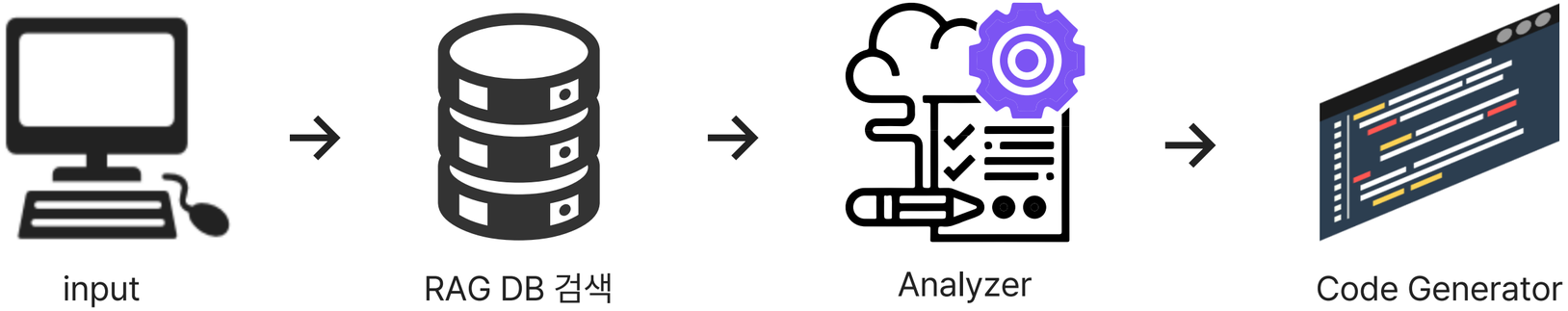
RAG DB 구축 방향성 구상

1. Analyzer 역할: input에 대한 모순 검증 및 추가적인 처리

→ (모순 검증은 되었다고 가정) Wifi(802.11ac) 프로토콜을 사용하고 이동 노드는 5개에 이동 패턴은 RandomWayPoint model, 연결 속도는 5mbps, 지연 속도는 최대 2ns, 애플리케이션은 단순한 Echo Server-Client로 해줘

⇒ Code Generation model에게 넘겨주고 code 생성

2. 구조



Data Cleaning

1. Data curation : 3.40 ns-3 documentation 수집

ns-3 | About | Research | Education | Releases | Documentation | Develop | Support | Search

- Documentation
  - Latest release
  - Older releases
  - Development tree
  - Presentations
  - Videos
- Releases
  - ns-3.44
  - ns-3.43
  - ns-3.42
  - ns-3.41
  - ns-3.40
    - Authors
    - Documentation
    - Download

Home > Releases > ns-3.40 > Documentation

Documentation

- **Tutorial:** an introduction to downloading and building ns-3 and running example programs: **HTML** format, **PDF** format, **HTML single-page** format
- **Installation Guide:** more complete documentation on how to install ns-3 including optional components: **HTML** format, **PDF** format, **HTML single-page** format
- **Manual:** an in-depth coverage of the architecture and core of ns-3: **HTML** format, **PDF** format, **HTML single-page** format
- **Model Library:** documentation on individual protocol and device models that build on the ns-3 core: **HTML** format, **PDF** format, **HTML single-page** format
- **Contributing Guide:** documentation on how to contribute to ns-3: **HTML** format, **PDF** format, **HTML single-page** format
- **Doxygen API Documentation**: Coverage of the C++ APIs using Doxygen.
- Documentation of the Bake build tool
- Release Errata

→ Tutorial, Manual, Model Library

Data Cleaning

2. Data Cleaning: 불필요한 정보 제거

→ 이미지, 표, 수식, 쪽수, 목차, 날짜, 시작부(~6p), 구분선 등

CHAPTER

TWO

ANIMATION

Animation is an important tool for network simulation. While *ns-3* does not contain a default graphical animation tool, we currently have two ways to provide animation, namely using the PyViz method or the NetAnim method. The PyViz method is described in http://www.nsnam.org/wiki/PyViz.

We will describe the NetAnim method briefly here.

