## 1. 라이브러리 및 데이터

- **Python 3.10**, **CUDA 12.1**, **A100 GPU**사용 (RTX 4090에서 구동을 위해 VRAM 사용량이 24GB를 넘지 않도록 설정하였음)
- **Hugging Face 라이브러리** 사용: transformers, datasets, peft
- 효율적인 **QLoRA 학습**을 위해 **unsloth 라이브러리**사용
- **ChatGPT-4o API**를 이용하여 **500개 대화(총 2500개의 문제)**를 증강하여 데이터셋으로 활용

## 2. 데이터 전처리

- 모델 학습 과정에서 대화 및 선택지 부분에 대한 전처리는 별도로 수행하지 않았음.
- 대화와 선택지를 모델에 구조화하여 입력하기 위해 사용하는 시스템 프롬프트와 인스트럭션 프롬프트는 아래의 목록을 사용함.
- 시스템 프롬프트는 대화형 모델 학습 시 사용된 것을 활용하였으며, 시스템 프롬프트가 없는 모델의 경우 인스트럭션 프롬프트만 사용하거나 자체적으로 작성하여 사용함.

- 인스트럭션 프롬프트의 경우, 베이스라인 코드에서 주어진 형식 이외에도 다양한 양식을 작성해 사용함:

```python
class Prompts:
  # allganize/Llama-3-Alpha-Ko-8B-Instruct
  LLAMA_ALPHA = '''당신은 인공지능 어시스턴트입니다. 묻는 말에 친절하고 정확하게 답변하세요.'''

  # yanolja/EEVE-Korean-Instruct-10.8B-v1.0
  EEVE = '''You are a helpful assistant.'''

  # Qwen/Qwen2-7B-Instruct
  QWEN2 = '''<|im_start|>system\nYou are a helpful assistant.<|im_end|>\n'''

  #davidkim205/Ko-Llama-3-8B-Instruct
  LLAMA3_DAVID = '''You are a helpful assistant.'''

  KIHOON_CUSTOM ='''주어진 대화를 읽고 자세하게 분석한 다음 논리적인 근거를 생각한 뒤 답변해주세요.'''
  KO_GEMMA = '''당신은 질문에 대해서 자세히 설명하는 AI입니다.'''
  YI_KO = '''친절한 챗봇으로서 상대방의 요청에 최대한 자세하고 친절하게 답하자. 모든 대답은 한국어(Korean)으로 대답해줘.'''
  EXAONE = '''You are EXAONE model from LG AI Research, a helpful assistant.'''
  # 기본 모델(blossom 8b)
  DEFAULT ="""You are a helpful AI assistant. Please answer the user's questions kindly. 당신은 유능한 AI 어시스턴트 입니다. 사용자의 질문에 대해 친절하게 답변해주세요."""
  # no system prompt
  NO = None

  @classmethod
  def get_prompt(cls, model_name):
      model_name = model_name.upper()
      if hasattr(cls, model_name):
          return getattr(cls, model_name)
      else:
          return cls.DEFAULT
```

그림 1. 학습에 사용되는 시스템 프롬프트, 인스트럭션 프롬프트 예시(출처:./src/prompt.py)

```python
def question_template(inp,custom_template):
    if custom_template=='default':
        question = f"[Question]\n위 대화의 {inp['category']}"
        if (ord(inp['category'][-1]) - ord("가")) % 28 > 0:
            question += "으로"
        else:
            question = "로"
        question += " 올바른 지문은?"
        return question

    elif custom_template=='skip_category':
        question = f"[Question]\n"
        question += "위 대화에서 알 수 있는 사실로 올바른 것은?"
        return question

    elif custom_template=='chain_of_thought':
        question = f"[Question]\n"
        question += "다음 단계를 따라 문제를 해결해 주세요:\n"
        question += "1. 대화의 주요 내용을 간단히 요약하세요.\n"
        question += "2. 대화에서 언급된 중요한 사실들을 나열하세요.\n"
        question += "3. 각 사실의 신뢰성과 관련성을 평가하세요.\n"
        question += "4. 가장 신뢰할 수 있고 관련성 높은 사실을 선택하세요.\n"
        question += "5. 선택한 사실이 왜 가장 적절한 답변인지 설명하세요.\n"
        question += "위의 단계를 따라 분석한 후, 최종적으로 다음 질문에 답하세요: 위 대화에서 알 수 있는 사실로 가장 올바른 것은 무엇인가요?"
        return question

    else :
        raise KeyError
```

그림 2. 추가로 사용한 인스트럭션 프롬프트 예시(출처:src/data.py)

- 인과추론의 유형을 언어 모델에게 제공하지 않는 프롬프트(skip_category)의 경우 기본 프롬프트보다 안 좋은 성능을 기록하는 것을 확인함.
- **모델이 순차적으로 판단할 수 있도록 가이드라인을 제시**하는 '**CoT**(Chain of Thought)' 프롬프트[1]의 경우 모델에 따라 더 좋은 성능을 기록하는 경우가 있어 해당 과제에 활용함.

