RAG 애플리케이션을 최적화하기 위한 모범 사례 작성

AWS 권장 가이드

AWS 권장 가이드: RAG 애플리케이션을 최적화하기 위한 모범 사례 작성

Table of Contents

RAG 애플리케이션을 최적화하기 위한 모범 사례 작성

Ivan Cui와 Samantha Stuart, Amazon Web Services

2025년 7월(문서 기록)

대규모 언어 모델(LLMs)은 인간과 유사한 텍스트를 이해하고 생성하는 뛰어난 능력을 갖춘 인공 지능 분야에 혁신을 가져왔습니다. 그러나 상당한 제한에 직면합니다. 훈련 데이터에 포함된 지식으로만 작업할 수 있습니다. 여기에서 Retrieval Augmented Generation(RAG)이 도움이 됩니다. LLMs을 조직의 데이터 및 문서와 같은 외부 지식 소스와 결합하는 솔루션을 제공합니다. 정보 검색 및 응답 생성과 관련된 2단계 프로세스를 통해 RAG를 통해 AI 시스템은 다양한 소스의 up-to-date 정보에 액세스하고 통합할 수 있으므로 정적 모델 지식과 동적 실제 정보 요구 사항 간의 차이를 연결하는 보다 정확하고 정보에 입각한 응답이 가능합니다.

RAG 기반 애플리케이션에서 콘텐츠를 검색하도록 최적화하려면 어떻게 해야 하나요? 이 가이드는 지식 기반에서 텍스트 기반 콘텐츠의 형식 지정 및 쓰기 스타일을 최적화하는 데 도움이 되는 모범 사례를 제공합니다. 콘텐츠를 최적화하면 컨텍스트가 향상되어 RAG 애플리케이션이 작업별 정보를 더 정확하게 이해하는 데 도움이 됩니다. 시스템에서 관련성이 높고 정확한 콘텐츠를 검색하면 LLM 응답의 품질이 향상됩니다. 시스템 수준에서 컨텍스트 전송 프로세스를 최적화하는 것을 컨텍스트 엔지니어링이라고 하며, 이는 에이전트 RAG 아키텍처의 필수적인 부분을 형성합니다. 에이전트 RAG에서 하나 이상의 추가 LLMs RAG 실행 전에 접수 요청을 고려하고 조치를 취합니다. 이렇게 하면 다단계 정보 전송 프로세스가 용이해집니다. RAG 아키텍처가 점점 더 복잡해짐에 따라 소스 콘텐츠 최적화는 LLMs. 이러한 모범 사례는 RAG 애플리케이션에 대한 조직의 투자를 극대화하는 데 도움이 되도록 설계되었습니다.

대상 독자

이 가이드는 하나 이상의 RAG 구성 요소로 LLM 애플리케이션을 구축하는 AI 엔지니어, 데이터 과학자, 데이터 엔지니어 또는 소프트웨어 개발자를 대상으로 합니다. 이 가이드의 개념과 권장 사항을 이해하려면 LLMs에 대한 벡터 데이터베이스 및 프롬프트를 숙지해야 합니다.

목표

이 가이드의 권장 사항은 다음을 달성하는 데 도움이 될 수 있습니다.

- 토큰 사용 및 중복성에 최적화된 체계적이고 의미론적으로 풍부한 소스 문서를 제공하여 RAG 애플리케이션에서 생성된 응답의 정확성과 관련성을 개선합니다.

