AI 개념

AI 개념

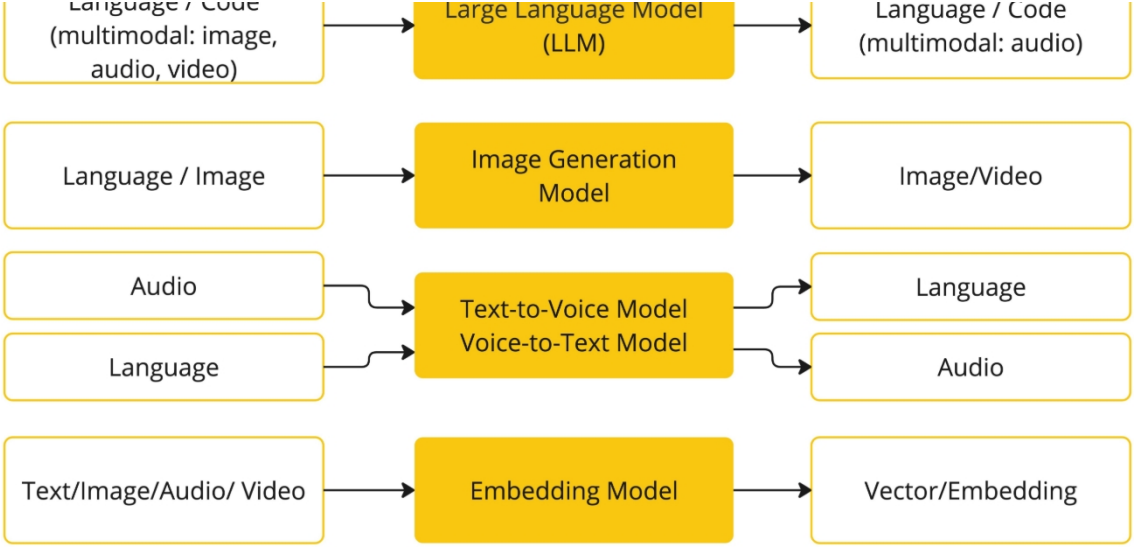
이 섹션에서는 Spring AI가 사용하는 핵심 개념에 대해 설명합니다. Spring AI가 구현되는 방식을 이해하려면 이 섹션을 자세히 읽어보시기 바랍니다.

모델

AI 모델은 정보를 처리하고 생성하도록 설계된 알고리즘으로, 인간의 인지 기능을 모방하는 경우가 많습니다. 이러한 모델은 대규모 데이터 세트에서 패턴과 인사이트를 학습함으로써 예측, 텍스트, 이미지 또는 기타 결과물을 생성하여 산업 전반의 다양한 애플리케이션을 향상시킬 수 있습니다.

AI 모델에는 다양한 유형이 있으며, 각 모델은 특정 사용 사례에 적합합니다. ChatGPT와 그 생성 AI 기능은 텍스트 입력과 출력을 통해 사용자를 사로잡았지만, 많은 모델과 회사들이 다양한 입력과 출력을 제공합니다. ChatGPT 이전에는 많은 사람들이 미드저니와 스테이블 디퓨전과 같은 텍스트-이미지 생성 모델에 매료되었습니다.

다음 표는 입력 및 출력 유형에 따라 여러 모델을 분류한 것입니다:



Spring AI는 현재 언어, 이미지, 오디오로 입력과 출력을 처리하는 모델을 지원합니다. 이전 표의 마지막 행은 텍스트를 입력으로 받아들이고 숫자를 출력하는 것으로, 더 일반적으로 임베딩 텍스트라고 하며 AI 모델에서 사용되는 내부 데이터 구조를 나타냅니다. Spring AI는 임베딩을 지원하여 보다 고급 사용 사례를 구현할 수 있습니다.

GPT와 같은 모델을 차별화하는 것은 GPT-Chat Gen- erative Pre-trained Transformer의 "P"에서 알 수 있듯이 사전 학습된 특성입니다. 이 사전 학습 기능은 AI를 광범위한 머신러닝이나 모델 학습 배경 지식이 필요 없는 일반 개발자 도구로 전환합니다.

