자연어처리 기술을 활용한 도메인 특화 언어모델 개발

2025-1 파란학기

지도 교수님 : 김영민 교수님
팀원 : 공종혁, 윤동현, 남현원, 한민규, 이승환, 정다훈
팀명 : 도해

목차

Part 1

선정 도메인

인공지능 관련 논문 이해 및 정리 지원 도메인

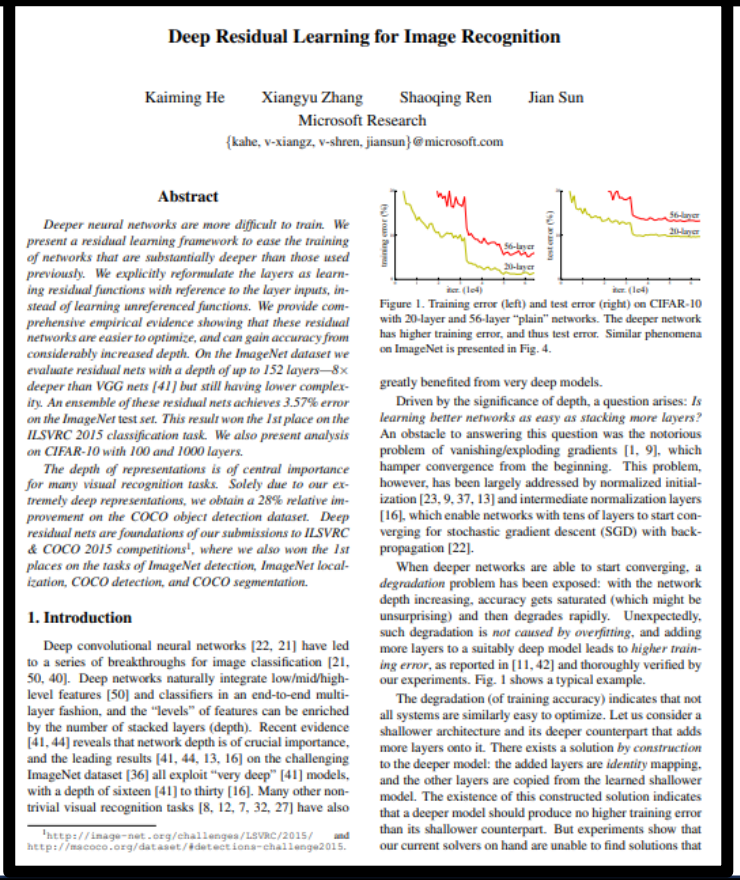
목적

- ✓ 논문을 보다 쉽게 이해할 수 있도록 주제별 기초 개념부터 최신 연구 흐름까지 아우르는 학습 루트를 체계적으로 제시
- ✓ 방대한 논문을 처음부터 끝까지 모두 읽지 않더라도, 핵심 아이디어와 기여점을 빠르게 파악할 수 있는 논문 요약 제공
- ✓ 논문 이해에 도움을 줄 수 있는 관련 연구, 배경 지식, 참고 논문 및 리뷰 논문까지 함께 추천하여 학습의 확장성 강화

Target

- ✓ 논문을 처음 접하는 대학생 및 대학원 진학 예정자 등, 논문 읽기에 어려움을 느끼는 학습 초심자
- ✓ 효율적인 시간 활용을 위해 논문의 핵심 내용을 빠르게 정리하고 싶은 사람, 논문을 바탕으로 과제나 발표 준비가 필요한 학부생/연구자
- ✓ 특정 논문이나 주제를 깊이 있게 이해하기 위해, 관련 기초 이론부터 실전 논문까지 체계적인 학습 경로가 필요한 학습자

예시 논문
Deep Residual Learning for Image Recognition



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1], where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

1. Introduction

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 50, 40]. Deep networks naturally integrate low/mid/high-level features [50] and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). Recent evidence [41, 44] reveals that network depth is of crucial importance, and the leading results [41, 44, 13, 16] on the challenging ImageNet dataset [36] all exploit "very deep" [41] models, with a depth of sixteen [41] to thirty [16]. Many other non-trivial visual recognition tasks [8, 12, 7, 32, 27] have also

[1]http://image-net.org/challenges/LSVRC/2015/ and http://mscoco.org/dataset/#detections-challenge2015.

