ChatGPT

GPT-4o를 활용한 한국어 문장 분류 프롬프트 엔지니어링 기법

여러 개의 한국어 문장을 **유형**, **극성**, **시제**, **확실성**과 같은 속성으로 일괄 분류하기 위해서는, 프롬프트 설계를 정교하게 최적화하는 것이 중요합니다. 최근 연구 사례와 실전 보고서에 따르면, **프롬프트 구조**, **출력 형식 지시**, **예시 제공 (Few-Shot)**, **모델 언어 설정**, **한국어 처리상의 유의점** 등 다양한 측면에서 기법이 개발되어 왔습니다. 아래에서는 각 사례의 프롬프트 구조와 설계 이유, 형식 및 내용 정확도를 높이는 기술, 한국어 LLM 분류 시 발생하는 오류와 완화 전략, 그리고 프롬프트 길이와 언어 비율 등에 따른 성능 변화를 정리합니다.

1. 분류 작업에 적합한 프롬프트 구조 설계

① **명확한 지시와 카테고리 제시:** 프롬프트에서는 모델에게 수행할 **작업**을 분명히 지시하고, **분류 범주(라벨)**를 명시적으로 제시하는 것이 핵심입니다. 예컨대 OpenAI GPT 계열을 활용한 분류에서는 다음과 같이 프롬프트를 구성할 수 있습니다:

- 예: “다음 문장의 유형, 극성, 시제, 확실성을 식별하세요. 가능한 유형은 [평서문, 의문문, 명령문, 감탄문], 극성은 [긍정, 부정], 시제는 [과거, 현재, 미래], 확실성은 [확실, 추측]입니다.”

이처럼 **모델이 출력해야 할 카테고리 명칭을 사전에 모두 나열**하면 모델이 임의의 표현 대신 정해진 라벨만 사용하게 되어 출력의 통일성이 높아집니다 [1] [2]. 실제 사례로, Hugman의 프롬프트 예시에서는 “Topic-class must be one of (World, Sports, Business, Sci/Tech)”처럼 분류 라벨들을 괄호 안에 제시하여 모델이 그중 하나를 고르게 했습니다 [1]. Kostas (2023)도 OpenAI GPT-3.5/4 분류 예시에 **Classes 리스트**를 제시해 놓고 “Classify the text into one of the above classes”라고 유도함으로써, 모델이 **지정된 클래스만으로 답하도록** 하였습니다 [3]. 이러한 **명시적 클래스 제시 전략**은 모델이 출력 포맷을 벗어나지 않도록 하는 데 효과적입니다.

② **입력-출력 구조 설정:** 여러 문장을 한꺼번에 분류하려면 **프롬프트 내에서 각 문장을 구분**하고, 모델이 각 문장에 대한 출력을 순서대로 생성하도록 유도해야 합니다. 일반적으로는 **각 문장에 번호를 매겨 나열**하는 방법이 활용됩니다. 예를 들어 감성 분류의 경우 다음과 같은 프롬프트가 사용되었습니다 [4]:

```
Classify sentiment as negative, positive, or neutral:
1. Amazingly crafted and good!
2. This thing broke after the second use.
3. Yes it looks good, but they don't feel good.
4. Can't wait to show them off to my partner!
```

위와 같이 번호를 붙여 여러 문장을 제시하면, GPT-4 계열 모델은 응답에서도 같은 번호 순서로 각 문장의 분류 결과를 나열합니다 [5]. Dr. Ernesto Lee의 GPT-4 API 예시에 따르면, 1~4번 문장에 대해 모델 출력이 `1. Positive`, `2. Negative`, … 형태로 일치되게 나왔습니다 [5]. **문장에 인덱스를 붙여 프롬프트를 구성**하면 모델이 입력과 매칭되는 다중 출력을 내놓기 쉽습니다.