- 충분한 수의 학습 데이터를 확보하기 위해 gpt-4o API를 활용하여 데이터를 증강하여 활용함.
  - 기존 학습 데이터를 one-shot으로 제공하여 데이터를 증강하였으며, 일상 대화 500개를 생성 후 각 대화 당 인과추론 유형 문제 각 5개를 생성하여 총 2500 문제를 증강함.

GPT-4o에게 제공한 프롬프트는 다음과 같음.

- 대화 생성 프롬프트:

```
두 명의 한국어 대화를 만들어 주세요. 다음 조건을 꼭 지켜주세요.

조건:
1. 현대 한국 사회의 특정 이슈나 트렌드를 반영한 상황을 가정하고,
   그 안에서 한국인 화자 두 명이 메신저 상에서 한국어로 대화하는 시나리오를 작성해 주세요.
2. 독특하고 흥미로운 주제를 선택하되, 너무 전문적이거나 특수한 상황은 피해주세요.
3. 화자 한 명의 발화를 1개의 턴이라고 정의했을 때, 대화의 길이는 20~30 턴 사이에서 작성해 주세요.
4. 대화는 자연스럽게 흘러가야 하며, 중간에서 시작하거나 끝나도 괜찮습니다.
   하지만 대화의 맥락이 명확히 드러나도록 해주세요.
5. 대화에서 다음 요소들이 간접적으로 드러나도록 작성해 주세요:
   - 대화의 배경이나 원인
   - 대화 이후에 일어날 수 있는 사건이나 결과
   - 대화의 전제 조건이나 상황
   - 화자들의 감정, 욕구, 의도
   - 화자들의 성격이나 관계
6. 대화 내용에 다음 요소들을 포함시켜 주세요:
   - 한국어 특유의 표현이나 관용구
   - 온라인 대화 특유의 표현
   - 세대나 직업을 반영한 말투나 어휘
7. [중요]다음의 형태를 가지는 JSON 형식으로 대화를 작성해 주세요.
   다른 문자는 포함시키지 않습니다.:
   {"conversations": [{'speaker': 1, 'utterance': "..."},
    {'speaker': 2, 'utterance': "..."}, ...]}

이 조건들을 충족하는 대화 시나리오를 1개 생성해 주세요.
각 시나리오는 현실적이면서도 흥미로운 한국의 일상을 반영해야 합니다.
```

그림 3. GPT-4o 대화 생성용 프롬프트 예시

- 문제 생성 프롬프트:

해당 프롬프트의 [[TYPE]]과 [[DESCRIPTION]]은 맥락 추론의 유형별 설명을 작성하여 각각 gpt API에 제공하였음:

{

**"원인"**:"각 선택지가 대화의 원인이 될 수 있는지 논리적으로 분석하세요. 대화의 맥락, 등장인물의 반응, 그리고 언급된 사건들과 선택지 간의 연관성을 고려하세요.",

**"후행사건"**:"각 선택지가 대화의 내용과 맥락을 고려했을 때 논리적으로 발생 가능한 후속 사건인지 분석하세요. 대화 참여자들의 의도, 감정 상태, 그리고 언급된 계획이나 예상되는 행동들과 선택지 간의 연관성을 평가하세요.",

**"전제"**:"각 선택지가 대화에서 언급된 사건이나 상황이 발생하기 위해 필요한 선행 조건인지 분석하세요. 대화의 맥락, 등장인물의 상황, 그리고 언급된 사건들이 성립하기 위해 반드시 충족되어야 할 조건들과 선택지 간의 논리적 연관성을 평가하세요.",

**"동기"**:"화자의 발언 내용, 어조, 그리고 대화의 전반적인 맥락을 분석하여 화자의 감정 상태와 기본적인 욕구를 파악하세요. 각 선택지가 화자의 행동이나 말의 근본적인 동기로 작용할 수 있는지 평가하고, 인간의 기본적인 심리적 욕구와 연관 지어 고려하세요.",

