

Solar Pro 2 Prompting Handbook

파라미터 세팅부터 RAG, 셀프검증(Self-verification)까지
진짜 성능을 이끌어내는 프롬프트 엔지니어링



Contents

1. 개요	3
1.1 추론(Reasoning)이란?	3
2. 파라미터 활용법	3
2.1 기본 파라미터	3
2.1.1 Temperature	3
2.1.2 Max Tokens	3
2.1.3 Top_p	4
2.2 Reasoning Effort	4
2.2.1 Low Reasoning Effort	4
2.2.2 High Reasoning Effort	5
2.2.3 High Reasoning Effort, 언제 선택할까요?	9
3. 프롬프트 구조 설계	10
3.1 핵심 요소	10
3.2 구조별 조합 패턴	12
3.2.1 기본 조합	12
3.2.2 출력 형태 고정	13
3.2.3 RAG (Retrieval-Augmented Generation)	13
3.3 요소 배치 순서의 중요성	13
3.4 구조화 포맷	14
3.4.1 Markdown	14
3.4.2 XML 태그	16
4. 세부 요소별 작성법	18
4.1 Role (역할) 정의	18
4.1.1 도메인 특화 전문가	18
4.1.2 태스크 수행 전문가	19

4.2 Instructions (지시사항)	21
4.2.1 요구사항 세분화	21
4.2.2 구체적인 수행 방법 제시	23
4.3 Constraints (제약사항)	24
4.3.1 긍정적 지시와 부정적 지시 분리	25
4.3.2 우선순위가 있는 제약사항	27
4.4 Format (출력 형식)	29
4.4.1 구조적 템플릿 제시	29
4.4.2 JSON 출력 형식	30
4.4.3 Few-Shot Learning	31
5. 꼭 따라야하는 지시사항 강조하기	34
5.1 셀프 검증 (Self-Verification)	34
5.2 맥락 관리 (Context Management)	36
5.2.1 긴 맥락에서의 핵심 정보 배치	36
6. 자주 발생하는 문제와 해결책	38
6.1 모호한 응답	38
6.2 일관적이지 않은 출력 형식	41
6.3 지시사항 불이행	44
6.4 할루시네이션	47
7. 마무리	51

1. 개요

solar-pro2는 고성능 추론과 다양한 태스크 수행에 최적화된 추론 모델입니다. 본 가이드는 개발자들이 solar-pro2의 향상된 능력을 최대한 활용할 수 있도록 체계적인 프롬프팅 방법론을 제시합니다.

solar-pro2는 명확하고 구조화된 지시사항에 매우 민감하게 반응하므로, 효과적인 프롬프트 설계가 성능에 직접적인 영향을 미칩니다. 기존의 일반적인 best practice와 함께, solar-pro2의 특성에 맞는 프롬프트 접근법이 필요합니다.

1.1 추론(Reasoning)이란?

solar-pro2는 모델이 응답하기 전에 내부적인 "사고 연쇄"를 생성할 수 있도록 하여 성능을 향상시킵니다. 이를 통해 모델은 복잡한 질문을 더 작은 단계로 나누고, 복잡한 질의를 논리적인 하위 작업으로 분해하며, 복잡하고 많은 지시사항을 정확히 수행할 수 있습니다. 복잡한 다단계 작업(예: 문서 분석, 코드 생성, 복잡한 논리적 문제)에서 모델의 구조화된 사고 연쇄 과정을 활용하세요.

2. 파라미터 활용법

2.1 기본 파라미터

2.1.1 Temperature

- 낮은 값 (0.0-0.3): 일관성이 중요한 태스크 (번역, 요약, 분석)
- 중간 값 (0.4-0.7): 균형잡힌 창의성과 정확성 (일반 대화, 설명)
- 높은 값 (0.8-1.0): 창의적 작업 (글쓰기, 브레인스토밍)

2.1.2 Max Tokens

- **짧은 응답:** 50-800 토큰 (키워드 추출, 분류)
- **중간 응답:** 800-4000 토큰 (요약, 설명)
- **긴 응답:** 4000+ 토큰 (상세 분석, 긴 글 작성)

2.1.3 Top_p

- **정밀한 응답:** 0.1-0.5 (기술 문서, 정확한 정보)
- **균형잡힌 응답:** 0.6-0.8 (일반적 용도)
- **다양한 응답:** 0.9-1.0 (창의적 작업)

2.2 Reasoning Effort

solar-pro2 모델은 작업 유형에 따라 추론(reasoning) 모드를 선택할 수 있는 하이브리드 모델입니다. reasoning_effort 파라미터는 low/high 두가지 설정이 가능합니다.

2.2.1 Low Reasoning Effort

reasoning_effort="low"로 설정하여 복잡한 추론 과정이 필요하지 않은 일반적인 태스크에 활용하는 모드입니다. 이 모드는 빠른 처리 속도와 비용 효율성이 장점이며, 대량의 단순 작업이나 실시간 응답이 필요한 서비스에 적합합니다.

- **일반적인 Q&A:** 사실 확인, 정의 설명
- **단순 번역:** 직역, 기본 언어 변환
- **정보 추출:** 텍스트에서 특정 데이터 추출
- **형식 변환:** 데이터 포맷 변경

Information Extraction 프롬프트 예제:

```
Python
system_prompt = """
Extract the following information from the [Text] I give and return as a
list:
- Person names
- Dates
- Locations
"""

user_prompt = """
Text: "John Smith visited Paris on March 15, 2024, and met with Sarah
Johnson at the Louvre Museum."
"""

temperature = 0.2
max_tokens = 300
reasoning_effort = "low"
```

답변:

```
Python
- Person names: John Smith, Sarah Johnson
- Dates: March 15, 2024
- Locations: Paris, Louvre Museum
```

2.2.2 High Reasoning Effort

`reasoning_effort="high"`로 설정하면 모델의 추론 모드를 활용할 수 있습니다. 이는 **모델이 답변을 생성하기 전에 필수적으로 추론 과정을 거치도록 하는 모드로**, 복잡한 분석과 구조적 사고가 필요한 작업에 최적화되어

있습니다. 이 모드에서는 모델이 단계적으로 문제를 분해하고, 각 단계에서 논리적 검증을 수행하여 더욱 정확하고 신뢰할 수 있는 결과를 생성합니다.

- **수학 문제 해결:** 다단계 계산, 증명
- **논리적 추론:** 삼단논법, 인과관계 분석
- **복합 분석:** 여러 관점에서의 종합적 검토
- **전략 수립:** 다양한 요소를 고려한 계획 수립

Legal Case Entity Extraction 프롬프트 예제:

```
Python
system_prompt = """
Extract property dispute details from the [Text] and return as lists only:
- Parties: [names and roles]
- Property details: [location, size, values]
- Damage calculations: [repair costs + lost rent with totals]
- Timeline: [key dates and duration of issues]
- Analysis: [verify if calculated total matches claimed amount, note any discrepancies]
"""

user_prompt = """
Text: "Landlord Sarah Kim is suing tenant Mike Johnson for $8,500 in damages to her downtown apartment. The lease shows monthly rent of $2,200. Johnson caused water damage in March 2024 requiring $3,500 in repairs. He also broke the lease early, leaving 2.5 months of unpaid rent before the July 2024 lease end date.\"
"""

temperature = 0.2
max_tokens = 8000
reasoning_effort = "high"
```

답변:

