

# KTds Azure AI 교육 교재 Day 4

## Azure OpenAI Services

Azure OpenAI는 개발자가 애플리케이션에 AI 기능을 추가하는 데 사용할 수 있는 언어별 SDK와 REST API를 모두 제공합니다. Azure OpenAI의 생성 AI 기능은 모델을 통해 제공됩니다. Azure OpenAI 서비스에서 사용할 수 있는 모델은 각각 포커스가 있는 서로 다른 제품군에 속합니다. 이러한 모델 중 하나를 사용하려면 Azure OpenAI 서비스를 통해 배포해야 합니다.

Azure OpenAI 리소스를 만들고 모델을 배포한 후에는 앱을 구성할 수 있습니다.

### 1. 사용 가능한 엔드포인트

Python, C#, JavaScript 등에 사용할 수 있는 REST API 또는 SDK를 통해 Azure OpenAI에 액세스할 수 있습니다. 배포된 모델과 상호 작용하는 데 사용할 수 있는 엔드포인트는 다르게 사용되며 특정 엔드포인트는 특정 모델만 사용할 수 있습니다. 사용 가능한 엔드포인트:

- **Completion** - 이 모델은 입력 프롬프트를 사용하고 하나 이상의 예측 완료를 생성합니다. 스튜디오에서 이 플레이그라운드를 볼 수 있지만 이 모듈에서는 자세히 다루지 않습니다.
- **ChatCompletion** - 이 모델은 채팅 대화 형식(보내는 메시지로 역할이 지정됨)을 입력하고, 다음 채팅 완료가 생성됩니다.
- **Embeddings** - 이 모델은 입력을 사용하고 해당 입력의 벡터 표현을 반환합니다.

예를 들어 ChatCompletion에 대한 입력은 각 메시지에 대해 명확하게 정의된 역할이 있는 대화입니다.

```
{"ROLE": "SYSTEM", "CONTENT": "YOU ARE A HELPFUL ASSISTANT, TEACHING PEOPLE ABOUT AI."},  
{"ROLE": "USER", "CONTENT": "DOES AZURE OPENAI SUPPORT MULTIPLE LANGUAGES?"},  
{"ROLE": "ASSISTANT", "CONTENT": "YES, AZURE OPENAI SUPPORTS SEVERAL LANGUAGES, AND CAN  
TRANSLATE BETWEEN THEM."},  
{"ROLE": "USER", "CONTENT": "DO OTHER AZURE AI SERVICES SUPPORT TRANSLATION TOO?"}
```

AI 모델에 실제 대화를 제공하면 보다 정확한 톤, 구문 및 컨텍스트로 더 나은 응답을 생성할 수 있습니다. ChatCompletion 엔드포인트를 사용하면 모델이 다음 사용자 메시지와 함께 채팅 기록을 전송하여 보다 현실적인 대화를 할 수 있게 됩니다.

ChatCompletion에서는 요약 또는 엔터티 추출과 같은 비채팅 시나리오도 허용합니다. 이 작업은 사용자 입력과 함께 시스템 정보 및 원하는 항목을 지정하여 간단한 대화를 제공함으로써 수행할 수 있습니다. 예를 들어 작업 설명을 생성하려면 다음 대화 입력과 같은 ChatCompletion을 제공하세요.

```
{"ROLE": "SYSTEM", "CONTENT": "YOU ARE AN ASSISTANT DESIGNED TO WRITE INTRIGUING JOB  
DESCRIPTIONS."},  
{"ROLE": "USER", "CONTENT": "WRITE A JOB DESCRIPTION FOR THE FOLLOWING JOB TITLE: 'BUSINESS  
INTELLIGENCE ANALYST'. IT SHOULD INCLUDE RESPONSIBILITIES, REQUIRED QUALIFICATIONS, AND  
HIGHLIGHT BENEFITS LIKE TIME OFF AND FLEXIBLE HOURS."}
```

## 참고

Completion은 이전 gpt-3 세대 모델에 사용할 수 있는 반면 ChatCompletion은 gpt-4 모델에서 지원되는 유일한 옵션이며 gpt-35-turbo 모델을 사용할 때 선호되는 엔드포인트입니다.

## 2. Azure OpenAI REST API 사용

Azure OpenAI는 개발자가 애플리케이션에 AI 기능을 추가하는 데 사용할 수 있는 응답을 상호 작용하고 생성하기 위한 REST API를 제공합니다. 이 단원에서는 API의 예제 사용, 입력 및 출력을 다룹니다.

REST API를 호출할 때마다 Azure OpenAI 리소스의 엔드포인트 및 키와 배포된 모델에 대해 지정한 이름이 필요합니다. 다음 예제에서는 다음 자리 표시자가 사용됩니다.

자리 표시자 이름	값
YOUR_ENDPOINT_NAME	이 기본 엔드포인트는 Azure Portal의 <b>키 및 엔드포인트</b> 섹션에서 찾을 수 있습니다. 리소스의 기본 엔드포인트 (예: <a href="https://sample.openai.azure.com/">https://sample.openai.azure.com/</a> )입니다.
YOUR_API_KEY	키는 Azure Portal의 <b>키 및 엔드포인트</b> 섹션에서 찾을 수 있습니다. 리소스에 대해 두 키를 사용할 수 있습니다.
YOUR_DEPLOYMENT_NAME	이 배포 이름은 Azure AI Foundry에서 모델을 배포할 때 제공되는 이름입니다.

### 3. ChatCompletion

Azure OpenAI 리소스에 모델을 배포한 후에는 POST 요청을 사용하여 서비스에 프롬프트를 보낼 수 있습니다.

CURL

```
HTTPS://YOUR_ENDPOINT_NAME.OPENAI.AZURE.COM/OPENAI/DEPLOYMENTS/YOUR_DEPLOYMENT_NAME/CHAT/COMPLETIONS?API-VERSION=2023-03-15-PREVIEW  
-H "CONTENT-TYPE: APPLICATION/JSON"  
-H "API-KEY: YOUR_API_KEY"  
-D '{"MESSAGES":[{"ROLE": "SYSTEM", "CONTENT": "YOU ARE A HELPFUL ASSISTANT, TEACHING PEOPLE ABOUT AI."},  
 {"ROLE": "USER", "CONTENT": "DOES AZURE OPENAI SUPPORT MULTIPLE LANGUAGES?"},  
 {"ROLE": "ASSISTANT", "CONTENT": "YES, AZURE OPENAI SUPPORTS SEVERAL LANGUAGES, AND CAN TRANSLATE BETWEEN THEM."},  
 {"ROLE": "USER", "CONTENT": "DO OTHER AZURE AI SERVICES SUPPORT TRANSLATION TOO?"}]}'
```

API의 응답은 다음 JSON과 비슷합니다.

```
{  
    "ID": "CHATCMPL-6v7MKQJ980V1YBEC6ETRKPRQFJNw9",  
    "OBJECT": "CHAT.COMPLETION",  
    "CREATED": 1679001781,  
    "MODEL": "GPT-35-TURBO",  
    "USAGE": {  
        "PROMPT_TOKENS": 95,  
        "COMPLETION_TOKENS": 84,  
        "TOTAL_TOKENS": 179  
    },  
    "CHOICES": [  
        {  
            "MESSAGE": {  
                "ROLE": "ASSISTANT",  
                "CONTENT": "YES, OTHER AZURE AI SERVICES ALSO SUPPORT TRANSLATION.  
AZURE AI SERVICES OFFER TRANSLATION BETWEEN MULTIPLE LANGUAGES FOR TEXT, DOCUMENTS, OR  
CUSTOM TRANSLATION THROUGH AZURE AI SERVICES TRANSLATOR."  
            },  
            "FINISH_REASON": "STOP",  
            "INDEX": 0  
        }  
    ]  
}
```

REST 엔드포인트를 사용하면 temperature, max\_tokens 등과 같은 다른 선택적 입력 매개

변수를 지정할 수 있습니다. 요청에 이러한 매개 변수를 포함하려면 요청을 사용하여 입력 데이터에 추가합니다.

## 4. Embedding

포함은 기계 학습 모델에서 쉽게 사용할 수 있는 특정 형식에 유용합니다. 입력 텍스트에서 포함을 생성하려면 POST은(는) embeddings 엔드포인트에 대한 요청입니다.

CURL

```
HTTPS://YOUR_ENDPOINT_NAME.OPENAI.AZURE.COM/OPENAI/DEPLOYMENTS/YOUR_DEPLOYMENT_NAME/EMBEDDINGS?API-VERSION=2022-12-01
```

```
-H "CONTENT-TYPE: APPLICATION/JSON"  
-H "API-KEY: YOUR_API_KEY"  
-D "{\"INPUT\": \"THE FOOD WAS DELICIOUS AND THE WAITER...\"}"
```

포함을 생성할 때 포함을 위한 Azure OpenAI의 모델을 사용해야 합니다. 그러한 모델은 원하는 기능에 따라 text-embedding 또는 text-similarity(으)로 시작합니다.

API의 응답은 다음 JSON과 비슷합니다.

```
{  
  "OBJECT": "LIST",  
  "DATA": [  
    {  
      "OBJECT": "EMBEDDING",  
      "EMBEDDING": [  
        0.0172990688066482523,  
        -0.0291879814639389515,  
        ....  
        0.0134544348834753042,
```

```

        ],
        "INDEX": 0
    }
],
"MODEL": "TEXT-EMBEDDING-ADA:002"
}

```

## 5. SDK에서 Azure OpenAI 사용

REST API 외에도 사용자는 C# 및 Python SDK를 통해 Azure OpenAI 모델에 액세스할 수 있습니다. REST 및 이러한 SDK를 통해 동일한 기능을 사용할 수 있습니다.

이 단원에서 다룬 두 SDK의 경우 Azure OpenAI 리소스의 엔드포인트 및 키와 배포된 모델에 대해 지정한 이름이 필요합니다. 다음 코드 조각에서는 다음 자리 표시자가 사용됩니다.

자리 표시자 이름	값
YOUR_ENDPOINT_NAME	이 기본 엔드포인트는 Azure Portal의 <b>키 및 엔드포인트</b> 섹션에서 찾을 수 있습니다. 리소스의 기본 엔드포인트 (예: <a href="https://sample.openai.azure.com/">https://sample.openai.azure.com/</a> )입니다.
YOUR_API_KEY	키는 Azure Portal의 <b>키 및 엔드포인트</b> 섹션에서 찾을 수 있습니다. 리소스에 대해 두 키를 사용할 수 있습니다.
YOUR_DEPLOYMENT_NAME	이 배포 이름은 모델을 배포할 때 제공되는 이름입니다.

## 6. 라이브러리 설치

먼저, 기본 설정 언어에 대한 클라이언트 라이브러리를 설치합니다. C# SDK는 REST API의 .NET 적응이며 Azure OpenAI용으로 특별히 빌드되지만 Azure OpenAI 리소스 또는 비 Azure OpenAI 엔드포인트에 연결하는 데 사용할 수 있습니다. Python SDK는 OpenAI에서 빌드 및 유지 관리됩니다.

PIP INSTALL OPENAI

## 7. Azure OpenAI 리소스에 액세스하도록 앱 구성

각 언어에 대한 구성은 약간 다르지만 둘 다 동일한 매개 변수를 설정해야 합니다. 필요한 매개 변수는 endpoint, key 및 deployment name입니다.

앱에 라이브러리를 추가하고 클라이언트에 필요한 매개 변수를 설정합니다.

```
# ADD OPENAI LIBRARY

FROM OPENAI IMPORT AZUREOPENAI


DEPLOYMENT_NAME = '<YOUR_DEPLOYMENT_NAME>'

# INITIALIZE THE AZURE OPENAI CLIENT

CLIENT = AZUREOPENAI(
    AZURE_ENDPOINT = '<YOUR_ENDPOINT_NAME>',
    API_KEY='<YOUR_API_KEY>',
    API_VERSION="20xx-xx-xx" # TARGET VERSION OF THE API, SUCH AS 2024-02-15-
PREVIEW
)
```

## 8. AZURE OPENAI 리소스 호출

Azure OpenAI에 대한 연결을 구성한 후 모델에 프롬프트를 보냅니다.

```
RESPONSE = CLIENT.CHAT.COMPLETIONS.CREATE(
    MODEL=DEPLOYMENT_NAME,
```

```
MESSAGES=[  
    {"ROLE": "SYSTEM", "CONTENT": "YOU ARE A HELPFUL ASSISTANT."},  
    {"ROLE": "USER", "CONTENT": "WHAT IS AZURE OPENAI?"}  
]  
)  
  
GENERATED_TEXT = RESPONSE.CHOICES[0].MESSAGE.CONTENT  
  
# PRINT THE RESPONSE  
  
PRINT("RESPONSE: " + GENERATED_TEXT + "₩N")
```

응답 개체에는 total\_tokens 및 finish\_reason과(와) 같은 여러 값이 포함됩니다. 응답 개체의 완성은 다음 완성과 유사합니다.

"AZURE OPENAI IS A CLOUD-BASED ARTIFICIAL INTELLIGENCE (AI) SERVICE THAT OFFERS A RANGE OF TOOLS AND SERVICES FOR DEVELOPING AND DEPLOYING AI APPLICATIONS. AZURE OPENAI PROVIDES A VARIETY OF SERVICES FOR TRAINING AND DEPLOYING MACHINE LEARNING MODELS, INCLUDING A MANAGED SERVICE FOR TRAINING AND DEPLOYING DEEP LEARNING MODELS, A MANAGED SERVICE FOR DEPLOYING MACHINE LEARNING MODELS, AND A MANAGED SERVICE FOR MANAGING AND DEPLOYING MACHINE LEARNING MODELS."

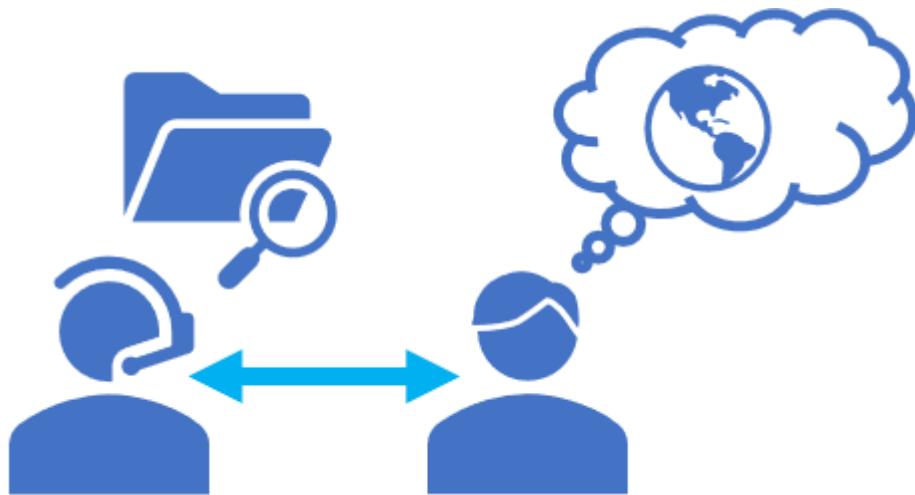
C# 및 Python 모두에서 호출에는 temperature 및 max\_tokens를 포함한 선택적 매개 변수가 포함될 수 있습니다.

# KTds Azure AI 교육 교재 Day 5

## 소개

모든 조직에서는 정보를 사용하여 의사 결정을 내리고, 질문에 답변하고, 효율적으로 업무를 수행합니다. 대부분의 조직에서 발생하는 문제는 정보 부족이 아니라 많은 문서, 데이터베이스 및 정보가 저장되는 기타 자료에서 정보를 찾고 추출하는 데 따르는 어려움입니다.

예를 들어 *Margie's Travel*은 전 세계 도시 여행 정보를 체계적으로 제공하는 데 특화된 여행사입니다. 시간이 지남에 따라 회사는 고객이 제출한 호텔 리뷰뿐만 아니라 브로슈어 등의 문서로 된 상당한 양의 정보를 축적했습니다. 이 데이터는 여행사와 고객이 여행을 계획할 때 유용한 소스가 되지만, 엄청난 양의 데이터로 인해 특정 고객의 질문에 답변할 때 관련 정보를 찾기가 어려워질 수 있습니다.



이러한 문제를 해결하기 위해 *Margie's Travel*에서는 문서를 인덱싱하여 검색을 쉽게 할 수 있도록 하는 솔루션을 구현할 수 있습니다. 이 솔루션을 사용하면 에이전트와 고객이 인덱스를 쿼리하여 관련 문서를 찾고 해당 문서에서 정보를 추출할 수 있습니다.

## Azure AI Search

Azure AI 검색은 광범위한 데이터 원본을 인덱싱 및 쿼리할 수 있고 스케일링 성능이 뛰어난 종합 검색 솔루션을 만들 수 있는 클라우드 기반 솔루션을 제공합니다. Azure AI 검색을 사용하면 다음을 수행할 수 있습니다.

다양한 원본의 데이터 및 문서를 인덱싱합니다.

인식 기술을 사용하여 인덱스 데이터를 보강합니다.

추출된 데이터를 분석하고 통합하기 위해 지식 저장소에 저장합니다.

## 용량 관리

Azure AI Search 솔루션을 만들려면 Azure 구독에서 Azure AI Search 리소스를 만들어야 합니다. 빌드하려는 특정 솔루션에 따라 데이터 스토리지 및 기타 애플리케이션 서비스에 대한 Azure 리소스가 필요할 수도 있습니다.

## 서비스 계층 및 용량 관리

Azure AI Search 리소스를 만들 때 가격 책정 계층을 지정해야 합니다. 선택한 가격 책정 계층에 따라 검색 서비스의 용량 제한과 사용 가능한 구성 옵션 그리고 서비스 비용이 결정됩니다. 사용 가능한 가격 책정 계층은 다음과 같습니다.

무료(F): 서비스를 살펴보거나 제품 설명서의 자습서를 사용해 보려면 이 계층을 사용합니다.

기본(B): 최대 15개의 인덱스와 5GB의 인덱스 데이터를 포함하는 소규모 검색 솔루션에 이 계층을 사용합니다.

표준(S): 엔터프라이즈 규모 솔루션에 이 계층을 사용합니다. 이 계층에는 S, S2 및 S3을 비롯한 여러 변형이 있습니다. 인덱스 및 스토리지 측면에서 더 많은 용량을 제공하는 S3HD와 더 적은 수의 인덱스에 대한 빠른 읽기 성능에 최적화된 S3HD.

스토리지 최적화(L): 쿼리 대기 시간이 더 긴 비용으로 큰 인덱스를 만들어야 하는 경우 스토리지 최적화 계층(L1 또는 L2)을 사용합니다.

## 참고

나중에 변경할 수 없으므로 솔루션에 가장 적합한 가격 책정 계층을 선택하는 것이 중요

합니다. 선택한 가격 책정 계층이 솔루션에 더 이상 적합하지 않은 경우 새 Azure AI 검색 리소스를 만들고 모든 인덱스와 개체를 다시 만들어야 합니다.

## 복제본 및 파티션

선택한 가격 책정 계층에 따라 '복제본' 및 '파티션'을 만들어 확장성과 사용 가능성을 높이도록 솔루션을 최적화할 수 있습니다.

- '복제본'은 검색 서비스의 인스턴스이며, 이를 클러스터의 노드로 간주할 수 있습니다. 복제본 수를 늘리면 진행 중인 인덱싱 작업을 관리하는 동시에 여러 쿼리 요청을 처리하기에 충분한 용량을 유지할 수 있습니다.
- '파티션'은 인덱스를 여러 스토리지 위치로 나누는 데 사용되며, 이를 통해 인덱스를 쿼리하거나 다시 빌드하는 등의 I/O 작업을 분할할 수 있습니다.

구성하는 복제본과 파티션의 조합에 따라 솔루션에서 사용하는 '검색 단위'가 결정됩니다. 간단히 말해서 검색 단위 수는 복제본 수와 파티션 수를 곱한 값( $R \times P = SU$ )입니다. 예를 들어 4개의 복제본이 있고 3개의 파티션이 있는 리소스는 12개의 검색 단위를 사용합니다.

## 검색 구성 요소 이해

AI 검색 솔루션은 데이터 추출, 보강, 인덱싱, 검색 프로세스에서 각각 중요한 역할을 하는 여러 구성 요소로 구성됩니다.

### 데이터 원본



대부분 검색 솔루션의 시작은 검색할 데이터를 포함하는 **데이터 원본**입니다. Azure AI 검색은 다음을 포함한 여러 형식의 데이터 원본을 지원합니다.

- Azure Blob 스토리지 컨테이너의 비구조적 파일
- Azure SQL Database의 테이블

- Cosmos DB의 문서

Azure AI 검색은 인덱싱을 위해 이러한 데이터 원본에서 데이터를 끌어올 수 있습니다.

또는 애플리케이션에서 JSON 데이터를 기존 데이터 원본에서 끌어오지 않고 인덱스에 직접 푸시할 수 있습니다.

## 기술 집합



기본 검색 솔루션에서는 데이터 원본에서 추출된 데이터를 인덱싱할 수 있습니다. 추출 할 수 있는 정보는 데이터 원본에 따라 달라집니다. 예를 들어 데이터베이스에서 데이터를 인덱싱할 때 데이터베이스 테이블의 필드를 추출할 수 있습니다. 아니면 문서 집합을 인덱싱할 때 파일 이름, 수정된 날짜, 크기 및 작성자와 같은 파일 메타데이터를 문서의 텍스트 내용과 함께 추출할 수 있습니다.

데이터 원본에서 직접 추출된 데이터 값을 인덱싱하는 기본 검색 솔루션도 유용할 수 있지만, 최신 애플리케이션 사용자의 기대로 인해 데이터에 대한 더 풍부한 인사이트가 필요해졌습니다. Azure AI 검색에서 인덱싱 프로세스의 일부로 AI(인공 지능) 기술을 적용하여 인덱스 필드에 매핑할 수 있는 새 정보로 원본 데이터를 보강할 수 있습니다. 인덱스에서 사용하는 기술은 '기술 세트'에 캡슐화되며, 이 기술 세트는 특정 AI 기술로 획득한 인사이트를 통해 단계마다 원본 데이터를 강화하는 보강 파이프라인을 정의합니다. AI 기술로 추출할 수 있는 정보 종류의 예는 다음과 같습니다.

- 문서 작성에 사용되는 언어
- 문서에서 설명하는 주요 테마 또는 항목을 확인하는 데 도움이 될 수 있는 주요 문구
- 문서를 양수 또는 음수로 수량화하는 감정 점수
- 콘텐츠에 언급된 특정 위치, 사람, 조직 또는 주요 사건
- AI에서 생성된 이미지의 설명 또는 광학 인식에서 추출한 이미지 텍스트

- 특정 요구 사항을 충족하기 위해 개발하는 사용자 지정 기술

## 인덱서



'인덱서'는 전체 인덱싱 프로세스를 구동하는 엔진입니다. 인덱서는 기술 세트의 기술을 사용하여 추출된 출력을 원래 데이터 원본에서 추출된 데이터 및 메타데이터 값과 함께 가져와 인덱스의 필드에 매핑합니다.

인덱서는 만들어지면 자동으로 실행되며 인덱스에 문서를 추가하기 위해 정기적으로 실행되거나 요청 시 실행되도록 예약할 수 있습니다. 인덱스에 새 필드를 추가하거나 기술 세트에 새 기술을 추가하는 등과 같은 일부 경우에 인덱서를 다시 실행하기 전에 인덱스를 다시 설정해야 할 수 있습니다.

## Index



인덱스는 검색 가능한 인덱싱 프로세스의 결과입니다. 인덱스는 인덱싱 중 추출된 값을 포함하는 필드가 있는 JSON 문서 컬렉션으로 구성됩니다. 클라이언트 애플리케이션은 인덱스를 쿼리하여 정보를 검색, 필터링, 정렬할 수 있습니다.

각 인덱스 필드는 다음 특성으로 구성될 수 있습니다.

- key: 인덱스 레코드에 대한 고유 키를 정의하는 필드.
- searchable: 전체 텍스트 검색을 사용하여 쿼리할 수 있는 필드.

- filterable: 지정된 제약 조건과 일치하는 문서만 반환하도록 필터 식에 포함할 수 있는 필드.
- sortable: 결과를 정렬하는 데 사용할 수 있는 필드.
- facetable: '패싯'(알려진 필드 값 목록을 기반으로 결과를 필터링하는 데 사용되는 사용자 인터페이스 요소)의 값을 결정하는 데 사용할 수 있는 필드.
- retrievable: 검색 결과에 포함될 수 있는 필드('기본적으로 모든 필드는 이 특성을 명시적으로 제거하지 않는 한 검색 가능함')

## 인덱싱 프로세스 이해

인덱싱 프로세스는 인덱싱된 각 엔티티에 대한 문서를 만들어 작동합니다. 인덱싱하는 동안 보강<sup>파이프라인</sup>은 데이터 원본의 메타데이터와 인지 기술에 의해 추출된 보강 필드를 결합하는 문서를 반복적으로 빌드합니다. 인덱싱된 각 문서는 처음에 원본 데이터에서 직접 추출된 필드에 매핑된 인덱스 필드가 있는 문서로 구성된 JSON 구조로 생각할 수 있습니다.