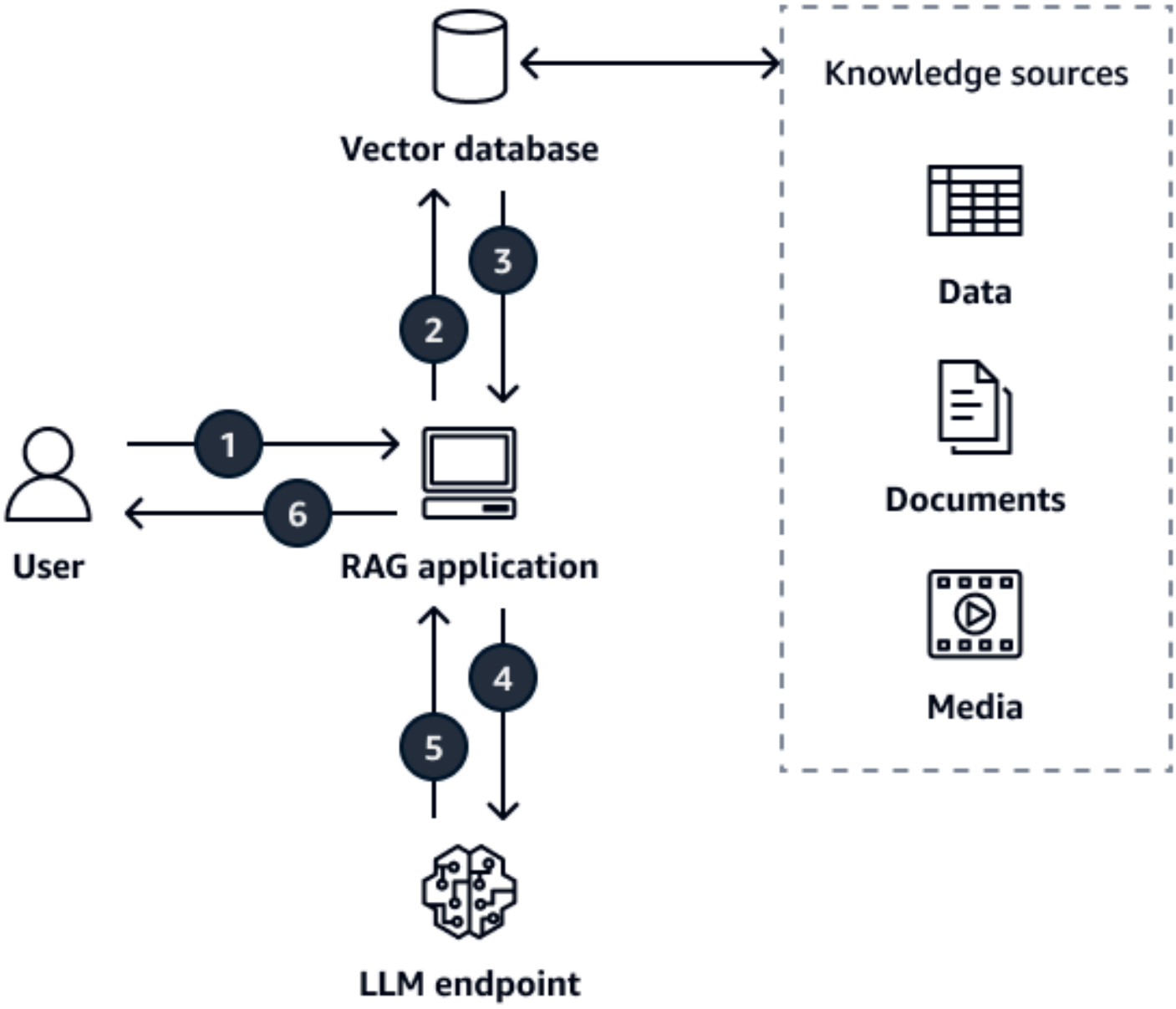
- 소스 문서 내에 명확한 정의와 설명을 제공하여 RAG 애플리케이션이 도메인별 지식과 컨텍스트를 더 잘 이해할 수 있도록 지원합니다.
- 소스 문서 전반에서 일관된 형식 지정 및 구조화 지침을 준수하여 RAG 애플리케이션의 유지 관리 및 지식 기반 업데이트를 더 쉽게 수행할 수 있습니다.
- 대규모 모놀리식 문서를 효율적으로 인덱싱하고 검색할 수 있는 더 작은 독립형 단위로 분류하여 RAG 솔루션의 확장성을 개선합니다.