Python

```
- **Parties**:
  - Sarah Kim (Landlord)
  - Mike Johnson (Tenant)

- **Property details**:
  - Location: Downtown apartment (specific address not provided)
  - Size: Not specified
  - Values: Monthly rent = $2,200 (lease value not explicitly stated)

- **Damage calculations**:
  - Repair costs: $3,500 (water damage repairs)
  - Lost rent: 2.5 months × $2,200 = **$5,500**
  - **Total calculated damages**: $3,500 + $5,500 = **$9,000**

- **Timeline**:
  - March 2024: Water damage caused by tenant
  - Mid-April 2024 (approx.): Tenant left apartment (2.5 months before lease end)
  - July 2024: Original lease end date
  - Duration of unpaid rent: ~2.5 months

- **Analysis**:
  - Calculated total damages ($9,000) **do not match** the claimed amount ($8,500).
  - Discrepancy of **$500** exists; landlord's claim is $500 less than the sum of repair costs and lost rent.
  - Possible reasons: Partial settlement, unaccounted deductions, or error in claim calculation.
```


추론 과정에는 다음과 같은 단계가 포함되어, 더 정확하고 높은 품질의 답변을 이끌어냅니다. reasoning 모드를 사용할 때는 답변을 도출하기 위한 충분한 추론과정을 포함하기 위해 **max_token을 4000이상으로 설정하는 것을 추천드립니다.**

단계별 계획 수립 및 작업 분해	"First, I'll identify the parties involved. The landlord is Sarah Kim and the tenant is Mike Johnson. Their roles are clear from the text."
순차적 정보 처리	"Next, property details. The text mentions a downtown apartment, but there's no info on size or specific values beyond rent."
수학적 계산과 검증	"For damage calculations, the repair costs are \$3,500 from water damage. The monthly rent is \$2,200, so 2.5 months would be \$5,500." "Let me double-check. 2.5 months at \$2,200 is $2.5 * 2200 = 5,500$. Plus 3,500 repairs equals 9,000."
추론 모순점 발견 및 원인 분석	"But the landlord is suing for \$8,500. There's a discrepancy here. The total calculated is \$9,000, but the claim is \$8,500." "Maybe the landlord is not claiming the full 2.5 months? Need to mention this discrepancy."
최종 검증과 구조화	"Make sure each section is clear and only includes the information from the text." "Alright, putting it all together now."

2.2.3 High Reasoning Effort, 언제 선택할까요?

기본 전략: 작업의 성격을 먼저 파악한 후, 품질이 절대적으로 중요한 경우에만 `reasoning_effort="high"` 사용을 권장합니다.

*작업 유형에 따라 속도 및 출력 토큰은 달라질 수 있습니다.

구분	<code>reasoning_effort="high"</code>	<code>reasoning_effort="low"</code>
처리 속도	상대적으로 느림	약 3배 빠른 속도로 처리 가능
출력 토큰	추론 과정 토큰 포함	출력 토큰 약 70% 감소 가능

다음 질문들을 통해 적절한 파라미터를 설정하세요:

품질 vs 효율성 우선순위

- ☐ 결과의 정확도가 절대적으로 중요한가? → `reasoning_effort="high"`
- ☐ 처리 속도와 비용이 더 중요한가? → `reasoning_effort="low"`

작업 복잡도 평가

- ☐ 여러 단계의 논리적 사고가 필요한가? → `reasoning_effort="high"`
- ☐ 단순한 정보 제공이나 변환 작업인가? → `reasoning_effort="low"`

사용 환경 고려

- ☐ 실시간 응답이 필요한 서비스인가? → `reasoning_effort="low"`

3. 프롬프트 구조 설계

3.1 핵심 요소

효과적인 프롬프트는 다음 요소들로 구성됩니다. 각 요소는 모델의 이해도와 출력 품질에 직접적인 영향을 미치므로, 작업의 특성에 맞게 세심하게 설계해야 합니다:

- **Role** (역할): 모델의 전문성과 관점 정의
 - 전문 영역 지식의 활성화
 - 응답 톤과 스타일의 설정
- **Task** (태스크): 수행할 작업의 명확한 정의
 - **Instructions** (지시사항): 구체적인 수행 방법
 - **Constraints** (제약사항): 준수해야 할 규칙과 한계
- **Output Format** (형식): 출력 구조와 스타일 정의
 - **Samples** (예제): 기대하는 출력의 구체적 예시
- **User Input** (사용자 입력): 실제 처리할 데이터
 - **Contexts** (맥락): 작업 수행에 필요한 배경 정보
 - **Query** (질의): 구체적인 요청 사항

Role

You are a professional email editor for business communications.

Role

Task

Instructions

- Correct grammar, spelling, and punctuation errors
- Improve sentence clarity and flow
- Maintain professional tone

Task

Instructions

Constraints

Constraints

- Keep original meaning intact
- Do not change technical terms
- Preserve sender's intent

Output Format

Return only the corrected email text without explanations.

Format

Instructions

Samples

Samples

Input: "Hi John, I wants to schedule meeting for discuss the project."

Output: "Hi John, I want to schedule a meeting to discuss the project."

User Input

Query

Correct this email for professional communication.

Use Input

Use Query

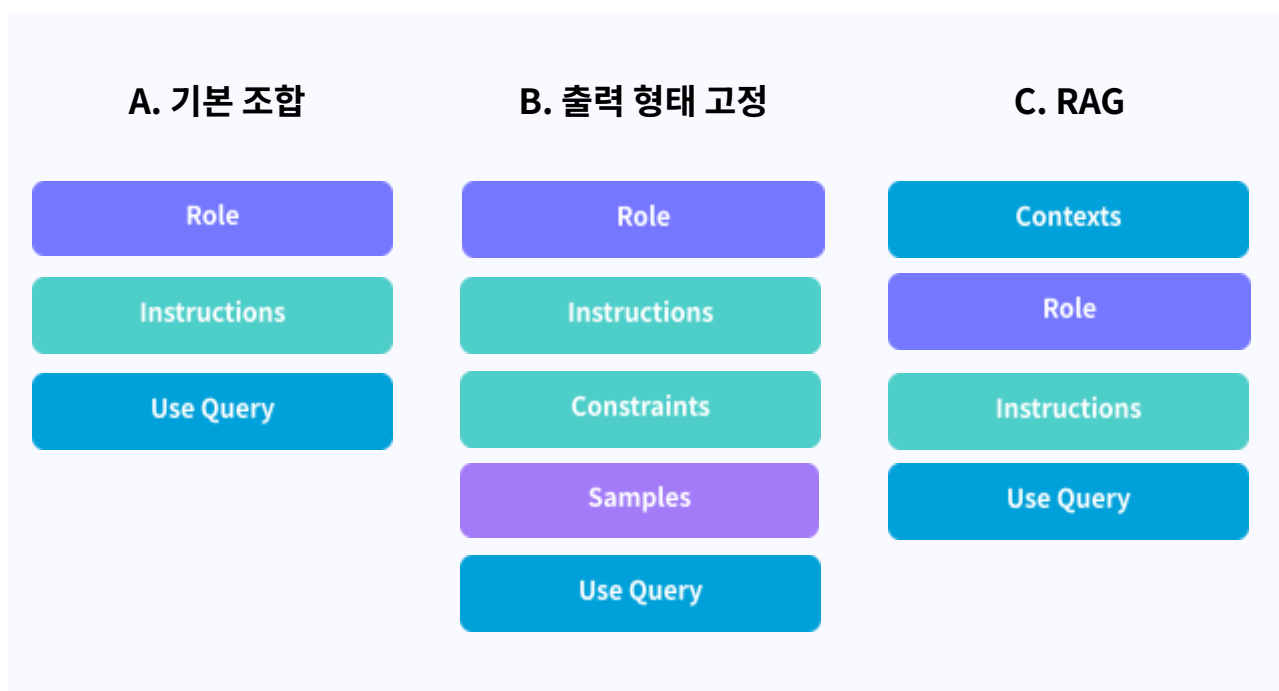
Contexts

Context

Internal company email to team member.