- 문서
  - metadata\_storage\_name
  - metadata\_author
  - 콘텐츠

데이터 원본의 문서에 이미지가 포함된 경우 다음과 같이 이미지 데이터를 추출하고 각 이미지를 normalized\_images 컬렉션에 배치하도록 인덱서를 구성할 수 있습니다.

- 문서
  - metadata\_storage\_name
  - metadata\_author
  - 콘텐츠
  - normalized\_images
    - image0

- image1

이러한 방식으로 이미지 데이터를 정규화하면 이미지 수집을 이미지 데이터에서 정보를 추출하는 기술에 대한 입력으로 사용할 수 있습니다.

각 기술은 문서에 필드를 추가하므로 예를 들어 문서가 작성된 언어를 감지하는 기술은 다음과 같이 해당 출력을 언어 필드에 저장할 수 있습니다.

- 문서
  - 메타데이터\_저장소\_이름
  - metadata\_author
  - 콘텐츠
  - normalized\_images
    - image0
    - image1
  - 언어

문서는 계층적으로 구조화되며, 기술은 계층 내의 특정 컨텍스트에 적용되므로 문서의 특정 수준에서 각 항목에 대한 기술을 실행할 수 있습니다. 예를 들어 정규화된 이미지 컬렉션의 각 이미지에 대해 *OCR*(광학 문자 인식) 기술을 실행하여 포함된 텍스트를 추출 할 수 있습니다.

- 문서
  - metadata\_storage\_name
  - metadata\_author
  - 콘텐츠
  - normalized\_images
    - image0
      - 문자 메시지
    - image1

- 문자 메시지

- 언어

각 기술의 출력 필드는 파이프라인의 뒷부분에 있는 다른 기술에 대한 입력으로 사용할 수 있으며, 그러면 해당 출력이 문서 구조에 저장됩니다. 예를 들어 병합 기술을 사용하여 원본 텍스트 콘텐츠를 각 이미지에서 추출한 텍스트와 결합하여 이미지 텍스트를 포함하여 문서의 모든 텍스트가 포함된 새 merged\_content 필드를 만들 수 있습니다.

- 문서

- metadata\_storage\_name

- metadata\_author

- 콘텐츠

- normalized\_images

- image0

- 문자 메시지

- image1

- 문자 메시지

- 언어

- 병합된\_콘텐츠

파이프라인 끝에 있는 최종 문서 구조의 필드는 다음 두 가지 방법 중 하나로 인덱서에 의해 인덱스 필드에 매핑됩니다.

1. 원본 데이터에서 직접 추출된 필드는 모두 인덱스 필드에 매핑됩니다. 이러한 매핑은 암시적일 수 있습니다(필드는 인덱스의 이름이 같은 필드에 자동으로 매핑됨) 또는 명시적(매핑은 원본 필드를 인덱스 필드와 일치하도록 정의되며, 종종 필드 이름을 더 유용한 것으로 바꾸거나 매핑될 때 데이터 값에 함수를 적용하기 위해 정의됨).
2. 기술 세트 내 기술의 출력 필드는 출력의 계층적 위치에서 대상 필드인 인덱스로 명시적으로 매핑됩니다.

## 필터링 및 정렬 적용

이는 사용자가 필드 값을 기준으로 필터링하고 정렬하여 쿼리 결과를 구체화할 수 있는 검색 솔루션에서 일반적으로 사용됩니다. Azure AI 검색은 검색 쿼리 API를 통해 이러한 기능을 모두 지원합니다.

### 결과 필터링

다음 두 가지 방법으로 쿼리에 필터를 적용할 수 있습니다.

- '단순' 검색 식에 필터 조건을 포함합니다.
- '전체' 구문 검색 식으로 **\$filter** 매개 변수에 OData 필터 식을 제공합니다.

인덱스의 모든 *filterable*에 필터를 적용할 수 있습니다.

예를 들어 **author** 필드 값이 'Reviewer'인 문서 중에서 'London'이라는 텍스트를 포함하고 있는 문서를 찾는다고 가정합니다.

다음과 같은 '단순' 검색 식을 제출하여 결과를 얻을 수 있습니다.

```
SEARCH=LONDON+AUTHOR='REVIEWER'
```

```
QUERYTYPE=SIMPLE
```

또는 다음과 같이 '전체' Lucene 검색 식으로 **\$filter** 매개 변수에 OData 필터를 사용할 수 있습니다.

```
SEARCH=LONDON
```

```
$FILTER=AUTHOR EQ 'REVIEWER'
```

```
QUERYTYPE=FULL
```

### 팁

OData **\$filter** 식은 대/소문자를 구분합니다.

### 패싯을 사용한 필터링

'패싯'은 결과 집합의 필드 값을 기준으로 필터링 조건에 해당하는 사용자를 표시하는 유용한 방법입니다. 필드에 사용자 인터페이스에 링크나 옵션으로 표시할 수 있는 개별 값이 적은 경우에 가장 적합합니다.

패싯을 사용하려면 초기 쿼리에서 가능한 값을 검색하는 *facetable* 필드를 지정해야 합니다.

다. 예를 들어 다음 매개 변수를 사용하여 **author** 필드에 사용할 수 있는 모든 값을 반환할 수 있습니다.

```
SEARCH=*
```

```
Facet=AUTHOR
```

이 쿼리의 결과는 사용자가 선택할 수 있도록 사용자 인터페이스에 표시할 수 있는 개별 패싯 값의 컬렉션을 포함합니다. 그런 다음 후속 쿼리에서 선택한 패싯 값을 사용하여 결과를 필터링할 수 있습니다.

```
SEARCH=*
```

```
$FILTER=AUTHOR EQ 'SELECTED-FACET-VALUE-HERE'
```

## 결과 정렬

기본적으로 결과는 쿼리 프로세스에서 할당된 관련성 점수를 기준으로 정렬되며 점수가 높은 순으로 나열됩니다. 단, 하나 이상의 *sortable* 필드와 정렬 순서(*asc* 또는 *desc*)를 지정하는 OData **orderby** 매개 변수를 포함하여 정렬 순서를 재정의할 수 있습니다.

예를 들어 가장 최근에 수정된 순서로 문서가 나열되도록 결과를 정렬하기 위해 다음 매개 변수 값을 사용할 수 있습니다.

```
SEARCH=*
```

```
$ORDERBY=LAST_MODIFIED DESC
```

## 인덱스 개선

기본 인덱스와 쿼리를 제출하고 결과를 표시할 수 있는 클라이언트를 사용하여 효율적인 검색 솔루션을 구현할 수 있습니다. 그러나 Azure AI 검색은 더 나은 사용자 환경을 제공하기 위해 인덱스를 개선하는 여러 가지 방법을 지원합니다. 이 토픽에서는 검색 솔루션을 확장할 수 있는 몇 가지 방법에 대해 설명합니다.

### 입력 기반 검색

인덱스에 '제안기'를 추가하여 사용자가 관련 결과를 보다 쉽게 찾을 수 있는 두 가지 입력 기반 검색 양식을 사용할 수 있습니다.

- 제안 - 검색 쿼리를 제출하지 않고도 검색 상자에 제안된 결과 목록을 검색하고

표시합니다.

- 자동 완성 - 인덱스 필드의 값에 따라 부분적으로 입력된 검색 용어를 완성합니다.

해당 기능 중 하나 또는 둘 다를 구현하려면 하나 이상의 필드에 대한 제안기를 정의하여 인덱스를 만들거나 업데이트합니다.

제안기를 추가한 후에는 suggestion 및 autocomplete REST API 엔드포인트 또는 .NET DocumentsOperationsExtensions.Suggest 및 DocumentsOperationsExtensions.AutoComplete 메서드를 사용하여 부분 검색 용어를 제출하고 제안된 결과 또는 자동 완성된 용어 목록을 검색하여 사용자 인터페이스에 표시할 수 있습니다.

## 참고

제안기에 대한 자세한 내용은 Azure AI 검색 설명서의 [클라이언트 앱에 자동 완성 및 제안 추가](#)를 참조하세요.

## 사용자 지정 점수 매기기 및 결과 향상

기본적으로 검색 결과는 TF/IDF(Term Frequency/Inverse Document Frequency) 알고리즘을 기반으로 계산되는 관련성 점수를 기준으로 정렬됩니다. 특정 필드에 가중치 값을 적용하는 '점수 매기기 프로필'을 정의하여 이 점수를 계산하는 방법을 사용자 지정할 수 있습니다. 해당 필드에 검색 용어가 있는 경우 문서에 대한 검색 점수가 늘어납니다. 또한 필드 값을 기준으로 결과를 '향상'할 수 있습니다. 예를 들어 최근에 수정한 날짜 또는 파일 크기에 따라 문서에 대한 관련성 점수를 늘릴 수 있습니다.

점수 매기기 프로필을 정의한 후 개별 검색에서 해당 프로필의 용도를 지정하거나, 기본적으로 사용자 지정 점수 매기기 프로필을 사용하도록 인덱스 정의를 수정할 수 있습니다.

## 동의어

여러 가지 방법으로 동일한 항목을 참조할 수 있는 경우가 많습니다. 예를 들어 영국에 대한 정보를 검색하는 사용자는 다음 조건 중 하나를 사용할 수 있습니다.

- 영국
- 영국
- 영국\*

- GB\*

\*엄밀하게 구분하자면 UK와 Great Britain은 서로 다른 엔터티입니다. 하지만 일반적으로 이 둘은 혼용되므로 "United Kingdom"을 검색하는 사람이 "Great Britain"을 참조하는 결과에 관심이 있을 수 있습니다.

필요한 정보를 찾을 수 있도록 관련 용어를 함께 연결하는 '동의어 맵'을 정의할 수 있습니다. 그런 다음 해당 동의어 맵을 인덱스의 개별 필드에 적용하면 사용자가 특정 용어를 검색할 때 용어 또는 해당 동의어를 포함하는 필드가 있는 문서가 결과에 포함됩니다.

# KTds Azure AI 교육 교재 Day 5

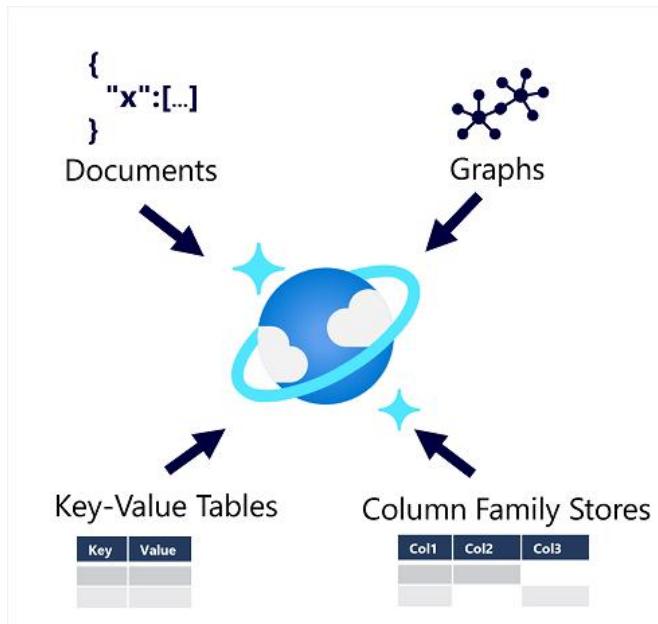
## Cosmos DB 소개

관계형 데이터베이스는 관계형 테이블에 데이터를 저장하지만 경우에 따라 이 모델이 적용하는 구조의 유연성이 부족할 수 있으며 시간을 들여 세부적 튜닝을 구현하지 않으면 성능이 저하되는 경우가 많습니다. NoSQL 데이터베이스로 통칭되는 다른 모델이 있습니다. 이러한 모델은 문서, 그래프, 키-값 저장소, 열 패밀리 저장소와 같은 다른 구조에 데이터를 저장합니다.

Azure Cosmos DB는 NoSQL 데이터를 위한 확장성이 뛰어난 클라우드 데이터베이스 서비스입니다.

이러한 문제를 해결하기 위해 Margie's Travel에서는 문서를 인덱싱하여 검색을 쉽게 할 수 있도록 하는 솔루션을 구현할 수 있습니다. 이 솔루션을 사용하면 에이전트와 고객이 인덱스를 쿼리하여 관련 문서를 찾고 해당 문서에서 정보를 추출할 수 있습니다.

## Azure Cosmos DB 설명



Azure Cosmos DB는 개발자가 여러 종류의 일반적인 데이터 저장소 프로그래밍 의미 체계를 사용하여 Cosmos DB 데이터베이스의 데이터로 작업할 수 있도록 하는 여러 API(애플리케이션 프로그래밍 인터페이스)를 지원합니다. 내부 데이터 구조가 추상화되었기 때문에 개발자는 Cosmos DB를 사용하여 이미 익숙한 API를 통해 데이터를 저장하고 쿼리할 수 있습니다.

## 참고

API는 애플리케이션 프로그래밍 인터페이스입니다. 데이터베이스 관리 시스템(및 기타 소프트웨어 프레임워크)은 데이터에 액세스해야 하는 프로그램을 작성하는 데 개발자가 사용할 수 있는 API 집합을 제공합니다. API는 데이터베이스 관리 시스템마다 다릅니다.

Cosmos DB는 인덱스와 분할을 사용하여 빠른 읽기 및 쓰기 성능을 제공하고 대용량 데이터로 확장할 수 있습니다. 전역적으로 분산된 사용자가 각각 로컬 복제본의 데이터로 작업할 수 있도록 선택한 Azure 지역을 Cosmos DB 계정에 추가한 후 다중 지역 쓰기를 사용하도록 설정할 수 있습니다.

## Cosmos DB를 사용하는 경우

Cosmos DB는 확장성이 뛰어난 데이터베이스 관리 시스템입니다. Cosmos DB는 컨테이너에 파티션 공간을 자동으로 할당하며, 각 파티션의 크기는 최대 10GB까지 늘릴 수 있습니다. 인덱스는 자동으로 생성 및 유지 관리됩니다. 사실상 관리 오버헤드가 없습니다.

Cosmos DB는 Azure의 기본 서비스입니다. Cosmos DB는 Skype, Xbox, Microsoft 365, Azure 등 글로벌 규모의 중요 업무용 애플리케이션을 위해 많은 Microsoft 제품에서 사

용되어 왔습니다. Cosmos DB는 다음과 같은 시나리오에 매우 적합합니다.

- IoT 및 텔레매틱스. 이러한 시스템은 일반적으로 자주 버스트되는 작업을 통해 많은 양의 데이터를 수집합니다. Cosmos DB는 이 정보를 빠르게 받아 저장할 수 있습니다. 그런 다음 Azure Machine Learning, Microsoft Fabric, Power BI와 같은 분석 서비스에서 데이터를 사용할 수 있습니다. 또한 데이터가 데이터베이스에 도착할 때 트리거되는 Azure Functions를 사용하여 실시간으로 데이터를 처리할 수 있습니다.
- 소매 및 마케팅. Microsoft는 Windows 스토어 및 Xbox Live의 일부로 실행되는 자체 전자 상거래 플랫폼에 CosmosDB를 사용합니다. 소매 업계의 경우 카탈로그 데이터 저장에, 주문 처리 파이프라인의 경우 이벤트 소싱에도 사용됩니다.
- 게임. 데이터베이스 계층은 게임 애플리케이션의 중요한 구성 요소입니다. 오늘날의 게임은 모바일/콘솔 클라이언트에서 그래픽 처리를 수행하지만 게임 내 통계, 소셜 미디어 통합 및 고득점 순위표와 같은 사용자 지정되고 개인 설정된 콘텐츠를 제공하기 위해 클라우드에 의존합니다. 매력적인 인게임 환경을 제공하기 위해서는 몇 밀리초 단위의 읽기 및 쓰기 대기 시간이 필요한 경우가 많습니다. 게임 데이터베이스는 속도가 빨라야 하며 신규 게임 출시 및 기능 업데이트 동안 요청 속도의 대량 스파이크를 처리할 수 있어야 합니다.
- 웹 및 모바일 애플리케이션. Azure Cosmos DB는 일반적으로 웹 및 모바일 애플리케이션 내에서 사용되며 소셜 상호 작용을 모델링하여 타사 서비스와 통합하고 풍부한 개인 설정 환경을 빌드하는 데 적합합니다. Cosmos DB SDK는 인기 있는 Xamarin 프레임워크를 사용하여 다양한 iOS 및 Android 애플리케이션을 빌드하는 데 사용할 수 있습니다.

## Azure Cosmos DB API 식별

Azure Cosmos DB는 관계형 및 비관계형 워크로드를 모두 지원하는 모든 크기 또는 규모의 애플리케이션을 위한 Microsoft의 완전 관리형 및 서비스 분산 데이터베이스입니다. 개발자는 PostgreSQL, MongoDB 및 Apache Cassandra를 비롯한 선호하는 오픈 소스 데이터베이스 엔진을 사용하여 애플리케이션을 빠르게 빌드하고 마이그레이션할 수 있습니다. 새 Cosmos DB 인스턴스를 프로비저닝할 때 사용할 데이터베이스 엔진을 선택합니다. 엔진 선택은 저장할 데이터 형식, 기존 애플리케이션을 지원해야 하는 필요성 및 데이터 저장소로 작업할 개발자의 기술을 비롯한 여러 요인에 따라 달라집니다.

## Azure Cosmos DB for NoSQL

Azure Cosmos DB for NoSQL는 문서 데이터 모델 작업을 위한 Microsoft의 네이티브 비 관계형 서비스입니다. 이 서비스는 JSON 문서 형식의 데이터를 관리하며 NoSQL 데이터 스토리지 솔루션임에도 불구하고 SQL 구문을 사용하여 데이터를 사용합니다.

고객 데이터를 포함하는 Azure Cosmos DB 데이터베이스에 대한 SQL 쿼리는 다음과 유사할 수 있습니다.

SQL

```
SELECT *
FROM CUSTOMERS C
WHERE C.ID = "JOE@LITWARE.COM"
```

이 쿼리의 결과는 다음과 같이 하나 이상의 JSON 문서로 구성됩니다.

JSON

```
{
    "ID": "JOE@LITWARE.COM",
    "NAME": "JOE JONES",
    "ADDRESS": {
        "STREET": "1 MAIN ST.",
        "CITY": "SEATTLE"
    }
}
```

## Azure Cosmos DB for MongoDB

MongoDB는 데이터가 BSON(Binary JSON) 형식으로 저장되는 인기 있는 오픈 소스 데이터베이스입니다. Azure Cosmos DB for MongoDB를 사용하면 개발자가 MongoDB 클라이

언트 라이브러리를 사용하여 Azure Cosmos DB의 데이터로 작업하고 코드를 사용할 수 있습니다.

MQL(MongoDB 쿼리 언어)은 개발자가 개체를 사용하여 메서드를 호출하는 컴팩트한 개체 지향 구문을 사용합니다. 예를 들어 다음 쿼리는 **찾기** 메서드를 사용하여 **db** 개체의 **제품** 컬렉션을 쿼리합니다.

JavaScript

```
DB.PRODUCTS.FIND({ID: 123})
```

이 쿼리의 결과는 다음과 유사한 JSON 문서로 구성됩니다.

JSON

```
{
  "ID": 123,
  "NAME": "HAMMER",
  "PRICE": 2.99
}
```

## Azure Cosmos DB for PostgreSQL

Azure Cosmos DB for PostgreSQL은 Azure의 분산형 PostgreSQL 옵션입니다. Azure Cosmos DB for PostgreSQL은 확장성 있는 앱을 빌드하는 데 도움이 되도록 데이터를 자동으로 분할하는 전역으로 분산된 관계형 데이터베이스인 네이티브 PostgreSQL입니다. 다른 데서도 PostgreSQL에서와 동일한 방식으로 단일 노드 서버 그룹에서 앱 빌드를 시작할 수 있습니다. 앱의 확장성 및 성능 요구 사항이 증가함에 따라 테이블을 투명하게 배포하여 여러 노드로 원활하게 확장할 수 있습니다. PostgreSQL은 데이터의 관계형 테이블을 정의하는 RDBMS(관계형 데이터베이스 관리 시스템)입니다. 예를 들어 다음과 같은 제품 테이블을 정의할 수 있습니다.

테이블 확장

ProductID	ProductName
123	망치
162	Screwdriver

그런 다음 이 테이블을 쿼리하여 다음과 같이 SQL을 사용하여 특정 제품의 이름과 가격을 검색할 수 있습니다.

SQL

```
SELECT PRODUCTNAME, PRICE
FROM PRODUCTS
WHERE PRODUCTID = 123;
```

이 쿼리의 결과에는 다음과 같이 제품 123에 대한 행이 포함됩니다.

테이블 확장

ProductName	가격
망치	2.99

### Azure Cosmos DB for Table

Azure Cosmos DB for Table은 Azure Table Storage와 유사한 키-값 테이블의 데이터로 작업하는 데 사용됩니다. Azure Table Storage보다 뛰어난 확장성과 성능을 제공합니다. 예를 들어 다음과 같이 Customers라는 테이블을 정의할 수 있습니다.

PartitionKey	RowKey	속성	Email
1	123	Joe Jones	joe@litware.com
1	124	Samir Nadoy	samir@northwind.com

그런 다음 언어별 SDK 중 하나를 통해 Table API를 사용하여 서비스 엔드포인트를 호출한 후 테이블에서 데이터를 검색할 수 있습니다. 예를 들어 다음 요청은 이전 표에

서 *Samir Nadoy*에 대한 레코드가 포함된 행을 반환합니다.

text

HTTPS://ENDPOINT/CUSTOMERS(PARTITIONKEY='1',ROWKEY='124')

### Azure Cosmos DB for Apache Cassandra

Azure Cosmos DB for Apache Cassandra는 열 패밀리 스토리지 구조를 사용하는 인기 있는 오픈 소스 데이터베이스인 Apache Cassandra와 호환됩니다. 열 패밀리는 관계형 데이터베이스의 테이블과 유사하며 모든 행에 동일한 열을 포함해야 하는 것은 아닙니다.

예를 들어 다음과 같이 **Employees** 테이블을 만들 수 있습니다.

ID	속성	Manager
1	Sue Smith	
2	Ben Chan	Sue Smith

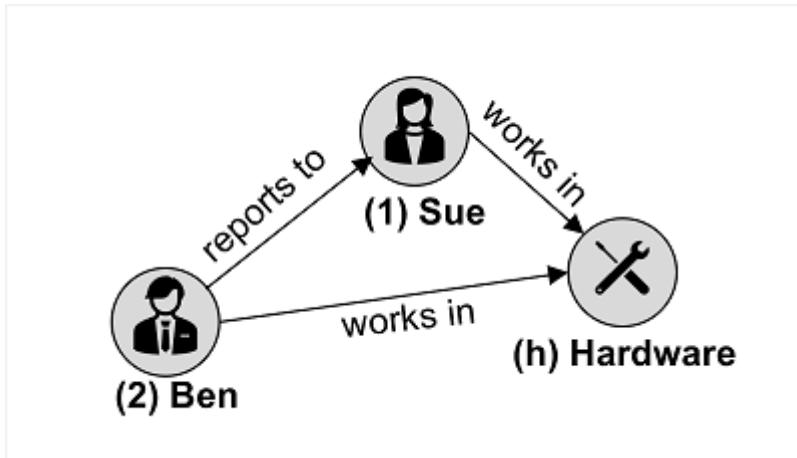
Cassandra는 SQL을 기반으로 하는 구문을 지원하므로 클라이언트 애플리케이션은 다음과 같이 Ben Chan에 대한 레코드를 검색할 수 있습니다.

SQL

SELECT \* FROM EMPLOYEES WHERE ID = 2

### Azure Cosmos DB for Apache Gremlin

Azure Cosmos DB for Apache Gremlin은 그래프 구조의 데이터와 함께 사용되며 엔터티가 연결된 그래프에서 노드를 형성하는 꼭짓점으로 정의됩니다. 노드는 다음과 같이 관계를 나타내는 에지에 의해 연결됩니다.



이미지의 예제에서는 두 종류의 꼭짓점(직원 및 부서)과 이를 연결하는 에지(직원 “Ben”은 직원 “Sue”에 보고하고 두 직원 모두 “하드웨어” 부서에서 근무)를 보여 줍니다.