2.1 NetAnim

NetAnim is a standalone, Qt5-based software executable that uses a trace file generated during an *ns-3* simulation to display the topology and animate the packet flow between nodes.

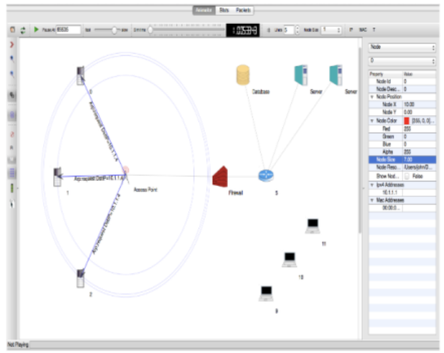


Fig. 1: An example of packet animation on wired-links

In addition, NetAnim also provides useful features such as tables to display meta-data of packets like the image below

5

→

This manual compiles documentation for ns-3 models and supporting software that enable users to construct network
simulations. It is important to distinguish between modules andmodels :
•ns-3 software is organized into separate modules that are each built as a separate software library. Individual
ns-3 programs can link the modules (libraries) they need to conduct their simulation.
•ns-3 models are abstract representations of real-world objects, protocols, devices, etc.
Anns-3 module may consist of more than one model (for instance, the internet module contains models for both
TCP and UDP). In general, ns-3 models do not span multiple software modules, however.
This manual provides documentation about the models of ns-3. It complements two other sources of documentation
concerning models:
• the model APIs are documented, from a programming perspective, using Doxygen. Doxygen for ns-3 models is
available on the project web server.
• the ns-3 core is documented in the developer's manual. ns-3 models make use of the facilities of the core, such
as attributes, default values, random numbers, test frameworks, etc. Consult the main web site to find copies of
the manual.
Finally, additional documentation about various aspects of ns-3 may exist on the project wiki.
A sample outline of how to write model library documentation can be found by executing the create-module.py
program and looking at the template created in the file new-module/doc/new-module.rst .
$ cd src
$ ./create-module.py new-module
The remainder of this document is organized alphabetically by module name.
If you are new to ns-3, you might first want to read below about the network module, which contains some fundamental
models for the simulator. The packet model, models for different address formats, and abstract base classes for objects
such as nodes, net devices, channels, sockets, and applications are discussed there.
Animation is an important tool for network simulation. While ns-3 does not contain a default graphical animation
tool, we currently have two ways to provide animation, namely using the PyViz method or the NetAnim method. The
PyViz method is described in http://www.nsnam.org/wiki/PyViz.
We will describe the NetAnim method briefly here.
NetAnim is a standalone, Qt5-based software executable that uses a trace file generated during an ns-3 simulation to
display the topology and animate the packet flow between nodes.
In addition, NetAnim also provides useful features such as tables to display meta-data of packets like the image below
A way to visualize the trajectory of a mobile node
A way to display the routing-tables of multiple nodes at various points in time
A way to display counters associated with multiple nodes as a chart or a table
A way to view the timeline of packet transmit and receive events
The class ns3::AnimationInterface is responsible for the creation the trace XML file. AnimationInterface uses the
tracing infrastructure to track packet flows between nodes. AnimationInterface registers itself as a