**"반응"**:"화자가 발언에 대해 가질 수 있는 감정적, 행동적 반응을키 분석하세요. 대화 맥락, 발화의 내용과 의도를 고려하여 청자의 입장에서 자연스럽고 논리적인 반응인지 평가하세요"

}

증강시킨 데이터를 학습에 활용했을 때, 더 좋은 성능을 보인 모델도 있었으나 일반적으로는 검증 데이터 세트보다 성능에 미치는 효과가 미미하였음.

```
[[DIALOGUE]]

위의 대화를 분석하고, 대화의 [[TYPE]]에 관한 고품질 추론 문제와 정답을 만들어 주세요.
다음 지침을 엄격히 따라주세요:

1. 대화의 [[TYPE]]에 대한 이해:
   [[TYPE]]이란 [[DESCRIPTION]]입니다. 이 정의를 바탕으로 문제를 구성하세요.

2. 문제 구성:
   - "화자1"과 "화자2"를 언급하는 3개의 서로 다른 문제를 만드세요.
   - 각 문제는 대화의 [[TYPE]]과 직접적으로 연관되어야 합니다.
   - 문제는 대화 내용에 대한 깊은 이해와 추론을 요구해야 합니다.

3. 답변 선택지:
   - 각 문제에 대해 3개의 선택지를 제시하세요.
   - 오직 1개의 선택지만 정답이어야 하며, 나머지 2개는 명확히 오답이어야 합니다.
   - 오답은 그럴듯해 보이지만, 주의 깊은 분석으로 배제할 수 있어야 합니다.

4. 문제 형식:
   - 각 문제는 명확하고 간결한 문장으로 작성하세요.
   - 예시 형식:
   "화자1의 [[TYPE]]에 대한 가장 정확한 설명은?",
   "대화에서 드러난 화자2의 [[TYPE]]은 무엇인가?",
   "이 대화의 [[TYPE]]를 가장 잘 나타내는 것은?"

5. 난이도와 다양성:
   - 쉬운 문제부터 복잡한 추론이 필요한 문제까지 다양한 난이도로 구성하세요.
   - 대화의 다른 측면을 다루는 문제들을 만들어 다양성을 확보하세요.
6. 다음의 형태를 가지는 JSON 형태의 데이터를 생성해 주세요.
{"category":[[TYPE]],
 "qusetion_1":qusetion_1,
  "qusetion_2":qusetion_2,
  "qusetion_3":qusetion_3,
  "answer":qusetion_x}

주의사항:
- 모든 문제와 선택지는 대화 내용을 기반으로 하며, 추가적인 정보 없이 해결 가능해야 합니다.
- 문제는 단순한 사실 확인이 아닌, 대화의 [[TYPE]]에 대한 깊은 이해와 분석을 요구해야 합니다.
- 선택지는 서로 명확히 구분되어야 하며, 모호성을 피해야 합니다.
```

그림 4. GPT-4o 문제 생성용 프롬프트 예시

이에 대한 이유는 다음의 관점에서 추론해볼 수 있음.

- gpt-4로 생성한 대화가 원본 대화 데이터의 특성을 충분히 반영하지 못함
- 주어진 훈련 데이터의 양이 이미 충분했기 때문에 추가적인 데이터가 성능에 영향을 끼치지 못함.

## 3. 데이터 분석

학습 데이터에 대한 특성을 파악하기 위해 다음과 같은 데이터 분석을 실시함.

| | |
|---|---|
| 최소 턴 길이 | 12 |
| 최대 턴 길이 | 30 |
| 평균 | 20.8 |
| 중간값 | 21 |
| 중위값 | 21 |

표 3. 제공받은 데이터의 턴 길이 분석

| 유형 | 문제수 |
|---|---|
| 원인 | 161 |
| 동기 | 156 |
| 전제 | 161 |
| 반응 | 162 |
| 후행사건 | 118 |

표 4. 문제 유형 분석

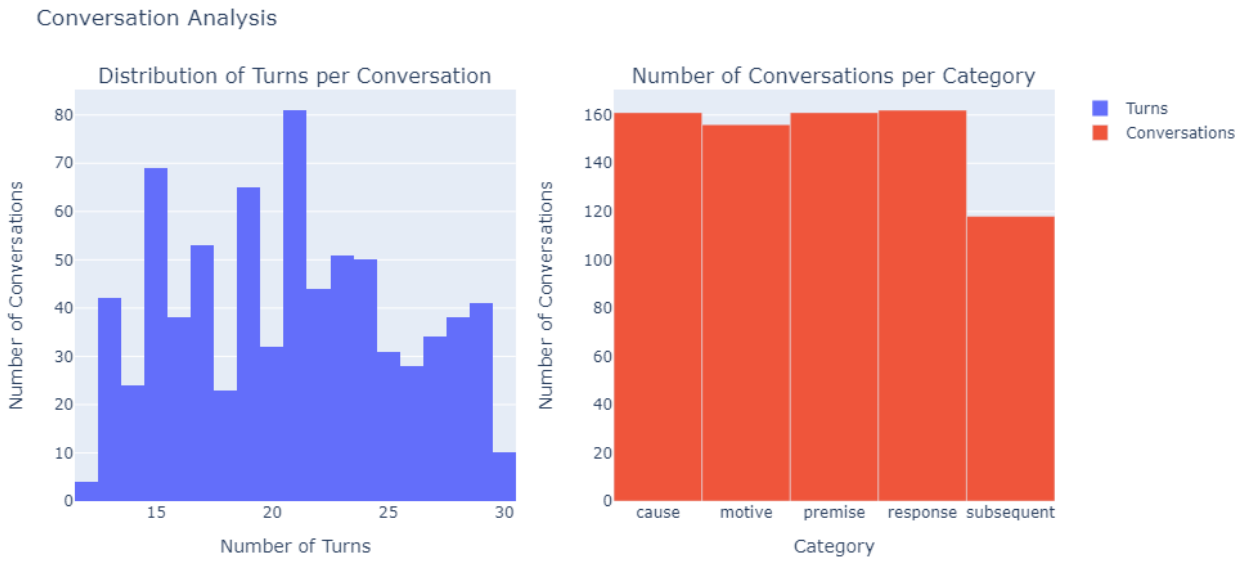


그림 5. 학습 데이터 분석 시각화 자료

## 4. 모델 개요

본 모델 기술서에서 다루고 있는 모델은 복수의 모델의 추론 결과를 앙상블 학습(하드보팅)으로 통합한 모델이다.