프롬프트

프롬프트는 AI 모델이 특정 출력을 생성하도록 안내하는 언어 기반 입력의 기초 역할을 합니다. ChatGPT에 익숙한 사람들에게 프롬프트는 단순히 API로 전송되는 대화 상자에 입력되는 텍스트처럼 보일 수 있습니다. 하지만 프롬프트에는 그 이상의 의미가 담겨 있습니다. 많은 AI 모델에서 프롬프트의 텍스트는 단순한 문자열이 아닙니다.

ChatGPT의 API에는 프롬프트 내에 여러 개의 텍스트 입력이 있으며, 각 텍스트 입력에는 역할이 할당됩니다. 예를 들어, 시스템 역할이 있는데, 이는 모델에 작동 방법을 알려주고 상호작용에 대한 컨텍스트를 설정합니다. 또한 일반적으로 사용자의 입력인 사용자 역할도 있습니다.

효과적인 프롬프트를 만드는 것은 예술이자 과학입니다. ChatGPT는 사람 간의 대화를 위해 설계되었습니다. 이는 SQL과 같은 것을 사용하여 "질문"하는 것과는 상당히 다릅니다. 다른 사람과 대화하는 것처럼 AI 모델과 소통해야 합니다.

이러한 상호작용 스타일의 중요성 때문에 '프롬프트 엔지니어링'이라는 용어가 독자적인 학문으로 등장했습니다. 효과를 개선하는 기법들이 급성장하고 있습니다. 프롬프트 제작에 시간을 투자하면 결과물을 크게 향상시킬 수 있습니다.

프롬프트 공유는 공동의 관행이 되었으며 대한 학술적 연구도 활발히 진행되고 있습니다. 효과적인 프롬프트를 만드는 것이 얼마나 직관적이지 않은지를 보여주는 예로 다음과 같은 경우를 들 수 있습니다.

충분한, SQL과는 대조적으로), 최근 연구 논문에 따르면 가장 효과적인 프롬프트 중 하나는 "심호흡을 하고 이 작업을 단계별로 진행하세요."라는 문구로 시작하는 것으로 나타났습니다. 언어가 왜 그렇게 중요한지 알 수 있을 것입니다. 새 버전은 말할 것도 없고 ChatGPT 3.5와 같은 이전 버전의 기술을 가장 효과적으로 활용하는 방법도 아직 완전히 이해하지 못했습니다.

를 개발 중입니다.

프롬프트 템플릿

효과적인 프롬프트를 만들려면 요청의 컨텍스트를 설정하고 요청의 일부를 사용자 입력에 맞는 값으로 대체해야 합니다.

이 프로세스는 신속한 생성 및 관리를 위해 기존의 텍스트 기반 템플릿 엔진을 사용합니다. Spring AI는 이를 OSS 라이브러리 StringTemplate을 사용합니다.

예를 들어 간단한 프롬프트 템플릿을 생각해 보세요:

```
콘텐츠}에 대한 {형용사} 농담을 말해 주세요.
```

Spring AI에서 프롬프트 템플릿은 Spring MVC 아키텍처의 "보기"에 비유할 수 있습니다. 템플릿 내에서 플레이스홀더를 채우기 위해 모델 객체(일반적으로 `java.util.Map`가 제공됩니다. "렌더링된" 문자열은 AI 모델에 제공되는 프롬프트의 콘텐츠가 됩니다.

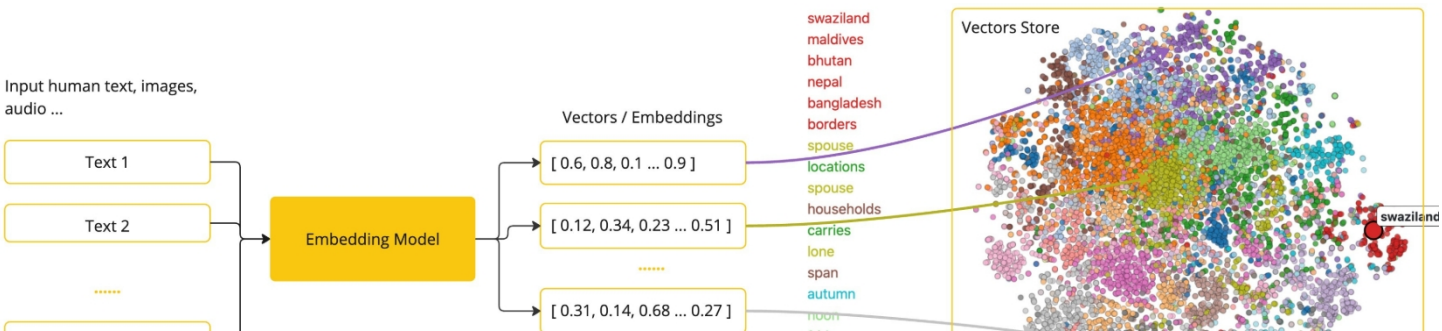
모델에 전송되는 프롬프트의 특정 데이터 형식에는 상당한 가변성이 있습니다. 처음에는 단순한 문자열로 시작했던 프롬프트는 여러 메시지를 포함하도록 발전했으며, 각 메시지의 각 문자열은 모델에 대한 고유한 역할을 나타냅니다.