LLMs 및 RAG 이해

소스 문서 품질을 개선하여 RAG 응답의 품질을 개선하는 방법을 이해하려면 LLM의 내부 작업을 이해해야 합니다. LLMs의 진정한 성능은 자체 주의 메커니즘과 변환기 아키텍처를 사용하는 능력에 있습니다. 이러한 고급 기술을 통해 모델은 텍스트 내 위치 또는 거리에 관계없이 입력 시퀀스의 다양한 부분을 효과적으로 처리하고 연결할 수 있습니다. 이 기능은 종종 장기 종속성과 컨텍스트 이해로 어려움을 겪는 기존 언어 모델과 극명한 대조를 이룹니다. 또한 LLMs은 전례 없는 규모로 훈련됩니다. 가장 큰 모델 중 일부는 수조 개의 파라미터로 구성되며 다양한 소스에서 수 테라바이트의 텍스트 데이터를 수집했습니다. 이 대규모 규모를 통해 LLMs 언어에 대한 풍부한 이해를 개발하고 이전에 AI 시스템에 어려운 미묘한 미묘한 뉘앙스, 관용구 및 상황별 신호를 포착할 수 있습니다. 그 결과 일관성 있고 유창한 텍스트를 생성하고 질문 답변, 텍스트 요약, 심지어 코드 생성과 같은 작업에서 주목할 만한 기능을 보여줄 수 있는 모델 클래스가 됩니다.

이러한 모델을 사용하기 위해 Amazon Bedrock과 같은 서비스를 사용할 수 있습니다.이 서비스는 Amazon 및 Anthropic, Cohere, Meta를 포함한 타사 공급자의 다양한 파운데이션 모델에 대한 액세스를 제공합니다. Amazon Bedrock을 사용하여 state-of-the-art 모델을 실험하거나, 모델을 사용자 지정 및 미세 조정하거나, 단일 API를 통해 생성형 AI 기반 솔루션에 통합할 수 있습니다.

LLMs 패턴을 캡처하고 일관성 있는 텍스트를 생성하는 데 뛰어나지만 up-to-date 또는 특수 정보에 대한 액세스 권한이 없는 경우가 많습니다. RAG는 LLMs의 생성 능력을 구체화된 LLM 프롬프트의 일부로 외부 소스의 관련 정보에 액세스하고 통합할 수 있는 검색 구성 요소와 결합합니다. 외부 소스의 예로는 Amazon Bedrock용 지식 기반, Amazon Kendra와 같은 지능형 검색 시스템 또는 Amazon OpenSearch Service와 같은 벡터 데이터베이스가 있습니다.



다이어그램은 다음 워크플로를 설명합니다.

1. 사용자가 RAG 애플리케이션에 쿼리를 제출합니다.
2. RAG 애플리케이션은 문서, 데이터 또는 미디어와 같은 지식 소스가 포함된 벡터 데이터베이스를 쿼리합니다.
3. RAG 애플리케이션은 쿼리와 저장된 문서 간의 의미론적 유사성을 기반으로 벡터 데이터베이스에서 관련 정보를 검색합니다.
4. RAG 애플리케이션은 검색된 컨텍스트로 원본 프롬프트를 보강하여 LLM 엔드포인트로 전송합니다.