Data Cleaning

2. Data Cleaning

data_cleaning.py

```python
import re
import textwrap
import unicodedata
import os
from PyPDF2 import PdfReader
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4

# 유니코드 정규화: 특수 문자 → 일반 문자
def normalize_text(text):
    return unicodedata.normalize('NFKC', text)

# 불필요한 줄 제거 규칙
def is_useless_line(line):
    line = line.strip()
    if re.search(r"\s+\d{1,3}$", line):
        return True
    return (
        re.match(r"^(one|two|three|four|five|six|seven|eight|nine|ten)\b", line.lower()) or
        re.match(r"^\d+(\.\d+)+\s+[A-Z].*", line) or
        line.isupper() and len(line.split()) <= 3 or
        line.lower().startswith("fig.") or
        line.lower().startswith("table") or
        "ns-3 model library" in line.lower() or
        "chapter" in line.lower() or
        "contents" in line.lower() or
        re.match(r"^\s*\d+\s*$", line) or
        re.match(r"^\s*sep\s+\d{1,2},\s+\d{4}\s*$", line.lower())
    )

# 단일 PDF 처리 함수
def clean_pdf(input_path, output_path):
    reader = PdfReader(input_path)
    cleaned_lines = []

    for i, page in enumerate(reader.pages):
        if i < 6:
            continue
        text = page.extract_text()
        if text:
            lines = text.split("\n")
            cleaned = [
                normalize_text(line.strip())
                for line in lines
                if line.strip() and not is_useless_line(line)
            ]
            cleaned_lines.extend(cleaned + [""])
```

```python
    # 새 PDF 파일로 작성
    c = canvas.Canvas(output_path, pagesize=A4)
    width, height = A4
    x_margin, y_margin = 40, 40
    y = height - y_margin
    c.setFont("Helvetica", 11)

    for line in cleaned_lines:
        wrapped_lines = textwrap.wrap(line, width=100)
        for wline in wrapped_lines:
            if y < y_margin:
                c.showPage()
                c.setFont("Helvetica", 11)
                y = height - y_margin
            c.drawString(x_margin, y, wline)
            y -= 15

    c.save()
    print(f"Cleaning completed: {output_path}")

# 폴더 전체 처리
input_dir = "/home/gpuadmin/ns3coder/psh/ns3.40_docs"
cleaned_dir = input_dir + "_cleaned"
os.makedirs(cleaned_dir, exist_ok=True)

# 모든 PDF 파일에 대해 반복 처리
for file in os.listdir(input_dir):
    if file.lower().endswith(".pdf"):
        input_path = os.path.join(input_dir, file)

        # 파일명 분리해서 _cleaned 붙이기
        name, ext = os.path.splitext(file)
        cleaned_filename = name + "_cleaned" + ext
        output_path = os.path.join(cleaned_dir, cleaned_filename)

        # PDF 정제 실행
        clean_pdf(input_path, output_path)
```

Data Cleaning

2. Data Cleaning

```python
# Step 1. Knowledge Base 구축
import os
import warnings
from langchain_community.document_loaders import PyMuPDFLoader
from PyPDF2 import PdfMerger
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_community.vectorstores import Chroma
from langchain.chains import RetrievalQA
from langchain_community.llms import HuggingFacePipeline

warnings.filterwarnings("ignore")

# 여러 PDF를 하나로 합치기 (./ns3coder/ns3.40_docs)
output = '/home/gpuadmin/ns3coder/psh/data_cleaning/O/model-library_clean_complete.pdf'

if os.path.exists('/home/gpuadmin/ns3coder/psh/data_cleaning/O/rlibDB') :
    print('A RAG DB already exists. Exist DB will be loaded.')
    # 임베딩 모델 초기화 (VectorDB 생성 시 사용한 설정과 동일하게)
    embeddings = HuggingFaceEmbeddings(
        model_name='BAAI/bge-m3',
        model_kwargs={'device':'cuda'},
        encode_kwargs={'normalize_embeddings': True},
    )

    # 저장된 Chroma VectorDB 경로 설정
    vectorstore_path = '/home/gpuadmin/ns3coder/psh/data_cleaning/O/rlibDB'

    # 저장된 VectorDB 불러오기
    vectorstore = Chroma(persist_directory=vectorstore_path, embedding_function=embeddings)
else :
    if os.path.exists(output):
        print('A merged PDF already exists.')
    else:
        pdf_folder = '/home/gpuadmin/ns3coder/psh/data_cleaning/O'  # PDF들이 들어 있는 폴더 경로
        pdf_files = [os.path.join(pdf_folder, file) for file in os.listdir(pdf_folder) if file.endswith('.pdf')]

        merger = PdfMerger()

        for pdf in pdf_files:
            merger.append(pdf)

        merger.write(output)
        merger.close()
        print('PDF was merged.')
```