각 모델의 간략한 특징은 다음과 같다.

- **STOCK_SOLAR-10.7B**[2]: SOLAR 10.7B 모델을 튜닝한 모델들을 STOCK기법을 사용하여 병합한 모델이다.
- **Mistral-Nemo-Instruct-2407**[3]: 미스트랄 AI와 엔비디아가 공동으로 제작한 12B 언어 모델이다.
- **gemma-2-27b-it**[4] : 질문 답변, 요약, 추론 등 다양한 텍스트 생성 작업에 적합한 Google사의 27B 언어 모델이다.
- **Yi-Ko-34B-Chat-Preview**[5]:01-ai의 Yi 34B[6] 모델에 한국어 어휘 확장과 추가적인 학습을 진행한 Yi-Ko[7] 모델에 ChatVector[8] 방법론을 적용하여 대화 기능을 이식한 34B 언어 모델이다.

앙상블 학습에 사용된 각 모델은 다음과 같은 실험들을 통해 선정되었다.

## 4.1. 모델 파라미터 수에 대한 실험

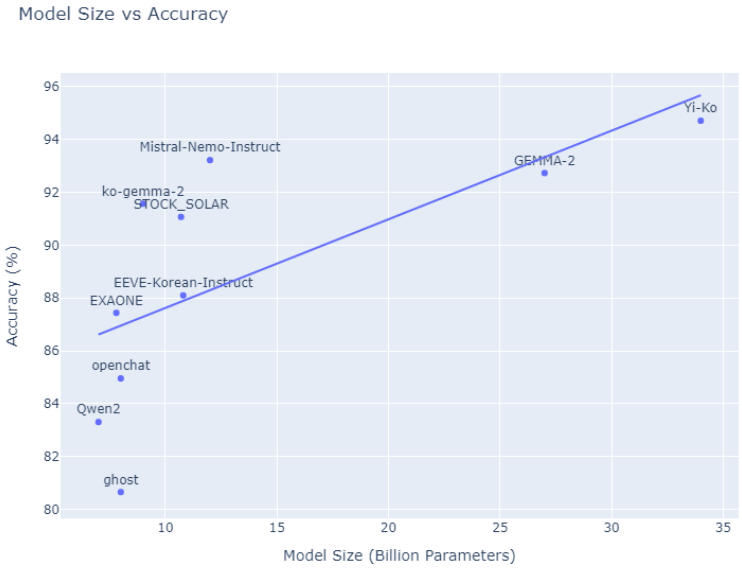


그림 6. LLM 파라미터 크기에 따른 정확도

- 10B 안팎의 파라미터 내에서는 성능의 편차가 크지만, 전반적인 추세를 확인했을 때 언어 모델의 매개변수 크기와 인과추론의 성능이 비례하는 것을 확인할 수 있음.
- 따라서 실험에는 주로 LoRA 학습을 한 성능이 좋았던 15B 미만 모델들과 QLoRA 학습을 한 15B 이상 모델을 사용함.

## 4.2. 학습 방법 실험

동일 모델에 대해 동일 조건하에서 학습 방법에 따른 성능 차이 검증을 진행함

본 실험에서 사용한 학습 기법은 다음과 같음.

- **SFT (Supervised Fine-Tuning) :** 미리 학습된 대형 모델을 특정 작업에 맞추기 위해, 레이블이 있는 데이터셋을 사용하여 모델의 파라미터를 추가로 학습시키는 기법
- **LoRA**[9] **:** 대형 모델의 일부 파라미터를 고정하고 저차원 행렬을 학습시켜 메모리와 연산 효율을 극대화하면서 성능을 유지하는 기법
- **QLoRA**[10] **:** LoRA에 파라미터 양자화를 추가해 메모리 사용량을 더욱 줄이면서도 미세 조정을 효율적으로 수행하는 기법

| 모델명 | SFT 성능 | LoRA 성능 |
|---|---|---|
| kihoonlee/STOCK_SOLAR-10.7B | 90.9 | **92.8** |
| yanolja/EEVE-Korean-Instruct-10.8B-v1.0 | 77.85 | **88.9** |
| rtzr/ko-gemma-2-9b-it | 72.89 | **91.5** |

표 5. 학습 방법에 따른 모델 성능 비교

- 전반적으로 SFT보다 LoRA 를 사용하여 학습했을 때 성능이 더 좋았던 것을 확인할 수 있었음.
- qlora를 사용하면 같은 학습 조건 하에서 더 큰 파라미터의 모델을 학습시킬 수 있음
- 4.1과 4.2의 결과를 종합해서 주어진 조건 하에서 최대한 큰 크기의 언어 모델을 사용할 수 있는 LoRA, QLoRA 학습 방법을 주로 사용하였음.