Gremlin 구문에는 꼭짓점 및 에지에서 작동하는 함수가 포함되어 있어 그래프에서 데이터를 삽입, 업데이트, 삭제 및 쿼리할 수 있습니다. 예를 들어 다음 코드를 사용하여 ID 가 1(Sue)인 직원에게 보고하는 Alice라는 새 직원을 추가할 수 있습니다.

apache

```

G.ADDV('EMPLOYEE').PROPERTY('ID', '3').PROPERTY('FIRSTNAME', 'ALICE')

G.V('3').ADDE('REPORTS TO').TO(G.V('1'))
  
```

다음 쿼리는 ID 순서로 모든 직원 꼭짓점을 반환합니다.

apache

```

G.V().HASLABEL('EMPLOYEE').ORDER().BY('ID')
  
```

# KTds Azure AI 교육 교재 Day 6

## Azure Document Intelligence 소개

양식은 매일 모든 산업의 정보를 전달하는 데 사용됩니다. 여전히 많은 사람들이 양식에서 데이터를 수동으로 추출하여 정보를 교환합니다.



사용자가 양식 데이터를 처리해야 하는 경우 몇 가지 인스턴스를 고려합니다.

- 클레임을 제출할 때
- 온라인 관리 시스템에 새 환자를 등록하는 경우
- 영수증에서 비용 보고서로 데이터를 입력하는 경우
- 작업 보고서를 검토하여 변칙을 검토하는 경우

- 보고서에서 이해 관계자에게 제공할 데이터를 선택하는 경우

AI 서비스가 없으면 사람들은 양식 문서를 수동으로 정렬하여 중요한 정보를 식별한 다음 데이터를 수동으로 다시 입력하여 기록해야 합니다. 일부는 고객과 실시간으로 이러한 작업을 완료해야 할 수도 있습니다.

Azure Document Intelligence 서비스는 지능형 서비스를 사용하여 대규모의 정확도로 데이터를 추출하여 자동화를 위한 구성 요소를 제공합니다. *Azure Document Intelligence*는 양식 문서에서 키-값 쌍 및 테이블 데이터를 추출하는 Vision API입니다.

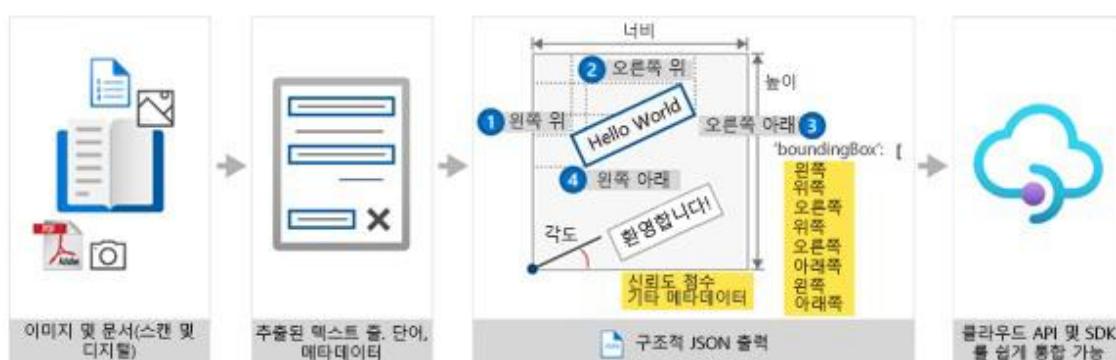
Azure Document Intelligence 서비스의 용도는 다음과 같습니다.

- 프로세스 자동화
- 정보 마이닝
- 산업별 애플리케이션

## Azure Document Intelligence란?

Azure Document Intelligence 는 애플리케이션에 인텔리전스를 빌드하는 데 사용할 수 있는 REST API 및 클라이언트 라이브러리 SDK를 사용하는 여러 Azure AI 서비스, 클라우드 기반 AI(인공 지능) 서비스 중 하나입니다.

Azure Document Intelligence는 OCR(광학 인식) 기능과 딥 러닝 모델을 사용하여 문서에서 텍스트, 키-값 쌍, 선택 표시 및 테이블을 추출합니다.



OCR은 이미지에서 검색된 개체 주위에 경계 상자를 만들어 문서 구조를 캡처합니다. 경계 상자의 위치는 페이지의 나머지 부분과 관련하여 좌표로 기록됩니다. Azure

Document Intelligence 서비스는 경계 상자 데이터 및 기타 정보를 원래 파일의 관계와 함께 구조화된 형식으로 반환합니다.



처음부터 높은 정확도 모델을 빌드하려면 사람들은 딥 러닝 모델을 빌드하고, 많은 양의 컴퓨팅 리소스를 사용하고, 긴 모델 학습 시간에 직면해야 합니다. 이러한 요인으로 인해 프로젝트를 불가능하게 만들 수 있습니다. Azure Document Intelligence는 수천 개의 양식 예제에서 학습된 기본 모델을 제공합니다. 기본 모델을 사용하면 모델 학습이 거의 또는 전혀 없이 양식에서 정확도 높은 데이터 추출을 수행할 수 있습니다.

#### Azure Document Intelligence 서비스 구성 요소

Azure Document Intelligence는 다음 서비스로 구성됩니다.

- 문서 분석 모델: JPEG, PNG, PDF 및 TIFF 파일의 입력을 받아 경계 상자, 텍스트 콘텐츠, 표, 선택 표시(확인란 또는 라디오 단추라고도 함) 및 문서 구조의 텍스트 위치가 있는 JSON 파일을 반환합니다.
- 미리 빌드된 모델: 문서 이미지에서 정보를 검색하고 추출한 데이터를 구조화된 JSON 출력으로 반환합니다. Azure Document Intelligence는 현재 다음을 비롯한 여러 양식에 대해 미리 빌드된 모델을 지원합니다.
  - W-2 양식

- 송장
  - 영수증
  - ID 문서
  - 명함
- 사용자 지정 모델: 사용자 지정 모델은 비즈니스와 관련된 양식에서 데이터를 추출합니다. 사용자 지정 모델은 Azure Document Intelligence Studio 통해 학습할 수 있습니다.

## Azure Document Intelligence 시작

Azure Document Intelligence 서비스를 사용하여 프로젝트를 시작하려면 데이터 추출을 위해 Azure 리소스 및 양식 파일 선택 항목이 필요합니다.

### 리소스 구독

다음을 통해 Azure Document Intelligence 서비스에 액세스할 수 있습니다.

- Azure AI 서비스 리소스: 다중 서비스 구독 키(여러 Azure AI 서비스에서 사용)

또는

- Azure Document Intelligence 리소스: 단일 서비스 구독 키(특정 Azure AI 서비스에서만 사용)

### 참고

단일 엔드포인트/키로 여러 Azure AI 서비스에 액세스하려는 경우 Azure AI Services 리소스를 만듭니다. Azure Document Intelligence 액세스 전용으로 Azure Document Intelligence 리소스를 만듭니다. Microsoft Entra 인증을 사용하려는 경우 단일 서비스 리소스가 필요합니다.

Azure Portal 또는 Azure CLI(명령줄 인터페이스)를 사용하여 서비스를 구독할 수 있습니다. 여기에서 CLI 명령에 대해 자세히 알아볼 수 [있습니다](#).

### Azure Document Intelligence 파일 입력 요구 사항 이해

Azure Document Intelligence는 다음 요구 사항을 충족하는 입력 문서에서 작동합니다.

- 형식은 JPG, PNG, BMP, PDF(텍스트 또는 스캔) 또는 TIFF여야 합니다.
- 파일 크기는 유료(S0) 계층의 경우 500MB 미만, 무료(F0) 계층의 경우 4MB 미만이어야 합니다.
- 이미지 차원은 50 x 50 픽셀에서 10,000 x 10,000 픽셀 사이여야 합니다.
- 학습 데이터 집합의 총 크기는 500페이지 이하여야 합니다.

### 사용할 Azure Document Intelligence의 구성 요소 결정

파일을 수집한 후 수행해야 할 작업을 결정합니다.

#### 테이블 확장

사용 사례	사용할 권장 기능
OCR 기능을 사용하여 문서 분석 캡처	<a href="#">레이아웃 모델</a> , <a href="#">읽기 모델</a> 또는 <a href="#">일반 문서 모델</a> 을 사용합니다.
W-2s, 청구서, 영수증, ID 문서, 건강 보험, 예방 접종 및 명함에서 데이터를 추출하는 애플리케이션 만들기	<a href="#">미리 빌드된 모델</a> 을 사용합니다. 이러한 모델은 학습할 필요가 없습니다. Azure Document Intelligence 서비스는 문서를 분석하고 JSON 출력을 반환합니다.
산업별 양식에서 데이터를 추출하는 애플리케이션 만들기	사용자 지정 모델을 만듭니다. 이 모델은 샘플 문서에 대해 학습해야 합니다. 사용자 지정 모델을 학습한 후 새 문서를 분석하고 JSON 출력을 반환할 수 있습니다.

### 사용자 지정 모델 학습

Azure의 Azure Document Intelligence 서비스는 감독된 기계 학습을 지원합니다. 사용자 지정 모델을 학습시키고 양식 문서 레이블이 지정된 필드가 포함된 JSON 문서를 복합 모델을 만들 수 있습니다.



사용자 지정 모델을 학습하려면 다음을 수행합니다.

1. 레이아웃 및 레이블 필드 정보를 포함하는 JSON 파일과 함께 Azure Blob 컨테이너에 샘플 양식을 저장합니다.
  - Azure Document Intelligence의 Analyze 문서 함수를 사용하여 각 샘플 양식에 대한 ocr.json 파일을 생성할 수 있습니다. 또한 추출할 필드를 설명하는 단일 fields.json 파일과 해당 양식의 해당 위치에 필드를 매핑하는 각 샘플 양식에 대한 labels.json 파일이 필요합니다.
2. 컨테이너에 대한 SAS(공유 액세스 보안) URL을 생성합니다.
3. Build 모델 REST API 함수(또는 동등한 SDK 메서드)를 사용합니다.
4. Get 모델 REST API 함수(또는 동등한 SDK 메서드)를 사용하여 학습된 모델 ID가 됩니다.
5. Azure Document Intelligence Studio를 사용하여 레이블을 지정하고 학습합니다.  
사용자 지정 템플릿 모델 또는 사용자 지정 신경망 모델 사용자 지정 양식에 대한 두 가지 기본 모델이 있습니다.
  - 사용자 지정 템플릿 모델은 문서에서 레이블이 지정된 키-값 쌍, 선택 표시, 테이블, 영역 및 서명을 정확하게 추출할 있습니다. 교육은 몇 분 밖에 걸리지 않으며 100개 이상의 언어가 지원됩니다.

- 사용자 지정 신경망 모델 레이아웃 및 언어 기능을 결합하여 문서에서 레이블이 지정된 필드를 정확하게 추출하는 심층 학습된 모델입니다. 이 모델은 반구조적 또는 구조화되지 않은 문서에 가장 적합합니다.

## Azure Document Intelligence 모델 사용

### API 사용

사용자 지정 모델을 사용하여 양식 데이터를 추출하려면 모델 ID(모델 학습 중에 생성됨)를 제공하면서 지원되는 SDK 또는 REST API의 문서 함수를 분석합니다. 이 함수는 양식 분석을 시작합니다. 그런 다음 결과를 요청하여 분석을 가져올 수 있습니다.

모델을 호출하는 예제 코드:

#### C#

```
STRING ENDPOINT = "<ENDPOINT>";

STRING APIKEY = "<APIKEY>";

AZUREKEYCREDENTIAL CREDENTIAL = NEW AZUREKEYCREDENTIAL(APIKEY);

DOCUMENTANALYSISCLIENT CLIENT = NEW DOCUMENTANALYSISCLIENT(NEW URI(ENDPOINT),
CREDENTIAL);

STRING MODELID = "<MODELID>";

URI FILEURI = NEW URI("<FILEURI>");

ANALYZEDOCUMENTOPERATION OPERATION = AWAIT
CLIENT.ANALYZEDOCUMENTFROMURIASYNC(WAITUNTIL.COMPLETED, MODELID, FILEURI);

ANALYZERESULT RESULT = OPERATION.VALUE;
```

파이썬

PYTHON복사

```
ENDPOINT = "YOUR_DOC_INTELLIGENCE_ENDPOINT"
```

```
KEY = "YOUR_DOC_INTELLIGENCE_KEY"
```

```
MODEL_ID = "YOUR_CUSTOM_BUILT_MODEL_ID"
```

```
FORMURL = "YOUR_DOCUMENT"
```

```
DOCUMENT_ANALYSIS_CLIENT = DOCUMENTANALYSISCLIENT(
```

```
    ENDPOINT=ENDPOINT, CREDENTIAL=AZUREKEYCREDENTIAL(KEY)
```

```
)
```

```
# MAKE SURE YOUR DOCUMENT'S TYPE IS INCLUDED IN THE LIST OF DOCUMENT TYPES THE CUSTOM  
MODEL CAN ANALYZE
```

```
TASK = DOCUMENT_ANALYSIS_CLIENT.BEGIN_ANALYZE_DOCUMENT_FROM_URL(MODEL_ID, FORMURL)
```

```
RESULT = TASK.RESULT()
```

성공적인 JSON 응답에는 추출된 콘텐츠와 문서 콘텐츠에 대한 정보가 포함된 페이지 배열이 포함된 **analyzeResult** 포함됩니다.

문서 **JSON** 응답을 분석하는 예제:

JSON

```
{
```

```
    "STATUS": "SUCCEEDED",
```

```
    "CREATEDDATETIME": "2023-10-18T23:39:50Z",
```

```
    "LASTUPDATEDDATETIME": "2023-10-18T23:39:54Z",
```

"ANALYZERESULT": {  
    "APIVERSION": "2022-08-31",  
    "MODELID": "DOCINTELMODEL",  
    "STRINGINDEXTYPE": "UTF16CODEUNIT",  
    "CONTENT": "PURCHASE ORDER\nHERO LIMITED\nCOMPANY PHONE: 555-348-6512  
WEBSITE: WWW.HEROLIMITED.COM EMAIL: ACCOUNTS@HEROLIMITED.COM  
PURCHASE ORDER\nDATED AS: 12/20/2020 PURCHASE ORDER #: 948284  
SHIPPED TO VENDOR NAME:  
BALOZI KHAMISI COMPANY NAME: HIGGLY WIGGLY BOOKS ADDRESS: 938 NE BURNER ROAD  
BOULDER CITY, CO 80284 PHONE: 938-294-2949  
SHIPPED FROM NAME: KIDANE TSEHAYE  
COMPANY NAME: JUPITER BOOK SUPPLY ADDRESS: 383 N KINNICK ROAD SEATTLE, WA  
38383  
PHONE: 932-299-0292  
DETAILS  
QUANTITY  
UNIT  
PRICE  
TOTAL  
BINDINGS  
1.00  
20.00  
COVERS  
SMALL  
1.00  
20.00  
FEATHER BOOKMARK  
20  
5.00  
100.00  
COPPER SWIRL  
MARKER  
20  
5.00  
100.00  
SUBTOTAL  
\$140.00  
TAX  
\$4.00  
TOTAL  
\$144.00  
KIDANE TSEHAYE  
MANAGER  
KIDANE TSEHAYE  
ADDITIONAL NOTES: DO NOT JOSTLE BOX.  
UNPACK CAREFULLY. ENJOY. JUPITER BOOK SUPPLY WILL REFUND YOU 50% PER BOOK IF RETURNED  
WITHIN 60 DAYS OF READING AND OFFER YOU 25% OFF YOUR NEXT TOTAL PURCHASE.",

"PAGES": [  
    {  
        "PAGENUMBER": 1,  
        "ANGLE": 0,  
        "WIDTH": 1159,  
        "HEIGHT": 1486,  
        "UNIT": "PIXEL",  
        "WORDS": [  
            {

        "CONTENT": "PURCHASE",

        "POLYGON": [  
            페이지 37 / 99

89,

90,

174,

91,

174,

112,

88,

112

],

"CONFIDENCE": 0.996,

"SPAN": {

"OFFSET": 0,

"LENGTH": 8

}

},

{

"CONTENT": "ORDER",

"POLYGON": [

178,

91,

237,

91,

236,

113,

```
    178,  
    112  
],  
    "CONFIDENCE": 0.997,  
    "SPAN": {  
        "OFFSET": 9,  
        "LENGTH": 5  
    }  
},  
...  
}
```

## 신뢰도 점수 이해

**analyzeResult** 신뢰도 값이 낮으면 입력 문서의 품질을 향상시킵니다.

신뢰도 값이 낮은 경우 분석하는 양식이 학습 집합의 양식과 유사한 모양인지 확인하려고 합니다. 양식 모양이 다른 경우 각 모델이 하나의 양식 형식에 초점을 맞춘 두 개 이상의 모델을 학습하는 것이 좋습니다.

사용 사례에 따라 위험 수준이 낮은 애플리케이션에 대해 신뢰도 점수가 80% 이상일 수 있습니다. 의료 기록 또는 청구 명세서를 읽는 것과 같은 더 민감한 경우 100% 점수가 권장됩니다.

## Azure Document Intelligence Studio 사용

SDK 및 REST API 외에도 Azure Document Intelligence 서비스는 Azure Document Intelligence 서비스의 기능을 시각적으로 탐색, 이해 및 통합하기 위한 온라인 도구인 Azure Document Intelligence Studio라는 사용자 인터페이스를 통해 액세스할 수 있습니다. Studio를 사용하여 양식 레이아웃을 분석하고, 미리 빌드된 모델에서 데이터를 추출

하고, 사용자 지정 모델을 학습할 수 있습니다.

Azure AI | Document Intelligence Studio

ⓘ Azure Form Recognizer is now Azure AI Document Intelligence. [Learn more](#) about the latest updates to the service and the Studio experience.

Form Recognizer Studio

### Get started with Document Intelligence Studio

Extract text, key-value pairs, tables, and structures from forms and documents using common layouts and prebuilt models, or create your own custom models. [Learn more](#)

#### Document analysis

Extract text, tables, structure, key-value pairs, and named entities from documents.



**Read**

Extract printed and handwritten text along with barcodes, formulas and font styles from images and documents.

[Try it out](#)



**Layout**

Extract tables, check boxes, and text from forms and documents.

[Try it out](#)



**General documents**

Extract key value pairs and structure like tables and selection marks from any form or document.

[Try it out](#)

#### Prebuilt models

Extract data from unique document types using the following prebuilt models.



**Invoices**

Extract invoice ID, customer details, vendor



**Receipts**

Extract time and date of the transaction,



**Business cards**

Extract person name, job title, address,



**Identity documents**

Extract name, expiration date, machine

Azure Document Intelligence Studio는 현재 다음 프로젝트를 지원합니다.

- 문서 분석 모델
  - 읽기: 문서 및 이미지에서 인쇄 및 필기 텍스트 줄, 단어, 위치 및 검색된 언어를 추출합니다.
  - 레이아웃: 문서(PDF 및 TIFF) 및 이미지(JPG, PNG 및 BMP)에서 텍스트, 표, 선택 표시 및 구조 정보를 추출합니다.
  - 일반 문서: 문서에서 키-값 쌍, 선택 표시 및 엔터티를 추출합니다.
- 미리 빌드된 모델
- 사용자 지정 모델

#### 문서 분석 모델 프로젝트 빌드

문서 분석 모델을 사용하여 텍스트, 테이블, 구조체, 키-값 쌍 및 명명된 엔터티를 추출 하려면 다음을 수행합니다.

- Azure Document Intelligence 또는 Azure AI Services 리소스 만들기
- 문서 분석 모델 범주에서 "읽기", "레이아웃" 또는 "일반 문서" 중 하나를 선택합니다.
- 문서를 분석합니다. Azure Document Intelligence 또는 Azure AI 서비스 엔드포인트 및 키가 필요합니다.

### 미리 빌드된 모델 프로젝트 빌드

미리 빌드된 모델을 사용하여 공통 양식에서 데이터를 추출하려면 다음을 수행합니다.

- Azure Document Intelligence 또는 Azure AI Services 리소스 만들기
- W-2, 송장, 영수증, ID 문서, 건강 보험, 예방 접종 및 명함을 포함하여 "미리 빌드된 모델" 중 하나를 선택합니다.
- 문서를 분석합니다. Azure Document Intelligence 또는 Azure AI 서비스 엔드포인트 및 키가 필요합니다.

### 사용자 지정 모델 프로젝트 빌드

사용자 지정 모델을 학습하고 테스트하는 전체 프로세스에 Azure Document Intelligence Studio의 사용자 지정 서비스를 사용할 수 있습니다.

Azure Document Intelligence Studio를 사용하여 사용자 지정 모델을 빌드하는 경우 학습에 필요한 **ocr.json** 파일, **labels.json** 파일 및 **fields.json** 파일이 자동으로 만들어지고 스토리지 계정에 저장됩니다.

사용자 지정 모델을 학습시키고 이를 사용하여 사용자 지정 모델을 사용하여 데이터를 추출하려면 다음을 수행합니다.

- Azure Document Intelligence 또는 Azure AI Services 리소스 만들기
- 학습을 위해 5-6개 이상의 샘플 양식을 수집하고 스토리지 계정 컨테이너에 업로드합니다.

- CORS(도메인 간 리소스 공유)를 구성합니다. CORS를 사용하면 Azure Document Intelligence Studio에서 레이블이 지정된 파일을 스토리지 컨테이너에 저장할 수 있습니다.
- Azure Document Intelligence Studio에서 사용자 지정 모델 프로젝트를 만듭니다. 스토리지 컨테이너와 Azure Document Intelligence 또는 Azure AI Service 리소스를 프로젝트에 연결하는構성을 제공해야 합니다.
- Azure Document Intelligence Studio를 사용하여 텍스트에 레이블을 적용합니다.
- 모델을 학습시킵니다. 모델이 학습되면 태그에 대한 모델 ID 및 평균 정확도를 받게 됩니다.
- 학습에 사용되지 않은 새 양식을 분석하여 모델을 테스트합니다.

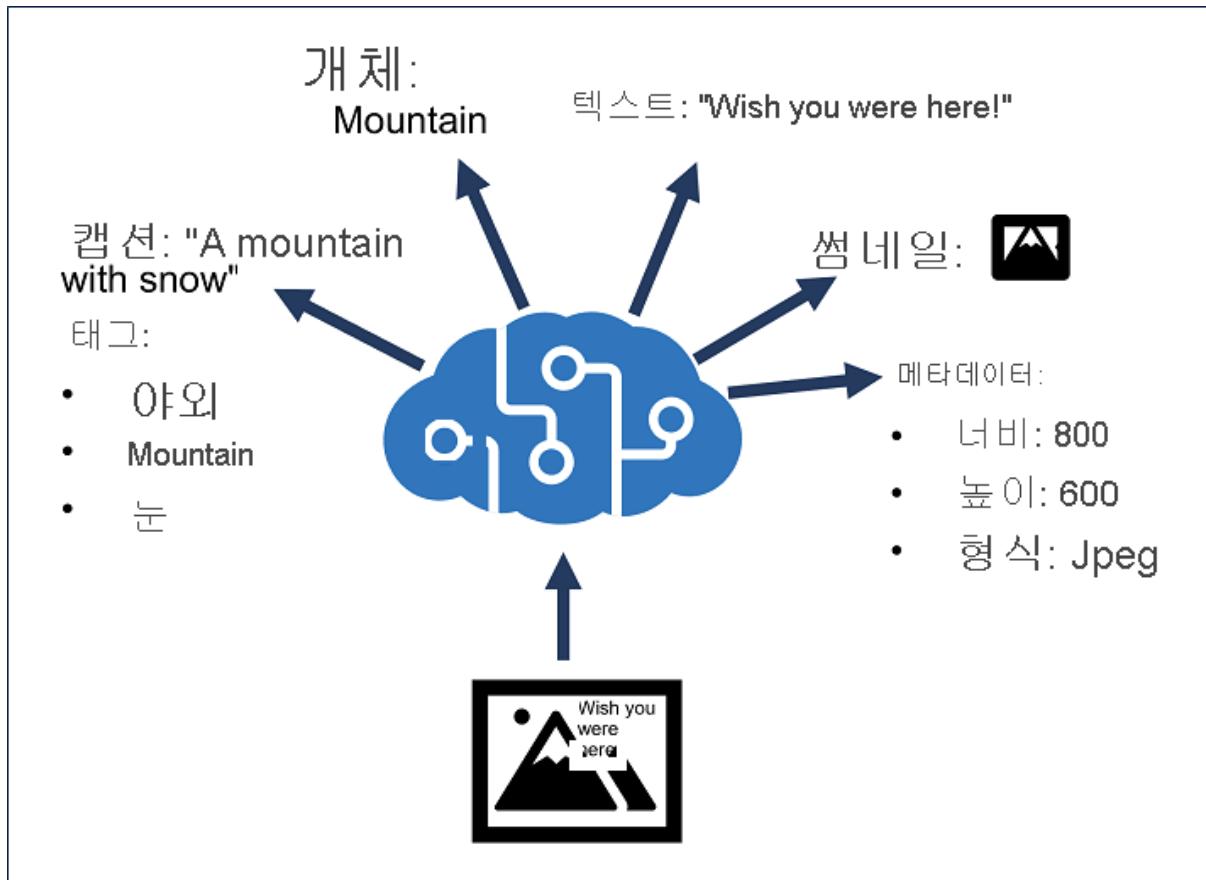
# KTds Azure AI 교육 교재 Day 6

## Azure Computer Vision 소개

Computer Vision은 소프트웨어가 이미지 또는 비디오 피드에서 시각적 입력을 해석하는 AI(인공 지능) 분기입니다. Microsoft Azure에서 **Azure AI Vision** 서비스를 사용하여 다음을 비롯한 여러 컴퓨터 비전 시나리오를 구현할 수 있습니다.