3.2 구조별 조합 패턴

프롬프트 구조는 작업의 복잡도와 요구사항에 따라 다르게 조합해야 합니다. solar-pro2는 구조화된 정보를 잘 이해하므로, 적절한 구조 선택이 성능 향상의 핵심입니다.



3.2.1 기본 조합

간단한 태스크나 일반적인 질의응답에 적합합니다. 빠른 응답이 필요하고 복잡한 제약이 없는 경우에 사용합니다.

None

Role → Instructions → Query

3.2.2 출력 형태 고정

정해진 형식의 출력이 필요한 경우에 사용합니다. API 응답, 데이터 변환, 정형화된 보고서 작성 등에 적합합니다.

None

Role → Instructions → Constraints → Samples → Query

3.2.3 RAG (Retrieval-Augmented Generation)

외부 정보를 활용한 답변 생성에 사용합니다. 문서 분석, 정보 검색 기반 응답, 지식 베이스 활용 등에 적합합니다.

None

Contexts → Role → Instructions → Query

3.3 요소 배치 순서의 중요성

프롬프트 내 요소의 순서가 모델의 출력 경향성에 큰 영향을 미치므로 중요한 정보를 프롬프트의 적절한 위치에 배치하는 것이 핵심입니다.

아래는 일반적인 각 요소 배치 순서이나, 동일한 요소라도 각 요소간의 위치를 변경하거나, 시스템 프롬프트와 유저 프롬프트 간에 위치를 변경하면서 성능 차이를 비교 분석해야 합니다. 이러한 체계적인 A/B 테스트를 통해 특정 작업에 최적화된 프롬프트 구조를 발견할 수 있습니다.

- **System Prompt**
 - **맥락 정보:** 가장 앞에 배치하여 전체 상황을 이해하도록 함
 - **역할 정의:** 맥락 다음에 배치하여 응답 관점을 설정
 - **구체적 지시사항:** 역할 정의 후 상세한 수행 방법 제시
 - **제약사항과 형식:** 지시사항 직후에 배치하여 출력 품질 관리
 - **예제:** 형식 설명 후 구체적 예시로 이해도 향상
- **User Prompt:** 구체적인 태스크와 입력 데이터

3.4 구조화 포맷

3.4.1 Markdown

주요 섹션과 하위 섹션을 위해 마크다운 제목을 사용하세요(H4+까지의 더 깊은 계층 포함). 필요에 따라 번호 매기기나 불릿 포인트를 사용하세요.

미팅 아젠다 생성 프롬프트 예제:

```
Python
system_prompt = """
# Role
You are a meeting organizer specializing in team communications.

## Instructions
Create structured meeting agendas with clear time management and actionable outcomes.

#### Example Format
**Topic Name [X min]**
- Discussion point 1
```

- Discussion point 2

Generate a clear meeting agenda following this structure."""

user_prompt = """

Create a 20-minute standup meeting agenda covering project updates and blockers.

"""

답변:

Python

Project Standup Meeting [20 min]

Focus: Progress updates, blockers, and alignment

1. Quick Check-in & Goals [2 min]

- Welcome and confirm attendance
- Recap meeting goals: 1) Share progress, 2) Identify blockers, 3) Align on priorities

2. Individual Updates [10 min]

- **Each team member (2 min):**
 - **Yesterday's Progress:** Key tasks completed
 - **Today's Priorities:** Immediate focus areas
 - **Blockers (brief):** Flag urgent issues (detailed in next section)

3. Blocker Resolution [5 min]

- Discuss raised blockers:
 - Clarify root causes
 - Brainstorm solutions or dependencies
 - Assign owners for follow-up (e.g., "Mark to contact vendor by EOD")

4. ****Action Items & Accountability [2 min]****

- Summarize assigned tasks, owners, and deadlines
- Confirm **next** steps (e.g., "Design review by 3 PM today")

5. ****Closing & Next Steps [1 min]****

- Schedule **next** **standup** (time/format)
- Thank participants and adjourn

****Outcome****: Clear visibility into progress, resolved blockers, and actionable **next** steps. Keep updates concise to stay within time limits!

3.4.2 XML 태그

시작과 끝을 포함하여 섹션을 정확하게 감싸야할 때 사용하세요. 추가 맥락을 위해 태그에 메타데이터를 추가할 수 있으며, 같은 태그 안에 새로운 태그를 넣어 중첩된 구조를 만들 수도 있습니다.

아래 예시는 보고서 작성 태스크를 위한 프롬프트입니다. XML 태그를 통해 구조를 명확히 표현하고, 각 구조별 글자수 제한을 메타데이터를 통해 표현할 수 있습니다.

글자수 제약이 있는 보고서 작성 프롬프트 예제:

```
Python
system_prompt = """<role>You are a business analyst.</role>
<instructions>
  <rule1>Generate a structured report with plain text following the hierarchy
  and 'max_chars' limits</rule1>
  <rule2>Count characters in each section and ensure none exceed their limits.
  If any section exceeds the limit, revise it to fit.</rule2>
  <rule3>Do not include character counts in your response.</rule3>
```

```
</instructions>
<report_structure>
  <summary max_chars="100">brief overview</summary>
  <key_finding max_chars="100">main insight</key_finding>
  <recommendation max_chars="100">suggested action</recommendation>
</report_structure>"""

user_prompt = """
Write a report about AI industry trends.
"""
```

답변:

```
Python
**Summary**
AI industry growth driven by generative AI, cross-industry adoption, and
ethical framework development.

**Key Finding**
Generative AI adoption surged 40% in 2023, revolutionizing content creation
and automation.

**Recommendation**
Prioritize ethical AI integration and upskill workforce to leverage
generative AI advancements sustainably.
```

4. 세부 요소별 작성법

4.1 Role (역할) 정의

역할 정의는 모델의 응답 품질과 일관성에 결정적인 영향을 미칩니다. solar-pro2는 역할에 따른 전문 지식과 관점을 매우 잘 적용하므로, 구체적이고 명확한 역할 설정이 중요합니다.

4.1.1 도메인 특화 전문가

전문 영역의 깊이 있는 지식과 경험이 필요한 작업에는 해당 분야의 실제 직업군을 모델에게 부여하세요. 아래 예제는 응급의학과 전문의라는 구체적인 역할을 부여하여 의학적 전문성을 활성화합니다. 15년 경력과 세부 전문 영역을 명시함으로써 모델이 응급상황에서의 체계적 진단 접근법과 임상 프로토콜을 적용하도록 유도합니다.

응급의학과 전문의 프롬프트 예제:

```
Python
system_prompt = """
# Role

You are a board-certified emergency medicine physician with 15 years of
clinical experience. You specialize in rapid assessment, differential
diagnosis, and emergency treatment protocols.

# Instructions
Provide brief medical assessment:
- Top 3 differential diagnoses
- Immediate action needed.
"""

user_prompt = """
```

A 45-year-old patient presents with chest pain, shortness of breath, and sweating. What's your initial assessment?

""

답변:

Python

****Summary****

AI industry growth driven by generative AI, cross-industry adoption, and ethical framework development.

****Key Finding****

Generative AI adoption surged **40% in 2023**, revolutionizing content creation and automation.

****Recommendation****

Prioritize ethical AI integration and upskill workforce to leverage generative AI advancements sustainably.

4.1.2 태스크 수행 전문가

특정 전문직이 존재하지 않는 작업이라도, 해당 태스크에 특화된 전문성을 가진 가상의 전문가 페르소나를 창조하여 부여하여 모델에게 맥락을 제공해줄 수 있습니다.