임베딩

임베딩은 입력 간의 관계를 캡처하는 텍스트, 이미지 또는 동영상의 숫자 표현입니다.

임베딩은 텍스트, 이미지, 동영상을 하는 부동 소수점 숫자 배열로 변환하는 방식으로 작동합니다. 이러한 벡터는 텍스트, 이미지 및 동영상의 의미를 포착하도록 설계되었습니다. 임베딩 배열의 길이를 벡터의 차원이라고 합니다.

두 텍스트 조각의 벡터 표현 사이의 수치적 거리를 계산하여 임베딩 벡터를 생성하는 데 사용된 개체 간의 유사성을 결정할 수 있습니다.



탐구하는 Java 개발자는 이러한 벡터 표현의 이면에 있는 복잡한 수학적 이론이나 구체적인 구현을 이해할 필요는 없습니다. 특히 애플리케이션에 AI 기능을 통합할 때는 AI 시스템 내에서의 역할과 기능에 대한 기본적인 이해만 있으면 충분합니다.

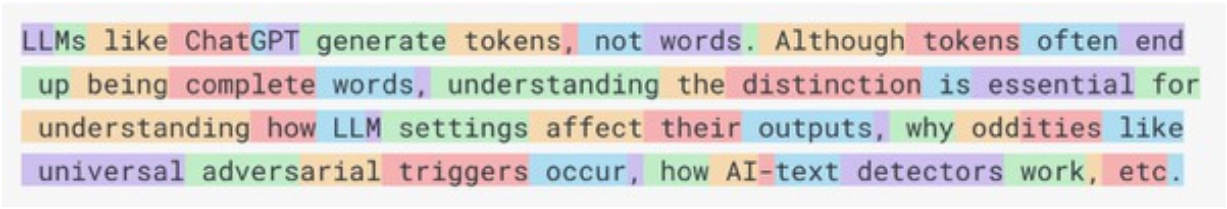
임베딩은 특히 검색 증강 생성(RAG) 패턴과 같은 실용적인 애플리케이션과 관련이 있습니다. 임베딩을 사용하면 유클리드 기하학의 2차원 공간과 유사하지만 더 높은 차원의 의미 공간에서 데이터를 점으로 표현할 수 있습니다. 즉, 유클리드 기하학에서 평면상의 점들이 좌표에 따라 가깝거나 먼 것처럼, 의미 공간에서는 점들의 근접성이 의미의 유사성을 반영합니다. 비슷한 주제에 대한 문장은 그래프에서 점들이 서로 가까이 있는 것처럼 이 다차원 공간에서 더 가깝게 배치됩니다. 이러한 근접성은 텍스트 분류, 의미론적 검색, 심지어 제품 추천과 같은 작업에 도움이 되는데, AI가 확장된 의미론적 환경에서의 '위치'를 기반으로 관련 개념을 식별하고 그룹화할 수 있게 해주기 때문입니다.

이 시맨틱 공간을 벡터로 생각할 수 있습니다.

토큰

토큰은 AI 모델이 작동하는 방식의 기본 구성 요소 역할을 합니다. 입력 시 모델은 단어를 토큰으로 변환합니다. 출력 시에는 토큰을 다시 단어로 변환합니다.

영어에서 1토큰은 대략 단어의 75%에 해당합니다. 참고로 셰익스피어의 전 작품은 약 U00,000개의 단어로 이루어져 있으며, 이는 약 120만 개의 토큰으로 환산됩니다.