- 이미지 분석
- OCR(광학 문자 인식)
- 얼굴 감지 및 분석
- 동영상 분석

이 모듈에서는 *이미지 분석*에 중점을 두고 Azure AI Vision 서비스를 사용하여 이미지에서 인사이트를 분석하고 추출하고 유추하는 애플리케이션을 빌드하는 방법을 살펴봅니다.

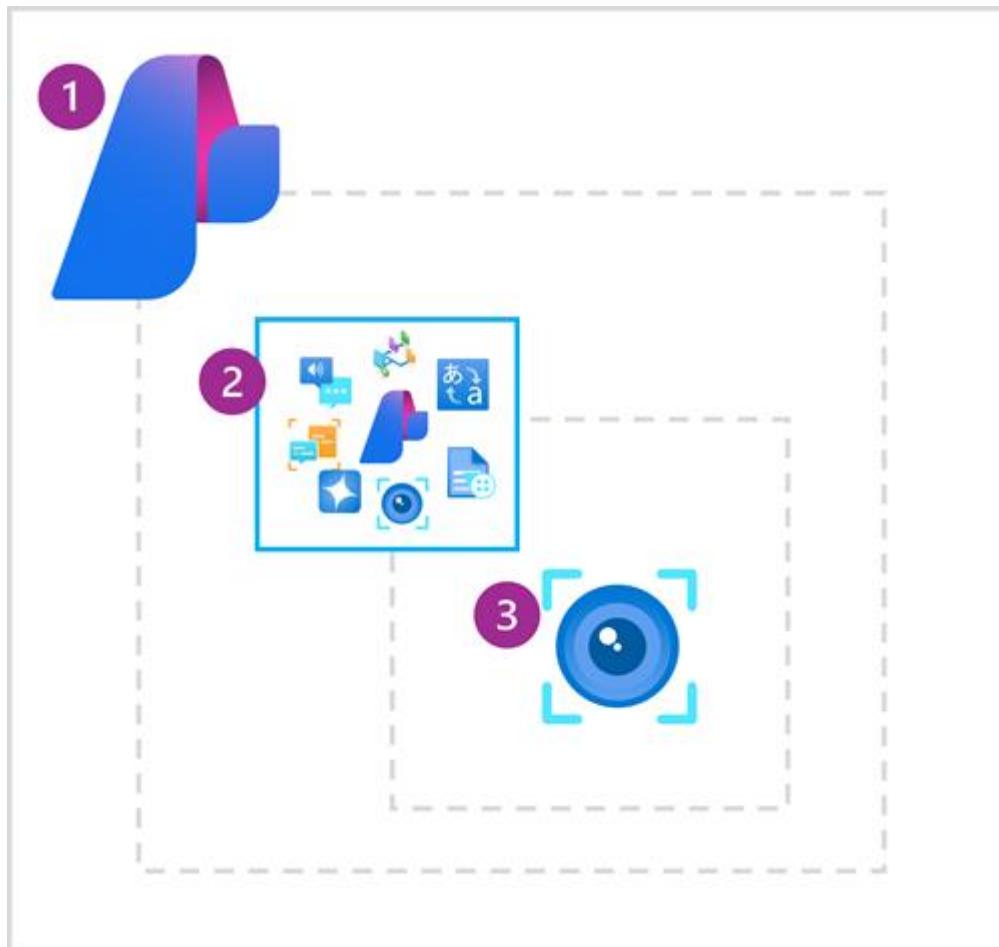


이 개념 다이어그램에 표시된 것처럼 Azure AI Vision 서비스는 이미지를 분석하고 다음을 분석하는 데 사용할 수 있는 서비스를 제공합니다.

- 내용에 따라 이미지에 대한 캡션을 생성합니다.
- 이미지와 연결할 적절한 태그를 제안합니다.
- 이미지에서 일반적인 개체를 검색하고 찾습니다.
- 이미지에서 사람을 검색하고 찾습니다.

## Azure AI Vision 리소스 설정

Azure AI Vision 이미지 분석 서비스를 사용하려면 Azure 구독에서 Azure AI Vision 리소스를 프로비전해야 합니다. 여러 프로비저닝 옵션 중에서 선택할 수 있습니다.



1. **Azure AI Foundry** 프로젝트 및 연결된 허브를 만듭니다. 기본적으로 Azure AI Foundry 허브에는 **Azure AI Vision**을 포함하는 Azure AI 서비스 다중 서비스 리소스가 포함됩니다. Azure AI Foundry 프로젝트는 생성 AI, 에이전트 및 미리 빌드된 Azure AI 서비스를 결합하거나 소프트웨어 엔지니어 및 서비스 운영자 팀이 공동으로 개발하는 Azure의 AI 솔루션 개발에 권장됩니다.
2. Azure AI Foundry 허브에서 모든 기능이 필요하지 않은 경우 Azure 구독에서 **Azure AI** 서비스 다중 서비스 리소스를 만들 수 있습니다. 그런 다음 이 리소스를 사용하여 단일 엔드포인트 및 키를 통해 Azure AI Vision 서비스 및 기타 AI 서비스에 액세스할 수 있습니다.
3. Azure AI Vision 기능만 사용해야 하거나 서비스를 실험하는 경우 Azure 구독에서 독립 실행형 **Computer Vision** 리소스를 만들 수 있습니다. 이 방법의 한 가지 이점은 독립 실행형 서비스가 무료로 서비스를 탐색하는 데 사용할 수 있는 무료 계층을 제공한다는 것입니다.

## 리소스에 연결

리소스를 배포한 후에 [는 Azure AI Vision REST API](#) 또는 언어별 SDK(예: [Python SDK](#) 또는 [Microsoft .NET SDK](#))를 사용하여 클라이언트 애플리케이션에서 연결할 수 있습니다.

모든 Azure AI Vision 리소스는 클라이언트 애플리케이션이 연결해야 하는 [엔드포인트](#)를 제공합니다. Azure Portal에서 리소스에 대한 엔드포인트를 찾거나 Azure AI Foundry 프로젝트에서 작업하는 경우 Azure AI Foundry 포털에서 찾을 수 있습니다. 엔드포인트는 URL 형식이며 일반적으로 다음과 같습니다.

### 복사

[https://<resource\\_name>.cognitiveservices.azure.com/](https://<resource_name>.cognitiveservices.azure.com/)

엔드포인트에 연결하려면 클라이언트 애플리케이션을 인증해야 합니다. 인증 옵션에는 다음이 포함됩니다.

- **키 기반 인증:** 클라이언트 애플리케이션은 인증 키를 전달하여 인증됩니다(포털에서 찾아서 관리할 수 있습니다).
- **Microsoft Entra ID 인증:** 클라이언트 애플리케이션은 Azure에서 Azure AI Vision 리소스에 액세스할 수 있는 권한이 있는 자격 증명에 대해 Microsoft Entra ID 토큰을 사용하여 인증됩니다.

애플리케이션을 개발하고 테스트할 때는 사용자 고유의 Azure 자격 증명을 기반으로 키 기반 인증 또는 Microsoft Entra ID 인증을 사용하는 것이 일반적입니다. 프로덕션 환경에서는 Azure 애플리케이션의 관리 ID를 기반으로 하는 Microsoft Entra ID 인증을 사용하거나 Azure Key Vault를 사용하여 권한 부여 키를 안전하게 저장하는 것이 좋습니다.

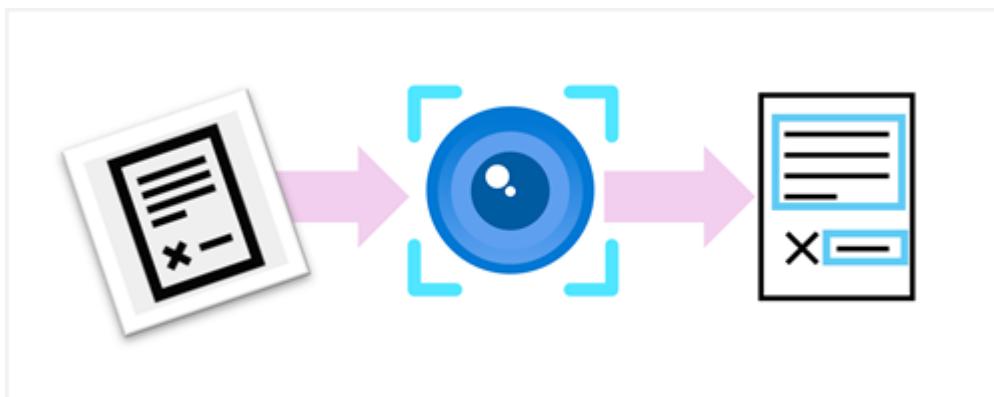
### 참고

Azure AI Foundry 프로젝트에서 Azure AI 서비스 리소스를 사용하는 경우 Azure AI Foundry SDK를 사용하여 Microsoft Entra ID 인증을 사용하여 프로젝트에 연결한 다음, 프로젝트에서 권한 부여 키를 포함한 Azure AI 서비스 리소스에 대한 연결 정보를 검색할 수 있습니다.

## OCR

우리는 데이터가 점점 더 이미지로 캡처되는 디지털 세계에 살고 있습니다. 이러한 이미지에는 처리, 인덱싱 및 기타 작업을 위해 이미지의 픽셀화된 형식에서 추출할 수 있어야 하는 텍스트가 포함되어 있는 경우가 많습니다. 일상적인 예는 다음과 같습니다.

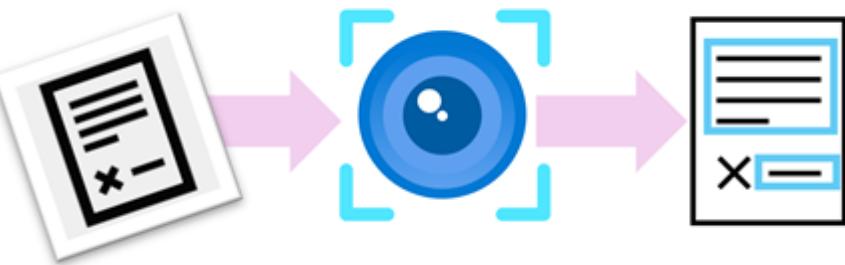
- 새 비즈니스 동료를 만나고 명함 사진을 찍어 연락처 세부 정보를 디지털로 저장합니다.
- 정부 또는 상용 서비스에 대한 응용 프로그램에 포함할 문서 또는 ID 카드를 스캔합니다.
- 디지털 전자 필기장의 메뉴 또는 레시피 사진을 찍어 저장합니다.
- 가로 표지판이나 상점 앞면을 촬영하여 포함된 텍스트를 번역 앱에 제출할 수 있습니다.
- 휴대폰 카메라를 사용하여 필기 노트 디지털화



우리는 데이터가 점점 더 이미지로 캡처되는 디지털 세계에 살고 있습니다. 이러한 이미지에는 처리, 인덱싱 및 기타 작업을 위해 이미지의 픽셀화된 형식에서 추출할 수 있어야 하는 텍스트가 포함되어 있는 경우가 많습니다. 일상적인 예는 다음과 같습니다.

- 새 비즈니스 동료를 만나고 명함 사진을 찍어 연락처 세부 정보를 디지털로 저장합니다.
- 정부 또는 상용 서비스에 대한 응용 프로그램에 포함할 문서 또는 ID 카드를 스캔합니다.

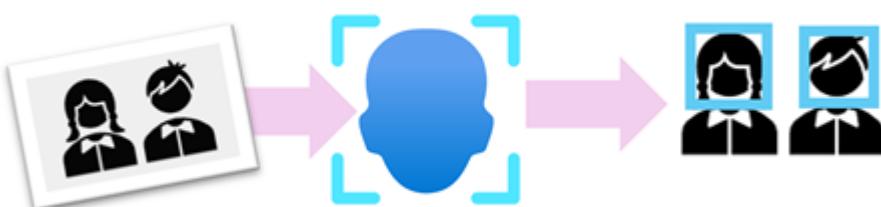
- 디지털 전자 필기장의 메뉴 또는 레시피 사진을 찍어 저장합니다.
- 가로 표지판이나 상점 앞면을 촬영하여 포함된 텍스트를 번역 앱에 제출할 수 있습니다.
- 휴대폰 카메라를 사용하여 필기 노트 디지털화



## Face API

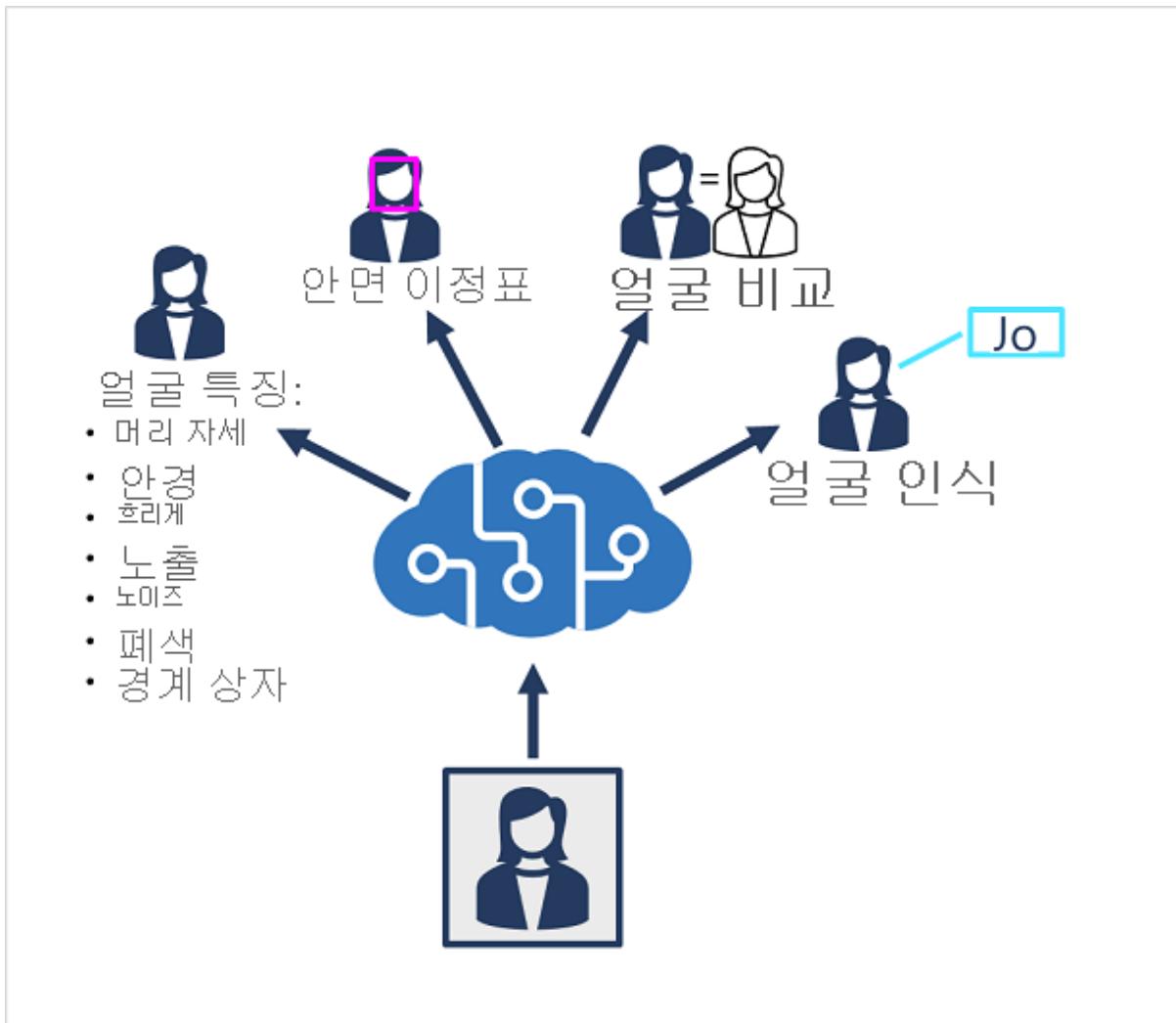
얼굴 감지, 분석 및 인식은 모두 AI 시스템의 일반적인 컴퓨터 비전 과제입니다. 사람이 있을 때를 감지하거나, 사람의 얼굴 특징을 분석하거나, 얼굴을 기반으로 개인을 인식하는 기능은 AI 시스템이 인간과 유사한 행동을 보이고 사용자와 공감할 수 있는 핵심적인 방법입니다.

이 모듈에서는 Azure AI Vision *Face API*를 사용하여 이미지에서 얼굴을 분석하는 솔루션을 빌드하는 방법을 살펴봅니다.



## 얼굴 감지, 분석 또는 인식 솔루션 계획

Face 서비스는 포괄적인 얼굴 감지, 분석 및 인식 기능을 제공합니다.



Face 서비스는 다음에 사용할 수 있는 기능을 제공합니다.

1. **얼굴 감지** - 감지된 각 얼굴에 대한 결과에는 얼굴을 식별하는 ID와 이미지의 위치를 나타내는 경계 상자 좌표가 포함됩니다.
2. **얼굴 특성 분석** - 다음을 포함하여 다양한 얼굴 특성을 반환할 수 있습니다.
  - 머리 포즈(3D 공간의 피치, 롤 및 요 방향 각도)
  - 안경(안경 없음, 독서용 안경, 선글라스 또는 수영 고글 없음)
  - 마스크(얼굴 마스크의 존재)

- 흐림(낮음, 중간 또는 높음)
  - 노출 (부족한 노출, 양호한 노출, 또는 과다 노출)
  - 노이즈(이미지의 시각적 노이즈)
  - 차폐(얼굴을 가리는 개체)
  - 액세서리(안경, 모자, 마스크)
  - QualityForRecognition(낮음, 중간 또는 높음)
3. 얼굴 랜드마크 위치 - 얼굴 특징과 관련된 주요 랜드마크 좌표(예: 눈 모서리, 동공, 코 끝 등)
  4. 얼굴 비교 - 여러 이미지에서 얼굴을 비교하여 유사성(비슷한 얼굴 특징을 가진 개인 찾기) 및 확인(한 이미지의 얼굴이 다른 이미지의 얼굴과 동일한지 확인)할 수 있습니다.
  5. 얼굴 인식 - 특정 개인에 속한 얼굴 컬렉션을 사용하여 모델을 학습시키고 모델을 사용하여 새 이미지에서 해당 사용자를 식별할 수 있습니다.
  6. 얼굴 생체 인식 - 생체 인식은 입력 비디오가 실제 스트림인지 가짜인지 확인하여, 악의적인 사람들이 얼굴 인식 시스템을 스푸핑하는 것을 방지하는 데 사용할 수 있습니다.

## 얼굴 감지 및 인식 모델

Azure AI Vision Face API는 미리 학습된 얼굴 감지 및 인식 모델을 기반으로 합니다. 이러한 모델의 여러 버전을 사용할 수 있으며, 각각에는 특정 강점과 기능이 있습니다. 예를 들어 최신 모델은 작은 이미지로 작업할 때 더 높은 정확도를 보여 줍니다. 그러나 얼굴 분석 기능의 동일한 폭을 제공하지 않을 수 있습니다. 애플리케이션에서 서비스를 사용하는 경우 요구 사항에 따라 사용하려는 모델을 선택해야 합니다.

## 얼굴 확인 및 식별

얼굴을 감지하고 분석하는 것 외에도 Azure AI Vision Face 서비스를 사용하여 얼굴을 비교하고 인식할 수 있습니다.

### 중요

얼굴 인식, 비교 및 확인을 사용하려면 [제한된 액세스 정책을](#) 통한 승인이 필요합니다.

## 얼굴 확인

Face 서비스에서 얼굴을 감지하면 고유한 ID가 할당되고 서비스 리소스에 24시간 동안 유지됩니다. ID는 얼굴 특징 외에 개인의 신원을 표시하지 않는 GUID입니다.

감지된 얼굴 ID가 캐시되는 동안 후속 이미지를 사용하여 새 얼굴을 캐시된 ID와 비교하고 유사한지(즉, 유사한 얼굴 특징을 공유) 확인하거나 동일한 사람이 두 이미지에 표시되는지 확인할 수 있습니다.



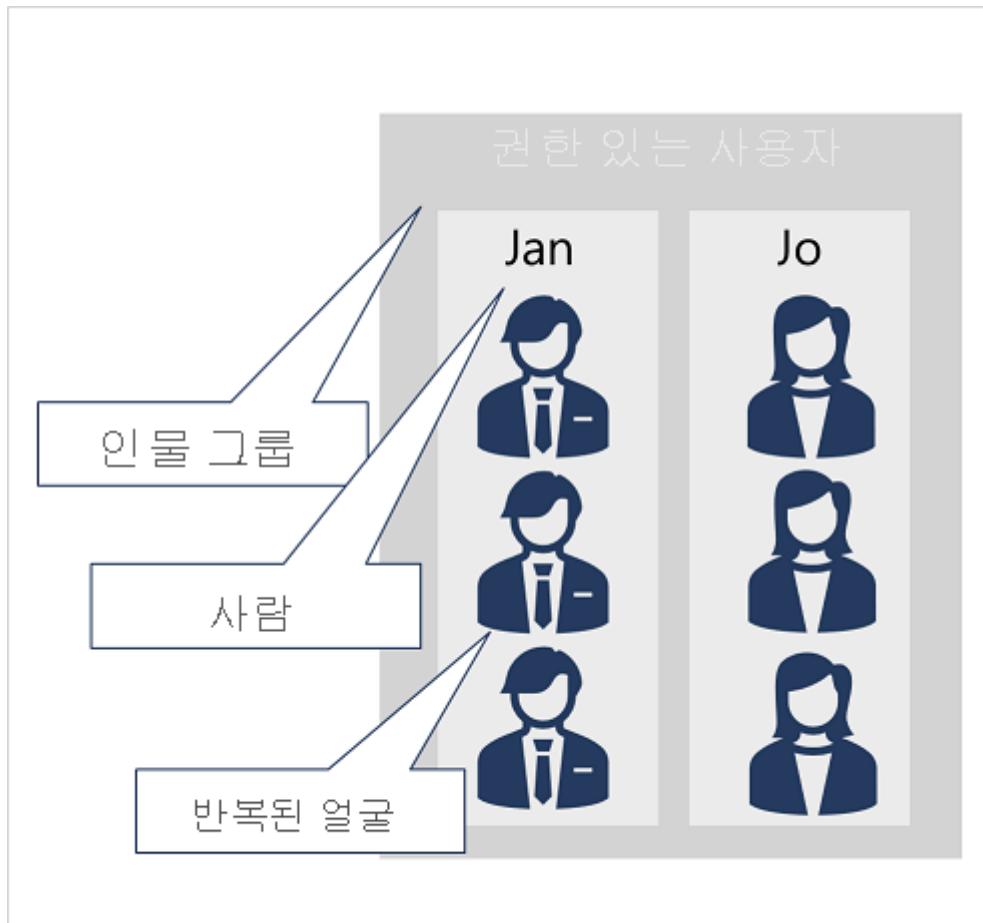
익명으로 얼굴을 비교하는 이 기능은 사람의 실제 신원을 알 필요 없이 동일한 사람이 두 번 있는지 확인하는 것이 중요한 시스템에서 유용할 수 있습니다. 예를 들어, 사람들이 들어오고 나갈 때 사람들의 사진을 찍어 보안 공간에 들어온 모든 사람이 나가는 것을 확인합니다.

## 얼굴 식별

개인을 긍정적으로 식별해야 하는 시나리오의 경우 얼굴 이미지를 사용하여 얼굴 인식 모델을 학습시킬 수 있습니다.

Face 서비스를 사용하여 얼굴 인식 모델을 학습하려면 다음을 수행합니다.

1. 식별하고자 하는 사람들의 집합(예: 직원)을 정의하는 **사람 그룹**을 만드십시오.
2. 식별하려는 각 개인에 대해 **개인**을 **개인 그룹으로** 추가합니다.
3. 여러 이미지에서 감지된 얼굴을 각 **사람**에게 추가하되, 다양한 포즈로 추가하는 것이 좋습니다. 이러한 얼굴의 ID는 더 이상 24시간 후에 만료되지 않으며(따라서 이제 **지속형** 얼굴이라고 불립니다).
4. 모델을 학습시킵니다.



학습된 모델은 Face(또는 Azure AI Services) 리소스에 저장되며 클라이언트 애플리케이션에서 다음을 위해 사용할 수 있습니다.