다만, 문장이 아주 많거나 길이가 길어진 경우 한 번의 프롬프트로 모두 처리하면 **중간 일부 문장에 대한 주의력이 떨어지는 문제**가 있을 수 있습니다. 연구에 따르면 LLM은 컨텍스트 창 내 **처음과 끝 부분 정보를 가장 잘 기억하고, 중간 부분 정보는 상대적으로 덜 정확히 처리하는 U자형 경향**이 있습니다 [6]. 따라서 **문장 수가 많다면 배치를 나누거나**, 중요

한 지침은 프롬프트 앞뒤에 배치하는 것이 좋습니다. 또한 각 문장을 가급적 **한 줄에 하나씩 분리**하고, 구두점이나 번호로 잘 구분하여 모델이 혼동하지 않도록 설계합니다.

③ 지시문의 언어 (한국어 vs 영어): 프롬프트 입력 형식을 설계할 때, **지시문을 어떤 언어로 쓰는지**도 성능에 영향을 줄 수 있습니다. 많은 LLM이 영어 데이터를 중심으로 학습되어 있기 때문에, **영어로 명령을 내릴 때 더 안정적인 수행을 보이는 경향**이 보고되었습니다 [7] [8]. 예컨데 아랍어 데이터셋에 대해 GPT-4o 등을 시험한 연구에서는, **프롬프트를 영어로 작성했을 때 성능이 가장 높고, 아랍어(원어)로 작성했을 때는 가장 떨어졌다**고 합니다 [2] [9]. 한국어에 대해서도 GPT-3.5 모델은 영어 지시문 사용 시 약간 정확도가 올라가는 사례가 있었고 [10], GPT-4 계열도 기본적으로 영어 활용도가 높으므로 **지시 자체는 영어로, 분류 대상 문장만 한국어로 제공**하는 전략을 고려할 수 있습니다. (다만 GPT-4는 한국어 이해력이 뛰어나 이러한 차이가 크지 않을 수도 있습니다 [9].) **혼용(Mixed) 프롬프트**도 한 방법인데, 실제 연구에서도 원어+영어를 섞어 쓴 혼용 프롬프트가 순수 원어보다 좋은 결과를 내는 등 긍정적 효과가 관찰되었습니다 [7]. 종합하면 **분류 라벨명을 영문으로 통일**하거나, **지시문은 영어로 간결하게 쓰는 것**이 모델의 이해를 높이고 불필요한 출력(엉뚱한 설명 등)을 줄이는 데 유리할 수 있습니다.

2. 출력 형식의 일관성 확보 기법

① 출력 포맷 명시와 예시: 모델의 **형식 정확도**를 높이려면, 원하는 **출력 포맷**을 구체적으로 알려주어야 합니다. 예를 들어 **"결과는 '유형/극성/시제/확실성' 순서로 콤마로 구분하여 출력하세요."** 혹은 **"아래와 같은 YAML 형식 블록으로만 답하세요:"** 등의 문구를 포함합니다. 프롬프트 내에 **예시 출력 형식**을 하나 넣어주는 것도 매우 효과적입니다 [11]. Huang(2025)는 구조화된 출력 받는 요령으로, **프롬프트에 원하는 출력의 예시 블럭** (예: YAML 또는 JSON)을 직접 제시하는 방법을 강조했습니다 [11]. 실제로 사람 이름과 나이를 추출하는 예시에서, `Example Output Format:` 아래에 `yaml\nname: Example Name\nage: 99\n` 같은 블럭을 보여주고 **"Your YAML output:"**으로 이어가자, 모델이 이를 따라 정확히 YAML 형식으로 답변하도록 유도할 수 있었습니다 [12] [11]. 이처럼 **출력 예시를 프롬프트에 포함**하면 형식 일관성이 크게 향상됩니다.

또한 JSON보다 YAML이 **LLM에게는 형식 오류 가능성이 적다**는 경험칙도 있습니다. JSON은 따옴표나 이스케이프 문자 규칙이 엄격해 모델이 종종 작은 실수를 하는 반면, YAML은 사람이 읽기 쉬운 구조로 줄바꿈이나 인용 부호 처리에 유연하여 LLM이 정확히 출력하기 수월합니다 [13] [14]. 예컨대 인용부호와 개행이 섞인 텍스트를 JSON으로 출력시키면 `\"`나 `\n` 등을 일일이 맞춰줘야 하지만, YAML의 멀티라인 문자열(`|` 사용)로 요청하면 훨씬 깨끗한 출력을 얻을 수 있습니다 [14] [15]. 따라서 **JSON보다 YAML 형식을 지시**하거나, 간단히 `key: value` 열거 형태로 답하게 하는 편이 좋습니다.