아마도 더 중요한 것은 토큰= 머니입니다. 호스팅된 AI 모델에서는 사용된 토큰 수에 따라 요금이 결정됩니다. 입력과 출력 모두 전체 토큰 기여합니다.

또한 모델에는 단일 API 호출에서 처리되는 텍스트의 양을 제한하는 토큰 제한이 적용됩니다. 이 임계값을 흔히 "컨텍스트 창"이라고 합니다. 모델은 다음과 같은 텍스트는 처리하지 않습니다.

이 한도를 초과합니다.

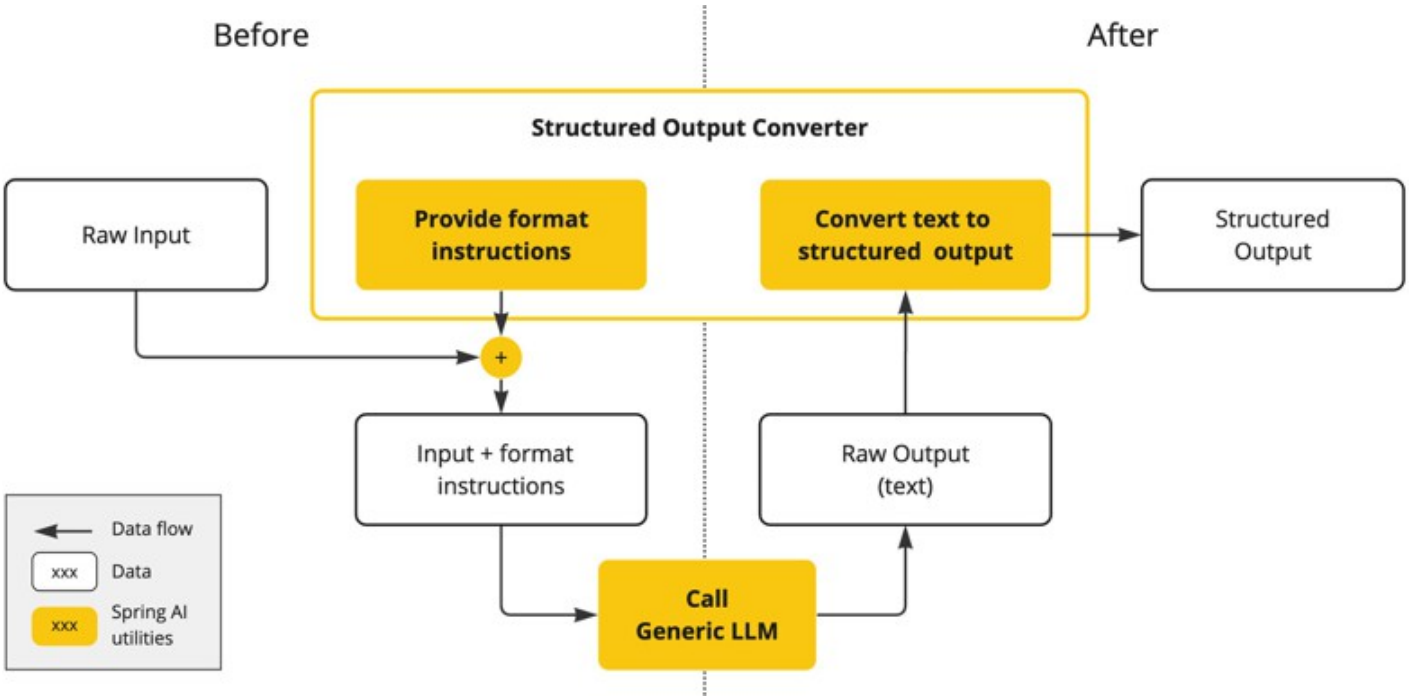
예를 들어 ChatGPT3는 4K 토큰 제한이 있는 반면, GPT4는 8K, 1GK, 32K 등 다양한 옵션을 제공합니다. Anthropic의 Claude AI 모델은 토큰 한도가 100,000개이며, Meta의 최근 연구에서는 토큰 한도가 1백만 개입니다.

수집된 셰익스피어의 작품을 GPT4로 요약하려면 데이터를 잘게 쪼개고 모델의 컨텍스트 창 제한 내에서 데이터를 표시하는 소프트웨어 엔지니어링 전략을 고안해야 합니다. Spring AI 프로젝트가 이 작업에 도움을 줍니다.