- 이미지에서 개인을 식별합니다.
- 감지된 얼굴의 ID를 확인합니다.
- 새 이미지를 분석하여 알려진 지속형 얼굴과 유사한 얼굴을 찾습니다.

# KTds Azure AI 교육 교재 Day 7

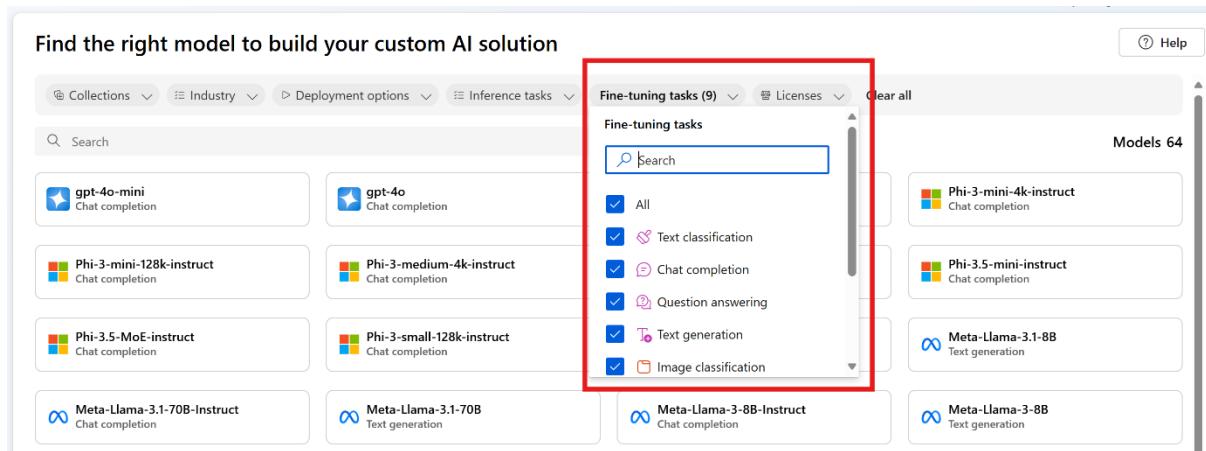
## Azure AI Foundry를 사용하여 모델 미세 조정

미세 조정은 특정 작업 또는 데이터 세트에 대한 추가 교육을 통해 미리 학습된 AI 모델을 사용자 지정하여 성능을 향상시키거나, 새 기술을 추가하거나, 정확도를 향상시킵니다. 결과는 제공된 예제를 기반으로 하는 새로운 최적화된 GenAI 모델입니다. 이 문서에서는 미세 조정을 위한 사용 사례와 GenAI 여정에 도움이 되는 방법을 안내합니다.

GenAI 모델을 다음과 같이 미세 조정하는 것이 좋습니다.

- 특정 엔터프라이즈 요구 사항에 맞게 크기 조정 및 조정
- 맞춤형 모델이 부정확하거나 관련이 없는 응답을 생성할 가능성이 적기 때문에 가양성 감소
- 도메인별 작업에 대한 모델의 정확도 향상
- 더 빠르고 정확한 결과를 통해 시간과 리소스 절약
- 모델이 특정 사용 사례에 맞게 미세 조정됨에 따라 더 관련성이 높고 컨텍스트 인식 결과를 얻습니다.

[Azure AI Foundry](#) 는 모델 공급자 간에 여러 모델을 제공하여 시장에서 가장 최신의 최신 모델에 액세스할 수 있도록 합니다. [자세한 내용은 이 목록을 참조하세요.](#)



## 미세 조정 시작

생성 AI 여정을 시작할 때는 기본 모델 및 해당 기능에 익숙해지도록 프롬프트 엔지니어링 및 RAG부터 시작하는 것이 좋습니다.

- 프롬프트 엔지니어링은 톤 및 스타일 세부 정보, 예제 응답 및 자연어 처리 모델에 대한 의도 매핑을 사용하여 프롬프트를 디자인하는 기술입니다. 이 프로세스는 응답의 정확도와 관련성을 향상시켜 모델의 성능을 최적화합니다.
- RAG(검색 보강 생성)은 외부 원본에서 데이터를 검색하고 프롬프트에 통합하여 LLM 성능을 향상시킵니다. RAG를 사용하면 기업은 데이터 관련성을 유지하고 비용을 최적화하면서 사용자 지정 솔루션을 달성할 수 있습니다.

편안하게 솔루션을 빌드하기 시작하면 프롬프트 엔지니어링이 부족한 위치와 미세 조정을 시도해야 하는 시기를 이해하는 것이 중요합니다.

- 기본 모델이 에지 사례 또는 예외에서 실패하나요?
- 기본 모델이 올바른 형식으로 출력을 일관되게 제공하지 않나요?
- 컨텍스트 창에서 모델을 조정하기에 충분한 예제를 맞추기가 어렵습니까?
- 대기 시간이 높습니까?

기본 모델 및 프롬프트 엔지니어링을 사용하는 오류의 예는 미세 조정을 위해 수집할 데이터를 식별하고 미세 조정된 모델을 평가하고 비교할 수 있는 성능 기준을 설정하는 데 도움이 될 수 있습니다. 미세 조정 없는 성능에 대한 기준을 설정하는 작업은 미세 조정이 모델 성능을 향상시켰는지 여부를 파악하는 데 필수적입니다.

예제는 다음과 같습니다.

고객은 GPT-4o-Mini를 사용하여 자연어 질문을 특정 비표준 쿼리 언어의 쿼리로 전환하려고 합니다. 고객은 프롬프트에 지침("항상 GQL 반환")을 제공하고 RAG를 사용하여 데이터베이스 스키마를 검색했습니다. 그러나 구문이 항상 올바른 것은 아니었으며 예지 사례에 대해 종종 실패했습니다. 고객은 모델이 이전에 실패한 경우를 포함하여 수천 개의 자연어 질문 예제와 데이터베이스에 대한 동등한 쿼리를 수집합니다. 그런 다음 고객은 해당 데이터를 사용하여 모델을 미세 조정합니다. 새로 미세 조정된 모델과 엔지니어링된 프롬프트 및 검색을 결합하면 모델 출력의 정확도가 허용 가능한 사용 기준에 도달합니다.

## 사용 사례

기본 모델은 이미 방대한 양의 데이터에 미리 학습되어 있습니다. 대부분의 경우 프롬프트에 지침과 예제를 추가하여 찾고 있는 품질 응답을 얻습니다. 이 프로세스를 "몇 번의 학습"이라고 합니다. 미세 조정을 사용하면 특정 사용 사례에 맞게 조정할 수 있는 더 많은 예제를 사용하여 모델을 학습할 수 있으므로 몇 번의 학습이 향상됩니다. 미세 조정은 프롬프트의 토큰 수를 줄여 잠재적인 비용 절감 및 대기 시간이 짧은 요청으로 이어질 수 있습니다.

자연어를 쿼리 언어로 전환하는 것은 모델이 어떻게 행동해야 하는지를 보여주는 하나의 사용 사례일 뿐입니다. 다음은 몇 가지 다른 사용 사례입니다.

- 검색된 데이터의 모델 처리 개선
- 특정 스타일, 톤 또는 형식으로 콘텐츠를 출력하도록 모델 조정
- 정보를 조회할 때 정확도 향상
- 프롬프트 길이 줄이기
- 새로운 기술 교육(즉, 코딩할 자연어)

비용이 주요 동기 부여자라고 판단할 경우 신중하게 진행합니다. 미세 조정은 프롬프트를 줄이거나 더 작은 모델을 사용할 수 있도록 하여 특정 사용 사례에 대한 비용을 줄일 수 있습니다. 그러나 학습에 대한 선불 비용이 더 높을 수 있으며 사용자 지정 모델을 호스팅하는 비용을 지불해야 합니다.

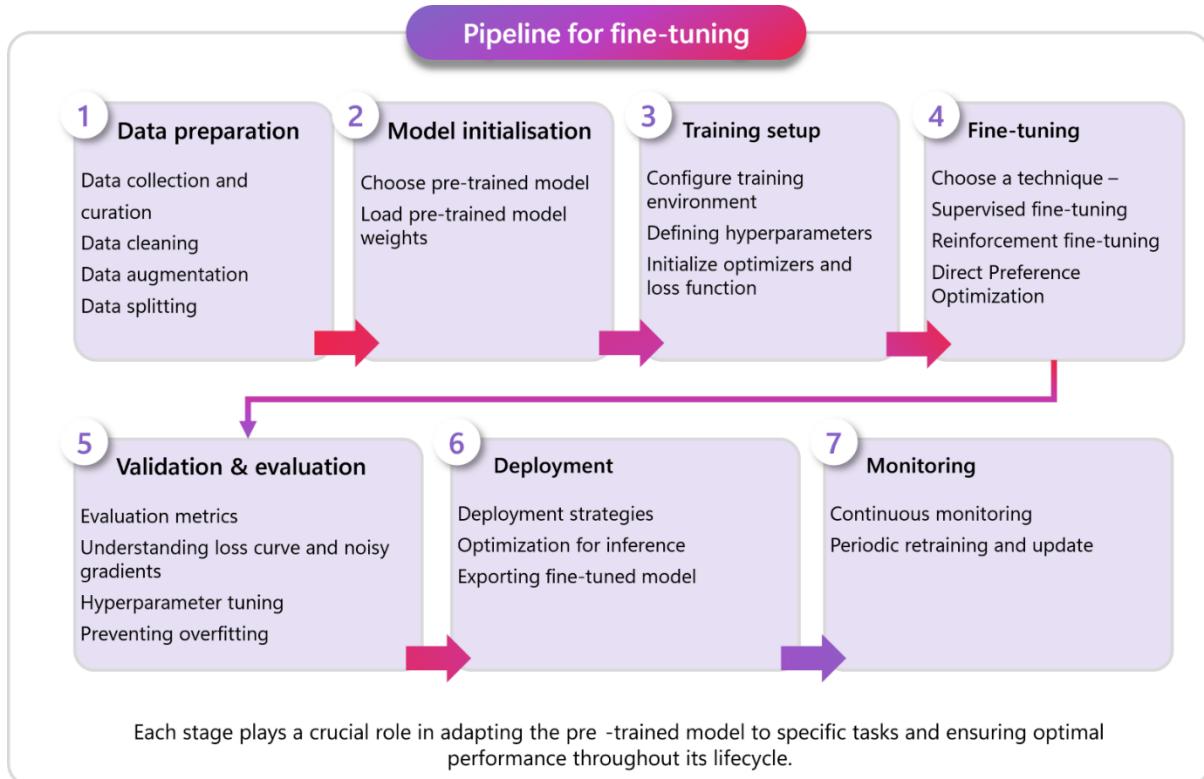
## 모델을 미세 조정하는 단계

대체적으로, 미세 조정을 수행하려면 다음과 같은 과정을 따라야 합니다.

- 학습 데이터 준비 및 업로드

- 새로운 세밀하게 조정된 모델 학습
- 새로 학습된 모델을 평가합니다.
- 추론을 위해 해당 모델을 배포하세요.
- 애플리케이션에서 미세 조정된 모델 사용

미세 조정은 제공할 수 있는 데이터의 품질에 크게 좌우됩니다. 성공하고 원하는 결과를 얻기 위해 수백 개의 학습 예제를 제공하는 것이 가장 좋습니다.



## 데이터 준비

### 미세 조정에 사용할 데이터는 무엇인가요?

미세 조정 프로세스는 미리 학습된 모델을 선택하고 대상 작업에 맞게 조정된 관련 데이터 세트를 준비하는 것으로 시작합니다. 이 데이터 세트는 모델이 배포에서 볼 수 있는 입력의 종류를 반영해야 합니다.

이 링크를 따라 [예제 데이터 세트를](#) 보고 다운로드하여 미세 조정을 시도합니다.

예를 들어 감정 분석을 위해 모델을 미세 조정하는 것이 목표인 경우 데이터 세트에는 감정(긍정, 부정, 중립)으로 분류된 레이블이 지정된 텍스트 예제가 포함됩니다. 그런 다

음 이 데이터 세트에서 모델을 다시 학습하여 새 작업에 더 잘 맞도록 매개 변수를 조정 합니다. 이 재학습 프로세스는 일반적으로 기존 기능을 기반으로 하므로 모델을 처음부터 학습하는 것에 비해 더 적은 계산 리소스가 필요합니다.

유용한 사용 사례에도 불구하고 미세 조정은 제공할 수 있는 데이터의 품질 정도로만 유용합니다. 다른 모델에는 서로 다른 데이터 볼륨이 필요하지만 올바른 형식으로 상당히 많은 양의 고품질 큐레이팅된 데이터를 제공해야 하는 경우가 많습니다. 이 샘플 리포지토리를 사용하여 서식 조건 및 데이터 준비를 이해할 수 있습니다.

채팅 또는 질문 답변에 대한 모델을 미세 조정하려면 학습 데이터 세트에 모델이 처리할 상호 작용 유형이 반영되어야 합니다. 데이터 세트에 포함해야 할 핵심 요소는 다음과 같습니다.

- **프롬프트 및 응답:** 각 항목에는 프롬프트(예: 사용자 질문) 및 해당 응답(예: 모델의 회신)이 포함되어야 합니다.
- **상황별 정보:** 다중 턴 대화의 경우 모델이 컨텍스트를 이해하고 일관성을 유지하는 데 도움이 되는 이전 교환을 포함합니다.
- **다양한 예:** 일반화 및 견고성을 개선하기 위한 다양한 토픽 및 시나리오를 다룹니다.
- **사람이 생성한 응답:** 인간이 작성한 응답을 사용하여 자연스럽고 정확한 회신을 생성하는 방법을 모델에 학습합니다.
- **서식 지정:** 명확한 구조를 사용하여 프롬프트와 응답을 구분합니다. 예를 들어 #n# 구분 기호가 콘텐츠에 표시되지 않는지 확인합니다.

## 데이터 준비 모범 사례

학습 사례가 많을수록 좋습니다. 10개 이상의 학습 예제가 없으면 미세 조정 작업이 진행되지 않지만 이러한 적은 수만으로는 모델 응답에 눈에 띄게 영향을 주지 않습니다. 성공하려면 수천 개는 아니더라도 수백 개의 학습 사례를 제공하는 것이 가장 좋습니다. 100개의 좋은 품질의 예는 1,000개의 가난한 예보다 낫습니다.

일반적으로 데이터 세트 크기가 두 배로 증가할 때마다 모델 품질이 선형적으로 증가할 수 있습니다. 그러나 저품질 예제는 성능에 부정적인 영향을 미칠 수 있습니다. 최고 품질의 예제에 대해서만 데이터 세트를 먼저 정리하지 않고 대량의 내부 데이터에서 모델을 학습하는 경우, 예상보다 훨씬 더 나쁜 성능을 발휘하는 모델이 될 수 있습니다.

## 데이터 레이블 지정 모범 사례

정확하고 일관된 레이블 지정은 모델 학습에 매우 중요합니다. 다음 모범 사례를 따르세요:

- **데이터 다양성 보장:** 문서 형식(디지털 및 스캔됨), 레이아웃 차이, 다양한 테이블 크기 및 선택적 필드와 같은 모든 일반적인 변형을 포함합니다.
- **필드를 명확하게 정의:** 사용자 지정 모델에서 특히 effective\_date와 같은 의미 있는 필드 이름을 사용하고, 패스칼 또는 낙타 대/소문자와 같은 일관된 명명 규칙을 따르십시오.
- **레이블 일관성 유지:** 특히 반복되는 값에 대해 문서 간에 균일한 레이블 지정을 보장합니다.
- **데이터 분할:** 학습 및 유효성 검사 집합을 분리하여 보이지 않는 데이터에 대한 모델을 평가하고 과잉 맞춤을 방지합니다.
- **대규모 레이블:** 해당하는 경우 클래스당 최소 50개의 레이블이 지정된 문서를 목표로 합니다.
- **자동화 및 검토 결합:** AI 생성 레이블을 사용하여 프로세스를 가속화하고 복잡하거나 중요한 필드에 수동 노력을 집중합니다.

## 모델 선택

미세 조정에 적합한 모델을 선택하는 것은 성능, 효율성 및 비용에 영향을 주는 중요한 결정입니다. 선택하기 전에 작업을 명확하게 정의하고 원하는 성능 메트릭을 설정하는 것이 중요합니다. 잘 정의된 작업은 선택한 모델이 특정 요구 사항에 맞게 조정되어 노력과 리소스를 최적화하도록 합니다.

## 모델 선택에 대한 모범 사례

- **도메인 특이성 및 사용 사례에 따라 모델 선택:** 일반 기능에 대한 업계 표준 모델을 평가한 다음 특정 사용 사례에 맞게 미세 조정된 모델을 평가합니다. 작업에 심층적인 도메인 전문 지식이 필요한 경우 업계에 맞는 모델을 선택하면 정확도와 효율성이 향상되는 동시에 광범위한 미세 조정의 필요성을 줄일 수 있습니다.
- **순위표에서 모델 성능 평가:** 벤치마크 순위표를 검토하여 사전 학습된 모델이 관련 작업에서 수행하는 방식을 평가합니다. 정확도, 일관성, 대기 시간 및 도메인별 벤치마크와 같은 주요 메트릭에 집중하여 미세 조정을 위한 강력한 기반을 파악

합니다.

- **모델 플레이그라운드 실험:** 대화형 테스트 환경을 활용하여 실제 사용 사례에서 기본 모델의 성능을 평가합니다. 프롬프트, 온도 및 기타 매개 변수를 조정하여 미세 조정에 투자하기 전에 성능 격차를 식별할 수 있습니다.
- **모델 크기, 복잡성, 비용 및 성능 간의 장차 가중치:** 더 큰 모델은 뛰어난 정확도를 제공하지만 더 높은 계산 비용과 대기 시간을 제공할 수 있습니다. 배포 요구 사항에 따라 효율성과 정밀도 간의 균형을 고려합니다.

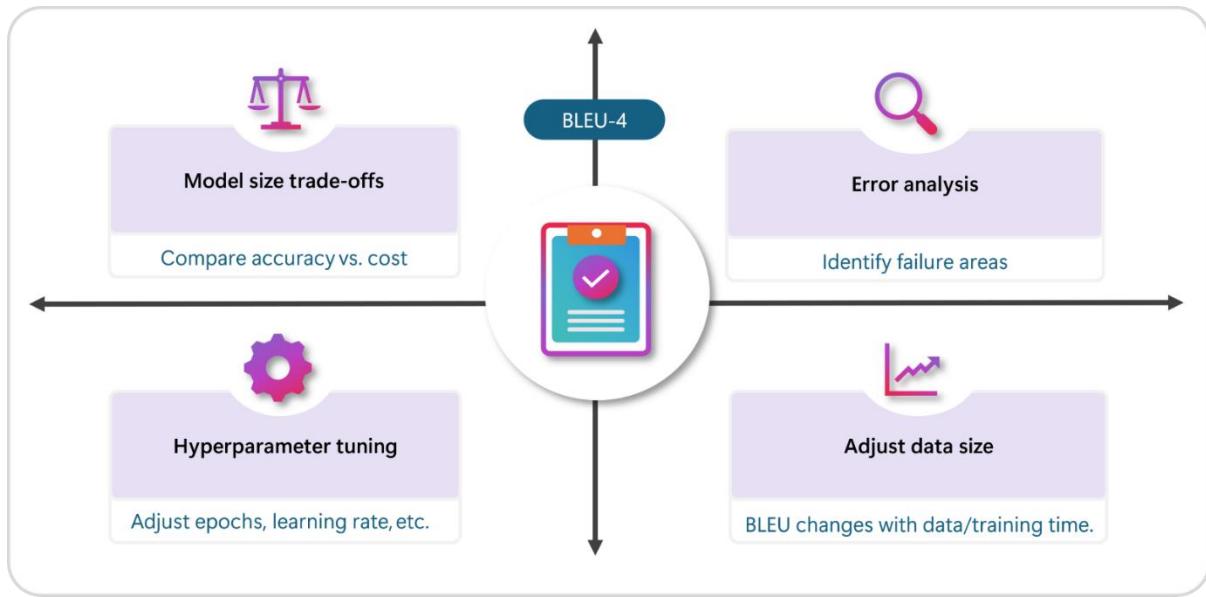
## 교육 및 평가

미세 조정은 단순히 새 데이터 세트에 대한 재학습의 문제가 아닙니다. 또한 정확도와 일반화의 균형을 맞추기 위해 다양한 하이퍼 매개 변수와 기술을 신중하게 고려해야 합니다. 주요 위험은 모델이 학습 데이터에 너무 좁게 조정되어 보이지 않는 입력에 대한 효율성을 줄이는 과잉 맞춤입니다. 과잉 맞춤을 완화하고 성능을 최적화하려면 학습 속도, 정규화, 일괄 처리 크기, epoch 수 및 시드 설정과 같은 매개 변수를 조정해야 합니다.

## 미세 조정에 평가 사용

미세 조정의 성공에 대한 목표를 명확하게 정의해야 합니다. 정성적 측정값을 넘어 훌드 아웃 유효성 검사 집합 사용, 사용자 승인 테스트 수행 또는 미세 조정된 모델을 기본 모델과 비교하는 A/B 테스트와 같은 양적 메트릭을 포함하는 것이 이상적입니다.

모델 학습은 메트릭을 통해 안내할 수 있습니다. 예를 들어 이 문서에서 볼 수 있듯이 BLEU-4는 모델을 미세 조정하여 흉부 X선 보고서를 생성할 때 학습을 평가하는 데 사용되었습니다. 또한 학습하는 동안 메트릭을 모니터링할 수도 있습니다. 손실 곡선이 예상대로 수렴되지 않는 경우 작업을 일시 중지하고 분석하고 다시 시작할 수 있습니다.



모델 선택 향상을 위해 중간 검사점을 사용합니다. 검사점을 정기적으로 저장하고(예: 몇 epoch마다) 성능을 평가합니다. 경우에 따라 중간 검사점이 최종 모델을 능가할 수 있으므로 마지막으로 학습된 반복에만 의존하지 않고 최상의 버전을 선택할 수 있습니다.

## 배포 및 모니터링

- 클라우드 기반 플랫폼 또는 온-프레미스 서버와 같은 적합한 배포 인프라를 선택합니다.
- 모델의 성능을 지속적으로 모니터링하고 최적의 성능을 보장하기 위해 필요한 조정을 수행합니다.
- 엔터프라이즈 SLA를 충족하기 위해 지역 배포 요구 사항 및 대기 시간 요구 사항을 고려합니다. 프라이빗 링크, 암호화 및 액세스 제어와 같은 보안 보호책을 구현하여 중요한 데이터를 보호하고 조직 정책 준수를 유지합니다.

## 미세 조정을 위해 지원되는 모델

이제 사용 사례에 미세 조정을 사용하는 시기를 알게 되었으므로 Azure AI Foundry로 이동하여 미세 조정에 사용할 수 있는 모델을 찾을 수 있습니다. 모델 카탈로그의 일부 모델의 경우 표준 배포 또는 관리형 컴퓨팅(미리 보기) 또는 둘 다를 사용하여 미세 조정을 사용할 수 있습니다.

세부 조정은 표준 배포를 통해 배포되는 일부 모델에 대해 특정 Azure 지역에서 사용할 수 있습니다.

미세 조정할 수 있는 Azure OpenAI 모델의 경우 미세 조정을 위해 지원되는 지역에는 미국 중북부, 스웨덴 중부 등이 포함됩니다.

## Azure OpenAI 모델 미세 조정

### 참고

gpt-35-turbo - 이 모델의 미세 조정은 하위 지역 집합으로 제한되며 기본 모델을 사용 할 수 있는 모든 지역에서는 사용할 수 없습니다.

Azure AI Foundry 프로젝트에서 Azure OpenAI 모델을 사용하는 경우와 프로젝트 외부를 사용하는 경우 미세 조정에 지원되는 지역이 달라질 수 있습니다.