## 4.3. 사용 데이터

| 모델명 | Train | Train+Vaild | Train+Vaild+Add |
|---|---|---|---|
| kihoonlee/STOCK_SOLAR-10.7B | 88.6 | **92.7** | 92.1 |
| beomi/Yi-Ko-34B-Chat-Preview | 93.7 | 94.7 | **95.5** |
| unsloth/Mistral-Nemo-Instruct-2407-bnb-4bit | 89.5 | **93.2** | 92.4 |

표 6. 학습 데이터에 따른 모델 성능 비교

- 대부분의 모델에 대해 실험한 결과, 학습 데이터셋만을 학습시켰을 때보다 학습 셋과 검증 셋을 합하여 같이 학습시켰을 때 일반적으로 성능이 높았음.
- chatgpt를 사용해 증강한 데이터셋은 일부 모델에서는 성능을 상승시키는 효과를 보였음.

## 4.4. 프롬프트

기술각 모델마다 그에 맞는 시스템 프롬프트와 문제 포메팅 방식을 달리 하였음.

- 모델별 사용 프롬프트 표(각 프롬프트 설명은 그림1, 그림2에서 확인 가능):

| 모델명 | 시스템 프롬프트 | 채팅 프롬프트 |
|---|---|---|
| STOCK_SOLAR-_dev | kihoon_custom | default |
| Mistral-Nemo-Instruct-2407-bnb-4bit-valdata-qlora | no | default |
| gemma-2-27b-it-bnb-4bit-valdata-qlora | no | default |
| Yi-Ko-34B-Chat-bnb-4bit-valdata-qlora | yi-ko | default |
| STOCK_SOLAR-10.7B-overfitting1 | no | skip_category |
| Yi-Ko-34B-Chat-bnb-4bit-valdata-adddata-qlora | yi-ko | default |

표 7. 사용한 모델들의 시스템, 채팅 프롬프트 설정

### 4.5. 문장 생성을 활용한 추론

추론 과정 데이터를 만든 뒤, 이를 label로 사용하여 모델을 학습하면 입력이 주어졌을 때 더욱 논리적이고 분석적인 결과를 생성해낼 것으로 예측함.

#### 4.5.1 CoT 추론

모델이 객관식 답만 출력하는 것이 아닌, CoT를 통해 충분히 논리적이고 분석적인 결과를 생성하여 추론하도록 만들었음.

```
...
        "category": "반응",
        "inference_1": "화자1은 조심하려고 노력하는 화자2를 보니 초조하다.",
        "inference_2": "화자1은 조심하려고 노력하는 화자2가 황당하다.",
        "inference_3": "화자1은 조심하려고 노력하는 화자2가 바람직스럽다."

    "cot": "이 대화는 새로 이사한 집의 문제들, 특히 베란다 누수와 주변 지역의 씽크홀 공사 문제에 대해 두 화자가 논의하는 내용입니다.\n\n 화자1은 새로 이사한 집에서 발생한 문제들을 해결하려 노력하고 있습니다.
        화자2는 화자1의 노력을 긍정적으로 평가하며, 상황이 더 악화되는 것을 막아주었다고 말하고 있습니다. 두 화자 모두 주변 지역의 공사 문제에 대해 우려를 표하고 있습니다.
        \n\n이를 종합해볼 때, 화자1은 조심하려고 노력하는 화자2의 태도를 바람직스럽게 여기고 있다고 볼 수 있습니다. \n\n따라서 정답은 c. 화자1은 조심하려고 노력하는 화자2가 바람직스럽다입니다."
    "output": "inference_3",
}
```

그림 7. CoT를 이용한 논리 추론 예시

해당 데이터를 사용해 학습을 진행하였음

```python
# 입력 템플릿 예시
def make_chat_template(self, inp):
  chat = [""]
  for cvt in inp['conversation']:
      speaker = cvt['speaker']
      utterance = cvt['utterance']
      chat.append(f"화자{speaker}: {utterance}")
  chat = "\n".join(chat)
  chat += "다음은 주어진 대화에 대해 틀린 답 두개와 정답 한개를 섞은 문제입니다.\n"
  chat += f"A. {inp['inference_1']}\n"
  chat += f"B. {inp['inference_2']}\n"
  chat += f"C. {inp['inference_3']}\n"
  chat += f"A,B,C중 올바른 정답은 무엇일까요? "
  return chat
```

그림 8. 학습 이후 user 입력 예시(출처:src/data.py)

```python
#모델을 통해 다음과 같이 결과가 나왔을 때, 이를 original file 형식에 맞추는 방안입니다.
#"reason": "문제의 맥락을 고려하면, 화자2가 뮤지컬을 관람한 경험이 있었고, 이 경험을 통해 영상과 관련된 작업에 대한 열정을 느꼈을 수 있습니다.
#          그러나 그가 다른 건설을 경부할 것인지, 영상 관련 작업을 갖는 것에 대해 다시 고민할 것인지, 또는 골학 전공자를 우대하는 회사에 지원할 것
#          따라서, 정확한 답을 제공하기 위해서는 화자2의 개인적인 생각과 결정에 대한 추가 정보가 필요합니다. 그러나 주어진 옵션 중에서 가장 가능성

def determine_inference(self, output):
    vote_count = {"A": 0, "B": 0, "C": 0, "None": 0}
    if "A" in output:
        vote_count["A"] += 1
    elif "B" in output:
        vote_count["B"] += 1
    elif "C" in output:
        vote_count["C"] += 1
    else:
        vote_count["None"] += 1

        #모델의 답변에서 가장 많이 언급된 정답 찾기
    selected_inference = max(vote_count, key=vote_count.get)

    if selected_inference == "A":
        return "inference_1"
    elif selected_inference == "B":
        return "inference_2"
    elif selected_inference == "C":
        return "inference_3"
    else:
        return str(vote_count)

# "output": "inference_2"
```