키워드 생성 프롬프트 예제:

```
Python
system_prompt = """
# Role

You are a search optimization specialist with expertise in keyword
extraction and information retrieval.

# Instructions

Extract 3 main keywords from the query and return them as a simple list
without any explanation:

1. [primary keyword]
2. [secondary keyword]
3. [related term]

Focus on the most important search terms for optimal information retrieval.
"""

user_prompt = """
I need help with machine learning algorithms for natural language processing
in Python.
"""
```

답변:

```
Python
1. machine learning algorithms
2. natural language processing
3. Python
```

4.2 Instructions (지시사항)

지시사항은 작업의 성공을 좌우하는 핵심 요소입니다. solar-pro2는 명확하고 구체적인 지시에 매우 잘 반응하므로, 모호함을 제거하고 단계별로 세분화된 지시를 제공해야 합니다.

4.2.1 요구사항 세분화

요구사항을 모델이 이해할 수 있도록 최대한 구체적으로 작성하는 것이 중요합니다. 만약 태스크의 시나리오가 다양하다면 User Input이나 Context의 유형에 따라 따라야 할 각각의 가이드를 작성해야 합니다. 모호한 지시문 대신 측정 가능하고 실행 가능한 기준을 제시하여 일관된 결과물을 보장하고, 다양한 상황에 대응할 수 있는 세부 가이드라인을 포함시켜야 합니다.

아래 프롬프트는 제목 생성 작업을 콘텐츠 유형별로 세분화한 예제입니다. 각 유형별로 구체적인 패턴 공식, 초점 영역, 톤, 글자 수 제한을 명시하여 정성적 요구사항인 "좋은 제목"을 구체적인 기준으로 변환하여 가이드를 구체화합니다.

글 유형별 제목 생성 프롬프트 예제:

```
Python
system_prompt = """
# Role
You are a title optimization specialist for different content platforms.

# Instructions
Create titles based on content type:

## Educational Titles
- Pattern: "How to [skill]" or "[Number] Steps to [outcome]"
- Focus: Clear learning outcomes, instructional tone

## Blog Titles
- Pattern: "[Number] [power word] [topic] [benefit]"
```

- Focus: Curiosity, engagement, personal benefit

News Titles

- Pattern: "[Who] [action] [what] [impact]"
- Focus: Facts, immediacy, authority

Constraints

- Title must be under 100 characters
- Return only the title matching the requested type.

"""

user_prompt = """

Working from home during the holiday season presents unique productivity challenges. Family gatherings, increased distractions, and holiday stress can derail your focus. However, with strategic time-blocking techniques, you can maintain peak performance. Set clear boundaries with family members about work hours, create a dedicated workspace away from holiday activities, and use the Pomodoro technique to stay focused during short bursts. These proven strategies help remote workers maintain productivity even during the busiest holiday periods.

"""

답변:

Python

5 Proven Strategies to Crush Holiday Productivity While Working Remotely

4.2.2 구체적인 수행 방법 제시

복잡하고 어려운 작업을 단계별로 분해하여 정확한 실행 방법을 가이드합니다. 아래 프롬프트는 한국어 숫자 단위를 영어로 변환하는 번역 작업 예제입니다. 한국어 숫자 단위(만, 억, 조) 식별, 기본 변환 공식과 복합 숫자 분해 방법 제시, 수학적 정확성 검증을 통해 체계적인 변환 과정을 안내합니다. 이러한 구체적인 수행 방법을 제시하여 모든 한국어 숫자 단위를 정확하게 영어로 변환할 수 있습니다.

숫자 번역 프롬프트 예제:

```
Python
system_prompt = """
# Role
You are a Korean-English business translator specializing in financial documents.

# Instructions
Translate Korean text to English with precise number unit conversion:

### Korean Number Units
- 만(10,000) → thousand format or exact number
- 억(100,000,000) → million
- 조(1,000,000,000,000) → trillion

### Complex Number Conversion
Break down compound numbers: 15조2천억 = 15조 + 2천억 = 15.2 trillion

### Format Requirements
- Use standard English notation (15.2 trillion, not 15,200 billion)
- Include commas for thousands when appropriate
```



```
CRITICAL: Convert ALL Korean numbers accurately. Return only the translated sentence.
```

```
"""
```

```
user_prompt = """
```

```
테크코퍼레이션은 올해 매출 8조3천억 원을 달성했으며, 이는 전년 대비 1조2천억 원 증가했습니다. AI 사업부가 2조7천억 원, 클라우드 사업부가 3천5백억 원의 수익을 올렸습니다. 순이익은 1조8천억 원을 기록했고, 연구개발에 4천2백억 원을 투자했습니다. 신규 채용으로 2만5천 명을 고용했으며, 평균 연봉은 8천5백만 원입니다.
```

```
"""
```

답변:

```
Python
```

```
Tech Corporation achieved sales of 8.3 trillion won this year, an increase of 1.2 trillion won compared to the previous year. The AI division generated 2.7 trillion won in revenue, and the cloud division generated 350 billion won. Net profit recorded 1.8 trillion won, and 420 billion won was invested in research and development. The company hired 25,000 new employees through new recruitment, and the average annual salary is 85 million won.
```

4.3 Constraints (제약사항)

제약사항은 출력의 품질과 일관성을 보장하는 중요한 요소입니다. solar-pro2에서는 명확한 제약사항이 복잡한 요구사항을 정확히 충족시키도록 가이드할 수 있습니다.

4.3.1 긍정적 지시와 부정적 지시 분리

해야 할 것과 하지 말아야 할 것을 명확히 구분하여 제시합니다. 아래는 뉴스레터 생성 태스크를 위한 프롬프트 예제입니다. 서비스 환경상 마크다운 출력이 아닌 플레인 텍스트로만 출력해야 하는 제약사항이 있다면, 다음과 같이 긍정적 지시와 부정적 지시를 분리하여 작성할 수 있습니다.

뉴스레터 작성 프롬프트 예제:

```
Python
system_prompt = """
# Role
You are a newsletter writer creating content for email distribution.

# DO:
- Use clear headlines and numbered lists
- Only use plain text
- Keep sentences short and conversational

# DON'T:
- NEVER use bold, italics, or # headers
- NEVER use bullet points with -, *, or + symbols
- NEVER use emojis
- NEVER add markdown formatting
"""

user_prompt = """
Create a tech newsletter about this week's AI developments.
"""
```

답변:

```
Python
Hi there,
```

Here are five key trends shaping the AI industry this quarter:

1. Generative AI expands into enterprise tools

Companies are integrating generative AI into project management software, CRM platforms, and design applications to automate content creation and data analysis. Expect more niche tools targeting specific industries like legal or healthcare.

2. Smaller, faster models challenge big AI

Efficiency-focused models like Microsoft's Phi-3 and Mistral's 7B are gaining traction for their ability to run locally on consumer hardware while matching performance of larger systems in specialized tasks.

3. AI regulation gains momentum

The EU's AI Act has entered final negotiations, with the US and UK also drafting sector-specific guidelines. Businesses should prepare for increased scrutiny around transparency and bias mitigation.

4. Multimodal AI breaks into mainstream apps

Tools combining text, image, and audio understanding—like OpenAI's GPT-4o—are being embedded in customer service chatbots, virtual assistants, and educational platforms.

Stay tuned for deeper dives into these topics next week. Let us know what you'd like us to cover!

Best

[Your Name]

4.3.2 우선순위가 있는 제약사항

지시사항 및 제약사항이 많아 상충할 수 있는 요구사항들의 우선순위를 명확히 설정합니다. 복잡한 태스크에서는 여러 목표가 동시에 달성되기 어려운 상황이 발생하므로, 명확한 우선순위 체계를 통해 일관된 의사결정 기준을 제공해야 합니다. 이는 모델이 딜레마 상황에서 적절한 선택을 할 수 있도록 도우며, 결과물의 품질과 일관성을 보장합니다.