### 테이블 확장

모델 아이디	표준 교육 글로벌 교육(미리 보기)	최대 요청(토큰)	학습 데이터 (최대)	형식
gpt-35-turbo(1106)	미국 동부2 - 미국 중북부 스웨덴 중부 스위스 서부	입력: 16,385 출력: 4,096	2021년 9월	텍스트 대 텍스트
gpt-35-turbo (0125)	미국 동부2 - 미국 중북부 스웨덴 중부 스위스 서부	16,385	2021년 9월	텍스트 대 텍스트
gpt-4o-mini(2024-07-18)	미국 중북부 스웨덴 중부	입력: 128,000 출력: 16,384 학습 예제 컨텍스트 길이: 65,536	2023년 10월	텍스트 대 텍스트
gpt-4o (2024-08-06)	미국 동부2 - 미국 중북부 스웨덴 중부	입력: 128,000 출력: 16,384 학습 예제 컨텍스트 길이: 65,536	2023년 10월	텍스트 및 비전에서 텍스트로

모델 아이디	표준 교육 글로벌 교육(미리 보기)	최대 요청(토큰)	학습 데이터 (최대)	형식
지역				
gpt-4.1 (2025-04-14)	미국 중북부	입력: 128,000 출력: 16,384 학습 예제 컨텍스트 길이: 65,536	2024년 5월	텍스트 및 비전에서 텍스트로
gpt-4.1-mini (2025-04-14)	미국 중북부	입력: 128,000 출력: 16,384 학습 예제 컨텍스트 길이: 65,536	2024년 5월	텍스트 대 텍스트
gpt-4.1-nano (2025-04-14)	미국 중북부	입력: 128,000 출력: 16,384 학습 예제 컨텍스트 길이: 32,768	2024년 5월	텍스트 대 텍스트
o4-mini (2025-04-16)	미국 동부 - 스웨덴 중부	입력: 128,000 출력: 16,384 학습 예제 컨텍스트 길이: 65,536	2024년 5월	텍스트 대 텍스트

## 참고

글로벌 교육(공개 미리 보기)은 토큰당 [더 저렴한](#) 교육을 제공하지만 [데이터 상주를](#) 제공하지는 않습니다. 현재 다음 지역의 Azure OpenAI 리소스에서 사용할 수 있으며 곧 더 많은 지역이 제공될 예정입니다.

- 오스트레일리아 동부
- 브라질 남부
- 프랑스 중부
- 독일 중서부
- 이탈리아 북부
- 일본 동부 (*비전 지원 없음*)
- 한국 중부

- 노르웨이 동부
- 폴란드 중부
- 동남아시아
- 스페인 중부
- 남아프리카 북부
- 

### 미세 조정 모범 사례

다음은 다양한 애플리케이션에 대한 LLM 미세 조정의 효율성과 효율성을 개선하는 데 도움이 되는 몇 가지 모범 사례입니다.

- **더 작은 모델로 시작:** 일반적인 실수는 애플리케이션에 가장 크고 비용이 많이 드는 최신 모델이 필요하다고 가정하는 것입니다. 특히 더 간단한 작업의 경우 더 작은 모델로 시작하고 필요한 경우에만 더 큰 모델을 사용해 보세요.
- **도메인 요구 사항에 따라 모델 선택:** 특정 사용 사례에 대해 미세 조정된 버전을 고려하기 전에 업계 표준 모델로 시작합니다. 벤치마크 순위표를 사용하여 모델 플레이그라운드에서 성능을 평가하고 실제 시나리오를 테스트합니다. 정확도, 비용 및 효율성의 균형을 유지하여 배포에 가장 적합합니다.
- **고품질의 대규모 데이터 세트 수집:** LLM은 많은 데이터를 필요로 하며, 보다 다양하고 대표적인 데이터를 사용하여 미세 조정할 수 있습니다. 그러나 큰 데이터 세트를 수집하고 주석을 추가하면 비용이 많이 들고 시간이 많이 걸릴 수 있습니다. 따라서 가상 데이터 생성 기술을 사용하여 데이터 세트의 크기와 다양성을 늘릴 수도 있습니다. 그러나 가상 데이터가 작업 및 도메인과 관련이 있고 일치하는지 확인해야 합니다. 또한 모델에 노이즈나 바이어스를 유발하지 않는지 확인합니다.
- **하위 집합 미세 조정 먼저 시도:** 더 많은 데이터를 가져오는 값을 평가하려면 현재 데이터 세트의 하위 집합에서 모델을 미세 조정하여 데이터 세트 크기로 성능이 어떻게 확장되는지 확인할 수 있습니다. 이 미세 조정은 모델의 학습 곡선을 예측하고 더 많은 데이터를 추가하는 것이 노력과 비용의 가치가 있는지 여부를 결정하는 데 도움이 될 수 있습니다. 모델의 성능을 미리 학습된 모델 또는 기준 선과 비교할 수도 있습니다. 이 비교는 미세 조정을 통해 얼마나 많은 개선을 달성할 수 있는지를 보여줍니다.

- **하이퍼 매개 변수 실험:** 하이퍼 매개 변수를 반복적으로 조정하여 모델 성능을 최적화합니다. 학습 속도, 일괄 처리 크기 및 Epoch 수와 같은 하이퍼 매개 변수는 모델의 성능에 상당한 영향을 줄 수 있습니다. 따라서 태스크 및 데이터 세트에 가장 적합한 값을 찾기 위해 하이퍼 매개 변수의 다양한 값과 조합을 실험해야 합니다.
- **다른 데이터 형식을 사용해 보세요.** 작업에 따라 데이터 형식이 다르면 모델 성능에 다른 영향을 미칠 수 있습니다. 예를 들어 분류 작업의 경우 {"prompt": "Paris##\\n", "completion": "city\\n##\\n"}과 같은 특수 토큰으로 프롬프트와 완성을 구분하는 형식을 사용할 수 있습니다. 애플리케이션에 적합한 형식을 사용해야 합니다.

# KTds Azure AI 교육 교재 Day 7

## Azure Speech 소개

AI 음성 기능을 사용하면 음성 명령으로 집과 자동차 시스템을 관리하고, 음성 질문에 대한 답변을 컴퓨터에서 가져오고, 오디오에서 캡션을 생성하는 등 다양한 작업을 수행할 수 있습니다.

이러한 종류의 상호 작용을 가능하게 하려면 AI 시스템이 최소한 두 가지 기능을 지원해야 합니다.

- **음성 인식** - 음성 입력을 감지하고 해석하는 기능.
- **음성 합성** - 음성 출력을 생성하는 기능.

**Azure AI 음성**은 음성 인식 및 합성을 통해 음성 텍스트 변환, 텍스트 음성 변환, 음성 번역 기능을 제공합니다. 오디오를 매우 정확하게 텍스트로 변환하는 것부터 대화에서 화자를 식별하고 사용자 지정 음성을 만드는 것까지 다양한 작업에 미리 빌드된 사용자 지정 음성 서비스 모델을 사용할 수 있습니다. 다음으로 AI 음성 기능이 어떻게 작동하는지 알아보겠습니다.

## 음성 인식 및 합성 이해

**음성 인식**은 말한 말을 받아 처리할 수 있는 데이터로 기록합니다. 이는 종종 말을 텍스트로 기록하는 작업을 통해 이루어집니다. 말해진 단어는 오디오 파일의 녹음된 음성 또는 마이크에서 나오는 라이브 오디오 형식이 될 수 있습니다. 음성 패턴은 오디오에서 분석되어 단어에 매핑되는 인식 가능한 패턴을 결정합니다. 이를 달성하기 위해 소프트웨어는 일반적으로 다음을 포함한 여러 모델을 사용합니다.

- 오디오 신호를 음소(특정 사운드를 나타내는 단위)로 변환하는 '음향' 모델.

- 음소를 단어로 매핑하는 '언어' 모델(일반적으로 음소에 따라 가장 가능성이 높은 단어 시퀀스를 예측하는 통계 알고리즘 사용).

인식된 단어는 일반적으로 다음과 같이 다양한 목적에 사용할 수 있는 텍스트로 변환됩니다.

- 녹화된 동영상 또는 라이브 비디오에 대한 자막 제공
- 전화 통화 또는 회의 내용 대본 만들기
- 자동화된 메모 받아쓰기
- 추가 처리를 위해 의도한 사용자 입력 결정

**음성 합성**은 일반적으로 텍스트를 음성으로 변환하여 데이터를 음성으로 표현하는 것과 관련이 있습니다. 음성 합성 솔루션에는 일반적으로 다음 정보가 필요합니다.

- 읽을 텍스트
- 말을 음성화하는 데 사용할 음성

음성을 합성하기 위해 시스템은 일반적으로 텍스트를 '토큰화'하여 개별 단어로 분할하고 각 단어에 음성 발음을 할당합니다. 그런 다음 오디오 형식으로 변환될 음소를 만들기 위해 음성 전사를 '운율' 단위(예: 구, 절 또는 문장)로 세분화합니다. 이러한 음소는 오디오로 합성되고 특정 음성, 말하는 속도, 음높이, 볼륨이 할당될 수 있습니다.

다음과 같이 다양한 목적으로 음성 합성의 출력력을 사용할 수 있습니다.

- 사용자 입력에 대한 음성 응답 생성
- 전화 시스템을 위한 음성 메뉴 만들기
- 핸즈프리 시나리오에서 메일 또는 문자 메시지를 소리 내어 읽기
- 기차역 또는 공항과 같은 공공장소에서 공지 사항 방송

## Azure에서 음성 시작

Microsoft Azure는 다음을 포함한 다양한 기능을 지원하는 **Azure AI 음성** 서비스를 통해 음성 인식 및 합성 기능을 제공합니다.

- 음성 텍스트 변환

- 텍스트 음성 변환

### 음성 텍스트 변환

Azure AI 음성 텍스트 변환 API를 사용하여 오디오의 대화 내용을 텍스트 형식으로 실시간 또는 일괄 기록할 수 있습니다. 전사용 오디오 소스는 마이크 또는 오디오 파일에서 나오는 실시간 오디오 스트림일 수 있습니다.

음성 텍스트 변환 API에서 사용하는 모델은 Microsoft에서 학습한 범용 언어 모델을 기반으로 합니다. 모델에 사용하는 데이터는 Microsoft 소유이며 Microsoft Azure에 배포됩니다. 이 모델은 대화 및 받아쓰기라는 두 가지 시나리오에 최적화되어 있습니다. Microsoft에서 미리 빌드된 모델이 필요한 항목을 제공하지 않는 경우 음향, 언어 및 발음을 포함한 사용자 지정 모델을 만들고 학습할 수도 있습니다.

**실시간 대화 내용 기록:** 실시간 음성 텍스트 변환을 이용하여 오디오 스트림에서 텍스트를 전사할 수 있습니다. 프레젠테이션, 데모 또는 사람이 말하는 다른 모든 시나리오에 실시간 전사를 사용할 수 있습니다.

실시간 대화 내용 기록이 작동하려면 마이크에서 수신 오디오 또는 오디오 파일과 같은 다른 오디오 입력 원본을 애플리케이션에서 수신 대기할 수 있어야 합니다. 애플리케이션 코드는 오디오를 서비스로 스트리밍하여 전사된 텍스트를 반환합니다.

**일괄 대화 내용 기록:** 모든 음성 텍스트 변환 시나리오가 실시간인 것은 아닙니다. 파일 공유, 원격 서버 또는 Azure Storage에 오디오 녹음이 저장되어 있을 수 있습니다. SAS(공유 액세스 서명) URI가 있는 오디오 파일을 가리키고 비동기적으로 전사 결과를 받을 수 있습니다.

일괄 작업이 '최선의 노력 기준'으로 예약되므로 전사 일괄 처리는 비동기 방식으로 실행해야 합니다. 일반적으로 작업은 요청 후 몇 분 이내에 실행을 시작하지만 작업이 실행 상태로 전환되는 시점에 대한 예상은 없습니다.

### 텍스트 음성 변환

텍스트 음성 변환 API를 사용하면 텍스트 입력을 가정 음성으로 변환할 수 있으며, 이를 컴퓨터 스피커를 통해 직접 재생하거나 오디오 파일에 쓸 수 있습니다.

**음성 합성 목소리:** 텍스트 음성 변환 API를 사용하는 경우 텍스트를 발음하는 데 사용할 음성을 지정할 수 있습니다. 이 기능을 통해 유연하게 음성 합성 솔루션을 개인화하고

개성을 부여할 수 있습니다.

이 서비스에는 '표준' 음성뿐만 아니라 **신경망**을 활용하여 억양과 관련된 음성 합성의 일반적인 한계를 극복하는 신경음성을 포함하여 여러 언어 및 지역 발음을 지원하는 여러 미리 정의된 음성이 포함되어 있어 보다 자연스러운 음성이 생성됩니다. 사용자 지정 음성을 개발하고 텍스트 음성 변환 API와 함께 사용할 수도 있습니다.

## Azure AI 음성 사용

**Azure AI 음성**은 다음을 비롯한 여러 도구 및 프로그래밍 언어를 통해 사용할 수 있습니다.

- 스튜디오 인터페이스
- CLI(명령줄 인터페이스)
- REST API 및 SDK(소프트웨어 개발 키트)

## 스튜디오 인터페이스

**Speech Studio** 또는 **Azure AI 스튜디오**의 사용자 인터페이스를 사용하여 Azure AI 음성 프로젝트를 만들 수 있습니다.

### 테이블 확장

The screenshot shows two side-by-side interfaces. On the left is the **Speech Studio**, which has a main heading "Get started with Azure Cognitive Services Speech" and several cards for "Captioning with speech to text", "Post call transcription and analysis", "Live chat avatar", and "Language learning practice". On the right is the **Azure AI Studio**, which has a main heading "Infuse your solutions with AI capabilities" and three cards for "Speech", "Language + Translator", and "Vision + Document". Each card includes a brief description and a "View all [service] capabilities" link.

## Azure AI Speech용 Azure 리소스

애플리케이션에서 Azure AI Speech를 사용하려면 Azure 구독에서 적절한 리소스를 만들

어야 합니다. 선택에 따라 다음과 같은 종류의 리소스 중 하나를 만들 수 있습니다.

- **Speech** 리소스 - Azure AI Speech만 사용할 계획이거나 다른 서비스와 별도로 리소스에 대한 액세스 및 청구를 관리하려는 경우 이 리소스 유형을 선택하세요.
- **Azure AI 서비스** 리소스 - 다른 Azure AI 서비스와 함께 Azure AI Speech를 사용하고 이러한 서비스에 대한 액세스 및 청구를 함께 관리하려는 경우 이 리소스 유형을 선택합니다.

# KTds Azure AI 교육 교재 Day 8

## Azure Cosmos DB NoSQL의 벡터 검색 및 스토리지 기능 구성

NoSQL용 Azure Cosmos DB에는 고차원 벡터를 저장하고 쿼리하는 강력한 방법을 제공하는 벡터 검색 기능이 포함되어 있습니다. 이 기능은 통합 벡터 검색 기능이 필요한 생성 AI 애플리케이션에 필수적입니다. NoSQL용 Azure Cosmos DB 벡터 데이터베이스를 사용하면 원본 데이터와 함께 포함을 저장, 인덱싱 및 쿼리할 수 있습니다. 즉, 데이터베이스의 각 문서에는 고차원 벡터와 기존 스키마 없는 데이터가 포함될 수 있습니다. 벡터 포함과 함께 나타내는 원래 데이터를 유지하면 다중 모달 데이터 작업이 향상되고 데이터 일관성, 크기 조정 및 성능이 향상됩니다. 또한 데이터 관리, AI 애플리케이션 아키텍처 및 벡터 기반 작업의 효율성을 간소화합니다.

### Azure Cosmos DB for NoSQL에서 벡터 검색 사용

Azure Cosmos DB에서 NoSQL API에 대한 벡터 검색 기능을 사용하도록 설정하는 작업은 [Azure Portal](#) 또는 [Azure CLI](#)(Azure 명령줄 인터페이스)를 통해 수행할 수 있습니다.

Azure Portal의 기능 아래에 나열됩니다.

Feature	Status
Full-Text & Hybrid Search for NoSQL API (preview)	Off
<b>Vector Search for NoSQL API</b>	Off
Dynamic Scaling (Per Region and Per Partition Autoscale)	On
Priority based execution (preview)	Off
Burst Capacity	Off
Partition merge (preview)	Off
Materialized Views for NoSQL API (preview)	Off
Diagnostics full-text query	Off

기능 설명을 검토한 후 **벡터 검색 for NoSQL API** 기능을 사용하도록 설정하여 계정에서 사용할 수 있도록 할 수 있습니다.

## Vector Search for NoSQL API

This enrolls the Azure Cosmos DB resource in the Vector Search feature, which enables you to perform vector similarity search on your documents. This includes capabilities such as:

- Defining container vector policies
- Adding vector indexes of type: flat, quantizedFlat, and DiskANN, to the Indexing Policy
- Using indexes in the VectorDistance() system function to perform vector search

**Important information:**

- After enrolling in the preview, **it may take up to 15 minutes to be applied to your Azure Cosmos DB for NoSQL resource.**
- Vector Search is only available in the Azure Cosmos DB for NoSQL API.
- This requires latest versions of the .NET, Python, JavaScript, and Java SDKs to use vector search.
- The vector search performance will be significantly improved by adding a vector index. **It's strongly recommended to create a vector index** to reduce RU cost and latency of vector search queries.
- Once this capability is enabled on an account, it cannot be disabled.
- These Azure Cosmos DB features are currently not supported with vector search: Shared Throughput and Analytical Store.

[Learn More about Vector Search in Azure Cosmos DB](#)

[Enable](#) [Close](#)

또는 다음 명령을 실행하여 Azure CLI를 통해 Vector Search를 사용하도록 설정할 수 있습니다.

Azure CLI 복사

```
az cosmosdb update ¶  
--resource-group <resource-group-name> ¶  
--name <cosmos-db-account-name> ¶  
--capabilities EnableNoSQLVectorSearch
```

## 컨테이너 벡터 정책 정의

NoSQL용 Azure Cosmos DB 계정에서 벡터 검색 기능을 사용하도록 설정하면 벡터를 저장할 컨테이너에 대한 벡터 포함 정책을 정의해야 합니다. 이 정책은 시스템 함수에서 벡터 속성을 처리하는 방법을 Azure Cosmos DB 쿼리 엔진에 VectorDistance 알려줍니다. 또한 벡터 인덱싱 정책에 필요한 세부 정보를 지정해야 합니다.

 벡터 검색 기능은 현재 기존 컨테이너에서 지원되지 않으므로 새 컨테이너를 만들고 컨테이너를 만들 때 컨테이너 수준 벡터 포함 정책 및 벡터 인덱싱 정책을 지정해야 합니다.

컨테이너 벡터 정책에는 다음 정보가 포함됩니다.

- path: 벡터 포함을 포함하는 속성의 경로입니다.
- datatype: 벡터에 있는 요소의 형식입니다. 기본값은 Float32입니다.
- dimensions: 이 속성은 각 벡터의 크기 또는 길이이며 포함을 만드는 데 사용되는 모델에 의해 구동됩니다.
- distanceFunction: 벡터 간의 거리 또는 유사성을 계산하는 데 사용되는 기술입니다. 사용 가능한 옵션은 유클리드산(기본값), 코사인 및 점 제품입니다.

벡터 검색에서 거리 함수는 다차원 공간 내에 있는 벡터와 유사하거나 다른 벡터를 결정합니다. 인기 있는 방법은 유클리드 거리, 코사인 유사성 및 맨해튼 거리를 포함하며, 각각 벡터를 비교하는 고유한 방법을 제공합니다.

## 벡터 인덱싱을 사용하여 벡터 검색 효율성 향상

NoSQL용 Azure Cosmos DB는 저장된 포함을 기반으로 벡터 검색 인덱스 만들기를 지원합니다. 벡터 검색 인덱스는 포함된 모든 데이터(벡터)에서 의미 체계 유사성 검색을 가능하게 하는 대기 시간 공간의 벡터 컨테이너입니다. 벡터 인덱스는 고차원 벡터 데이터에 특수화되어 NoSQL용 Azure Cosmos DB의 시스템 함수를 사용하여 VectorDistance 벡터 검색을 수행하는 효율성을 높입니다. 지원되는 벡터 인덱스 정책은 다음과 같습니다.

### 테이블 확장

유형	설명	최대 차원
플랫	다른 인덱싱된 속성과 동일한 인덱스에 벡터를 저장합니다.	505

유형	설명	최대 차원
quantizedFlat	인덱스에 저장하기 전에 벡터를 양자화(압축)합니다. 이 정책은 적은 양의 정확도로 대기 시간 및 처리량을 향상시킬 수 있습니다.	4096
diskANN	빠르고 효율적인 근사 검색을 위해 DiskANN을 기반으로 인덱스를 만듭니다.	4096

벡터 검색은 벡터 인덱스 사용 시 대기 시간을 줄이고 처리량을 늘리며 RU 사용량을 줄입니다. 벡터 인덱스 검색을 사용하면 정보를 인덱스에서 의미 체계적으로 검색한 다음 LLM(큰 언어 모델)에서 추론할 정확한 프롬프트를 생성하는 데 사용할 수 있는 프롬프트 전처리 단계를 수행할 수 있습니다. 이 프로세스는 LLM에 대한 지식 보강 및 도메인별 포커스를 제공합니다. 벡터 검색 인덱스는 벡터 필드가 들어오는 메시지와 의미상 유사한 문서 목록을 반환합니다. 동일한 문서에 저장된 원본 텍스트는 LLM의 프롬프트를 보강하며, LLM은 제공된 정보에 따라 요청자에 응답하는 데 사용합니다.

## Azure OpenAI Service를 사용하여 임베딩 생성

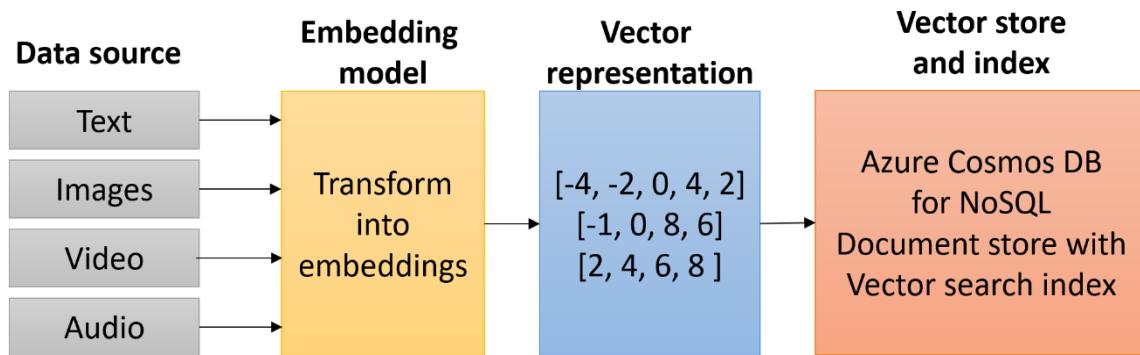
포함 또는 벡터 포함이라고도 하는 벡터는 고차원 공간에서 데이터를 수학적으로 표현한 것입니다. 각 차원은 이 공간의 데이터 기능에 해당하며, 수만 개의 차원을 사용하여 정교한 데이터를 나타낼 수 있습니다. 이 공간에서 벡터의 위치는 해당 특성을 나타냅니다. 단어, 구 또는 전체 문서, 이미지, 오디오 및 기타 유형의 데이터를 모두 벡터화할 수 있습니다. 데이터는 벡터로 표시되므로 벡터 검색은 여러 데이터 형식에서 일치하는 데이터를 식별할 수 있습니다.

포함은 기계 학습 모델 및 알고리즘이 효율적으로 활용할 수 있는 데이터 표현 형식입니다. 임베딩은 텍스트의 의미론적 의미를 정보 밀도가 높은 방식으로 표현한 것입니다. 각 임베딩은 부동 소수점 숫자의 벡터입니다. 따라서 벡터 공간의 두 포함 사이의 거리는 원래 형식의 두 입력 간의 의미 체계 유사성과 상관 관계가 있습니다.

## Azure OpenAI 를 사용하여 임베딩 생성

Azure OpenAI 는 OpenAI 의 고급 언어 모델을 Microsoft 의 Azure 플랫폼과 통합하여 개발자에게 안전하고 확장 가능한 환경을 제공하는 최첨단 서비스입니다. 이 강력한 조합을 사용하면 자연어 처리, 텍스트 요약 및 감정 분석과 같은 작업에 필수적인 인간과 유사한 텍스트를 이해하고 생성할 수 있는 지능형 애플리케이션을 만들 수 있습니다. Azure OpenAI 의 기본 애플리케이션은 복잡한 쿼리를 처리하고, 개인 설정된

응답을 제공하고, 다른 서비스와 원활하게 통합할 수 있는 정교한 가상 도우미 또는 생성 AI 애플리케이션을 빌드하는 것입니다. 개발자는 포함 모델을 사용하여 텍스트 데이터의 벡터 표현을 생성하고 NoSQL용 Azure Cosmos DB와 같은 벡터 저장소에 저장할 수 있습니다. 이 방법은 효율적이고 정확한 유사성 검색을 용이하게 하여 관련 정보를 검색하고 상황에 맞는 풍부한 상호 작용을 제공하는 생성 AI 애플리케이션의 기능을 크게 향상합니다.



포함은 임베딩 모델에 데이터를 전송하여 생성되며, 여기서 벡터로 변환됩니다. Azure OpenAI는 text-embedding-ada-002, text-embedding-3-small, text-embedding-3-large 모델을 포함하여 임베딩 생성을 위한 여러 모델을 제공합니다. Python용 Azure OpenAI SDK를 사용하여 포함을 생성하는 방법을 제공하는 Azure OpenAI 클라이언트를 만들 수 있습니다.