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [1, 9], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 13] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more layers to a suitably deep model leads to *higher training error*, as reported in [11, 42] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

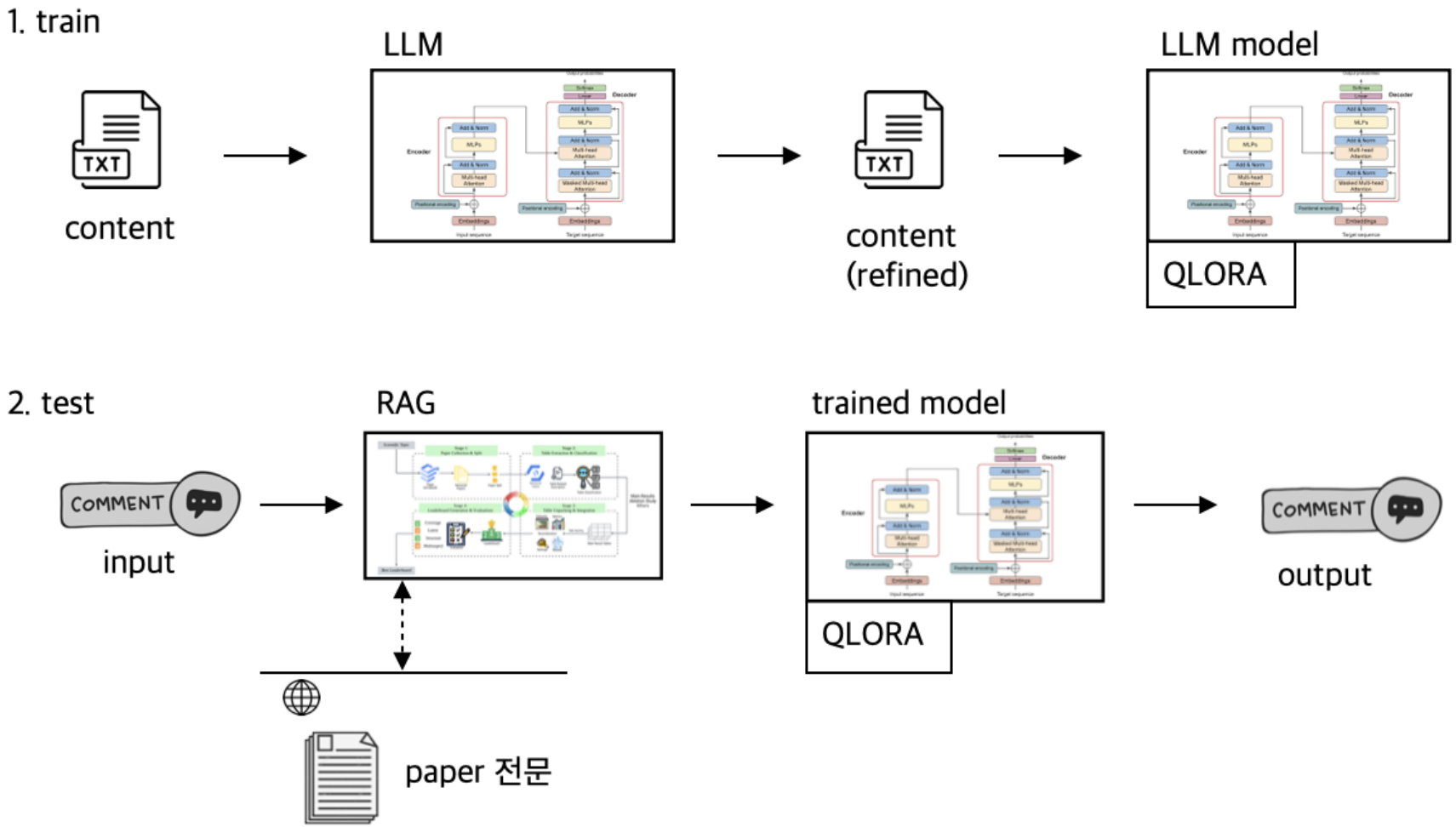
The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution *by construction* to the deeper model: the added layers are *identity* mapping, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that