② 불필요한 수식어 억제: 분류 작업에서는 **라벨 자체만 깔끔히 출력**하는 것이 중요하므로, 프롬프트에 **"라벨 이외의 불필요한 말은 하지 말 것"**을 명시해야 합니다. 예를 들어 "각 문장에 대해 `[유형], [극성], [시제], [확실성]`만 콤마로 구분하여 출력하세요. 다른 설명은 하지 마세요." 같이 요구합니다. 실전 실험에서는 이러한 지시가 없을 경우, 모델이 "문장의 극성은 긍정입니다."처럼 문장 형태 응답을 하거나 설명을 덧붙이는 사례가 많았습니다 [16]. Hugman의 AG News 분류 예시에서도 **원래는 라벨만 달라고 했지만** GPT-3.5 모델이 "The topic is Sci/Tech, as it discusses..."처럼 문장 형태로 장황하게 답하는 경우가 빈번했습니다 [16]. 이에 따라 출력 처리를 위해 별도의 후처리 코드가 필요했는데, 이상적으로는 **프롬프트 단계에서 이런 현상을 줄이는 것이 좋습니다** [17] [18]. 즉, "Output ONLY the labels in the format X"처럼 **'ONLY' 등의 강조어를 사용**하고, 아예 잘못된 출력 예를 하나 보여주고 "이와 같이 말하지 말 것"이라고 못박는 방법도 있습니다. 모델 응답의 **일관성 부족**은 프롬프트 기반 분류의 대표적인 단점으로 지적되므로 [18], 이를 완화하려면 지시문을 엄격히 하고 가능한 **출력 예시와 금지 예시**까지 주는 것이 권장됩니다.

③ OpenAI 기능 활용 (선택 사항): OpenAI API를 사용할 경우, 2023년 이후 도입된 **함수 호출(Function Calling)**이나 **JSON 모드**를 활용하면 모델이 **지정한 스키마에 꼭 맞는 JSON 출력**을 내놓게 강제할 수도 있습니다 [19]. 이 기능을 쓰면 프롬프트 엔지니어링만으로 형식을 맞추는 것보다 훨씬 정확하게 구조적 출력이 옵니다. 다만 해당 기능을 사용하지 않는 순수 프롬프트 환경에서도, 위에서 언급한대로 **명시적 형식 지시와 예시 제시만으로도** 상당히 **clean한 출력**을 얻을 수 있다고 보고됩니다 [20]. 즉 **"친절하지만 단호하게 원하는 포맷을 요구"**하는 것이 핵심이라는 지침입니다 [21].

3. 내용 정확도를 높이는 테크닉 (한국어 LLM 분류 최적화)

① **제로샷 vs. 몇-샷 프롬프트:** 현재 모델(GPT-4o)이 사전학습된 지식을 활용해 **즉석 분류**를 잘하겠지만, **Few-Shot 프롬프팅**을 도입하면 정확도를 더 끌어올릴 수 있습니다. 예를 들어 문장 몇 개와 정답 라벨을 예시로 보여준 뒤 새로운 문장을 제시하면, **일관성 있게 맞는 라벨을 예측**하는 경향이 강화됩니다. 실제 연구들도 **Few-Shot 예시 추가 시 분류 성능 향상**을 관측했습니다 [9]. Chungnam대 Hugman 교수의 실험에서도 초기엔 **Zero-Shot**으로 했지만, **"유사 예시를 포함한 Few-Shot 형태로 개선하면 성능이 꽤 높아질 것"**이라 권장하고 있습니다 [22]. 다만 Few-Shot 활용 시 주의할 점은, **예시와 실제 입력을 명확히 구분**하지 않으면 모델이 예시까지 분류하려 들거나 혼동할 수 있다는 것입니다. 일부 개방형 한국어 LLM에서, 한국어로 프롬프트를 줄 경우 **예시 문장들을 실제 분류 대상으로 오인**하여 잘못된 출력을 내놓는 오류도 보고되었습니다 [23]. 이를 방지하려면 예시 섹션과 실제 질문 섹션을 구분하는 구획(**구분선**, `예시 ---`, `질문 ---` 등)이나 역할 지시를 사용해야 합니다. 정리하면, **소량의 한국어 예시**를 프롬프트에 포함하되 경계가 모호하지 않게 구성하면, **내용 및 형식 양면에서 더 나은 결과**를 얻을 수 있습니다.

