1. 유사도 평가 모듈

유사도 평가는 `BERT` 또는 `Sentence-BERT` 임베딩을 활용하여 생성된 질문과 요약본 간의 유사도를 계산,

Cosine Similarity를 사용하여 점수를 산출한다.

```
python코드 복사from sentence_transformers import SentenceTransformer, util

class SimilarityScorer:
    def __init__(self):
        self.model = SentenceTransformer('all-MiniLM-L6-v2')  # BERT 임베딩 모델

    def compute_similarity_score(self, generated_question, summary):
        # 질문과 요약본의 임베딩 생성
        question_embedding = self.model.encode(generated_question,
convert_to_tensor=True)
        summary_embedding = self.model.encode(summary, convert_to_tensor=True)

        # Cosine Similarity 계산
        similarity_score = util.cos_sim(question_embedding,
summary_embedding).item()
        return similarity_score
```

결과: 높은 유사도 점수는 질문이 요약본과 관련성이 높다는 의미이며, 유사도 점수가 낮을수록 할루시네이션 가능성이 있다고 볼 수 있습니다.

2. 지식 검증 모듈

지식 검증은 `Zero-Shot NLI` 모델을 사용해 생성된 질문이 원문과 일치하는지를 평가합니다. 각 질문이 요약본과 일치한다고 평가될 확률을 스코어로 부여합니다.

```
python코드 복사from transformers import pipeline

class KnowledgeVerifier:
    def __init__(self):
        # Zero-Shot 텍스트 분류 파이프라인 로드
        self.classifier = pipeline("zero-shot-classification",
model="facebook/bart-large-mnli")

    def verify_knowledge(self, generated_question, summary):
        # 요약본이 질문과 일치하는지 추론
        result = self.classifier(generated_question, candidate_labels=
["entailment", "neutral", "contradiction"])

        entailment_score = result['scores'][result['labels'].index("entailment")]
        return entailment_score
```

결과: `entailment_score`가 높을수록 질문이 원문 내용과 일치하므로 할루시네이션 가능성이 낮다고 평가할 수 있습니다.

3. 관계 및 인과 관계 분석 모듈

관계 및 인과 관계 분석은 질문이 요약본과의 논리적 관계나 인과 관계에 맞는지를 평가합니다. 관계 추론을 통해 질문이 요약본의 논리적 전개에 일치하는지를 판단합니다.

```
python코드 복사from transformers import AutoModelForSequenceClassification,
AutoTokenizer
import torch

class CausalRelationshipAnalyzer:
    def __init__(self):
        self.tokenizer = AutoTokenizer.from_pretrained("roberta-large-mnli")
        self.model = AutoModelForSequenceClassification.from_pretrained("roberta-
large-mnli")

    def analyze_relationship(self, generated_question, summary):
        # 입력 데이터 토큰화
        inputs = self.tokenizer(generated_question, summary, return_tensors="pt",
truncation=True)
        outputs = self.model(**inputs)

        # 추론 스코어 추출
        logits = outputs.logits
        entailment_score = torch.softmax(logits, dim=-1)[0][2].item()  #
"entailment" 스코어
        return entailment_score
```

결과: 관계 점수가 높을수록 요약본과의 논리적 연관성이 높으며, 할루시네이션 가능성이 낮다고 평가됩니다.

최종 할루시네이션 스코어링 모듈

각 모듈에서 나온 점수를 종합하여 최종 할루시네이션 가능성 스코어를 계산합니다. 일정 기준 이하의 스코어를 가진 질문만 최종적으로 채택하는 방식입니다.

```
python코드 복사class HallucinationScorer:
    def __init__(self):
        self.similarity_scorer = SimilarityScorer()
        self.knowledge_verifier = KnowledgeVerifier()
        self.relationship_analyzer = CausalRelationshipAnalyzer()

    def compute_hallucination_score(self, generated_question, summary):
        # 각 모듈에서 점수 계산
        similarity_score =
self.similarity_scorer.compute_similarity_score(generated_question, summary)
        knowledge_score =
self.knowledge_verifier.verify_knowledge(generated_question, summary)
        relationship_score =
self.relationship_analyzer.analyze_relationship(generated_question, summary)

        # 가중 평균 또는 합산하여 최종 점수 산출 (가중치는 실험을 통해 최적화)
        final_score = (similarity_score * 0.4) + (knowledge_score * 0.3) +
(relationship_score * 0.3)
        return final_score
```