중요도 순서 표현:

- CRITICAL / HIGHEST / MOST IMPORTANT → HIGH → MEDIUM → LOW / LOWEST
- Priority 1 (절대적) → Priority 2 (중요) → Priority 3 (선택적)
- Non-negotiable (타협 불가) → Essential (필수) → Preferred (선호) → Optional (선택)

데이터 표 변환 프롬프트 예제:

```
Python
system_prompt = """
You are a data formatter converting text to structured tables.

## Formatting Guidelines

### PRIORITY 1: Only response in markdown table format(CRITICAL)
- NEVER add title of the table
- Do not add any explanation text

### PRIORITY 2: Table Format Compliance
- All numerical data MUST be in markdown table format
- Use exact syntax: | Header | Value | with proper alignment
- Include every single number from source text

### PRIORITY 3: Complete Coverage
- No information from source should be omitted
- Add category groupings if needed
```

Convert the provided text into structured table format.

"""

user_prompt = """

GlobalTech reported exceptional Q4 performance with total revenue reaching \$89.4 billion, representing 15.2% year-over-year growth. Operating income was \$24.7 billion, up 18.5% from last year. Net income climbed to \$19.3 billion, a 22.1% increase. The company's profit margin improved to 21.6%, while return on equity reached 28.9%. Cash flow from operations totaled \$32.1 billion.

"""

답변:

Python

Header	Value
Total revenue	\$89.4 billion
Year-over-year growth	15.2%
Operating income	\$24.7 billion
Increase from last year	18.5%
Net income	\$19.3 billion
Increase	22.1%
Profit margin	21.6%
Return on equity	28.9%
Cash flow from operations	\$32.1 billion

4.4 Format (출력 형식)

출력 형식의 정의는 결과물의 활용도와 일관성을 결정합니다. 기대하는 출력 형식을 구조적 템플릿과 예제를 활용하여 가이드할 수 있습니다. 명확한 형식 지정은 후속 처리나 시스템 통합을 용이하게 하고, 사용자가 일관된 품질의 결과물을 기대할 수 있도록 돕습니다.

4.4.1 구조적 템플릿 제시

기대하는 출력물의 형태에서 모델이 응답하여 채워 넣어야 하는 부분을 대괄호를 사용해 가이드합니다. 이 방법은 모델에게 정확한 구조와 각 섹션에 포함되어야 할 내용의 성격을 명확히 전달하며, 결과물의 일관성과 완성도를 보장합니다. 대괄호 안의 설명은 해당 위치에 들어갈 내용의 유형과 목적을 구체적으로 제시하여 모델이 적절한 정보를 생성할 수 있도록 돕습니다.

출력 템플릿 프롬프트 예제:

```
Python
system_prompt = """
Create a brief business summary using this template:
## 🛠️ Issue: [main problem]
## 💡 Impact: [consequence]
## ✅ Solution: [recommended action]

Keep each section to 1-2 sentences."""

user_prompt = """
Situation: Website loading speed decreased by 40%, causing 15% drop in user
engagement.
"""
```

답변:

```
Python
## 🛠️ Issue: Website loading speed has decreased by 40%, significantly
slowing user access.
## 💡 Impact: The slowdown has caused a 15% drop in user engagement, risking
revenue and retention.
## ✅ Solution: Optimize images, enable caching, and upgrade hosting to
improve load times and recover engagement.
```

4.4.2 JSON 출력 형식

필요한 형식과 모델이 채워 넣어야 하는 부분을 가이드합니다. 출력물 이외의 다른 텍스트를 포함하지 않도록 지시하는 것이 중요합니다. 특히 "Respond ONLY the JSON object"와 같은 제약사항을 명시하여 추가적인 설명이나 주석이 포함되지 않도록 하고, 시스템이 직접 처리할 수 있는 순수한 데이터 형태로 출력되도록 보장해야 합니다.

JSON 출력 가이드 프롬프트 예제:

```
Python
system_prompt = """
Extract key information and return as JSON only:
{
  "company": "company name",
  "revenue": "revenue amount",
  "growth": "growth percentage"
}

Return ONLY the JSON object."""
```

```
user_prompt = """
TechCorp reported $2.5 billion revenue, up 18% from last year.
"""
```

답변:

```
Python
{
  "company": "TechCorp",
  "revenue": "$2.5 billion",
  "growth": "18%"
}
```

4.4.3 Few-Shot Learning

Few-Shot Learning은 모델에게 소수의 구체적인 예제를 제시하여 원하는 출력 패턴을 학습시키는 기법입니다. 긴 설명 대신 실제 입출력 예제를 통해 태스크의 요구사항과 기대 결과를 직관적으로 전달할 수 있어 매우 효과적입니다. 구체적이고 실제적인 예제를 통해 기대하는 출력 패턴을 학습시키며, 간단한 예제부터 복잡한 상황까지 단계적으로 제시하여 학습 효과를 극대화합니다.

- **점진적 복잡도:** 기본적인 단순 케이스부터 시작하여 예외 상황이나 복합적인 문제까지 단계적으로 난이도를 높여 모델의 학습 곡선을 최적화합니다.
- **유형별 분류:** 오류나 상황의 유형별로 예제를 그룹화하여 각 카테고리의 특성과 처리 방법을 명확히 학습시킵니다.
- **실용적 예제:** 실제 서비스 환경에서 발생할 수 있는 구체적이고 현실적인 상황을 예제로 사용하여 모델의 실용성을 높입니다.

- **처리 가이드 포함:** 단순히 입출력 예제만 제시하는 것이 아니라, 각 유형별로 어떻게 접근하고 처리해야 하는지에 대한 구체적인 지침을 함께 제공합니다.

아래는 간단한 교열기 프롬프트 예제입니다. 교열해야하는 각 에러별로 가이드를 나누고, 가이드별로 적절한 예제를 포함하여 모델이 다양한 오류 유형을 체계적으로 인식하고 수정할 수 있도록 돕습니다.

간단한 한국어 교열 프롬프트 예제:

```
Python
system_prompt = """
# Role
You are a professional Korean proofreader specializing in business and
formal documents.

# Instructions
Correct Korean text based on error type.
Return only the corrected sentence without explanations.

# 맞춤법 오류
- 올바른 한글 맞춤법으로 수정
Input: "안녕하세요. 회의 일정을 알려드리겠습니다."
Output: "안녕하세요. 회의 일정을 알려드리겠습니다."

# 띄어쓰기 오류
- 표준 띄어쓰기 규칙 적용
Input: "다음주월요일에 회의가있습니다."
Output: "다음 주 월요일에 회의가 있습니다."

# 존댓말 통일
- 일관된 높임말 사용으로 수정
Input: "보고서를 검토해 주세요. 그리고 피드백 좀 줘."
Output: "보고서를 검토해 주세요. 그리고 피드백을 부탁드립니다."
```

문장 구조 개선

- 명확하고 자연스러운 문장으로 다시 작성

Input: "그 것은 우리가 해야 할 일이 맞다고 생각이 됩니다."

Output: "그것은 우리가 해야 할 일이라고 생각합니다."

어색한 표현 수정

- 자연스러운 한국어 표현으로 개선

Input: "미팅을 진행하겠습니다."

Output: "회의를 진행하겠습니다."

"""

user_prompt = """

Input : 안녕하세요. 다음주화요일에회의를개최할예정이니 참석해줘. 그리고 자료도 아직준비하세요.

Output :

"""

답변:

Python

안녕하세요. 다음 주 화요일에 회의를 개최할 예정이니 참석해 주시기 바랍니다. 그리고 자료도 아직 준비하지 않으셨으니 준비해 주세요.