그림 9. LLM 생성 이후 출력 결과 포멧팅 코드 예시

train과 dev 데이터셋을 사용해서 “STOCK-SOLAR-10.8B” 모델을 LoRA로 튜닝하였고, MMRazor와 MMDeploy 팀이 개발한 LLM 압축, 배포, 서빙을 위한 툴킷 LMDeploy[11]를 활용해 추론한 결과 94.38의

결과를 얻었음.
- 생성 인자값 실험 결과:
temperature와 top_p 각각 0.2로 설정, 나머지는 default 유지 시 더욱 효과적인 추론 결과 얻음

### 4.5.2 CoT 데이터 증강

CoT 데이터는 기존의 학습 세트에 있는 대화와 맥락 추론 정답을 gpt-4o에게 제공하고, 정답을 맞히는 과정에 필요한 논리적 추론 과정을 생성하는 방식으로 제공하였음.
사용 프롬프트와 데이터 예시는 다음과 같음.

```
[[DIALOGUE]]

위의 대화와 문제, 정답을 보고 정답을 추론하기 위한 추론 과정을 작성해 주세요. 다음 지시사항을 지켜주세요.

1. 대화의 주요 내용을 간단히 요약하세요.
2. [[TYPE_INST]]
3. 가장 관련성 높은 적절한 정답 문항을 선택하세요.
4. 작성 내용은 길어도 5문장을 넘지 말아야 합니다.
5. 추론 과정의 마지막 문장은 "따라서 정답은 <A/B/C>입니다."와 같은 형태로 마무리해 주세요.
확실한 정답이 없더라도 가능성이 가장 높은 한가지를 골라주세요.
```

그림 10. CoT에 필요한 논리적 추론 과정을 요청하는 프롬프트 예시.

[[DIALOGUE]]에는 대화 내용, [[TYPE_INST]]에는 맥락 추론 유형 별 지시사항을 다음과 같이 부여하였음.

TYPE_MAP = {
"원인":"각 선택지가 대화의 원인이 될 수 있는지 논리적으로 분석하세요. 대화의 맥락, 등장인물의 반응, 그리고 언급된 사건들과 선택지 간의 연관성을 고려하세요.",

**"후행사건"**:"각 선택지가 대화의 내용과 맥락을 고려했을 때 논리적으로 발생 가능한 후속 사건인지 분석하세요. 대화 참여자들의 의도, 감정 상태, 그리고 언급된 계획이나 예상되는 행동들과 선택지 간의 연관성을 평가하세요.",

**"전제"**:"각 선택지가 대화에서 언급된 사건이나 상황이 발생하기 위해 필요한 선행 조건인지 분석하세요. 대화의 맥락, 등장인물의 상황, 그리고 언급된 사건들이 성립하기 위해 반드시 충족되어야 할 조건들과 선택지 간의 논리적 연관성을 평가하세요.",

**"동기"**:"화자의 발언 내용, 어조, 그리고 대화의 전반적인 맥락을 분석하여 화자의 감정 상태와 기본적인 욕구를 파악하세요. 각 선택지가 화자의 행동이나 말의 근본적인 동기로 작용할 수 있는지 평가하고, 인간의 기본적인 심리적 욕구와 연관 지어 고려하세요.",

**"반응"**:"화자가 발언에 대해 가질 수 있는 감정적, 행동적 반응을 분석하세요. 대화 맥락, 발화의 내용과 의도를 고려하여 청자의 입장에서 자연스럽고 논리적인 반응인지 평가하세요"
}

```
{
    "id": "nikluge-2024-대화 맥락 추론-train-000001",
    "input": {
        "conversation": [
            {
                "speaker": 2,
                "utterance": "진짜 신의 한수",
                "utterance_id": "MDRW2100003410.1.1"
            },{...}
        ],
        "reference_id": [
            "MDRW2100003410.1.11"
        ],
        "category": "원인",
        "inference_1": "화자2가 사는 곳 근처에서 베란다 보수 공사가 진행되고 있다.",
        "inference_2": "화자2가 사는 곳 근처에서 싱크홀 보수 공사가 진행되고 있다.",
        "inference_3": "화자2가 사는 곳 근처에서 싱크홀 보수 공사가 중단되었다."
    },
    "output": "inference_2",
    "cot": "1. 대화의 주요 내용은 화자2가 사는 곳 근처에서 싱크홀 공사가 진행되고 있으며,
    이로 인해 비가 많이 올 때 문제가 발생하고 있다는 것입니다.\n\n
    2. 선택지 A는 대화의 내용과 부합하지 않습니다.
    대화에서는 베란다 보수 공사에 대한 언급이 없습니다.
    선택지 B는 대화의 내용과 일치합니다. 화자들은 싱크홀 공사에 대해 언급하고 있으며,
    비가 많이 오면 문제가 발생한다고 말하고 있습니다.
    선택지 C는 대화에서 싱크홀 공사가 중단되었다는 내용이 없으므로 적절하지 않습니다.\n\n
    3. 대화의 내용을 종합해볼 때 선택지 B가 가장 관련성이 높습니다.\n\n
    4. 따라서 정답은 B입니다."
}
```