구조화된 출력

응답을 JSON으로 요청하더라도 AI 모델의 출력은 일반적으로 `java.lang.String`으로 도착합니다. 이는 올바른 JSON일 수 있지만 JSON 데이터 구조가 아닙니다. 그냥 문자열일 뿐입니다. 또한 프롬프트의 일부로 "for JSON"을 요청하는 것은 100% 정확하지 않습니다.

이러한 복잡성으로 인해 의도한 출력을 얻기 위해 프롬프트를 생성하고, 그 결과 생성된 간단한 문자열을 애플리케이션 통합에 사용할 수 있는 데이터 구조로 변환하는 전문 분야가 등장하게 되었습니다.



구조화된 출력 변환은 세심하게 제작된 프롬프트를 사용하므로 원하는 형식을 얻기 위해 모델과 여러 번 상호 작용해야 하는 경우가 많습니다.

데이터 및 API를 AI 모델로 가져오기

AI 모델에 학습되지 않은 정보를 어떻게 장착할 수 있나요?

GPT 3.5/4.0 데이터 세트는 2021년 9월까지만 확장된다는 점에 유의하세요. 따라서 이 모델은 해당 날짜 이후의 지식이 필요한 질문에 대한 답을 알지 못한다고 말합니다. 흥미로운

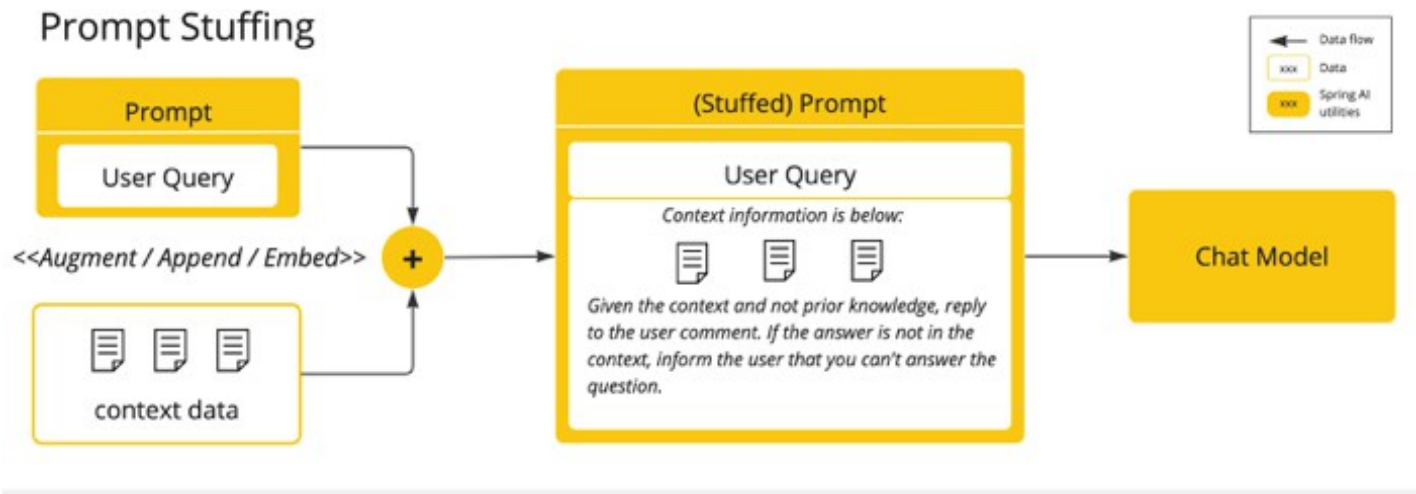
이 데이터 세트의 용량은 약 G50GB입니다.

데이터를 통합하기 위해 AI 모델을 사용자 지정하는 세 가지 기술이 있습니다:

- 미세 조정: 이 전통적인 머신 러닝 기법에는 모델을 조정하고 내부 가중치를 변경하는 것이 포함됩니다. 하지만 머신 러닝 전문가에게는 어려운 과정이며, GPT와 같은 모델의 경우 그 크기 때문에 리소스 집약도가 높습니다. 또한 일부 모델에서는 이 옵션을 제공하지 않을 수도 있습니다.
- 프롬프트 스터핑: 보다 실용적인 대안은 모델에 제공되는 프롬프트에 데이터를 임베드하는 것입니다. 모델의 토큰 제한이 주어지면 모델의

컨텍스트 창 내에 관련 데이터를 표시하는 기술이 필요합니다. 이 접근 방식을 구어체로 '스터핑'이라고 합니다.