5. 꼭 따라야하는 지시사항 강조하기

중요한 규칙이나 제약사항이 무시되지 않도록 특별한 강조 방법 및 강조 단어를 사용하여 모델의 주의를 집중시킵니다. 일반적인 지시사항과 달리 절대적으로 준수해야 하는 요구사항은 시각적 강조와 반복을 사용합니다.

시각적 강조 방법

- **대문자 사용:** CRITICAL, MANDATORY, WARNING, IMPORTANT로 중요도 표시
- **구분선과 박스:** #, ---, ===를 활용해 중요 섹션 분리
- **반복 강조:** 핵심 지시사항을 프롬프트 시작과 끝에 재배치
- **번호와 체크리스트:** 필수 요구사항을 순서대로 나열하여 누락 방지

강조 단어 활용

- **절대적 준수:** CRITICAL, MANDATORY, REQUIRED, MUST, NEVER, ALWAYS
- **경고 표시:** WARNING, ALERT, CAUTION, FAILURE TO COMPLY
- **우선순위:** PRIORITY, ESSENTIAL, VITAL, CRUCIAL, NON-NEGOTIABLE
- **검증 유도:** VERIFY, CHECK, CONFIRM, VALIDATE, ENSURE

5.1 셀프 검증 (Self-Verification)

셀프 검증은 모델이 작업을 완료한 후 스스로 결과의 정확성과 완성도를 점검하도록 유도하는 기법입니다. 이는 인간이 중요한 업무를 마친 후 재검토하는 과정과 유사하게, 모델에게도 체계적인 확인 절차를 수행하도록 지시하여 오류를 사전에 발견하고 수정할 수 있게 합니다. 모델이 작업 필수 요구사항들을 하나씩 확인할 수 있는 구체적인 체크리스트를 제공하고, 각 항목의 준수 여부를 점검하도록 프롬프트에 명시합니다.

환율 계산 프롬프트 예제:

```
Python
system_prompt = """
Convert USD amounts to multiple currencies using specified exchange rates.

# Exchange Rates (Use these exact rates)
- 1 USD = 1,320 KRW (Korean Won)
- 1 USD = 150 JPY (Japanese Yen)
- 1 USD = 7.2 CNY (Chinese Yuan)

# Format Requirements
- USD: $ symbol with commas and 2 decimals
- KRW: ₩ symbol with commas, no decimals
- JPY: ¥ symbol with commas, no decimals
- CNY: ¥ symbol with commas and 2 decimals

# VERIFICATION CHECKLIST
Before submitting your response, verify:
1. USD amount × 1,320 = KRW calculation is correct
2. USD amount × 150 = JPY calculation is correct
3. USD amount × 7.2 = CNY calculation is correct
4. All currency symbols are properly used (₩, ¥, ¥, $)
5. KRW and JPY show no decimal places
6. USD and CNY show exactly 2 decimal places
7. All amounts include proper comma separators

CRITICAL WARNING: Failure to follow these formatting rules will result in
unusable output for financial systems.

Convert the USD amount and return only the converted amounts without any
explanations.
"""
```

```
user_prompt = """
$2,850.75 USD
"""
```

답변:

```
Python
$2,850.75
₩3,762,990
¥427,613
¥20,525.40
```

5.2 맥락 관리 (Context Management)

5.2.1 긴 맥락에서의 핵심 정보 배치

긴 맥락에서의 핵심 정보 배치를 통해 중요한 지시사항이 무시되지 않도록 합니다. 중요한 지시사항을 프롬프트 시작과 끝에 배치하여 정보 처리 과정에서 지속적으로 상기시킬 수 있습니다.

아래 프롬프트는 {DOCUMENT}라는 변수에 저장된 9000자 이상의 긴 재무 문서에서 수치 데이터만 정확히 추출하는 RAG 시스템 예제입니다. 문서 처리 시 핵심 제약사항인 "숫자만 추출"과 "출처 인용 필수"를 프롬프트 시작과 끝에 배치하여 긴 문서 맥락에서도 지시사항을 잘 따를 수 있습니다.

**DOCUMENT 변수에 저장된 문서 예시는 코드 파일에 포함되어 있습니다.*

재무 문서 수치 추출 프롬프트 예제:

```
Python
system_prompt = """
=== CRITICAL: Extract Numbers Only ===
```

```
# Role
You are a financial data extraction specialist.

# Context Documents
<document>
{DOCUMENT}
</document>

# Instructions
Extract ONLY numerical data from the provided document. Return numbers with
their units but NO explanatory text.

# MANDATORY Format
- Use this exact format: Revenue: $34.7 billion (Source: Executive Summary)
- Include dollar amounts, percentages, and growth rates ONLY
- No sentences, descriptions, or analysis
- One number per line

=== REMINDER: Numbers Only ===
Extract financial figures without any explanatory text or analysis.
"""

user_prompt = """
Extract TechCorp's 2024 total revenue, growth rate, and revenue breakdown by
division with their growth rates.
"""
```

답변:

```
Python
Revenue: $34.7 billion (Source: Executive Summary)
```

Growth Rate: 28% (Source: Executive Summary)

Software Division Revenue: \$12.8 billion (Source: Revenue Breakdown by Division)

Software Division Growth Rate: 35% (Source: Revenue Breakdown by Division)

Hardware Division Revenue: \$10.2 billion (Source: Revenue Breakdown by Division)

Hardware Division Growth Rate: 22% (Source: Revenue Breakdown by Division)

Cloud Services Revenue: \$7.8 billion (Source: Revenue Breakdown by Division)

Cloud Services Growth Rate: 41% (Source: Revenue Breakdown by Division)

Consulting Services Revenue: \$3.9 billion (Source: Revenue Breakdown by Division)

Consulting Services Growth Rate: 18% (Source: Revenue Breakdown by Division)

6. 자주 발생하는 문제와 해결책

6.1 모호한 응답

문제: 명확하지 않거나 일관성 없는 답변

분석 기준이 불명확한 프롬프트로 인해 모델이 일관성 없고 모호한 응답을 제공하는 문제 상황입니다. 시스템 프롬프트에서 단순히 "데이터를 분석하고 인사이트를 제공하라"는 추상적 지시만 제공하고, 유저 프롬프트에서도 구체적인 맥락이나 분석 기준 없이 "우리 매출에 대해 어떻게 생각하는가"라는 모호한 질문을 던집니다. 이로 인해 모델은 어떤 관점에서 분석해야 할지, 무엇을 기준으로 평가해야 할지 알 수 없어 동일한 데이터에 대해서도 실행할 때마다 다른 응답을 생성하거나, 구체적이지 않은 일반론적 답변을 제공하게 됩니다.

문제 프롬프트 예제:

```
Python
system_prompt = "Analyze the data and provide insights in three korean sentences."
user_prompt = "What do you think about our sales? Revenue: $500K, customers: 200, churn: 5%"
```

답변:

```
Python
매출 $500K와 고객 200명을 고려할 때, 고객당 평균 수익은 $2,500으로 상당히 높은 편입니다.
5%의 낮은 이탈률은 고객 유지 전략이 효과적으로 작동하고 있음을 시사합니다.
다만, 고객 확장 또는 업셀링 기회를 모색하면 매출 성장을 더욱 가속화할 수 있을 것입니다.
```

해결: 구체적인 지시사항 및 충분한 맥락 제공

모호한 응답은 불충분한 지시사항으로 모델이 사용자의 요구사항을 정확히 파악하지 못했을 때 발생합니다.

모델이 수행해야 하는 작업을 구체적이고 세분화된 지시사항으로 제공하는 것이 핵심입니다.