- ✓ 해당 논문에서 Reference 한 논문의 수 : 50
- ✓ 해당 논문을 인용한 논문의 수 : 269230

Part 2

Whole Process

인공지능 논문 관련 이해 및 정리 지원 도메인 특화 언어 모델 개발 과정



Part 3

데이터 수집 | Semantic Scholar API

✓ **Semantic Scholar API를 활용하여 유명 인공지능 학회(CVPR, ICCV, ECCV) 에 속해 있는 논문 정보 수집**

Semantic Scholar API

Semantic Scholar API는 논문의 제목, 요약, 저자, 인용 수 같은 정보를 프로그램이 자동으로 가져올 수 있게 해주는 도구

SEMANTIC SCHOLAR

API 활용 방법

- 논문 검색 keyword 지정
  ex) `"image generation", "GAN", "StyleGAN"`
- 지정 학회 인지 filtering
  ex) `"Computer Vision and Pattern Recognition"`
- 일정 Citation이 넘는 논문인지 filtering
  ex) `paper["citationCount"] > 5`


```
{
    "title": "CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification",
    "authors": [
        {
            "authorId": "48239920",
            "name": "Chun-Fu Chen"
        },
        {
            "authorId": "33421444",
            "name": "Quanfu Fan"
        },
        {
            "authorId": "1819152",
            "name": "Rameswar Panda"
        }
    ],
    "abstract": "The recently developed vision transformer (ViT) has achieved promising results on
    "citationCount": 1429,
    "year": 2021,
    "venue": "IEEE International Conference on Computer Vision",
    "fieldsOfStudy": [
        "Computer Science"
    ],
    "externalIds": "2103.14899"
},
```

Part 3

데이터 수집 | arXiv API

✓ **Semantic Scholar API에 있는 논문 고유 번호를 통해 arXiv API를 통해 논문 원본 source를 획득**

Semantic Scholar API

arXiv API는 논문 정보를 찾을 수 있을 뿐 아니라, 논문의 원본 파일(LaTeX 형식)도 자동으로 가져올 수 있어 연구 자료 수집에 유용

arXiv API는 학회 필터링에 문제가 있음
따라서, Semantic Scholar API를 통해 학회 필터링 후 수집한 논문 고유 번호를 통해 arXiv API 활용

API 활용 방법

- 논문 고유 번호를 통해 논문 검색
  ex) `url = f"https://arxiv.org/e-print/{paper_id}"`
- 논문 원문 Source 파일 Download
  ex)

```
with open(f"{paper_id}.tar.gz", "wb") as f:
    f.write(response.content)
```

논문 Latex 파일

```
\begin{table}[!ht]
\centering
\caption{Precision and Recall score per model. \textbf{Bold} indicates
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{|c|c|c|}
\hline
\textbf{Model} & \textbf{Precision} & \textbf{Recall} \\
\hline
GraspCheckNet & 0.678 & 0.749\\
GPT-4o        & \textbf{0.739}& \textbf{0.95}\\
Llama 3.2     & \underline{0.357}& \underline{0.513}\\
\hline
\end{tabular}
\label{tab:model_metrics}
\end{table}

\subsection{Visual Question Answering}
In order to establish a baseline to which compare our GraspCheckNet mod
Our goal is to leverage their state-of-the-art performance in visual re


We follow a visual-question-answering approach in which an LLM is prompt
The same prompt is used for all instances and no concrete information a
We evaluate two LLMs to compare the effects of the model size and whethe
GPT4-o is tested using OpenIA's API while Llama 3.2 Vision is used thro
We use both the state of the art GPT4-o, which is closed-source and has
This latter model can be run in consumer devices with approximately 6GB
```