## Python

```

FROM AZURE.IDENTITY IMPORT DEFAULTAZURECREDENTIAL, GET_BEARER_TOKEN_PROVIDER
FROM OPENAI IMPORT AZUREOPENAI

```

```

# ENABLE MICROSOFT ENTRA ID RBAC AUTHENTICATION
CREDENTIAL = DEFAULTAZURECREDENTIAL()
TOKEN_PROVIDER = GET_BEARER_TOKEN_PROVIDER(
    CREDENTIAL,
    "HTTPS://COGNITIVESERVICES.AZURE.COM/.DEFAULT"
)

```

```

# INSTANTIATE AN AZURE OPENAI CLIENT

CLIENT = AZUREOPENAI(
    API_VERSION = AZURE_OPENAI_API_VERSION,
    AZURE_ENDPOINT = AZURE_OPENAI_ENDPOINT,
    AZURE_AD_TOKEN_PROVIDER = TOKEN_PROVIDER
)

# GENERATE EMBEDDINGS FOR INPUT TEXT

RESPONSE = CLIENT.EMBEDDINGS.CREATE(
    INPUT = "BUILD GENERATIVE AI APPLICATIONS WITH PYTHON AND AZURE COSMOS DB FOR
NoSQL",
    MODEL = "TEXT-EMBEDDING-3-LARGE"
)

# RETRIEVE THE GENERATED EMBEDDING

EMBEDDING = RESPONSE.DATA[0].EMBEDDING

```

Azure Cosmos DB for NoSQL에서 컨테이너 벡터 정책을 정의할 때 모델이 생성하는 차원 수를 아는 것이 중요합니다. 또한 적절한 벡터 인덱싱 정책을 선택하는 역할을 합니다. 만든 벡터의 차원은 벡터를 생성하는 데 사용되는 모델에 의해 결정됩니다. 이전 예제 text-embedding-3-large에서는 기본적으로 3,072 개의 차원이 포함된 벡터를 만든 모델을 사용했습니다. 컨테이너 벡터 정책을 정의할 때 **Dimensions** 속성에서 해당 번호를 지정해야 합니다. 마찬가지로 사용되는 차원 수를 지원하는 인덱스 형식을 선택해야 합니다.

만든 후에는 포함을 벡터 데이터베이스(예: NoSQL 용 Azure Cosmos DB)에 저장할 수 있습니다.

## 벡터 생성 및 저장을 위한 일반적인 개발 패턴

Azure OpenAI 및 NoSQL 용 Azure Cosmos DB 를 통합하여 벡터 포함을 생성하고 저장하려면 일반적으로 효율성, 실시간 처리 및 원활한 통합을 보장하는 자동화된 작업자 프로세스가 필요합니다. 이 통합을 수행하는 데 사용되는 몇 가지 일반적인 패턴은 다음과 같습니다.

### 1. **Cosmos DB 변경 피드 트리거를 사용하는 Azure Function**

데이터가 Cosmos DB 에 삽입될 때 포함을 생성하는 효과적인 패턴에는 Cosmos DB 변경 피드와 함께 Azure Function 을 사용하는 것이 포함됩니다. 문서가 컨테이너에 삽입되거나 업데이트될 때 함수의 Cosmos DB 트리거가 호출됩니다. 이 함수는 Azure OpenAI 포함 모델을 호출하여 벡터를 만들고, 이러한 포함으로 문서를 업데이트하고, Cosmos DB 에 다시 씁니다. 이 방법은 후속 검색을 위해 벡터 포함의 실시간 처리 및 즉시 가용성을 보장합니다.

### 2. **Azure Data Factory 를 사용하여 일괄 처리**

일괄 처리는 Cosmos DB 컨테이너에 이미 있거나 다른 데이터 저장소에 상주하는 문서에 대한 대량 데이터 작업을 위한 효율적인 방법이 될 수 있습니다. Azure Data Factory 를 사용하여 벡터 포함을 생성하는 프로세스를 오케스트레이션할 수 있습니다. Data Factory 는 원본에서 데이터를 추출하고, 포함 생성을 위해 Azure OpenAI 로 보낸 다음, 보강된 데이터를 Cosmos DB 에 쓸 수 있습니다. 이 방법은 실시간 처리가 중요하지 않은 초기 데이터 로드 또는 정기 업데이트에 유용합니다.

### 3. **マイクロ 서비스 아키텍처**

マイクロ 서비스 아키텍처는 관련된 프로세스에 대한 보다 세분화된 제어를 제공하는 솔루션을 제공할 수 있습니다. 포함 생성은 전용 마이크로 서비스 내에 캡슐화할 수 있습니다. 이 서비스는 Azure OpenAI 및 Cosmos DB 와 상호 작용하여 필요에 따라 벡터 포함을 생성합니다. 다른 서비스는 임베딩을 업데이트하거나 검색해야 할 때마다 이 마이크로 서비스를 호출하여 모듈식이고 유지 관리가 용이한 시스템 디자인을 보장합니다.

이러한 패턴을 채택하면 애플리케이션은 Azure OpenAI 및 Cosmos DB for NoSQL의 기능을 사용하여 벡터 포함을 효율적으로 통합하여 검색 및 데이터 분석을 향상시킬 수 있습니다.

## Azure Cosmos DB NoSQL 및 Python을 사용하여 생성 AI 애플리케이션 빌드

차세대 지능형 도우미인 생성 AI 애플리케이션은 컨텍스트 인식 지원을 제공하고 의사 결정 프로세스를 향상하며 복잡한 워크플로를 자동화하여 생산성을 재정의하고 있습니다. NoSQL Python 용 Azure Cosmos DB 벡터 검색을 통해 개발자는 Python의 기능과 방대한 라이브러리 및 도구 구색을 통합하여 정확하고 효율적인 솔루션을 제공하는 고급 생성 AI 애플리케이션을 빌드할 수 있습니다. Azure Cosmos DB의 강력한 벡터 검색 기능과 통합된 Python의 다양한 데이터 조작 기능을 통해 생성 AI 애플리케이션은 복잡한 데이터 쿼리를 처리하고 실시간 인사이트를 효율적으로 제공할 수 있습니다. 또한 벡터 검색은 RAG(Retrieval-Augmented 생성) 패턴을 구현할 때 중요한 역할을 합니다. 이를 통해 AI는 입력 쿼리와 유사성을 기반으로 방대한 모음에서 가장 관련성이 큰 문서를 검색할 수 있으므로 정확하고 상황에 맞는 응답을 생성할 수 있습니다. 이 시너지 효과를 통해 사용자는 전략적 작업에 집중할 수 있으며 AI 생성 AI 애플리케이션은 데이터 처리 및 분석의 많은 부분을 관리할 수 있습니다.

### Python 가상 환경 구성

Python의 가상 환경은 깨끗하고 체계적인 개발 환경을 유지하는 데 매우 중요하며 코딩 환경을 크게 향상시킬 수 있는 다양한 이점을 제공합니다. 각 프로젝트에는 충돌을 방지하고 일관된 개발 워크플로를 보장하는 다른 프로젝트와 격리된 자체 종속성 집합을 가질 수 있습니다. 이러한 격리는 개발 중에 사용되는 정확한 종속성 버전이 유지 관리되므로 예기치 않은 버그 및 비호환성의 위험을 줄여 프로젝트를 프로덕션에 배포할 때 유용합니다. 또한 가상 환경을 사용하면 여러 컴퓨터 및 개발 단계에서 일관된 설정을 제공하여 다른 개발자와 쉽게 공동 작업할 수 있습니다. 가상 환경을 사용하면 패키지 버전을 쉽게 관리하고 종속성 충돌을 방지하며 프로젝트가 원활하게 실행되도록 할 수 있습니다. 이 모범 사례는 안정적이고 신뢰할 수 있는 코딩 환경에 필수적이므로 개발 프로세스를 보다 효율적이고 문제가 발생하기 쉽습니다.

다음 명령과 유사한 명령을 사용하여 Python 가상 환경을 쉽게 만들 수 있습니다. 그러면 명령이 실행되는 디렉터리에 이름이 지정된 .venv 가상 환경이 만들어집니다.

## Bash

```
PYTHON -M VENV .VENV
```

만든 후에는 다음 표에서 OS 및 셸에 대한 적절한 명령을 선택하여 가상 환경을 활성화할 수 있습니다.

## 테이블 확장

플랫폼	Shell	가상 환경을 활성화하는 명령
POSIX	bash/zsh	source .venv/bin/activate
	물고기	source .venv/bin/activate.fish
	csh/tcsh	source .venv/bin/activate.csh
	pwsh	.venv/bin/Activate.ps1
윈도우즈	cmd.exe	.venv\Scripts\activate.bat
	PowerShell	.venv\Scripts\Activate.ps1

가상 환경을 활성화한 후에는 명령을 사용하여 필요한 모든 Python 라이브러리를 pip install 설치할 수 있습니다. 일반적으로 필수 라이브러리 및 해당 버전은 파일에서 requirements.txt 유지 관리됩니다. 이 파일을 사용하면 버전이 프로젝트 내에서 지정되고 유지되므로 환경 간에 모든 라이브러리와 종속성이 유지됩니다. 이러한 파일은 다음 형식으로 종속성을 저장합니다.

## ini

```
AZURE-COSMOS==4.9.0
```

```
AZURE-IDENTITY==1.19.0
```

```
FASTAPI==0.115.5
```

```
OPENAI==1.55.2
```

PYDANTIC==2.10.2

REQUESTS==2.32.3

STREAMLIT==1.40.2

UVICORN==0.32.1

이 파일에 나열된 모든 라이브러리를 추가하는 작업은 다음을 실행하여 명령을 사용하여 pip install 수행할 수도 있습니다.

Bash 복사

```
pip install -r requirements.txt
```

### Entra ID RBAC 를 사용하여 Azure 리소스에 안전하게 액세스

Azure OpenAI 및 Azure Cosmos DB 와 같은 Azure 서비스에 대해 인증하기 위해 Microsoft Entra ID 의 RBAC(Role-Based Access Control)를 활용하면 키 기반 메서드보다 몇 가지 주요 이점이 제공됩니다. Entra ID RBAC 는 사용자 역할에 맞게 조정된 정확한 액세스 제어를 통해 보안을 강화하여 무단 액세스 위험을 효과적으로 줄입니다. 또한 사용자 관리를 간소화하여 관리자가 암호화 키를 배포하고 유지 관리하는 번거로움 없이 권한을 동적으로 할당하고 수정할 수 있습니다. 또한 이 접근 방식은 조직 정책에 부합하고 포괄적인 액세스 모니터링 및 검토를 촉진하여 규정 준수 및 감사 가능성을 향상시킵니다. Entra ID RBAC 는 보안 액세스 관리를 간소화하여 Azure 서비스를 사용하기 위한 보다 효율적이고 확장 가능한 솔루션을 만듭니다.

다음 코드 조각은 Microsoft Entra ID RBAC(역할 기반 액세스 제어) 인증을 사용하여 Azure OpenAI 서비스와 상호 작용하기 위해 클라이언트를 인증하고 구성하는 방법을 보여 줍니다.

Python

```
FROM AZURE.IDENTITY IMPORT DEFAULTAZURECREDENTIAL, GET_BEARER_TOKEN_PROVIDER  
FROM OPENAI IMPORT AZUREOPENAI
```

```

# ENABLE MICROSOFT ENTRA ID RBAC AUTHENTICATION

TOKEN_PROVIDER = GET_BEARER_TOKEN_PROVIDER(
    DEFAULTAZURECREDENTIAL(),
    "HTTPS://COGNITIVESERVICES.AZURE.COM/.DEFAULT"
)

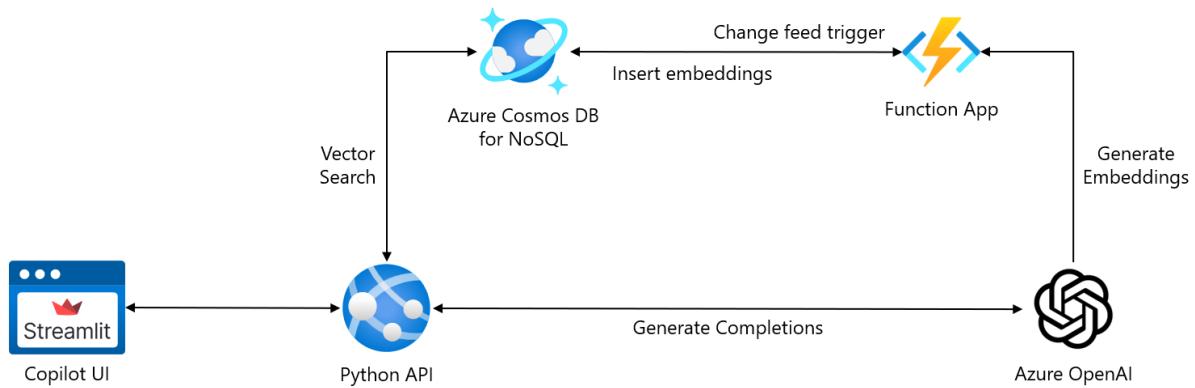
CLIENT = AZUREOPENAI(
    API_VERSION = AZURE_OPENAI_API_VERSION,
    AZURE_ENDPOINT = AZURE_OPENAI_ENDPOINT,
    AZURE_AD_TOKEN_PROVIDER = TOKEN_PROVIDER
)

```

이전 줄은 `token_provider = get_bearer_token_provider(DefaultAzureCredential(), "https://cognitiveservices.azure.com/.default")` Azure Cognitive Services 범위 URL 을 DefaultAzureCredential 사용하여 토큰 공급자를 만듭니다. DefaultAzureCredential 는 인증 프로세스를 처리하고 유ти리티 함수는 `get_bearer_token_provider` Azure 서비스에 대한 액세스 토큰을 가져오는 토큰 공급자를 반환합니다. 그런 다음, 클라이언트는 AzureOpenAI 만든 `token_provider` 항목을 사용하여 Microsoft Entra ID RBAC 를 사용하여 Azure OpenAI 서비스에 대한 요청을 인증합니다.

## 생성 AI 애플리케이션 아키텍처

별도의 프런트 엔드 및 백 엔드 계층을 사용하는 아키텍처는 상당한 확장성을 제공하므로 시간이 지남에 따라 추가 기능이 원활하게 통합됩니다. 초기 생성 AI 애플리케이션이 개발되면 LangChain 오케스트레이션과 같은 새로운 기능을 API 에 쉽게 통합할 수 있습니다.



이 모듈식 디자인을 통해 개발자는 기존 프런트 엔드를 방해하지 않고 고급 기능으로 백엔드를 향상시킬 수 있습니다. 예를 들어 LangChain을 추가하여 복잡한 워크플로를 처리하고 여러 작업을 연결하여 생성 AI 애플리케이션의 기능을 강화할 수 있습니다. 이러한 유연성을 통해 시스템은 확장성과 적응성을 유지하고 향후 발전을 통합할 준비가 되어 있으며 진화하는 사용자 요구를 효율적으로 충족할 수 있습니다.

## Streamlit 를 사용하여 UI 만들기

Streamlit는 대화형 웹 애플리케이션의 신속한 개발을 가능하게 하는 강력한 오픈 소스 Python 라이브러리로, 생성 AI 애플리케이션의 사용자 인터페이스를 빌드하는 데 이상적인 선택입니다. 개발자는 직관적인 API를 사용하여 실시간 상호 작용을 용이하게 하는 동적 응답성 채팅 인터페이스를 빠르게 만들 수 있습니다. 다양한 위젯에 대한 Streamlit의 기본 제공 지원과 인기 있는 데이터 시각화 도구와의 원활한 통합을 통해 채팅 기능을 쉽게 통합할 수 있으므로 사용자가 생성 AI 애플리케이션과 효율적으로 통신할 수 있습니다. Streamlit를 사용하면 사용자에게 친숙하고 매력적인 인터페이스를 생성하는 프로세스를 간소화하여 생성 AI 애플리케이션과 상호 작용하는 사용자에게 원활하고 생산적인 환경을 보장할 수 있습니다.

## Python

```
# CREATE A GENERATIVE AI APPLICATION UI WITH STREAMLIT
```

```
IMPORT STREAMLIT AS ST
```

```
IMPORT REQUESTS
```

```
ST.SET_PAGE_CONFIG(PAGE_TITLE="COSMIC WORKS GENERATIVE AI APPLICATION", LAYOUT="WIDE")
```

```
DEF SEND_MESSAGE_TO_COPILOT(MESSAGE: STR, CHAT_HISTORY: LIST = []) -> STR:  
    """SEND A MESSAGE TO THE GENERATIVE AI APPLICATION CHAT ENDPOINT."""  
  
    TRY:  
  
        API_ENDPOINT = "HTTP://LOCALHOST:8000"  
  
        REQUEST = {"MESSAGE": MESSAGE, "CHAT_HISTORY": CHAT_HISTORY}  
  
        RESPONSE = REQUESTS.POST(F"{API_ENDPOINT}/CHAT", JSON=REQUEST, TIMEOUT=60)  
  
        PRINT('RESPONSE:', RESPONSE.CONTENT)  
  
        RETURN RESPONSE.JSON()  
  
    EXCEPT EXCEPTION AS E:  
  
        ST.ERROR(F"AN ERROR OCCURRED: {E}")  
  
        RETURN ""  
  
  
DEF MAIN():  
    """MAIN FUNCTION FOR THE COSMIC WORKS PRODUCT MANAGEMENT GENERATIVE AI  
APPLICATION UI."""  
  
    ST.WRITE(  
        """  
        # COSMIC WORKS PRODUCT MANAGEMENT GENERATIVE AI APPLICATION  
  
  
        WELCOME TO COSMIC WORKS PRODUCT MANAGEMENT GENERATIVE AI APPLICATION, A TOOL  
FOR MANAGING AND FINDING BICYCLE-RELATED PRODUCTS IN THE COSMIC WORKS SYSTEM.  
    """
```

\*\*ASK THE GENERATIVE AI APPLICATION TO APPLY OR REMOVE A DISCOUNT ON A CATEGORY OF PRODUCTS OR TO FIND PRODUCTS.\*\*

.....

)

IF "MESSAGES" NOT IN ST.SESSION\_STATE:

ST.SESSION\_STATE.MESSAGES = []

# DISPLAY MESSAGE FROM THE HISTORY ON APP RERUN.

FOR MESSAGE IN ST.SESSION\_STATE.MESSAGES:

WITH ST.CHAT\_MESSAGE(MESSAGE["ROLE"]):

ST.MARKDOWN(MESSAGE["CONTENT"])

# REACT TO USER INPUT

IF PROMPT := ST.CHAT\_INPUT("WHAT CAN I HELP YOU WITH TODAY?"):

WITH ST.SPINNER("AWAITING THE GENERATIVE AI APPLICATION'S RESPONSE TO YOUR QUESTION..."):

# DISPLAY USER MESSAGE IN CHAT MESSAGE CONTAINER

WITH ST.CHAT\_MESSAGE("USER"):

ST.MARKDOWN(PROMPT)

# SEND USER MESSAGE TO GENERATIVE AI APPLICATION AND GET RESPONSE

RESPONSE = SEND\_MESSAGE\_TO\_COPILOT(PROMPT, ST.SESSION\_STATE.MESSAGES)

# DISPLAY ASSISTANT RESPONSE IN CHAT MESSAGE CONTAINER

```

WITH ST.CHAT_MESSAGE("ASSISTANT"):

    ST.MARKDOWN(RESPONSE)

# ADD THE CURRENT USER MESSAGE AND ASSISTANT RESPONSE MESSAGES TO CHAT
HISTORY

ST.SESSION_STATE.MESSAGES.APPEND({"ROLE": "USER", "CONTENT": PROMPT})

ST.SESSION_STATE.MESSAGES.APPEND({"ROLE": "ASSISTANT", "CONTENT": RESPONSE})

```

## Python 및 FastAPI 를 사용하여 백 엔드 API 빌드

FastAPI 는 Python 을 사용하여 API 를 개발하기 위한 최신 프레임워크로, 강력한 백 엔드 API 를 만드는 데 적합합니다. Streamlit 를 사용하여 생성 AI 애플리케이션 UI 를 디자인할 때 FastAPI 는 Azure OpenAI 및 Cosmos DB 와 같은 Azure 서비스와의 상호 작용을 처리하기 위한 강력한 백 엔드 엔진 역할을 할 수 있습니다. FastAPI 의 효율적인 요청 처리를 사용하여 Streamlit 프런트 엔드와 Azure 서비스 간의 통신을 가능하게 하는 엔드포인트를 신속하게 빌드할 수 있습니다. 이 설정을 사용하면 생성 AI 애플리케이션에 대한 사용자 쿼리가 원활하게 처리되어 실시간 응답과 효율적인 데이터 관리가 가능합니다. FastAPI 의 단순성과 고성능은 고급 생성 AI 애플리케이션을 지원하는 데 필요한 백 엔드 인프라를 빌드하기 위한 훌륭한 선택입니다.

### Python

```

FROM FASTAPI IMPORT FASTAPI

FROM AZURE.IDENTITY IMPORT DEFAULTAZURECREDENTIAL, GET_BEARER_TOKEN_PROVIDER

FROM OPENAI IMPORT AZUREOPENAI

APP = FASTAPI()

```

```

# ENABLE MICROSOFT ENTRA ID RBAC AUTHENTICATION

TOKEN_PROVIDER = GET_BEARER_TOKEN_PROVIDER(

```

```

DEFAULTAZURECREDENTIAL(),
"HTTPS://COGNITIVESERVICES.AZURE.COM/.DEFAULT"

)

CLIENT = AZUREOPENAI(
    API_VERSION = AZURE_OPENAI_API_VERSION,
    AZURE_ENDPOINT = AZURE_OPENAI_ENDPOINT,
    AZURE_AD_TOKEN_PROVIDER = TOKEN_PROVIDER
)

@app.get("/CHAT")

async def root(message: str, deployment_name: str = "GPT-4O"):

    messages = [{"role": "USER", "content": message}]

    completion = client.chat.completions.create(
        model=deployment_name,
        messages=messages,
    )

    return completion

```

## 함수 호출을 통해 프라이빗 데이터 활용

Azure OpenAI에서 함수 호출을 사용하면 외부 API 또는 도구를 모델의 출력에 직접 원활하게 통합할 수 있습니다. 모델이 관련 요청을 검색하면 필요한 매개 변수를 사용하여 JSON 개체를 생성한 다음, 실행합니다. 결과는 모델에 반환되므로 외부 데이터로 보강된 포괄적인 최종 응답을 제공할 수 있습니다.

함수 호출을 사용하는 경우 코드에서 수행해야 하는 몇 가지 단계가 있습니다. 먼저 작업을 수행하기 위해 호출되는 함수를 만들어야 합니다. 이 메서드는 일반 Python 함수입니다. 예를 들어 다음 함수는 apply\_discount10 개%대해 0.1 과 같은 할인 금액을 허용하고 해당 범주와 일치하는 모든 제품에 해당 할인 금액을 적용합니다.

## Python

```
ASYNC DEF APPLY_DISCOUNT(DISCOUNT: FLOAT, PRODUCT_CATEGORY: STR) -> STR:  
    """APPLY A DISCOUNT TO PRODUCTS IN THE SPECIFIED CATEGORY."""  
  
    # LOAD THE DATABASE  
  
    DATABASE = COSMOS_CLIENT.GET_DATABASE_CLIENT(DATABASE_NAME)  
  
    # RETRIEVE THE CONTAINER  
  
    CONTAINER = DATABASE.GET_CONTAINER_CLIENT(CONTAINER_NAME)  
  
  
    QUERY_RESULTS = CONTAINER.QUERY_ITEMS(  
  
        QUERY = """  
            SELECT * FROM PRODUCTS P WHERE LOWER(P.CATEGORY_NAME) =  
            LOWER(@PRODUCT_CATEGORY)  
  
            """,  
  
        PARAMETERS = [  
  
            {"NAME": "@PRODUCT_CATEGORY", "VALUE": PRODUCT_CATEGORY}  
  
        ]  
  
    )  
  
  
    # APPLY THE DISCOUNT TO THE PRODUCTS  
  
    ASYNC FOR ITEM IN QUERY_RESULTS:  
  
        ITEM['DISCOUNT'] = DISCOUNT
```

```
ITEM['SALE_PRICE'] = ITEM['PRICE'] * (1 - DISCOUNT) IF DISCOUNT > 0 ELSE ITEM['PRICE']
```

```
AWAIT CONTAINER.UPSERT_ITEM(ITEM)
```

```
RETURN F"A {DISCOUNT}% DISCOUNT WAS SUCCESSFULLY APPLIED TO {PRODUCT_CATEGORY}." IF  
DISCOUNT > 0 ELSE F"DISCOUNTS ON {PRODUCT_CATEGORY} REMOVED SUCCESSFULLY."
```

다음으로 LLM(더 큰 언어 모델)이 함수와 상호 작용하는 방법을 이해하는 데 사용하는 JSON 형식 함수 정의를 제공해야 합니다.