또한 모델에게 충분한 맥락을 제공하는 것이 중요합니다. 단순히 데이터만 제시하는 것이 아니라 비즈니스 상황, 업계 기준, 목표 등 모델이 작업을 수행하기 위해 필요한 배경 정보를 포함시켜야 합니다.

해결 프롬프트 예제:

```
Python
system_prompt = """
# Role: Sales Performance Analyst

# Required Analysis (exactly 3 korean sentences):
```


1. **Revenue per Customer**: Calculate and assess ($\$500K \div 200$ customers)
2. **Churn Impact**: Evaluate 5% churn rate and monthly revenue loss
3. **Priority Action**: Recommend most critical improvement with specific target

Use specific numbers and clear recommendations.

"""

```
user_prompt = """
## Current Metrics
- Monthly revenue: $500K
- Active customers: 200
- Monthly churn rate: 5%

## Business Context
- Industry: B2B SaaS (project management tools)
- Company stage: Series A startup
- Target: $1M monthly revenue by year-end

## Question
What's your performance assessment with specific recommendations?
"""
```

답변:

Python

1. **Revenue per Customer**: 현재 고객당 월 매출은 $\$2,500 (\$500K \div 200\text{명})$ 로, 목표 매출($\$1M$) 달성 시 400명 고객 확보가 필요합니다.
2. **Churn Impact**: 월 5% 이탈률로 인해 매월 $\$25K (\$500K \times 5\%)$ 의 수익이 손실되며, 이탈률 1%p 감소 시 연간 $\$300K$ 추가 수익 창출이 가능합니다.

3. ****Priority Action****: ****이탈률 3%로 감축****을 최우선 과제로 권장하며, 이를 위해 고객 성공 팀 강화(CSM 1명 추가 채용) 및 주요 고객 대상 맞춤형 온보딩 프로그램 도입이 필요합니다. (현재 200명 기준 CSM 1명당 100명 담당 → 50명으로 조정)

6.2 일관적이지 않은 출력 형식

문제: 요청한 형식과 다른 출력

JSON 형식이나 특정 구조로 출력을 요청했음에도 불구하고 모델이 일반 텍스트나 다른 형식으로 응답하는 문제입니다. 특히 API 연동, 데이터 처리, 자동화 시스템에서 정확한 형식은 필수적이지만, 모호한 형식 지시로 인해 파싱 오류나 시스템 중단이 발생할 수 있습니다. 예를 들어 "JSON으로 응답해달라"고 요청했지만 모델이 설명 텍스트와 함께 출력하거나, 요청한 키 구조와 다른 형태로 응답하는 경우입니다.

문제 프롬프트 예제:

```
Python
system_prompt = "Return the analysis in JSON format"

user_prompt = "Analyze customer feedback: "Great product but expensive pricing" - Rating: 4/5"
```

답변:

```
Python
Here is the analysis of the customer feedback in JSON format:

```json
{
```

```

"feedback": "Great product but expensive pricing",
"rating": {
 "score": 4,
 "out_of": 5,
 "sentiment": "positive"
},
"aspects": [
 {
 "aspect": "Product Quality",
 "sentiment": "positive",
 "confidence": 0.95,
 "keywords": ["Great product"]
 },
 {
 "aspect": "Pricing",
 "sentiment": "negative",
 "confidence": 0.85,
 "keywords": ["expensive pricing"]
 }
],
"overall_sentiment": "mixed",
"recommendation": "Consider addressing pricing concerns while maintaining
product quality to improve overall satisfaction"
}
...

```

### ### Breakdown of the analysis:

1. **Overall Sentiment**: Mixed (positive product experience but negative pricing sentiment)
2. **Product Quality**: Strongly positive ("Great product")
3. **Pricing**: Negative concern ("expensive pricing")
4. **Rating**: 4/5 indicates high satisfaction despite one major drawback

5. **Actionable Insight**: The business should explore ways to justify the pricing (e.g., adding value) or consider tiered pricing options while maintaining the product's quality.

### 해결: 정확한 구조 명세 및 구체적인 예시 제공

형식 불일치는 출력 요구사항이 불명확하거나 예시가 부족할 때 자주 발생합니다. 단순히 "JSON으로 응답하라"가 아닌 키 이름, 데이터 타입, 필수/선택 필드를 명확히 정의하고, 추가 설명 텍스트 금지와 같은 제약사항을 명시해야 합니다.

"Output format example"을 제공하는 것만으로도 모델에게 정확한 출력 구조, 키 순서, 값 형태에 대한 명확한 가이드라인을 전달할 수 있어 일관성을 크게 향상시킬 수 있으며, temperature를 낮게 설정하여 형식 준수를 더욱 엄격하게 요구할 수 있습니다.

### 해결 프롬프트 예제:

```
Python
system_prompt = """
Return the analysis in JSON format without any explanation.
Return ONLY a valid JSON object with these EXACT keys:
{
 "sentiment_score": number (1-5 scale),
 "sentiment_label": string ("positive" or "negative" or "neutral"),
 "main_topics": array of strings (max 3 items),
 "improvement_area": string (single most critical issue),
 "confidence_level": number (0.0-1.0)
}
#Output Format Example
```

```

{"sentiment_score":3,"sentiment_label":"neutral","main_topics":["Service
Speed","Staff Attitude"],"improvement_area":"Response
Time","confidence_level":0.78}
"""

```

```

user_prompt = "Analyze customer feedback: "Great product but expensive
pricing" - Rating: 4/5"

```

**답변:**

```

Python
{"sentiment_score":4,"sentiment_label":"positive","main_topics":["Product
Quality","Pricing"],"improvement_area":"Pricing","confidence_level":0.95}

```

## 6.3 지시사항 불이행

**문제:** 요청한 구성 요소나 필수 섹션이 누락된 불완전한 답변

지시사항 불이행은 복잡한 요청에서 일부 요구사항이 간과되거나 체계적인 확인 과정이 없을 때 발생합니다. 아래 문서 윤문 예시에서도 어미 변경, 용어 교체, 외래어 변환 등 다양한 편집 규칙을 동시에 적용해야 하므로 모델이 일부 규칙만 수행하기 쉽습니다. ‘매니저’, ‘프로젝트’ 단어를 적절한 한국 명사로 변환하지 않았으며, 윤문한 결과 이외의 텍스트를 포함하여 출력하는 오류가 있습니다.

## 문제 프롬프트 예제:

```
Python
system_prompt = "Edit the text to formal style, change all endings to
습니다/됩니다, and convert every English terms to Korean without any
explanation."

user_prompt = "우리 회사의 매니저들은 매주 미팅을 진행해서 프로젝트 진행 상황을 체크하고 있어요. 각
팀의 리포트를 검토하고, 데드라인에 맞춰서 솔루션을 준비하고 있어요. 다음 주까지 새로운 스케줄을 정리해서
회사 전체에 공유할 예정이에요."
```

## 답변:

```
Python
본사의 매니저들은 매주 회의를 진행하여 프로젝트 진행 상황을 확인하고 있습니다. 각 팀의 보고서를 검토하며,
마감일에 맞춰 해결 방안을 준비하고 있습니다. 다음 주까지 새로운 일정을 정리하여 회사 전체에 공유할
예정입니다.
```

(수정 사항)

- "미팅" → "회의"
- "체크" → "확인"
- "리포트" → "보고서"
- "데드라인" → "마감일"
- "솔루션" → "해결 방안"
- "스케줄" → "일정"
- 모든 서술어 끝을 "-합니다" 형식으로 통일하여 공식적 어투 적용

## 해결: 작업 순서 및 검증 절차 포함

지시사항 불이행은 복잡한 요청에서 일부 요구사항이 간과되거나 체계적인 확인 과정이 없을 때 발생합니다. 다양한 규칙을 동시에 적용해야 하는 복잡한 태스크의 경우, 모델이 일부 규칙만 수행하기 쉽습니다. 효과적인 해결을 위해서는 각 편집 작업을 명확히 구분하여 번호를 매기고, 구체적인 작업 예시를 제공하여 혼동을 방지해야 합니다. 또한 체크리스트를 통해 모델이 후 모든 규칙의 적용 여부를 자체 검증하도록 유도합니다.

### 해결 프롬프트 예제:

```
Python
system_prompt = """
MANDATORY Editing Requirements
Apply ALL 4 editing rules to the text:

REQUIRED EDITING TASKS (ALL must be applied):
1. **Formality**: Change all informal endings (해요/어요) to formal
(습니다/됩니다)
2. **Term Replacement**: Replace "회사" with "기업" everywhere
3. **Foreign Words**: Convert ALL foreign terms to Korean (check every word
carefully):
- 미팅 → 회의
- 프로젝트 → 사업
- 매니저 → 관리자
- 스케줄 → 일정
4. **Style**: Maintain professional business tone throughout

VERIFICATION CHECKLIST
Before submitting, confirm ALL applied:
☐ All endings changed to 습니다/됩니다
☐ All foreign terms converted to Korean
☐ Professional tone maintained
☐ Only edited text provided

```

```
OUTPUT FORMAT:
[Only response with edited text]
"""
```

```
user_prompt = "우리 회사의 매니저들은 매주 미팅을 진행해서 프로젝트 진행 상황을 체크하고 있어요. 각 팀의 리포트를 검토하고, 데드라인에 맞춰서 솔루션을 준비하고 있어요. 다음 주까지 새로운 스케줄을 정리해서 회사 전체에 공유할 예정이에요."
```

**답변:**

```
Python
우리 기업의 관리자들은 매주 회의를 진행하여 사업 진행 상황을 확인하고 있습니다. 각 팀의 보고서를 검토하고, 마감일에 맞춰 해결책을 준비하고 있습니다. 다음 주까지 새로운 일정을 정리하여 기업 전체에 공유할 예정입니다.
```

## 6.4 할루시네이션

**문제:** 주어진 정보를 넘어선 사실 제공 혹은 잘못된 정보 제공

고객 문의에 대한 챗봇 응답에서 명확한 원인을 파악할 수 없는 상황임에도 모델이 추측에 기반해 확정적인 원인을 제시하거나 구체적인 해결방법을 단언하는 문제입니다. 예를 들어 "앱이 사진 업로드 시 종료된다"는 피드백에 대해 충분한 정보가 없음에도 특정 원인을 확정적으로 제시하고 임의의 해결책을 단언하여 고객에게 잘못된 기대를 주거나 부적절한 해결 시도로 유도할 수 있습니다.

**문제 프롬프트 예제:**

```
Python
system_prompt = """
```



You are a customer support chatbot for a mobile app.

# Known Technical Issues Context:

- Login failures often occur due to incorrect credentials, expired sessions, browser cache issues, server maintenance, or account lockouts
- Typical solutions include password reset, cache clearing, trying different browsers, or checking account status

Provide helpful troubleshooting advice based on the customer's issue.

"""

user\_prompt = """

Customer message: "The app keeps crashing every time I try to upload a photo. What's wrong?".

"""

**답변:**

Python

I'm sorry to hear you're experiencing this issue! Let's troubleshoot the photo upload crash step by step:

#### 1. **\*\*Check App Updates\*\***

Ensure you're using the latest app version. Outdated versions may have unresolved bugs—update via your device's app store.

#### 2. **\*\*Free Up Storage Space\*\***

Low storage can cause crashes during uploads. Delete unnecessary files or apps to free up space, then [try](#) again.

#### 3. **\*\*Try a Smaller/Simpler Photo\*\***

Large files or high-resolution images may overwhelm the app. Test [with](#) a smaller photo or compressed file (e.g., JPEG instead of RAW).

#### 4. **\*\*Clear App Cache\*\***

- Go to **\*\*Settings > Apps > [Your App Name] > Storage > Clear Cache\*\***.
- If the issue persists, [try](#) **\*\*Clear Data\*\*** (note: this will log you out).

#### 5. **\*\*Restart Your Device\*\***

Background processes might interfere. Restart your phone/tablet and reopen the app.

#### 6. **\*\*Check Internet Connection\*\***

A unstable connection during uploads can trigger crashes. Ensure you're on a strong Wi-Fi/[4G](#) network.

#### 7. **\*\*Reinstall the App\*\***

If crashes [continue](#), uninstall and reinstall the app [from](#) the app store.

#### 8. **\*\*Test on Another Device\*\***

If possible, [try](#) uploading the photo [from](#) a different device to isolate the issue.

If none of these steps work, please share:

- Your **\*\*device model\*\*** and **\*\*OS version\*\***
- Whether the crash happens [with](#) **\*\*all photos\*\*** or specific ones
- A screenshot/error message ([if](#) available)

This will [help](#) us investigate further! 😊

## 해결: 모르는 것 인정하게 하기

할루시네이션은 모델이 제한된 정보를 바탕으로 확실하지 않은 결론을 도출하거나 존재하지 않는 세부사항을 생성할 때 발생합니다. 고객 지원 챗봇에서는 특히 잘못된 정보 제공이 고객 불만으로 이어질 수 있어 신중한 접근이 필요합니다. 효과적인 해결을 위해서는 "정확한 원인을 파악할 수 없다면 명시적으로 불확실성을 인정하라"는 지시를 제공해야 합니다. 주어진 맥락 내에서 답변할 수 없을 때 "I'm not sure of the exact cause"와 같은 한정적 표현을 사용하도록 가이드합니다.

## 해결 프롬프트 예제:

```
Python
system_prompt = """
You are a customer support chatbot for an e-commerce platform.

Known Technical Issues Context:
- Login failures often occur due to incorrect credentials, expired sessions,
 browser cache issues, server maintenance, or account lockouts
- Typical solutions include password reset, cache clearing, trying different
 browsers, or checking account status

VERIFICATION BEFORE RESPONDING:
- Can I identify the specific cause from the customer's message? If NO →
 ONLY respond "Sorry, I don't know the exact cause."
- Am I about to guess or assume? If YES → ONLY respond "Sorry, I don't know
 the exact cause."

IMPORTANT: When the cause is unclear, admit uncertainty rather than
guessing.
"""

user_prompt = """
Customer message: "The app keeps crashing every time I try to upload a
photo. What's wrong?".
```

```
"""
```

답변:

```
Python
Sorry, I don't know the exact cause.
```

## 7. 마무리

solar-pro2는 명확하고 구조화된 프롬프트에 매우 반응적인 모델입니다. 이 가이드의 원칙들을 따라 체계적으로 프롬프트를 설계하면, 모델의 뛰어난 성능을 최대한 활용할 수 있습니다.

핵심은 **명확성**, **구조화**, **구체성**입니다. 모호함 없이 정확한 지시를 제공하고, 논리적인 구조로 프롬프트를 구성하며, 구체적인 예시와 검증 단계를 포함하여 일관된 고품질 결과를 얻으시기 바랍니다.

지속적인 테스트와 개선을 통해 각자의 사용 사례에 최적화된 프롬프트를 개발하시길 권장합니다.

이제 직접 실행해볼 차례입니다. 아래 링크에서 실습 코드를 확인해보세요:

[\[GitHub에서 실습 코드 확인하기\]](#)

# Building intelligence for the future of work

---

비즈니스 고민을 해결할 생성형 AI 모델이 필요하신가요?  
업스테이지의 AI 전문가와 함께 시작하세요!

콘솔에서 바로 시작하기

제품 문의 바로 가기