5. LLM 엔드포인트는 응답을 생성하여 RAG 애플리케이션에 반환합니다.
6. RAG 애플리케이션은 생성된 응답을 사용자에게 반환합니다.

RAG는 핵심에서 2단계 프로세스를 사용합니다. 첫 번째 단계에서 검색 모델은 입력 쿼리를 기반으로 관련 문서 또는 구절을 식별하고 검색합니다. 이 검색 모델은 기존 정보 검색 시스템, 밀집 검색 모델 또는 둘 다일 수 있습니다. 두 번째 단계에서는 검색된 정보와 원본 쿼리가 완전히 구체화된 프롬프트 템플릿으로 LLM에 공급됩니다. LLMs 리트리버 구성 요소에서 제공하는 소스 콘텐츠의 품질에 크게 의존합니다. 자체 주의 메커니즘을 적용하여 검색된 콘텐츠가 작업과 어떤 관련이 있는지 수학적으로 인코딩합니다. 그런 다음 LLM은 쿼리와 검색된 정보를 모두 기반으로 응답을 생성합니다. RAG에서 검색된 소스 문서의 품질을 제어하는 것은 작업에 대한 LLM의 내부 표현을 개선하는 직접적인 수단을 나타냅니다. RAG는 관련 외부 데이터로 LLM의 훈련 데이터를 효과적으로 보강합니다. 이 접근 방식을 통해 RAG는 LLMs과 검색 시스템의 장점을 모두 활용하여 현재 및 전문 지식을 통합하는 보다 정확하고 정보에 입각한 응답을 생성할 수 있습니다.

벡터 및 임베딩

벡터 및 임베딩은 기계 학습 및 자연어 처리의 기본 개념입니다. 벡터는 크기와 방향이 모두 있는 수량을 나타내는 수학적 객체입니다. 자연어 처리(NLP)의 맥락에서 단어, 문장 또는 문서는 고차원 벡터 공간에서 벡터로 표현되는 경우가 많습니다. 반면 임베딩은 벡터 간의 관계가 의미 또는 구문 유사성을 캡처하는 저차원 벡터 공간에서 단어 또는 문서와 같은 객체를 나타내는 방법입니다. 예를 들어 단어 임베딩은 유사한 의미의 단어가 유사한 벡터 표현을 갖도록 허용합니다. 이를 통해 알고리즘은 언어를 더 효과적으로 이해하고 처리할 수 있습니다.

벡터 데이터베이스

생성형 AI에서 벡터 데이터베이스는 문서, 쿼리 또는 기타 객체의 벡터 표현을 저장하고 관리하는 데이터베이스입니다. 벡터를 효율적으로 저장하고 검색하도록 설계되었습니다. 이는 의미 검색 및 유사성 일치와 같은 빠르고 확장 가능한 작업을 지원합니다. 벡터 데이터베이스는 계층적 탐색 가능 스몰 월드(HNSW) 그래프 또는 K-Nearest Neighbors(KNN) 알고리즘과 같은 특수 데이터 구조를 사용하여 벡터를 인덱싱합니다. 이러한 데이터 구조를 통해 가장 가까운 이웃을 빠르게 검색할 수 있으므로 데이터베이스에서 유사한 벡터를 빠르게 찾을 수 있습니다.

시맨틱 검색

의미 체계 검색은 단순히 키워드를 일치시키는 대신 쿼리의 의도와 컨텍스트를 이해하여 검색 결과의 관련성을 개선하는 기법입니다. 기술적으로 의미 체계 검색에는 쿼리의 벡터 표현과 데이터베이스의

문서를 비교하여 가장 관련성이 높은 일치 항목을 찾는 작업이 포함됩니다. 의미 체계 검색에는 다음을 포함하되 이에 국한되지 않는 다양한 검색 전략을 사용할 수 있습니다.

- HNSW - 가장 가까운 이웃을 효율적으로 검색할 수 있는 방식으로 벡터를 구성하는 그래프 기반 데이터 구조입니다.
- KNN - 코사인 유사성과 같은 거리 지표를 기반으로 쿼리 벡터에 가장 가까운 K 벡터를 찾는 알고리즘입니다.
- 코사인 유사성 - 두 벡터 간의 각도 코사인을 측정하는 0이 아닌 두 벡터 간의 유사성 척도입니다. 시맨틱 검색에서 고차원 공간에서 벡터의 방향을 비교하는 데 자주 사용됩니다.
- 로컬리티 구분 해싱(LSH) - 확률이 높은 동일하거나 가까운 버킷에 유사한 벡터를 해시하는 기법입니다. 이렇게 하면 대략적인 가장 가까운 이웃 검색이 가능하므로 고차원 공간에서의 정확한 검색보다 빠를 수 있습니다.