② **라벨 정의와 기준 설명:** 모델이 각 속성을 올바로 분류하게 하려면, **라벨의 의미나 판정 기준을 사전에 설명**하는 것도 좋은 전략입니다. 예를 들어 "극성은 문장이 긍정적인지 부정적인지를 나타냅니다. 긍정 표현 예: 기쁘다, 좋다 / 부정 표현 예: 싫다, 아니다", "시제는 문장의 서술어 어미로 판정합니다 (-았다: 과거, -는다: 현재, -겠다: 미래)" 처럼 **라벨별로 결정 기준**을 알려주면, 모델이 더 일관된 판단을 내릴 수 있습니다. 실제로 LLM을 이용한 오류 분류 연구에서는 **프롬프트에 각 클래스의 정의와 예시**를 충분히 담았더니 분류 정확도가 크게 향상된 사례가 있습니다. 예컨대 프로그래밍 논리 오류 10가지를 정의하고 예제를 보여준 뒤 분류시켰더니, **오류 유형 설명을 포함한 프롬프트에서는 평균 21%p 정확도가 상승**했다고 보고되었습니다 [24]. 마찬가지로 한국어 문장 유형/극성 등의 분류에서도, **"확실성=화자가 해당 사실을 확실히 알고 있음을 나타내는지 여부 ('~다'로 단정하면 확실, '~일 것이다/같다' 등 추측 어미 사용하면 추측)"**처럼 기준을 제시하면 모델의 판단 근거가 명확해져 성능 향상을 기대할 수 있습니다.

③ **Chain-of-Thought 활용:** 복잡한 분류 기준이 적용될수록, **모델이 내부적으로 사고 과정을 거쳐 답하도록 유도**하는 기법이 효과적입니다. 이를 위해 명시적으로 "step-by-step으로 생각하고 최종 답만 말해"라고 할 수도 있지만, 한국어 문장 분류에서는 대개 최종 정형 출력만 원하기 때문에 **은근히 사고를 유도하는 기법**이 선호됩니다. 최근 제안된 방법 중 하나는 **출력에 `#` 주석 형태로 이유를 적게 한 뒤 정답을 구조화**하는 것입니다 [25] [26]. Huang(2025)은 이를 "Embed Reasoning with Comments"라 부르며, YAML 출력에서 한 줄짜리 주석으로 모델의 판단 근거를 쓰도록 지시하면 **모델이 실제 구조화 데이터를 내기 직전에 한 번 더 자기검증을 수행**하게 되어 최종 정답의 정확성이 올라간다고 설명합니다 [26] [27]. 예를 들면 모델 보고:

```
- # 문장1은 '~인가?'로 끝나므로 질문임. 부정어 없으니 극성은 긍정. 과거형 어미 '였다'가 있어 시제=과거. 단정 표현이라 확실.
- 유형: 의문문, 극성: 긍정, 시제: 과거, 확실성: 확실
```

이렇게 출력하게 프롬프트를 설계하는 것입니다. 첫 줄의 `#` 주석은 사람이 읽을 때 이유를 설명하지만, 파싱 시는 무시할 수 있습니다. 이 **Chain-of-Thought** 삽입 기법을 쓰면 모델이 곧바로 레이블을 찍을 때 생기는 실수를 줄이고, 특히 복합 분류에서 **누락이나 잘못된 지정 오류를 줄여주는 효과**가 있습니다 [28] [29]. 실제 예에서 이유 달기를 추가하자 놓치는 스팸 리뷰가 줄고 정확도가 향상되었다는 보고도 있습니다 [27]. 단, 이 방법을 쓸 때는 최종 출력에서 주석을 제거하거나, 주석을 허용하는 파서(YAML 등)를 사용하는 것이 필요합니다.