그림 11. 출력된 논리적 추론 텍스트 예시

기존 베이스라인 학습 방법에서 입력 부분은 그대로 유지하고 출력 부분을 선택지 3개 중 하나를 선택하는 기존의 방법 대신 생성한 논리적 추론 과정 텍스트로 대체하여 학습시켜 성능을 측정하였음.

최종 정답 선택은 5.1과 동일하게 생성한 추론 결과에 등장하는 (A,B,C)중 가장 빈번하게 등장한 선택지를 모델이 추론한 정답으로 간주하였음.

성능은 다음과 같음.

| 모델명 | 기존 방식 성능 | CoT Train 실험 |
|---|---|---|
| yanolja/EEVE-Korean-Instruct-10.8B-v1.0 | **88.9** | 85.9 |
| rtzr/ko-gemma-2-9b-it | **91.5** | 70.5 |

표 9. 기존 방식 대비 CoT 학습 비교 결과.

5.1과 5.2 방법 모두 기존 방식보다 안좋은 효과를 기록했기 때문에(5.1에서 사용한 방법으로는 94.38로 단일 모델로써는 나쁘지 않은 성능을 기록했지만 앙상블 학습에 사용했을 시 부정적인 영향을 미침) 최종적으로는 기존 방법+프롬프트 수정을 채택하였음.

### 4.5.3 앙상블 추론

좋은 성능을 보이는 모델에 대해 결과를 하드보팅하는 식으로 앙상블을 진행했을 때 성능이 1~2% 올라가는 효과를 보임.

조합을 바꾸어가며 리더보드 기준으로 가장 높은 성능을 기록한 모델 조합을 최종 제출 모델로 선정함. 앙상블에 사용할 모델 결과 파일을 yaml 파일 형태로 관리하여 실험하였음. 앙상블 학습 코드와 모델 조합은 다음과 같음.

| 모델명 | 베이스 모델 | 학습 방법 | 학습 데이터 | 비고 |
|---|---|---|---|---|
| STOCK_SOLAR-_dev | kihoonlee/STOCK_SOLAR-10.7B | LoRA | train+valid | |
| Mistral-Nemo-Instruct-2407-bnb-4bit-valdata-qlora | unsloth/Mistral-Nemo-Instruct-2407-bnb-4bit | QLoRA | train+valid | |
| gemma-2-27b-it-bnb-4bit-valdata-qlora | unsloth/gemma-2-27b-it-bnb-4bit | QLoRA | train+valid | |
| STOCK_SOLAR-10.7B-overfitting1 | kihoonlee/STOCK_SOLAR-10.7B | LoRA | train+valid | target_modules="all-linear" |
| Yi-Ko-34B-Chat-bnb-4bit-valdata-adddata-qlora | beomi/Yi-Ko-34B-Chat-Preview | QLoRA | train+valid+증강데이터 | |
| Yi-Ko-34B-Chat-bnb-4bit-valdata-qlora | beomi/Yi-Ko-34B-Chat-Preview | QLoRA | train+valid | |

```python
def hard_voting(paths, output_file):
    file1 = './resource/data/대화맥락추론_test.json'

    with open(file1, 'r', encoding='utf-8') as f1:
        data1 = json.load(f1)

    data_list = []
    for path in paths:
        with open(path, 'r', encoding='utf-8') as f:
            data_list.append(json.load(f))

    if any(len(data1) != len(data) for data in data_list):
        raise ValueError("The files do not contain the same number of entries.")

    total = len(data1)

    hard_voting_results = []

    for i in range(total):
        outputs = [data[i]['output'] for data in data_list]
        most_common_output = Counter(outputs).most_common(1)[0][0]
        result_entry = data1[i].copy()
        result_entry['output'] = most_common_output
        hard_voting_results.append(result_entry)

    with open(output_file, 'w', encoding='utf-8') as f:
        json.dump(hard_voting_results, f, ensure_ascii=False, indent=4)

    print(f"Hard voting results saved to {output_file}")
```