RAG 애플리케이션에 영향을 미치는 소스 데이터의 문제

최적의 검색 증강 생성(RAG) 애플리케이션을 개발하는 데 있어 중요한 과제 중 하나는 사용되는 원시 데이터 또는 문서의 특성에 있습니다. 기업은 인간 참조용으로 생성된 기존 문서를 사용하는 경우가 많습니다. 이러한 문서에는 이해를 돕기 위한 하이퍼링크와 이미지 스크린샷이 포함되는 경우가 많습니다. 그러나 이러한 요소는 발췌 토큰 제한으로 인해 의미 체계 검색을 방해합니다. 이로 인해 리트리버 성능이 저하됩니다.

다음은 최적의 RAG 애플리케이션에서 가장 일반적인 원시 문서 과제입니다.

- 구조화된 형식 및 메타데이터 부족 - 원시 문서에는 명확한 섹션 제목, 부제목 또는 메타데이터가 없을 수 있습니다. 따라서 관련 정보를 식별하고 추출하기가 어렵습니다. 예를 들어 명확한 제목이 없는 긴 문서를 사용하면 특정 정보의 컨텍스트를 파악하기 어려울 수 있습니다.
- 비공식 및 일관되지 않은 언어 - 원시 문서에는 비공식 언어 또는 일관되지 않은 용어가 포함되는 경우가 많습니다. 이로 인해 RAG 모델이 혼동될 수 있습니다. 예를 들어 문서에 정의되지 않았거나 LLM에서 이미 알고 있는 약어가 문서 전체에서 사용될 수 있습니다.
- 세부 정보 및 중복성 - 원시 문서는 상세하고 불필요하거나 중복된 정보를 포함할 수 있습니다. 이로 인해 RAG 모델이 압도되어 간결하고 관련성이 높은 응답이 줄어들 수 있습니다. 예를 들어 동일한 정보를 여러 번 반복하는 문서 또는 유사하거나 모순되는 정보가 포함된 여러 문서가 있습니다.
- 모호한 용어 및 문구 - 원시 문서에는 여러 방식으로 해석될 수 있는 모호한 용어 또는 문구가 포함될 수 있습니다. 이러한 모호함으로 인해 RAG 모델에 의한 잘못된 해석과 부정확한 응답이 발생할 수 있습니다. 예를 들어 여러 의미의 용어를 사용하는 문서는 의도한 의미와 일치하지 않는 응답을 생성할 수 있습니다.
- 그래픽 및 하이퍼링크 요소 삽입 - 그래픽 및 하이퍼링크 정보가 포함된 원시 문서는 사람이 사용하기에 적합합니다. 그러나 이러한 요소는 검색 토큰 제한을 사용할 수 있습니다. 그 결과 발췌문이 불완전할 수 있습니다. 예를 들어 그래픽 및 하이퍼링크 URLs은 검색 토큰을 사용하는 검색의 일부로 반환되며, 후속 단락의 키 정보가 누락됩니다.
- 도메인별 지식 또는 컨텍스트 부족 - 원시 문서에는 정확한 생성에 필요한 도메인별 지식 또는 컨텍스트가 부족할 수 있습니다. 이렇게 하면 RAG 모델이 적절하고 정확한 응답을 생성하는 능력이 제한될 수 있습니다. 예를 들어 컨텍스트를 제공하지 않고 특수 개념을 참조하는 문서가 있습니다. 이로 인해 지정된 도메인에서 의미가 없는 응답이 발생할 수 있습니다.

이 목록은 포괄적이지 않지만 기업이 작동하지 않는 부분과 그 이유를 생각할 수 있는 출발점을 제공합니다. 문서에는 이러한 문제가 하나 이상 있을 수 있습니다. RAG 애플리케이션을 최적화하는 핵심은 검색을 최적화하는 쓰기 모범 사례를 준수하는 문서 세트를 사용하는 것입니다.