cleaning_rag.py

```python
    #NS 3.40 Docs pdf를 PyMuPDFLoader를 이용해 로드
    loader = PyMuPDFLoader(output)
    pages = loader.load()

    #Docs를 문장으로 분리 (Chunk size : 500, 각 청크의 50자씩은 겹치게 나눌 것임(overlap))
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=500,
        chunk_overlap=50,
    )
    docs = text_splitter.split_documents(pages)

    # Embedding
    embeddings = HuggingFaceEmbeddings(
        model_name='BAAI/bge-m3',
        model_kwargs={'device':'cuda'},
        encode_kwargs={'normalize_embeddings':True},
    )

    # Chroma Vector DB 구축
    vectorstore = Chroma.from_documents(docs, embeddings)
    ## 경로 설정
    vectorstore_path = '/home/gpuadmin/ns3coder/psh/data_cleaning/O/rlibDB'
    os.makedirs(vectorstore_path, exist_ok=True)
    ### Vector DB 생성 및 저장
    vectorstore = Chroma.from_documents(docs, embeddings, persist_directory=vectorstore_path)
    vectorstore.persist()
    print("Vector DB created and saved.")

# Step 2. Retriever 정의
retriever = vectorstore.as_retriever(search_kwargs={'k':3})
```

Data Cleaning

2. Data Cleaning

```python
# Step 3. LangChain & Generate
from transformers import AutoTokenizer, AutoModelForCausalLM, TextStreamer, BitsAndBytesConfig
import torch
import transformers

torch.cuda.empty_cache()

quantization_config = BitsAndBytesConfig(
    load_in_8bit = True
)

model_id = "meta-llama/Llama-3.3-70B-Instruct"

tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    quantization_config=quantization_config,
    device_map="auto"
)
print(f"{model_id} loaded sucessfully.")
pipe = transformers.pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    max_new_tokens=5000,
    do_sample=True,
    temperature=0.7
)

# LangChain에서 사용할 LLM 인터페이스로 변환
llm = HuggingFacePipeline(pipeline=pipe)

qa_chain = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",  # "stuff" 방식은 검색된 문서 전체를 하나의 prompt에 넣어 처리
    retriever=retriever
)
```

cleaning_rag.py

```python
# 예시 질의
input_text = "What is the purpose of the NetAnim tool in the NS-3 model library, and how is it typically used in simulations?"
input_prompt = f"Let's think step by step. You're NS3 coder. {input_text}"
answer = qa_chain.run(input_prompt)

print("Answer:", answer)
```

Data Cleaning

2. Data Cleaning - 결과 비교 (근거)

Data Cleaning 적용 전

▼ 근거

CHAPTER
TWO
ANIMATION
Animation is an important tool for network simulation. While ns-3 does not contain a default graphical animation
tool, we currently have two ways to provide animation, namely using the PyViz method or the NetAnim method. The
PyViz method is described in http://www.nsnam.org/wiki/PyViz.
We will describe the NetAnim method briefly here.
2.1 NetAnim
NetAnim is a standalone, Qt5-based software executable that uses a trace file generated during an ns-3 simulation to

ns-3 Model Library, Release ns-3-dev
2.1.3 Building NetAnim
Prerequisites
Qt5 (5.4 and over) is required to build NetAnim. The ns-3 Installation Guide lists some packages to install for some
Linux systems, for macOS https://www.nsnam.org/docs/installation/html/macos.html#optional, and for Windows.
The Qt site also provides download options.
Build steps
To build NetAnim use the following commands:
$ cd netanim
$ make clean
$ qmake NetAnim.pro
$ make

ns-3 Model Library, Release ns-3-dev

1. Initial boilerplate:
   #include <ns3/core-module.h>
   #include <ns3/network-module.h>
   #include <ns3/mobility-module.h>
   #include <ns3/lte-module.h>
   using namespace ns3;
   int main(int argc, char *argv[])
   {
   // the rest of the simulation program follows
2. Create an LteHelper object:
   Ptr<LteHelper> lteHelper = CreateObject<LteHelper>();
   This will instantiate some common objects (e.g., the Channel object) and provide the methods to add eNBs and