Part 3

데이터 전처리 | .Latex → .txt

✓ 모델이 논문의 내용을 잘 이해하려면 복잡한 LaTeX 형식을 단순한 텍스트로 바꿔주는 전처리가 필요

.latex file → .txt file

LaTeX 원문을 그대로 학습하기 어려운 이유

1. ₩section{}, ₩begin{equation} 같은 명령어와 태그로 구성되어 있음
2. 자연어가 아닌 수학 수식 ($E = mc^2$), 표 형식, 인용 정보 등은 모델이 문맥 파악이나 의미 분석하기 어렵게 함
3. 줄 바꿈, 주석, 특수문자 등이 모델 입자에서 Noise로 작용하여 성능을 저하시킬 수 있음

Pandoc 문서 변환 프로그램을 통해 변환

- Pandoc은 문서를 다른 형태로 바꿔주는 변역기 같은 역할을 하며, 복잡한 논문 파일을 모델이 이해할 수 있는 txt로 바꿔줌

Code)

```
tex_path = all_tex_files[0]
txt_path = tex_path.replace(".tex", ".txt")
os.system(f'pandoc "{tex_path}" -t plain -o "{txt_path}"')
```

.tex file

```
\section{INTRODUCTION}
% need to add citations from the papers on the cha
Deformable object manipulation is a growing field of
Deformable objects are a common occurrence in both in
Their deformation and varying response to traditional

One critical aspect of deformable object manipulation

In this context, computer vision has emerged as a pro
These approaches use vision in combination with other
However, most proposed methods focus on the grasping
These constraints and their complexity make these mod


This paper explores the application of computer visio
\begin{enumerate}
\item We introduce a two-stage vision-based grasp ver
\item We present HSR-GraspSynth, a synthetic dataset
\item We investigate the integration of Multimodal La
\end{enumerate}

%Our approach, which can be easily adapted to differe
%With this goal, we introduce a synthetic dataset des

% Here on the results
```

.txt file

```
INTRODUCTION

Deformable object manipulation is a growing field o
robotics due to its relevance in a wide range of ta
objects are a common occurrence in both industrial
environments, and their manipulation poses challeng
rigid objects. Their deformation and varying respons
force and tactile sensing methods during the graspi
significant uncertainty, making it a more challengi

One critical aspect of deformable object manipulati
verification of successful grasping. Traditional me
rely on the object's geometry and force and tactile
account for the deformation of the object and its l
structure and resistance . This requires the use of
sensors and control algorithms, which are often rob
specific.

In this context, computer vision has emerged as a p
address these challenges. Various methods have been
and 3D vision during the grasping process for tasks
cloth manipulation . These approaches use vision in
other input modalities such as tactile sensing to e
deformation during the grasping procedure. However,
methods focus on the grasping control feedback and
specific. These constraints and their complexity ma
unsuitable for the task of verifying a successful g
```

Part 3

데이터 전처리 | 질문/답 형식의 데이터 셋 만들기

✓ **Deepseek오픈 LLM 모델을 활용하여 논문 내용을 토대로 {질문-답} 형태의 데이터 셋 생성**

```python
def call_deepseek_api(title, content, question, api_key):
    url = "https://api.deepseek.com/v1/chat/completions"

    headers = {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

    prompt = f"논문 제목: {title}\n\n논문 내용:\n{content}\n\n질문: {question}"

    data = {
        "model": "deepseek-chat",
        "messages": [
            {"role": "system", "content": "당신은 논문을 한국어로 요약해주는 전문가입니다. 반드시 모든 답변은 한국어로 작성해주세요."},
            {"role": "user", "content": prompt}
        ],
        "temperature": 0.5,
        "max_tokens": 1024
    }
```