JSON

```
{
  "TYPE": "FUNCTION",
  "FUNCTION": {
    "NAME": "APPLY_DISCOUNT",
    "DESCRIPTION": "APPLY A DISCOUNT TO PRODUCTS IN THE SPECIFIED CATEGORY",
    "PARAMETERS": {
      "TYPE": "OBJECT",
      "PROPERTIES": {
        "DISCOUNT": {"TYPE": "NUMBER", "DESCRIPTION": "THE PERCENT DISCOUNT TO APPLY."},
        "PRODUCT_CATEGORY": {"TYPE": "STRING", "DESCRIPTION": "THE CATEGORY OF PRODUCTS TO WHICH THE DISCOUNT SHOULD BE APPLIED."}
      },
      "REQUIRED": ["DISCOUNT", "PRODUCT_CATEGORY"]
    }
  }
}
```

함수 정의를 LLM 에 전달할 수 있는 배열 tools 에 추가해야 합니다. 모델에 사용할 수 있도록 하려는 모든 함수에 대한 정의가 포함되어 있습니다. 예를 들어 다음 tools 배열에는 두 함수 및 에 대한 정의가 포함되어 있습니다 apply\_discountget\_category\_names.

Python

```
# DEFINE FUNCTION CALLING TOOLS

TOOLS = [
    {
        "TYPE": "FUNCTION",
        "FUNCTION": {
            "NAME": "APPLY_DISCOUNT",
            "DESCRIPTION": "APPLY A DISCOUNT TO PRODUCTS IN THE SPECIFIED CATEGORY",
            "PARAMETERS": {
                "TYPE": "OBJECT",
                "PROPERTIES": {
                    "DISCOUNT": {"TYPE": "NUMBER", "DESCRIPTION": "THE PERCENT DISCOUNT TO APPLY."},
                    "PRODUCT_CATEGORY": {"TYPE": "STRING", "DESCRIPTION": "THE CATEGORY OF PRODUCTS TO WHICH THE DISCOUNT SHOULD BE APPLIED."}
                },
                "REQUIRED": ["DISCOUNT", "PRODUCT_CATEGORY"]
            }
        }
    },
},
```

```

{
  "TYPE": "FUNCTION",
  "FUNCTION": {
    "NAME": "GET_CATEGORY_NAMES",
    "DESCRIPTION": "RETRIEVES THE NAMES OF ALL PRODUCT CATEGORIES"
  }
}

]

```

Azure OpenAI에서 함수 호출을 사용하는 경우 LLM을 두 차례 호출해야 합니다. 첫 번째는 LLM이 사용할 "도구"를 결정할 수 있도록 하고, 두 번째 도구는 함수 호출의 출력으로 보강된 프롬프트를 사용하여 완성을 생성합니다.

첫 번째 호출에서는 Azure OpenAI 클라이언트에 도구 배열을 제공한 다음, 해당 호출의 응답에서 도구 출력을 캡처해야 합니다.

Python

```

# FIRST API CALL, PROVIDING THE MODEL TO THE DEFINED FUNCTIONS

RESPONSE = AOAI_CLIENT.CHAT.COMPLETIONS.CREATE(
  MODEL = COMPLETION_DEPLOYMENT_NAME,
  MESSAGES = MESSAGES,
  TOOLS = TOOLS,
  TOOL_CHOICE = "AUTO"
)

```

```

# PROCESS THE MODEL'S RESPONSE

RESPONSE_MESSAGE = RESPONSE.CHOICES[0].MESSAGE

```

```
MESSAGES.APPEND(RESPONSE_MESSAGE)
```

그런 다음 요청된 함수 호출을 사용하여 LLM에서 요청한 함수를 호출하는 응답을 "처리"해야 합니다. 이 처리기를 사용하면 함수를 호출하고 해당 출력을 캡처하여 대화에 기록하고 다음 API 호출에서 LLM에서 사용할 수 있습니다.

Python

```
# HANDLE FUNCTION CALL OUTPUTS

IF RESPONSE_MESSAGE.TOOL_CALLS:

    FOR CALL IN RESPONSE_MESSAGE.TOOL_CALLS:

        IF CALL.FUNCTION.NAME == "APPLY_DISCOUNT":

            FUNC_RESPONSE = APPLY_DISCOUNT(**JSON.LOADS(CALL.FUNCTION.ARGS))

            MESSAGES.APPEND(

                {

                    "ROLE": "TOOL",

                    "TOOL_CALL_ID": CALL.ID,

                    "NAME": CALL.FUNCTION.NAME,

                    "CONTENT": FUNC_RESPONSE

                }

            )

        ELIF CALL.FUNCTION.NAME == "GET_CATEGORY_NAMES":

            FUNC_RESPONSE = GET_CATEGORY_NAMES()

            MESSAGES.APPEND(
```

```
{

    "ROLE": "TOOL",

    "TOOL_CALL_ID": CALL.ID,
```

```

        "NAME": CALL.FUNCTION.NAME,
        "CONTENT": JSON.DUMPS(FUNC_RESPONSE)
    }
)

ELSE:
    PRINT("NO FUNCTION CALLS WERE MADE BY THE MODEL.")

```

마지막으로 LLM에 대한 두 번째 호출을 실행하여 함수 호출의 messages 출력으로 보강된 컬렉션을 전달합니다.

### Python

```

# SECOND API CALL, ASKING THE MODEL TO GENERATE A RESPONSE

FINAL_RESPONSE = AOAIClient.CHAT.COMPLETIONS.CREATE(
    MODEL = COMPLETION_DEPLOYMENT_NAME,
    MESSAGES = MESSAGES
)

# OUTPUT THE COMPLETION RESPONSE

PRINT(FINAL_RESPONSE.CHOICES[0].MESSAGE.CONTENT)

```

### 프롬프트 엔지니어링을 사용하여 생성 AI 애플리케이션에 가상 사용자 제공

프롬프트 엔지니어링은 인공 지능 및 자연어 처리에서 중요한 기술입니다. 정확하고 관련 있는 결과를 달성하기 위해 AI의 응답을 조정하는 특정하고 잘 구조화된 입력(프롬프트)을 만들어 원하는 출력을 생성하는 AI 모델을 안내합니다.

효과적인 프롬프트 엔지니어링을 사용하려면 AI 모델의 기능과 제한 사항 및 언어의 뉴앙스를 이해해야 합니다. 개발자는 프롬프트 내에서 명확한 지침, 컨텍스트 및 예제를

제공하여 AI에 영향을 미침으로써 고품질 콘텐츠를 생성하거나 복잡한 문제를 해결하거나 특정 작업을 수행할 수 있습니다. 이 방법은 AI가 다양한 쿼리를 이해하고 응답하는 기능을 향상시켜 상호 작용을 더 유용하고 매력적으로 만듭니다.

생성 AI 애플리케이션을 빌드할 때 프롬프트 엔지니어링을 사용하면 도우미에 대한 "가상 사용자"를 정의할 수 있습니다. 가상 사용자는 입력을 처리하고 출력을 만드는 방법을 설명하는 생성 AI 애플리케이션 사용자와 상호 작용하는 방법에 대해 LLM에 지시합니다. 가상 사용자를 만드는 작업은 Azure OpenAI 클라이언트 및 기본 언어 모델에서 사용하는 메시지 컬렉션에 추가하는 시스템 프롬프트를 사용하여 수행됩니다.

## Python

```
# DEFINE THE SYSTEM PROMPT THAT CONTAINS THE ASSISTANT'S PERSONA.
```

```
SYSTEM_PROMPT = """
```

```
YOU ARE AN INTELLIGENT GENERATIVE AI APPLICATION FOR COSMIC WORKS DESIGNED TO HELP USERS  
MANAGE AND FIND BICYCLE-RELATED PRODUCTS.
```

```
YOU ARE HELPFUL, FRIENDLY, AND KNOWLEDGEABLE, BUT CAN ONLY ANSWER QUESTIONS ABOUT  
COSMIC WORKS PRODUCTS.
```

IF ASKED TO APPLY A DISCOUNT:

- APPLY THE SPECIFIED DISCOUNT TO ALL PRODUCTS IN THE SPECIFIED CATEGORY. IF THE USER DID NOT PROVIDE YOU WITH A DISCOUNT PERCENTAGE AND A PRODUCT CATEGORY, PROMPT THEM FOR THE DETAILS YOU NEED TO APPLY A DISCOUNT.

- DISCOUNT AMOUNTS SHOULD BE SPECIFIED AS A DECIMAL VALUE (E.G., 0.1 FOR 10% OFF).

IF ASKED TO REMOVE DISCOUNTS FROM A CATEGORY:

- REMOVE ANY DISCOUNTS APPLIED TO PRODUCTS IN THE SPECIFIED CATEGORY BY SETTING THE DISCOUNT VALUE TO 0.

WHEN ASKED TO PROVIDE A LIST OF PRODUCTS, YOU SHOULD:

- PROVIDE AT LEAST 3 CANDIDATE PRODUCTS UNLESS THE USER ASKS FOR MORE OR LESS, THEN USE THAT NUMBER. ALWAYS INCLUDE EACH PRODUCT'S NAME, DESCRIPTION, PRICE, AND SKU. IF THE PRODUCT HAS A DISCOUNT, INCLUDE IT AS A PERCENTAGE AND THE ASSOCIATED SALE PRICE.

# PROVIDE THE GENERATIVE AI APPLICATION WITH A PERSONA USING THE SYSTEM PROMPT.

```
MESSAGES = [{ "ROLE": "SYSTEM", "CONTENT": SYSTEM_PROMPT }]
```

## 생성 AI 애플리케이션에서 NoSQL 용 Azure Cosmos DB 를 사용하여 벡터 검색 수행

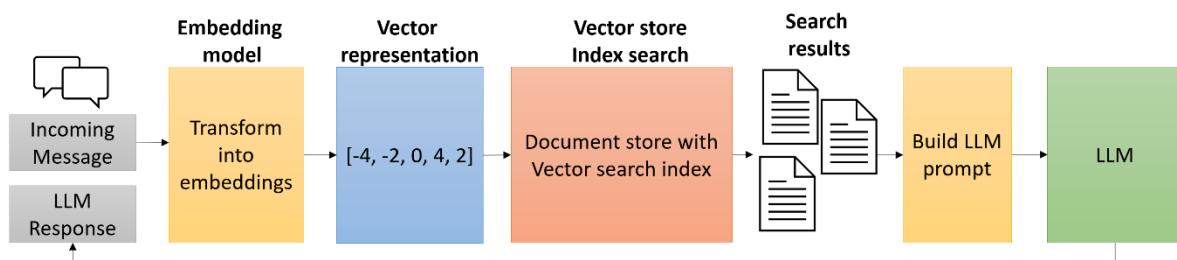
완료됨 100 XP

- 4 분

Azure Cosmos DB for NoSQL의 벡터 검색 기능을 Python 기반 생성 AI 애플리케이션에 통합하면 유사성 검색을 효율적으로 수행하는 생성 AI 애플리케이션의 기능이 크게 향상됩니다. 벡터 검색을 사용하면 벡터 포함을 기반으로 의미상 유사한 항목을 검색할 수 있으므로 생성 AI 애플리케이션의 자연어 처리 작업에 적합합니다. 생성 AI 애플리케이션은 벡터 검색을 사용하여 대규모 데이터 세트의 관련 정보를 신속하게 찾아 검색하여 정확하고 상황에 맞는 데이터로 응답을 보강할 수 있습니다. 이 기능은 보다 정확하고 의미 있는 지원을 제공하여 사용자 환경을 향상시켜 궁극적으로 생성 AI 애플리케이션을 보다 지능적이고 효과적으로 만듭니다.

### 벡터 검색이란?

벡터 검색은 특정 속성 또는 필드에서 정확히 일치하는 항목이 아닌 데이터 특성을 기반으로 항목을 찾을 수 있도록 하는 기술입니다. 정확한 일치를 요구하는 대신 벡터 검색을 사용하면 벡터 표현을 기반으로 일치 항목을 식별할 수 있습니다. 이 기술은 유사성 검색을 수행할 때 유용하며 텍스트의 큰 블록 내에서 정보를 검색해야 하는 애플리케이션에서 유용합니다.



이는 독점 데이터를 LLM(대규모 언어 모델) 응답에 안전하고 효율적으로 통합하여 생성 AI 애플리케이션이 중요한 정보의 기밀성과 무결성을 유지하면서 정확한 도메인별 답변을 제공할 수 있도록 함으로써 생성 AI 애플리케이션에서 자주 사용하는 RAG(검색 보강 세대) 패턴의 구현에서 중요한 역할을 합니다.

## 벡터 검색과 관련된 단계 이해

NoSQL 용 Azure Cosmos DB 를 사용하여 벡터 검색을 실행하려면 다음 단계가 포함됩니다.

1. 유사성 검색을 수행할 필드에 대한 벡터 포함을 만들고 저장합니다.
2. 컨테이너의 벡터 포함 정책에 벡터 포함 경로를 지정합니다.
3. 컨테이너의 인덱싱 정책에 원하는 벡터 인덱스를 포함합니다.
4. 컨테이너를 벡터 포함이 포함된 문서로 채웁니다.
5. Azure OpenAI 또는 다른 서비스를 사용하여 검색 쿼리를 나타내는 포함 항목을 생성합니다.
6. 함수를 VectorDistance 사용하여 쿼리를 실행하여 검색 쿼리 포함의 유사성을 Cosmos DB 컨테이너에 저장된 벡터의 포함과 비교합니다.
- 7.

## VectorDistance 함수를 사용하여 벡터 검색 수행

NoSQL 용 Azure Cosmos DB 의 함수는 VectorDistance 코사인 유사성, 점 제품 또는 유clidean 거리와 같은 메트릭을 사용하여 거리를 계산하여 두 벡터 간의 유사성을 측정합니다. 이 함수는 자연어 처리 또는 권장 사항 시스템과 같이 빠르고 정확한 유사성 검색이 필요한 애플리케이션에 매우 중요합니다. 활용하면 VectorDistance 고차원 벡터 쿼리를 효율적으로 처리하여 AI 기반 애플리케이션의 관련성과 성능을 크게 향상시킬 수 있습니다.

예를 들어 제품 설명을 확인하여 자전거 페달에 대한 제품을 검색하려는 경우를 가정해 보겠습니다. 먼저 쿼리 텍스트에 대한 포함을 생성해야 합니다. 이 경우 쿼리 텍스트 "클립인 페달"에 대한 포함을 만들 수 있습니다. 벡터 검색 쿼리의 함수에 검색 쿼리 텍스트 VectorDistance 포함을 제공하면 다음 쿼리와 같이 쿼리와 유사한 제품을 식별하고 검색할 수 있습니다.

## SQL

```
SELECT TOP 5 C.NAME, C.DESCRIPTION, VECTORDISTANCE(C.EMBEDDING, [1,2,3]) AS  
SIMILARITYSCORE  
  
FROM C  
  
ORDER BY VECTORDISTANCE(C.EMBEDDING, [1,2,3])
```

이전 예제에서는 유사성 점수를 별칭 SimilarityScore 으로 투영하고 가장 유사한 쿼리에서 가장 유사하게 정렬하는 NoSQL 쿼리를 보여 줍니다.

**!** 벡터 검색을 TOP N 수행할 때는 항상 쿼리에 SELECT 절을 사용해야 합니다. 벡터 검색이 없으면 더 많은 일치 항목을 반환하려고 시도하므로 쿼리 비용이 더 많이 들고 필요한 것보다 대기 시간이 더 깁니다.

생성 AI 애플리케이션에 대한 Python 코드에서 벡터 검색을 구현하는 것은 다음과 유사합니다.

## Python 복사

```
# CREATE A COSMOS DB CLIENT  
  
COSMOS_CLIENT = COSMOSCLIENT(URL=AZURE_COSMOSDB_ENDPOINT,  
CREDENTIAL=CREDENTIAL)  
  
# LOAD THE DATABASE  
  
DATABASE = COSMOS_CLIENT.GET_DATABASE_CLIENT(DATABASE_NAME)  
  
# RETRIEVE THE CONTAINER  
  
CONTAINER = DATABASE.GET_CONTAINER_CLIENT(CONTAINER_NAME)  
  
  
# FUNCTION FOR GENERATING EMBEDDINGS USING AZURE OPENAI'S TEXT-EMBEDDING-3-SMALL  
MODEL  
  
DEF GENERATE_EMBEDDINGS(TEXT: STR):  
  
    CLIENT = AZUREOPENAI(  
  
        API_VERSION = AZURE_OPENAI_API_VERSION,
```

```

        AZURE_ENDPOINT = AZURE_OPENAI_ENDPOINT,
        AZURE_AD_TOKEN_PROVIDER = TOKEN_PROVIDER
    )

RESPONSE = CLIENT.EMBEDDINGS.CREATE(INPUT = TEXT, MODEL = "TEXT-EMBEDDING-3-SMALL")
RETURN RESPONSE.DATA[0].EMBEDDING

DEF VECTOR_SEARCH(QUERY_EMBEDDING: LIST, NUM_RESULTS: INT = 3, SIMILARITY_SCORE: FLOAT =
0.25):
    """SEARCH FOR SIMILAR PRODUCT VECTORS IN AZURE COSMOS DB"""

RESULTS = CONTAINER.QUERY_ITEMS(
    QUERY = """
        SELECT TOP @NUM_RESULTS P.NAME, P.DESCRIPTION, P.SKU, P.PRICE, P.DISCOUNT,
P.SALE_PRICE, VECTORDISTANCE(P.EMBEDDING, @QUERY_EMBEDDING) AS SIMILARITY_SCORE
        FROM PRODUCTS P
        WHERE VECTORDISTANCE(P.EMBEDDING, @QUERY_EMBEDDING) > @SIMILARITY_SCORE
        ORDER BY VECTORDISTANCE(P.EMBEDDING, @QUERY_EMBEDDING)
    """
    ,
    PARAMETERS = [
        {"NAME": "@QUERY_EMBEDDING", "VALUE": QUERY_EMBEDDING},
        {"NAME": "@NUM_RESULTS", "VALUE": NUM_RESULTS},
        {"NAME": "@SIMILARITY_SCORE", "VALUE": SIMILARITY_SCORE}
    ],
    ENABLE_CROSS_PARTITION_QUERY = TRUE
)

RESULTS = LIST(RESULTS)

```

```
    FORMATTED_RESULTS = [ {'SIMILARITY_SCORE': RESULT.pop('SIMILARITY_SCORE'), 'PRODUCT': RESULT} for RESULT in RESULTS]

    RETURN FORMATTED_RESULTS

# GET EMBEDDINGS FOR SEARCH QUERY TEXT

QUERY_EMBEDDING = GENERATE_EMBEDDINGS(QUERY_TEXT)

# PERFORM A VECTOR SEARCH
```

## Python 생성 AI 애플리케이션에서 LangChain 오케스트레이션을 통합하여 효율성 및 코드 유지 관리 효율성 향상

완료됨 100 XP

- 5 분

Azure Cosmos DB는 의미 체계 커널 및 LangChain과 같은 주요 LLM(대규모 언어 모델) 오케스트레이션 패키지와 원활하게 통합되므로 애플리케이션 내에서 고급 AI 기능의 기능을 활용할 수 있습니다. 이러한 오케스트레이션 패키지는 LLM, 포함 모델 및 데이터베이스의 관리 및 사용을 간소화하여 고급 AI 생성 AI 애플리케이션을 더욱 쉽게 개발할 수 있습니다.

### LangChain 오케스트레이션 통합

LangChain은 복잡하고 역동적인 AI 애플리케이션을 만들기 위해 여러 AI 모델 및 도구의 통합 및 조정을 향상시키는 강력한 도구입니다. LangChain은 오케스트레이션 기능을 사용하여 다양한 언어 모델, API 및 사용자 지정 구성 요소를 통합 워크플로로 원활하게 결합할 수 있습니다. 이 오케스트레이션을 통해 각 요소가 효율적으로 함께 작동하여 자연어 이해 및 생성부터 정보 검색 및 데이터 분석에 이르기까지 다양한 작업을 수행할 수 있는 정교한 애플리케이션을 만들 수 있습니다.

LangChain의 오케스트레이션 기능은 Python 및 NoSQL용 Azure Cosmos DB를 사용하여 생성 AI 애플리케이션을 빌드할 때 유용합니다. 생성 AI 애플리케이션은 정확하고 상황에 맞는 응답을 제공하기 위해 종종 NLP(자연어 처리) 모델, 지식 검색

시스템 및 사용자 지정 논리를 결합해야 합니다. LangChain은 다양한 NLP 모델 및 API를 오케스트레이션하여 이 프로세스를 용이하게 하여 생성 AI 애플리케이션이 사용자 쿼리에 대한 응답을 효과적으로 이해하고 생성할 수 있도록 합니다.

또한 Azure Cosmos DB for NoSQL을 LangChain과 통합하면 짧은 대기 시간으로 대량의 데이터를 처리할 수 있는 확장 가능하고 유연한 데이터베이스 솔루션을 제공합니다. Cosmos DB 벡터 검색 기능을 사용하면 데이터의 의미 체계 유사성을 기반으로 관련 정보를 고성능으로 검색할 수 있으며 이는 NLP 애플리케이션에 특히 유용합니다. 즉, 생성 AI 애플리케이션은 대규모 데이터 세트에 대해 정교한 검색을 수행하여 사용자 쿼리에 대한 컨텍스트 관련 정보를 검색할 수 있습니다.

LangChain의 오케스트레이션을 통해 Azure Cosmos DB의 벡터 검색 데이터가 AI 모델과 원활하게 통합되어 생성 AI 애플리케이션이 시기 적절하게 정확한 응답을 제공할 수 있습니다. LangChain의 오케스트레이션과 Cosmos DB의 고급 검색 기능을 결합하여 생성 AI 애플리케이션이 사용자를 보다 효과적으로 이해하고 상호 작용하는 기능을 향상시킵니다.

## LangChain을 사용하여 RAG(검색 보강된 생성)

RAG(검색 보강 생성)은 검색 및 생성을 결합하여 AI 애플리케이션의 성능과 정확도를 향상시키는 패턴으로, NoSQL의 벡터 검색 기능을 위한 LangChain 및 Azure Cosmos DB와 통합될 때 강력한 도구입니다. LangChain의 오케스트레이션 기능을 사용하여 RAG는 관련 정보 검색을 AI 모델의 생성 능력과 원활하게 결합할 수 있습니다. Azure Cosmos DB의 벡터 검색 기능은 대규모 데이터 세트의 의미상 유사한 데이터를 고성능으로 검색할 수 있도록 하여 이 프로세스에서 매우 중요합니다. 이 검색 프로세스는 생성 AI 애플리케이션이 가장 관련성이 높은 정보에 빠르고 효율적으로 액세스하고 활용할 수 있도록 합니다. 사용자가 쿼리를 발생시키는 경우 RAG 모델은 벡터 검색을 사용하여 Cosmos DB에서 상황에 맞는 데이터를 검색한 다음 해당 데이터를 기반으로 포괄적인 일관된 응답을 생성할 수 있습니다. 검색 및 생성의 이러한 조합은 생성 AI 애플리케이션이 정확하고 컨텍스트 인식적인 답변을 제공하는 기능을 크게 향상시켜 보다 강력하고 사용자 친화적인 환경을 제공합니다.

## LangChain의 함수 호출 및 도구 이해

LangChain에서 함수 호출은 Python에서 직접 Azure OpenAI 클라이언트를 사용하는 것에 비해 보다 구조적이고 유연한 접근 방식을 제공합니다. LangChain에서 함수를

쉽게 재사용하고 유지 관리할 수 있는 모듈식 구성 요소로 정의하고 관리할 수 있습니다. 이 방법을 사용하면 각 함수가 특정 작업을 캡슐화하여 복잡성을 줄이고 개발 프로세스를 보다 효율적으로 만드는 보다 체계적인 코드를 사용할 수 있습니다.

Python에서 Azure OpenAI 클라이언트를 사용하는 경우 함수 호출은 일반적으로 직접 API 상호 작용으로 제한됩니다. 여전히 복잡한 워크플로를 빌드할 수 있지만, 비동기 작업을 수동으로 오케스트레이션하고 처리해야 하는 경우가 많으므로 애플리케이션이 증가함에 따라 번거롭고 유지 관리하기가 더 어려워질 수 있습니다.

LangChain의 도구는 함수 호출을 향상시키는 데 중요한 역할을 합니다. 다양한 기본 제공 도구와 외부 도구를 통합하는 기능을 통해 LangChain을 사용하면 함수가 도구를 호출하여 데이터 검색, 처리 또는 변환과 같은 특정 작업을 수행할 수 있는 정교한 파이프라인을 만들 수 있습니다. 이러한 도구는 조건부 또는 병렬로 작동하도록 구성하여 애플리케이션의 성능을 더욱 최적화할 수 있습니다. 또한 LangChain의 도구는 함수와 도구를 격리하여 오류 처리 및 디버깅을 간소화하여 문제를 보다 쉽게 식별하고 해결할 수 있도록 합니다.