Data Cleaning 적용 후

▼ 근거

We will describe the NetAnim method briefly here.
NetAnim is a standalone, Qt5-based software executable that uses a trace file generated during an
ns-3 simulation to
display the topology and animate the packet flow between nodes.
In addition, NetAnim also provides useful features such as tables to display meta-data of packets
like the image below
A way to visualize the trajectory of a mobile node
A way to display the routing-tables of multiple nodes at various points in time

classes for objects
such as nodes, net devices, channels, sockets, and applications are discussed there.
Animation is an important tool for network simulation. While ns-3 does not contain a default graphical animation
tool, we currently have two ways to provide animation, namely using the PyViz method or the NetAnim
method. The
PyViz method is described in http://www.nsnam.org/wiki/PyViz.
We will describe the NetAnim method briefly here.

later section. It is
important to note that AnimationInterface records a packet only if the rx trace hooks are called.
Every tx event must
be matched by an rx event.
If NetAnim is not already available in the ns-3 package you downloaded, you can do the following:
The latest version of NetAnim can be downloaded using git with the following command:
$ git clone https://gitlab.com/nsnam/netanim.git
Prerequisites

Data Cleaning

2. Data Cleaning - 결과 비교 (답변)

Data Cleaning 적용 전

▼ 답변

▼ 영어 ver

NetAnim is a tool for visualizing simulations in NS-3. It reads a trace file generated during a simulation and provides a graphical representation of the simulation. It is typically used to visualize the movement of nodes, the transmission of packets, and other simulation events. This allows users to better understand the behavior of their simulation and identify potential issues or areas for improvement. In the context provided, NetAnim is built using Qt5 and can be used in conjunction with other NS-3 tools and methods for analyzing and visualizing simulation results.

▼ 한글 ver

NetAnim은 NS-3에서 시뮬레이션을 시각화하기 위한 도구입니다. 이 도구는 시뮬레이션 중에 생성된 추적 파일을 읽고 시뮬레이션의 그래픽 표현을 제공합니다. 일반적으로 노드의 움직임, 패킷 전송 및 기타 시뮬레이션 이벤트를 시각화하는 데 사용됩니다. 이를 통해 사용자는 시뮬레이션의 동작을 더 잘 이해하고 잠재적인 문제나 개선 사항을 식별할 수 있습니다. 제공된 컨텍스트에서 NetAnim은 Qt5를 사용하여 구축되었으며, 시뮬레이션 결과를 분석하고 시각화하기 위한 다른 NS-3 도구 및 방법과 함께 사용할 수 있습니다.

Data Cleaning 적용 후

▼ 답변

▼ 영어 ver

NetAnim is a visualization tool used to display the topology and animate the packet flow between nodes in an ns-3 simulation. It is typically used to visualize the trajectory of mobile nodes, display routing tables of multiple nodes at various points in time, and provide tables to display meta-data of packets. NetAnim is not a built-in feature of ns-3 but can be downloaded separately and used to provide a graphical representation of simulation results. It relies on trace files generated during an ns-3 simulation to function. The primary purpose of NetAnim is to facilitate understanding and debugging of network simulations by providing a visual representation of packet flow and network behavior.

▼ 한글 ver

NetAnim은 ns-3 시뮬레이션에서 노드 간의 토폴로지를 표시하고 패킷 흐름을 애니메이션화하는 데 사용되는 시각화 도구입니다. 일반적으로 모바일 노드의 궤적을 시각화하고, 여러 시점에서 여러 노드의 라우팅 테이블을 표시하며, 패킷의 메타데이터를 표시하는 테이블을 제공하는 데 사용됩니다. NetAnim은 ns-3의 내장 기능은 아니지만, 별도로 다운로드하여 시뮬레이션 결과를 그래픽으로 표현하는 데 사용할 수 있습니다. NetAnim은 ns-3 시뮬레이션 중에 생성된 추적 파일에 의존하여 작동합니다. NetAnim의 주요 목적은 패킷 흐름과 네트워크 동작을 시각적으로 표현하여 네트워크 시뮬레이션의 이해와 디버깅을 용이하게 하는 것입니다.