④ **형태소 정보 활용:** 한국어 특성상 **어미 변화나 부정 소사** 등이 분류 결정에 중요 단서가 됩니다. 예를 들어 "않는다", "못했다" 같은 부정어, "-겠다", "-리라" 같은 추측/미래 시제 어미 등을 모델이 올바로 잡아내야 극성이나 시제를 정확히 분류할 수 있습니다. GPT-4 정도의 거대 모델은 한국어 형태 변화도 대부분 이해하지만, **특정 접사가 있는지 여부를 명시적으로 언급**해 주면 더 정확할 수 있습니다. 예컨대 프롬프트에 "부정 여부는 '안/못/아니-' 등의 단어 사용으로 판단", "시제는 '-었/았-' 있으면 과거" 등 **힌트를 기술**할 수 있습니다. 혹은 **형태소 분석 결과를 함께 제공**하는 방안도 생각해볼 수 있습니다. 한 예로 문장 옆에 **"(형태소: 먹/VV + 었/EP + 다/EF)"** 같은 형태소열을 추가 제공하면 작은 한국어 모델의 경우 도움이 된다는 의견도 있으나, GPT-4급에 대해 연구된 바는 많지 않습니다. 현재까지는 **추가적인 형태소 태깅 없이도 GPT-4 계열은 한국어 문법 요소를 상당히 잘 파악**하는 것으로 알려져 있으므로, 오히려 형태소 표시를

넣으면 모델 혼란을 야기할 위험도 있습니다. 따라서 실무적으로는 모델이 헷갈려 하는 패턴(예: 이중 부정문 등)이 발견되었을 때 그 때 **해당 패턴을 간단히 설명**해 주는 식으로 보완하는 것이 좋아 보입니다.

⑤ **앙상블과 반복 질의:** 마지막으로, 하나의 모델 예측에 전적으로 의존하지 않고 **여러 번 답변을 생성하여 투표**하거나, **여러 서로 다른 모델의 결과를 조합**하면 정확도를 높일 수도 있습니다. Hugman도 프롬프트 분류기 개선책으로 **"하나의 LLM에 여러 번 쿼리하거나, 여러 모델을 활용해 다양한 생성 결과를 종합하는 앙상블"**을 제안했습니다 [30]. 예를 들어 동일 문장을 약간 다른 프롬프트로 두세 번 물어보고 다수결로 라벨을 결정하거나, GPT-4와 다른 한국어 특화 LLM 결과를 함께 고려하는 방식입니다. 이는 특히 경계가 모호한 분류일 때 오분류를 줄이고 확실성을 높여주는 전략입니다. 다만 API 비용과 시간 면의 부담이 커질 수 있어, 일반적으로는 **프롬프트 자체를 최적화**하는 접근을 우선 시도하고, 그래도 어려운 경우 보완책으로 앙상블을 활용합니다.

4. 한국어 LLM 분류에서 발견된 오류 사례 및 완화

① **출력 누락/과잉:** 여러 항목을 한 번에 출력하게 할 때 일부 속성을 빼먹거나 엉뚱한 값을 채우는 오류가 있습니다. 예를 들어 4가지 속성을 요구했는데 모델이 3가지만 답한다던가, 존재하지 않는 카테고리를 만들어낸다던가 하는 경우입니다. 이를 막으려면 앞서 언급한 **포맷 예시 제시**와 **필드 누락 없도록 강조**가 필요합니다. OpenAI 함수 기능을 쓰면 아예 스키마에 없는 값은 못 내놓게 되지만, 순수 프롬프트 환경에서는 **"모든 속성을 빠짐없이 하나씩 제공"**하라는 지시를 꼭 포함합니다. 그리고 모델 출력 후 **후처리 검증 단계**를 추가하여, 만약 형식이 잘못되었으면 재시도하는 체계를 갖출 수 있습니다 [31] [32]. 예컨대 Python에서 YAML 파싱 후 키가 누락되었으면 모델에게 다시 요청(retry)하는 식입니다 [32] [33].