검색 증강 생성(RAG)으로도 알려진 기술입니다.



- 도구 호출: 이 기술을 사용하면 대규모 언어 모델을 외부 시스템의 API에 연결하는 도구(사용자 정의 서비스)를 등록할 수 있습니다.

  Spring AI는 도구 호출을 지원하기 위해 작성해야 하는 코드를 크게 간소화합니다.

검색 증강 세대

정확한 AI 모델 응답을 위해 관련 데이터를 프롬프트에 통합하는 문제를 해결하기 위해 검색 증강 생성(RAG)이라는 기술이 등장했습니다.

이 접근 방식에는 문서에서 비정형 데이터를 읽고 변환한 다음 벡터 데이터베이스에 기록하는 일괄 처리 스타일의 프로그래밍 모델이 포함됩니다. 높은 수준에서 이것은 ETL(추출, 변환 및 로드) 파이프라인입니다. 벡터 데이터베이스는 RAG 기술의 검색 부분에서 사용됩니다.

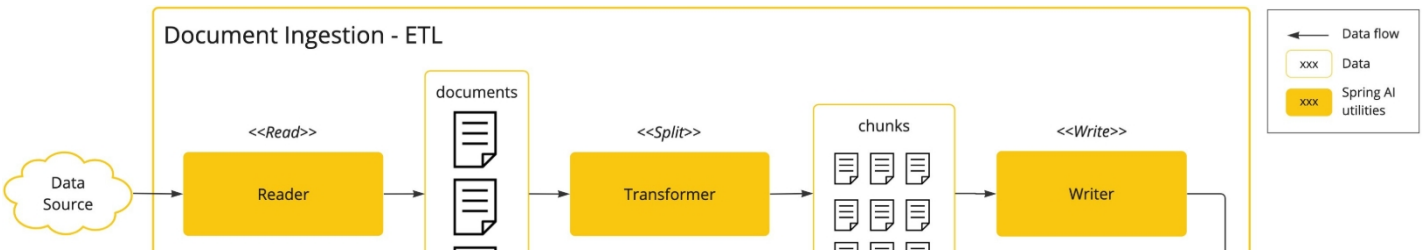
비정형 데이터를 벡터 데이터베이스에 로드할 때 가장 중요한 변환 작업 중 하나는 원본 문서를 더 작은 조각으로 분할하는 것입니다. 원본 문서를 더 작은 조각으로 분할하는 절차에는 두 가지 중요한 단계가 있습니다:

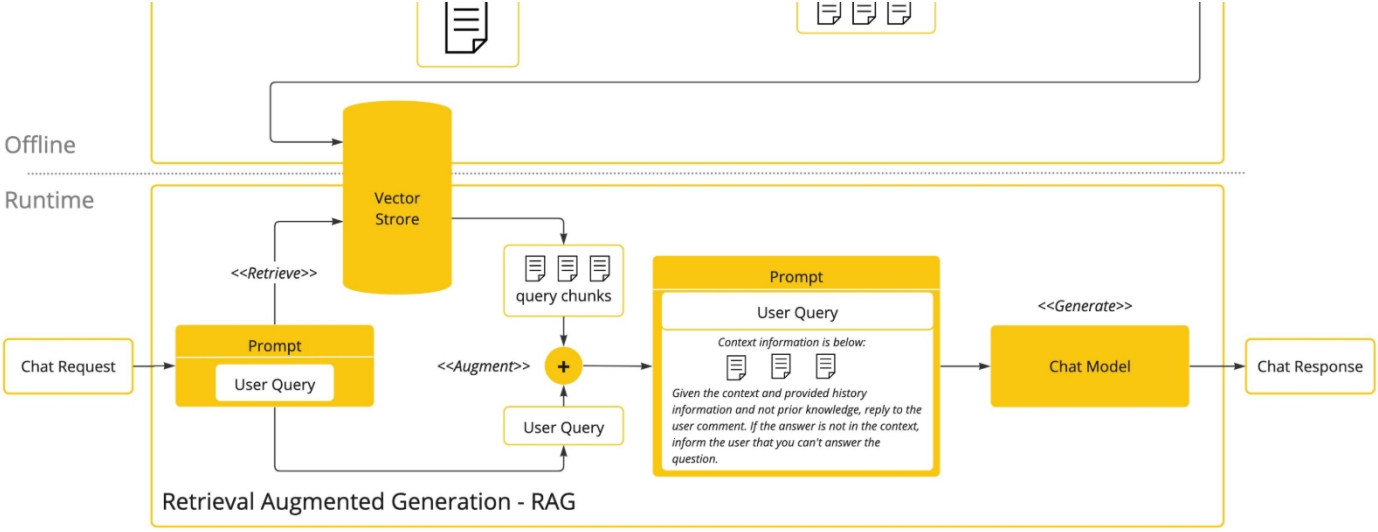
1. 콘텐츠의 의미적 경계를 유지하면서 문서를 여러 부분으로 분할합니다. 예를 들어 단락과 표가 있는 문서의 경우 문서를 분할하지 않아야 합니다.