RAG 애플리케이션에 대한 설명서 모범 사례

성공적인 검색 증강 생성(RAG) 애플리케이션을 개발하려면 성능을 최적화하기 위해 다양한 문서 관련 요소를 신중하게 고려해야 합니다. 이 섹션의 모범 사례는 많은 조직 리더와 함께 RAG 시스템을 구축한 경험을 기반으로 큐레이션됩니다. 다음은 RAG 애플리케이션의 효과를 개선하기 위한 문서의 몇 가지 주요 모범 사례입니다.

- 제목 및 부제목을 올바르게 사용 - 명확한 제목 및 부제목으로 콘텐츠를 구성하면 가독성이 향상되고 RAG 모델이 문서의 구조를 이해하는 데 도움이 됩니다. 이 방법을 사용하면 모델이 문서에서 정보를 더 잘 탐색하고 추출할 수 있으므로 생성된 응답의 품질이 향상됩니다.
- 번호 지정이 순차적인지 확인 - 번호가 매겨진 목록을 사용할 때는 혼동을 방지하기 위해 적절한 번호 지정을 유지하는 것이 중요합니다. 번호를 건너뛰지 않고 각 목록 항목의 번호가 순차적으로 지정되었는지 확인합니다. 이를 통해 콘텐츠의 명확성과 일관성을 유지할 수 있습니다.
- 목록 항목 간 전환 추가 - 글머리 기호 또는 번호가 매겨진 목록의 항목 간 전환을 제공하면 LLM이 콘텐츠를 안내하는 데 도움이 됩니다. 예를 들어 "2단계를 완료한 후...합니다"와 같은 문구를 사용하여 아이디어를 연결하고 정보 흐름을 개선할 수 있습니다.
- 테이블 교체 - 테이블을 사용하지 마세요. 이 정보의 형식을 다중 수준 글머리표 목록 또는 플랫 수준 구문으로 지정합니다. 플랫 레벨 구문은 중첩된 하위 수준 없이 동일한 계층적 수준에서 요소 또는 항목을 정렬합니다. 이러한 구조는 LLMs 정보를 다이제스트하는 데 도움이 됩니다. 대부분의 인덱싱된 문서는 왼쪽에서 오른쪽으로 읽히기 때문에 플랫 레벨 구문을 사용하면 추가 차원을 참조할 필요 없이 정보를 보다 일관성 있게 따를 수 있습니다. 이 형식은 구조화되고 쉽게 이해할 수 있는 방식으로 정보를 제공하기 때문에 RAG 애플리케이션에 더 유용합니다.
- 효율성을 위한 그래픽 정보 사전 처리 - 다중 모달 LLMs 이미지와 텍스트를 모두 수집할 수 있습니다. 이미지 해상도를 줄이고, 중복 이미지를 제거하고, 그래픽 요소의 내용을 텍스트 형식으로 설명합니다. 이러한 조치는 의미 있는 컨텍스트를 개선하고, 토큰을 불필요하게 사용하지 않으며, RAG 모델의 접근성을 개선합니다.
- 일반적인 쿼리를 위한 세션 스타터 추가 - "소프트웨어를 주문하려면 어떻게 해야 합니까?"와 같은 일반적인 질문이나 작업을 처리할 때 리더를 프로세스로 전환하는 세션 스타터를 추가합니다. 예를 들어 "소프트웨어를 주문하려면 아래 단계를 따르세요."를 추가할 수 있습니다. 이렇게 하면 높은 의미 체계 일치를 생성하여 LLM이 일관된 응답을 구성하는 데 도움이 됩니다.
- 각 섹션에 요약 추가 - 각 제목 또는 부제목 뒤에 해당 섹션의 콘텐츠에 대한 간결하고 간략한 요약을 추가합니다. 이렇게 하면 의미 체계 적용 범위를 늘리고 핵심 사항을 강화할 수 있습니다. 이렇게 하면 임베딩 공간 내에서 유사성 검색의 정확도가 향상되어 RAG 애플리케이션의 성능이 향상됩니다. 이는 문서가 LLM 및 인적 사용을 위한 것이거나 테이블 및 그래픽 요소가 필요한 경우에 특히 유용합니다.