DeepseekAPI를 활용

Input : txt파일로 변환한 논문 내용

Output : {질문- 답}

각 논문에서 생성된 {질문-답}쌍을 논문 내용과 함께 저장

```
{
  "title": "Residual Attention Network for Image Classification",
  "content": "Introduction\n\n[image]\n\nNot only a friendly face but also red color will draw our atten
  "question": "이 논문의 핵심 기여는 무엇인가요?",
  "answer": "이 논문의 핵심 기여는 다음과 같습니다:\n\n1. **Residual Attention Network 제안**: 이미지 분류
},
```

Output 예시)
예시 논문 : Residual Attention Network for Image Classification

Part 3

데이터 전처리 | 질문/답 형식의 데이터 셋 만들기

✓ **Deepseek오픈 LLM 모델을 활용하여 논문 내용을 토대로 {질문-답} 형태의 데이터 셋 생성**

```python
def call_deepseek_api(title, content, question, api_key):
    url = "https://api.deepseek.com/v1/chat/completions"

    headers = {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

    prompt = f"논문 제목: {title}\n\n논문 내용:\n{content}\n\n질문: {question}"

    data = {
        "model": "deepseek-chat",
        "messages": [
            {"role": "system", "content": "당신은 논문을 한국어로 요약해주는 전문가입니다. 반드시 모든 답변은 한국어로 작성해주세요."},
            {"role": "user", "content": prompt}
        ],
        "temperature": 0.5,
        "max_tokens": 1024
    }
```

DeepseekAPI를 활용

Input : txt파일로 변환한 논문 내용

Output : {질문- 답}

각 논문에서 생성된 {질문-답}쌍을 논문 내용과 함께 저장

```
{
  "title": "Residual Attention Network for Image Classification",
  "content": "Introduction\n\n[image]\n\nNot only a friendly face but also red color will draw our atten
  "question": "이 논문의 핵심 기여는 무엇인가요?",
  "answer": "이 논문의 핵심 기여는 다음과 같습니다:\n\n1. **Residual Attention Network 제안**: 이미지 분류
},
```

Output 예시)
예시 논문 : Residual Attention Network for Image Classification

Part 4

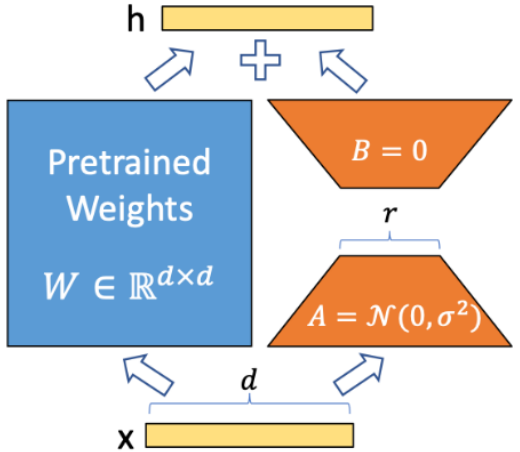
구현 과정 / 오픈 LLM 모델 Fine tuning

- ✓ **qLoRA 학습을 통해 모델의 메모리 사용을 줄이고 효율적으로 fine tuning**
- ✓ **Teacher Forcing Learning을 통해 오류 누적을 막는다.**