   단락이나 표의 중간에 삽입하지 마세요. 코드의 경우 메서드 구현 중간에 코드를 분할하지 마세요.

2. 문서의 파트를 AI 모델의 토큰 제한에 비해 크기가 작은 부분으로 더 분할합니다.

RAG의 다음 단계는 사용자 입력을 처리하는 것입니다. 사용자의 질문에 AI 모델이 답변해야 하는 경우, 해당 질문과 모든 "유사한" 문서 조각이 AI 모델에 전송되는 프롬프트에 배치됩니다. 이것이 바로 벡터 데이터베이스를 사용하는 이유입니다. 벡터 데이터베이스는 유사한 콘텐츠를 찾는 데 매우 능숙합니다.





- ETL 파이프라인은 데이터 소스에서 데이터를 추출하고 구조화된 벡터 저장소에 저장하는 흐름을 조율하여 데이터를 AI 모델에 전달할 때 검색을 위한 최적의 형식으로 데이터를 확보하는 방법에 대한 자세한 정보를 제공합니다.
- ChatClient - RAG에서는 애플리케이션에서 RAG 기능을 활성화하기 위해 `QuestionAnswerAdvisor`를 사용하는 방법을 설명합니다.

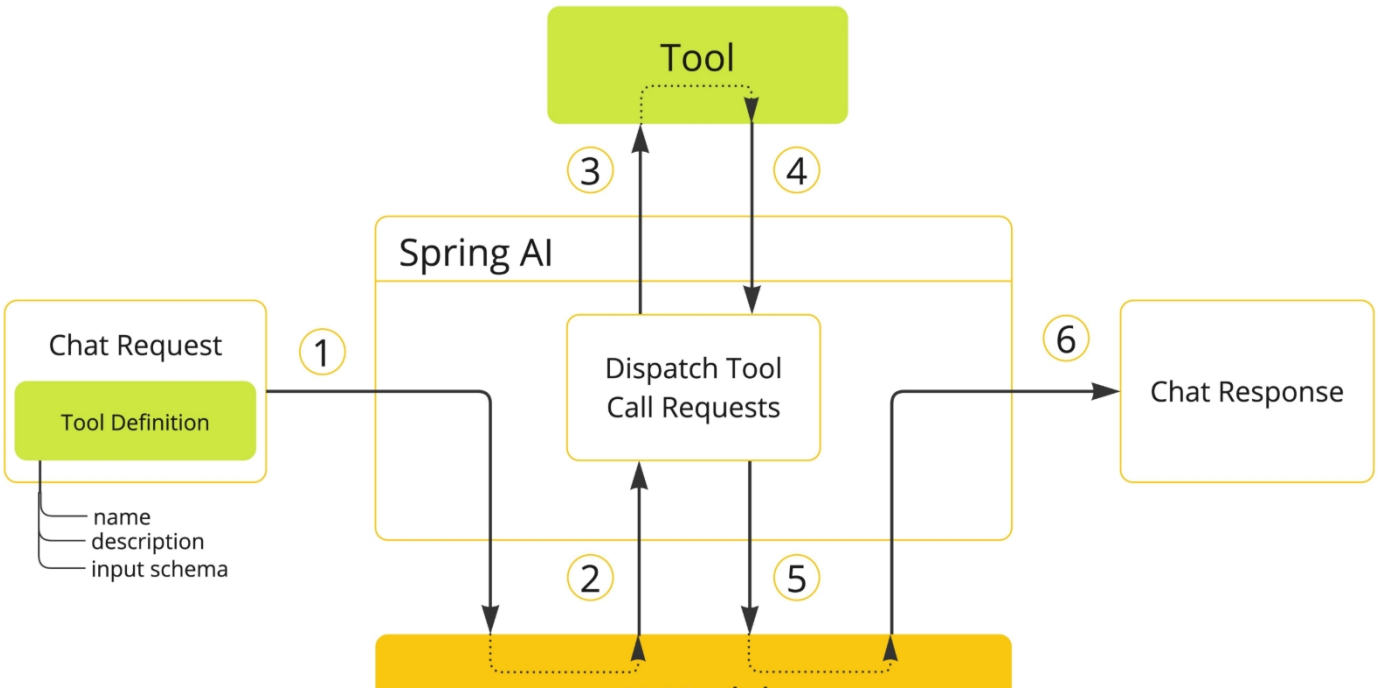
도구 호출

대규모 언어 모델(LLM)은 학습 후 동결되어 지식이 오래되어 외부 데이터에 액세스하거나 수정할 수 없게 됩니다.

도구 호출 메커니즘은 이러한 단점을 해결합니다. 이를 통해 자체 서비스를 도구로 등록하여 대규모 언어 모델을 외부 시스템의 API에 연결할 수 있습니다. 이러한 시스템은 LLM에 실시간 데이터를 제공하고 대신 데이터 처리 작업을 수행할 수 있습니다.

Spring AI는 도구 호출을 지원하기 위해 작성해야 하는 코드를 크게 간소화합니다. 도구 호출 대화를 대신 처리해 줍니다. 도구를 `@Tool` 주석 처리된 메서드로 제공하고 프롬프트 옵션에 제공하여 모델에서 사용할 수 있도록 할 수 있습니다. 또한 여러 도구를 정의하고 참조할 수 있습니다.