② **한국어 지시문 해석 오류:** 한국어로 프롬프트를 쓸 때 종종 모델이 미묘한 뉘앙스를 오해하거나, 질문 의도를 다르게 받아들여 틀린 답을 하는 경우가 있습니다. 앞서 인용한 연구에서는 아랍어로 지시했을 때 **엉뚱한 출력(irrelevant results)**이 늘었다고 합니다 [34]. 한국어도 유사한 맥락일 수 있습니다. 이런 오류를 피하려면 **중요 단어에 영어 병기**(예: 확실성(certainty)), **문장부호를 활용한 강조**(예: `**중요**`), 또는 아예 **System 메시지를 활용한 역할 지시**(예: "Assistant는 엄격한 분류기처럼 행동하며, 사용자 입력에 대해 지정된 형식으로만 응답하세요.") 등을 사용할 수 있습니다. 영어로 핵심 지시를 내리고 한국어 예문을 분리해 보여주는 것도 방법입니다. 실제로 GPT-4o를 비롯한 다국어 LLM 실험에서 **비영어 프롬프트의 평균 성능이 낮았지만, 영어로 동일 지시를 하면 훨씬 안정적인 결과**가 나온 바 있습니다 [2] [7].

③ **예시 혼동:** Few-shot 예시를 넣은 경우, 드물게 모델이 **예시 자체를 분류하거나 답변에 재언급**하는 실수를 합니다. 이는 예시와 실제 입력 경계를 모델이 인지하지 못할 때 발생합니다. 해결책으로는 **구분선을 명확히**하거나, 예시 부분은 System나 Assistant 메시지로 제공하고 최종 질문만 User 메시지로 주는(ChatGPT API 사용 시) 방법이 있습니다. 또는 예시 끝부분에 "위는 예시였습니다. 이제 실제 문장을 분류하십시오:"라고 분명히 적어주는 것도 좋습니다. Jais 같은 모델의 사례에서 **아랍어 예시를 주었더니 그것까지 출력에 포함하는 오류**가 있었는데, 이 역시 프롬프트 구성을 개선하여 해결 가능하다고 합니다 [35].

④ **모델 편향 및 한계:** 한국어 분류 시 모델이 데이터 편향 때문에 특정 라벨을 과다하게 예측하는 현상도 있을 수 있습니다. 예를 들어 확실성 분류에서 애매한 경우 매번 "확실"로 치우친다든지 하는 경우입니다. 이럴 때는 **추가 지시**로 "애매하면 추측으로 분류"와 같이 정책을 명시하거나, **temperature 조절**을 통해 모델의 확신도를 낮추는 것도 고려됩니다. GPT 계열은 기본적으로 일정 정도 **확률적 출력**을 내므로, **temperature=0**으로 두면 항상 가장 가능성 높은 답을 선택해 **일관성은 높지만 오류 시 반복될 위험**이 있고, **temperature 약간 (>0)**으로 두면 가끔 다른 관점 시도를 해서 오류 수정 기회는 있지만 흔들릴 수도 있습니다. 중요한 시스템에서는 이 값을 조정해보며 최적점을 찾거나 아예 **다회 출력 후 다수결** 전략(앞서 앙상블로 언급)을 사용하기도 합니다.

⑤ **속도/비용 이슈:** 마지막으로, 프롬프트 기반 분류는 **속도가 느리고 비용이 증가**할 수 있다는 현실적인 단점이 있습니다 [18]. 문장 100개를 한꺼번에 프롬프트로 보내면 한 번의 API 호출로 끝나지만, 너무 길면 성능이 떨어질 수 있고, 하나씩 보내면 100번 호출해야 하므로 비용 문제가 됩니다. 이 균형을 위해 **적절한 batch 크기 조절**이 필요합니다. 예컨대 한번에 5~10문장씩 묶어 처리하면 성능과 속도의 trade-off를 맞출 수 있습니다. 또한 **토큰 최적화**를 위해 불필요한 설명은 최대한 줄이고, 특히 **한국어 문장 자체가 영어 대비 토큰을 더 많이 차지**하는 경향이 있으므로 (예: 공백 단위가

아니라 서브워드 단위 토크나이즈 시 길어짐), 프롬프트 길이를 관리해야 합니다. Microsoft 리서치에 따르면 GPT-4 모델은 한국어 입력의 토큰 효율화가 과거보다 개선되었으나 여전히 영어보다 토큰 소비가 높다고도 합니다 [36]. 따라서 **짧고 명료한 지시, 필요한 정보만 포함한 프롬프트**가 비용 면에서도 유리합니다.