- 모호하지 않음 - 문서는 간결하고 초점을 맞춰야 합니다. LLMs 검색된 발췌문을 기반으로 응답을 생성하므로 모호성을 제거하면 모델이 명확하고 관련 있는 정보를 사용하는 데 도움이 됩니다. 이렇게 하면 더 정확하고 유용한 응답이 생성됩니다.
- 약어 정의 및 컨텍스트 설정 - LLMs은 대량의 인터넷 데이터에 대해 훈련되며 대부분의 경우 엔터프라이즈 내부 문서의 컨텍스트가 없습니다. 따라서 컨텍스트를 설정하고, 약어를 정의하고, 회사별 용어를 피하거나 정의하면 LLM이 엔터프라이즈 데이터를 이해하는 데 도움이 됩니다. 이렇게 하면 LLM이 질문에 더 정확하게 답변하고 할루시네이션을 방지하는 데 도움이 됩니다.
- 효율적인 태그 지정 및 인덱싱을 위해 대형 문서를 더 작은 문서로 재구성 - 여러 하위 주제가 포함된 대형 문서를 인덱싱하지 마세요. 대용량 문서를 명확한 제목이 있는 더 작고 독립적인 문서로 나누는 것이 좋습니다. 이렇게 하면 인덱싱 및 태그 지정이 개선됩니다.

FAQ

RAG 애플리케이션용 문서를 최적화하는 것이 중요한 이유는 무엇입니까?

원시 문서는 검색 증강 생성(RAG) 애플리케이션과 같은 고급 AI 시스템의 요구 사항을 고려하지 않고 사람이 사용할 수 있도록 작성되는 경우가 많습니다. 모범 사례를 따라 문서를 최적화하면 모델에 체계적이고 모호하지 않은 관련 정보를 제공하여 RAG 애플리케이션의 성능과 정확도를 크게 개선할 수 있습니다.

RAG 성능을 저해할 수 있는 원시 문서의 일반적인 문제는 무엇입니까?

일부 주요 과제로는 구조화된 형식 및 메타데이터 부족, 비공식적이거나 일관되지 않은 언어, 세부 정보 및 중복성, 모호한 용어 및 문구, 하이퍼링크 요소 포함, 도메인별 컨텍스트 부족 등이 있습니다. 이러한 문제는 RAG 모델을 혼동하여 부정확하거나 관련이 없는 응답으로 이어질 수 있습니다. 자세한 내용은이 설명서의 RAG 애플리케이션에 영향을 미치는 소스 데이터의 문제를 참조하세요.

제목 및 하위 제목을 사용하면 RAG 성능이 어떻게 향상될 수 있습니까?

명확한 제목과 하위 제목은 RAG 모델이 콘텐츠의 구조와 컨텍스트를 이해하는 데 도움이 됩니다. 이를 통해 문서에서 관련 정보를 더 잘 탐색하고 추출할 수 있으며 생성된 응답의 품질을 개선할 수 있습니다. 자세한 내용은이 가이드의 RAG 애플리케이션에 대한 설명서 모범 사례를 참조하세요.

테이블 정보를 플랫 레벨 구문으로 바꾸는 것이 권장되는 이유는 무엇입니까?

RAG 모델은 2차원 구조를 이해해야 하므로 테이블을 해석하기 어려울 수 있습니다. 플랫 레벨 구문 또는 글머리 기호 목록에 테이블 정보를 제공하면 모델이 정보를 보다 쉽게 처리할 수 있으므로 성능이 향상됩니다. 자세한 내용은이 가이드의 RAG 애플리케이션에 대한 설명서 모범 사례를 참조하세요.