도구를 하나의 프롬프트에서 사용할 수 있습니다.



AI Model

1. 모델이 도구를 사용할 수 있게 하려면 채팅 요청에 해당 도구의 정의를 포함합니다. 각 도구 정의는 이름, 설명, 입력 매개변수의 스키마로 구성됩니다.
2. 모델이 도구를 호출하기로 결정하면 정의된 스키마에 따라 모델링된 도구 이름과 입력 파라미터가 포함된 응답을 보냅니다.
3. 애플리케이션은 도구 이름을 사용하여 프로비저닝된 입력 매개변수로 도구를 식별하고 실행할 책임이 있습니다.
4. 도구 호출의 결과는 애플리케이션에서 처리됩니다.
5. 애플리케이션은 도구 호출 결과를 모델로 다시 보냅니다.
6. 모델은 도구 호출 결과를 추가 컨텍스트로 사용하여 최종 응답을 생성합니다.

다양한 AI 모델에서 이 기능을 사용하는 방법에 대한 자세한 내용은 도구 호출 문서를 참조하세요.

AI 응답 평가

최종 애플리케이션의 정확성과 유용성을 보장하기 위해서는 사용자 요청에 대한 AI 시스템의 출력을 효과적으로 평가하는 것이 매우 중요합니다. 이러한 목적으로 사전 학습된 모델 자체를 사용할 수 있는 몇 가지 새로운 기술이 등장하고 있습니다.

이 평가 프로세스에는 생성된 응답이 사용자의 의도와 쿼리의 맥락에 부합하는지 여부를 분석하는 작업이 포함됩니다. 관련성, 일관성, 사실 정확성 등의 메트릭을 사용하여 AI가 생성한 품질을 측정합니다.

한 가지 접근 방식은 사용자의 요청과 AI 모델의 응답을 모두 모델에 제시하고 응답이 제공된 데이터와 일치하는지 쿼리하는 것입니다.

또한, 벡터 데이터베이스에 저장된 정보를 보조 데이터로 활용하면 평가 프로세스를 개선하여 응답 관련성을 판단하는 데 도움이 될 수 있습니다.

Spring AI 프로젝트는 현재 모델 응답을 평가하기 위한 기본 전략에 액세스할 수 있는 평가자 API를 제공합니다. 자세한 내용은 평가 테스트 문서를 참조하세요.