qLoRA

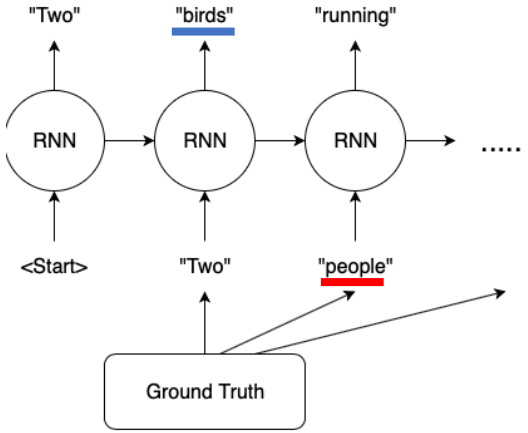
LoRA(Low-Rank Adaptation):
기존 모델의 큰 파라미터를 전부 학습시키지 않고, 작은 추가 모듈(Low-Rank Adapter)만 학습하는 방법

qLoRA:
LoRA에 4비트(저정밀도) 양자화를 결합 → GPU 메모리 사용량 확 줄임
즉, 저렴한 환경에서도 거대 모델 튜닝 가능



Teacher Forcing

모델 output 대신 정답 토큰(ground truth)을 다음 입력으로 사용
(모델이 틀리든 맞든 다음 입력은 무조건 정답으로
→ 초기 학습 시 모델이 잘 못할 때 오류 누적을 맞기 위해서이다)



deepseek

deepseek-ai/**DeepSeek-R1-Distill-Llama-8B**

DeepSeek-R1이 학습한 추론 능력을 LLaMA-3.1-8B 같은 작은 모델에 이식(distill)
큰 모델이 잘 푸는 문제들을 작은 모델이 모방 학습함

원본 LLaMA보다 훨씬 강한 추론 능력을 갖지만,
메모리/속도는 LLaMA-8B 수준

제약적인 환경을 고려하여
효율적인 모델을 가져와
효율적인 Fine tuning

hwnam1129/**deepseek-qlora-paper-search**

Transformers | Safetensors | arxiv:1910.09700

Part 4

구현 과정 / RAG

✓ **Inference 시, 2개의 단계로 나누어서 모델의 부족한 부분을 보완**

Inference 1단계

논문 검색과 **RAG**에 쓸 2가지 쿼리를 생성

생성 방법
기존 사용하는 모델을 사용하여 두 쿼리를 zero-shot으로 생성
별도 fine tuning 없이 기존 모델의 자연어 처리 능력만으로 두 형태의 쿼리 생성

구분	목적	형태
ArXiv API용 쿼리	논문을 가져오기 위한 **검색 쿼리**	**짧고 핵심적인 키워드** (ex: "Vision Transformer ViT")
RAG용 쿼리	LLM이 논문과 유사도를 비교할 **자연어 문장**	**자연스러운 질문 형태** (ex: "What are the architectural advantages of ViT models?")

Inference 2단계

1. 논문 검색 (Arxiv API 사용)

- 1단계에서 만든 Arxiv 쿼리(짧은 키워드)를 넣어서 논문 검색
- (ex: "Vision Transformer ViT")
- 최대 3~5개의 논문만 선택 (속도 고려)

2. 논문 임시 벡터화 (SPECTER 사용)

- 검색된 논문들을 SPECTER라는 문서 임베딩 모델을 이용해 벡터로 변환
- 이 벡터들은 임시 벡터 데이터베이스(DB)에 저장됨

3. RAG 쿼리로 top-k 유사 논문

- 1단계에서 만든 RAG 쿼리를 SPECTER vector DB에 넣고
- 가장 유사한 논문 3~5개를 top-k로 검색

4. 최종 답변 생성 (LLM)

➢ **더 자세하고 보완된 답변 사용자에게 제공**

Part 5

서버

✓ **Flask를 사용해서 AWS 서버에 모델을 배포**

```python
app = Flask(__name__)

# 모델 로드
model_name = "hwnam1129/deepseek-qlora-paper-search"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

@app.route("/generate", methods=["POST"])
def generate():
    data = request.get_json()
    prompt = data.get("prompt", "")
    inputs = tokenizer(prompt, return_tensors="pt")
    outputs = model.generate(**inputs, max_new_tokens=100)
    result = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return jsonify({"response": result})

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

감사합니다.