요약을 추가하면 RAG 성능이 어떻게 향상될 수 있나요?

각 섹션 또는 하위 섹션의 시작 부분에 간결한 요약을 포함하면 의미 체계 적용 범위를 늘리고 핵심 사항을 강화할 수 있습니다. 이렇게 하면 임베딩 공간 내에서 유사성 검색의 정확도가 향상되어 궁극적으로 RAG 애플리케이션의 성능이 향상됩니다. 자세한 내용은이 가이드의 RAG 애플리케이션에 대한 설명서 모범 사례를 참조하세요.

약어를 정의하고 LLMs에 대한 컨텍스트를 설정하는 것이 중요한 이유는 무엇입니까?

LLMs은 광범위한 데이터에 대해 훈련되지만 엔터프라이즈별 약어 또는 용어에 대한 컨텍스트가 부족합니다. 약어를 정의하고 컨텍스트를 제공하면 LLMs 더 정확하게 이해하고 대응할 수 있습니다. 이렇게 하면 할루시네이션이나 오해를 방지하는 데 도움이 될 수 있습니다. 자세한 내용은이 가이드의 RAG 애플리케이션에 대한 설명서 모범 사례를 참조하세요.

다음 단계 및 리소스

이 가이드에서는 RAG 애플리케이션의 문서 수준 문제 세트와 이를 완화하기 위한 모범 사례를 설명합니다. 이러한 학습은 업계 리더와의 인터뷰 및 논의를 통해 큐레이션되며 엔터프라이즈 사용 사례를 기반으로 합니다.

RAG 애플리케이션을 위한 문서 최적화를 시작하려면 기존 문서에 대한 감사를 수행하는 것이 좋습니다. RAG 애플리케이션에 문제를 일으키는 영역을 식별합니다. 예를 들어 구조 부족, 모호한 언어 또는 그래픽 요소의 과도한 사용이 있습니다. 비즈니스 운영에 자주 액세스하거나 중요한 문서의 우선순위를 지정합니다. 주제 전문가와 협력하여이 가이드의 모범 사례를 구현합니다. 문서가 명확한 제목, 간결한 언어 및 컨텍스트 설정 요소로 재구성되었는지 확인합니다. 새 문서의 경우 일관성을 보장하고 작성자가 모범 사례를 준수할 수 있도록 지침과 템플릿을 설정합니다. 또한 생성형 AI를 사용하여 문서를 재구성하는 등 문서 최적화 프로세스의 측면을 자동화할 수 있는 도구 또는 서비스에 투자하는 것도 고려해 보세요. 문서 최적화에 대한 선제적 접근 방식을 취하면 RAG 애플리케이션의 잠재력을 최대한 활용하고 조직 전체에서 더 정확하고 통찰력 있는 결과를 도출할 수 있습니다.

다음 리소스는 조직의 RAG 애플리케이션을 이해하고 구축하는 데 도움이 될 수 있습니다.

리소스

AWS 설명서

- RAG 사용 사례를 위한 AWS 벡터 데이터베이스 선택(AWS 권장 가이드)
- Terraform 및 Amazon Bedrock을 AWS 사용하여에 RAG 사용 사례 배포(AWS 권고 가이드)
- RAG 및 ReAct 프롬프트를 사용하여 고급 생성형 AI 채팅 기반 어시스턴트 개발(AWS 권고 가이드)
- Amazon Bedrock 지식 기반을 사용하여 데이터 검색 및 AI 응답 생성(Amazon Bedrock 설명서)
- 검색 증강 생성(Amazon SageMaker AI 설명서)
- 의 Augmented Generation 옵션 및 아키텍처 검색 AWS(AWS 권고 가이드)

기타 AWS 리소스

- 고급 RAG 및 Amazon Bedrock을 사용하여 멀티모달 어시스턴트 생성(AWS 블로그 게시물)

문서 기록

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
최초 게시	—	2025년 7월 24일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.