그림 12. 하드보팅 코드 (출처:run/voting.py)

```yaml
paths:
  - '/workspace/CCI_2024/outputs/stock/STOCK_SOLAR-_dev.json'
  - '/workspace/CCI_2024/outputs/mistral/Mistral-Nemo-Instruct-2407-bnb-4bit-valdata-qlora.json'
  - '/workspace/CCI_2024/outputs/gemma2/gemma-2-27b-it-bnb-4bit-valdata-qlora.json'
  - '/workspace/CCI_2024/outputs/yi/Yi-Ko-34B-Chat-bnb-4bit-valdata-qlora.json'
  - '/workspace/CCI_2024/outputs/stock/STOCK_SOLAR-10.7B-overfitting1.json'
  - '/workspace/CCI_2024/outputs/yi/Yi-Ko-34B-Chat-bnb-4bit-valdata-adddata-qlora.json'
```

그림 13. 앙상블에 사용한 모델 결과 파일 경로 (출처:scripts/vote/voting.yaml)

## 5. 평가 결과

최종적으로 본 참가팀의 모델은 리더보드 상에서 96.85점을 기록하였음.
이는 앙상블하기 전 각각의 모델 성능보다 평균적으로 2.9점 향상된 성능임.

### 5.1. 의의

최신 학습 방법론과 추론 프레임워크, 프롬프트 엔지니어링 기법들을 적용하여 비교 분석을 진행하였음. (QLoRA,MoRA, Merging, LMdeploy, unsloth, etc.) 이를 통해 "맥락 추론" 작업에서 어떤 방안이 왜 효과적인지 확인할 수 있었음.
본 방안은 여러 개의 모델 응답 중 가장 높은 비율의 정답을 고르는 방식으로 높은 정확성을 보이며, 정확성이 중요한 작업에서 고려할만한 방안이라 생각됨. 효율성을 중요시하는 작업에서는 단일모델을 사용하는 것이 좋으며, 일반적으로 파라미터의 크기가 큰 모델을 양자화와 같이 압축시켜 최소한의 자원으로 학습하고 생성하는 것이 효과적임을 증명함.(sLLM+Full-Finetune < LLM+QLoRA)

### 5.2. 한계점 및 보완 사항

- 본 경진대회에서 성능을 높이기 위해 학습 방법 탐색, 훈련 데이터 증강 이외에 텍스트 생성 방식을 이용한 인과 추론, CoT 프롬프트 활용 등 여러 새로운 방법론을 고안하여 시도해 봤으나 효과적이지 않았음.
- 하드보팅 앙상블 방법의 특성 상 하나의 결과를 추론하기 위해 복수의 모델을 학습-추론하는 과정이 필요하고, 이는 시간과 자원을 많이 소모하는 방식이고, 실제 응용 환경에는 적합하지 않을 수 있음.
- 추후 다음과 같은 연구를 통해 추가적인 성능 향상을 기대할 수 있음.
  - 하드 보팅 방식이 아닌 각 모델의 선택지에 대한 logit을 기반으로 soft voting을 시행
  - 맥락 추론의 각 유형에 대한 정보를 언어 모델에게 추가로 전달하여 추론하도록 함
  - 언어 모델에 대화 전체 정보를 전달하는 대신 중요 정보만 요약하여 전달함

### 5.3. 모델 활용 방안

본 참가팀이 제안한 맥락 추론 모델은 다음과 같은 분야에 활용할 수 있을 것으로 기대됨.

- 인공지능 기반 컨택트 센터(AICC)
- AI 에이전트 대화 시스템에서의 대화 상태 추적(Dialogue State Tracking), 질문 추천(Question Recommandation)

## 6. 모델 사용설명서

### 6.1. 환경 설정

docker를 구동할 수 있는 환경을 기준으로 실험 환경을 설정하였음.

```
# 저장소 클론
git clone https://github.com/overfit-brothers/CCI-2024.git /overfitting-brothers
cd /overfitting-brothers
# 이미지 빌드
docker build -t overfitting-brothers:latest .
#컨테이너 실행
docker compose up -d
#컨테이너 내부에서 작업 진행
cd /overfitting-brothers
```

### 6.2. 모델 학습

```
# 해당 스크립트를 train 폴더에 있는 각 모델에 대해 실행
sh scripts/train/MODEL_NAME.sh
```

### 6.3. 모델 추론

```
# 학습된 각각의 모델에 대하여 추론 실행
sh scripts/test/MODEL_NAME.sh
```

### 6.4. 앙상블 추론

```
# 결과 json 파일들을 사용하여 Hard Voting을 진행하여 최종 결과 파일 생성
sh scripts/vote/voting.sh
```

---

1) Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022): 24824-24837.
2) https://huggingface.co/kihoonlee/STOCK_SOLAR-10.7B
3) https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407
4) https://huggingface.co/google/gemma-2-27b-it
5) https://huggingface.co/beomi/Yi-Ko-34B-Chat-Preview
6) https://huggingface.co/01-ai/Yi-34B
7) https://huggingface.co/beomi/Yi-Ko-34B
8) Huang, S. C., Li, P. Z., Hsu, Y. C., Chen, K. M., Lin, Y. T., Hsiao, S. K., ... & Lee, H. Y. (2024, August). Chat vector: A simple approach to equip llms with instruction following and model alignment in new languages. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 10943-10959).
9) Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.
10) Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2024). Qlora: Efficient finetuning of

quantized llms. Advances in Neural Information Processing Systems, 36.
11) https://github.com/InternLM/lmdeploy