以上의 전략들을 종합하면, ① **분류 작업과 카테고리를 명확히 정의하고 제시**하며 ② **여러 문장 입력 시 구조적 배열과 포맷 예시로 모델을 가이드**하고 ③ **한국어의 특성을 고려한 추가 정보나 영어 활용**을 통해 GPT-4o 모델을 최적 활용하는 것이 가능합니다. 실제 사례들은 이러한 기법으로 **형식 일치율**과 **내용 분류 정확도** 모두 개선되는 것을 보여줍니다. 앞으로도 프롬프트 엔지니어링 기법은 계속 고도화되고 있으며, 한국어같이 복잡한 언어에 대한 LLM 분류 성능 역시 모델 발전과 함께 더욱 향상될 것으로 기대됩니다.

참고 자료: 최신 프롬프트 엔지니어링 팁과 사례들 [4] [5] [1] [16], 프롬프트 기반 분류 실험 보고 [22] [18], LLM 출력 구조화 기법 [11] [26], 다국어 프롬프트 연구 결과 [2] [9] [7] 등. 각각의 출처에서 보다 상세한 설명과 데이터를 확인할 수 있습니다.

---

[1] [16] [17] [18] [22] [30] [Hands-On] 거대언어모델을 활용한 프롬프트 기반 텍스트 분류 | by Hugman Sangkeun Jung | Medium
https://medium.com/@hugmanskj/hands-on-%EA%B1%B0%EB%8C%80%EC%96%B8%EC%96%B4%EB%AA%A8%EB%8D%B8%EC%9D%84-%ED%99%9C%EC%9A%A9%ED%95%9C-%ED%94%84%EB%A1%AC%ED%94%84%ED%8A%B8-%EA%B8%B0%EB%B0%98-%ED%85%8D%EC%8A%A4%ED%8A%B8-%EB%B6%84%EB%A5%98-6e9537243eec

[2] [7] [8] [9] [23] [34] [35] Native vs Non-Native Language Prompting: A Comparative Analysis
https://arxiv.org/html/2409.07054v1

[3] [19] How to use GPT-4 and OpenAI's functions for text classification | by Kostas Stathoulopoulos | Discovery at Nesta | Medium
https://medium.com/discovery-at-nesta/how-to-use-gpt-4-and-openais-functions-for-text-classification-ad0957be9b25

[4] [5] How to Use OpenAI's GPT-4 API for Sentiment Classification | by Dr. Ernesto Lee | Medium
https://drlee.io/how-to-use-openais-gpt-4-api-for-sentiment-classification-ba7deed49428?gi=8f53799cdd4c

[6] LLM Prompt Best Practices for Large Context Windows
https://winder.ai/llm-prompt-best-practices-large-context-windows/

[10] Performance of ChatGPT on the National Korean Occupational ...
https://pmc.ncbi.nlm.nih.gov/articles/PMC10908230/

[11] [12] [13] [14] [15] [20] [21] [25] [26] [27] [28] [29] [31] [32] [33] Structured Output for Beginners: 3 Must-Know Prompting Tips | by Zachary Huang | Medium
https://medium.com/@zh2408/structured-output-for-beginners-3-must-know-prompting-tips-45a28aa643c6

[24] [논문 리뷰] Improving LLM Classification of Logical Errors by Integrating Error Relationship into Prompts
https://www.themoonlight.io/ko/review/improving-llm-classification-of-logical-errors-by-integrating-error-relationship-into-prompts

[36] 마이크로소프트-오픈AI "GPT-4, 한국어 토큰 효율화 달성"
https://zdnet.co.kr/view/?no=20240430131643