aws

개발자 가이드

Amazon Comprehend

Amazon Comprehend: 개발자 가이드

Table of Contents

Amazon Comprehend란?

Amazon Comprehend는 자연어 처리를 사용하여 문서 내용에 대한 인사이트를 추출합니다. 문서에 있는 개체, 핵심 문구, 언어, 감정 및 기타 일반적인 요소를 인식하여 인사이트를 개발합니다. Amazon Comprehend를 사용하면 문서 구조에 대한 이해를 바탕으로 새로운 상품을 생성할 수 있습니다. 예를 들어 Amazon Comprehend를 사용하면 소셜 네트워킹 피드에서 제품에 대한 멘션을 검색하거나 전체 문서 리포지토리에서 주요 문구를 스캔할 수 있습니다.

Amazon Comprehend 콘솔 또는 Amazon Comprehend API를 사용하여 Amazon Comprehend 문서 분석 기능에 액세스할 수 있습니다. 소규모 워크로드에 대해 실시간 분석을 실행하거나 대규모 문서 세트에 대해 비동기 분석 작업을 시작할 수 있습니다. Amazon Comprehend에서 제공하는 사전 학습된 모델을 사용하거나 분류 및 개체 인식을 위해 자체 사용자 정의 모델을 교육할 수 있습니다.

Amazon Comprehend는 분석 모델의 품질을 지속적으로 개선하기 위해 사용자의 콘텐츠를 저장할 수 있습니다. 자세한 내용은 Amazon Comprehend 요금을 참조하세요.

모든 Amazon Comprehend 기능은 UTF-8 텍스트 문서를 입력으로 받아들입니다. 또한 사용자 정의 분류 및 사용자 정의 개체 인식은 이미지 파일, PDF 파일 및 Word 파일을 입력으로 받아들입니다.

Amazon Comprehend는 특정 기능에 따라 다양한 언어로 문서를 검사하고 분석할 수 있습니다. 자세한 내용은 Amazon Comprehend에서 지원되는 언어를 참조하십시오. Amazon Comprehend의 지배적 언어 기능은 문서를 검사하고 훨씬 더 다양한 언어에 사용할 수 있는 지배적 언어를 결정할 수 있습니다.

주제

- Amazon Comprehend 인사이트
- Amazon Comprehend 사용자 정의
- 플라이 휠
- 문서 클러스터링 (주제 모델링)
- 예제
- 이점
- Amazon Comprehend 요금
- Amazon Comprehend를 처음 사용하십니까?

Amazon Comprehend 인사이트

Amazon Comprehend는 사전 학습된 모델을 사용하여 문서 또는 문서 세트를 검토 및 분석하여 이에 대한 인사이트를 수집합니다. 대량의 텍스트를 기반으로 이 모델을 지속적으로 학습시키기 때문에 학습 데이터를 제공할 필요가 없습니다.

Amazon Comprehend는 다음과 같은 유형의 인사이트를 분석합니다.

- 개체 — 문서에 포함된 사람, 장소, 항목 및 위치의 이름을 가리키는 참조입니다.
- 핵심 문구 — 문서에 나타나는 문구. 예를 들어 농구 경기에 관한 문서에는 팀 이름, 경기장 이름, 최종 점수를 보낼 수 있습니다.
- 개인 식별 정보 (PII) — 주소, 은행 계좌 번호, 전화번호 등 개인을 식별할 수 있는 개인 데이터입니다.
- 언어 — 문서의 주요 언어입니다.
- 감성 — 문서의 주된 감정으로 긍정적, 중립적, 부정적 또는 혼합적일 수 있습니다.
- 대상 감성 — 문서 내 특정 개체와 관련된 감정. 각 개체에 대한 감정은 긍정적, 부정적, 중립적 또는 혼합적일 수 있습니다.
- 구문 — 문서 내 각 단어의 품사.

자세한 내용은 인사이트을 참조하십시오.

Amazon Comprehend 사용자 정의

기계 교육 기반 NLP 솔루션을 구축하는 데 필요한 기술이 없어도 특정 요구 사항에 맞게 Amazon Comprehend를 사용자 정의할 수 있습니다. Amazon Comprehend 사용자 정의는 자동 기계 학습 또는 AutoML을 사용하여 이미 보유한 데이터로 사용자 대신 사용자 정의 NLP 모델을 구축합니다.

사용자 정의 분류 — 사용자 정의 분류 모델(분류기)을 생성하여 문서를 고유한 범주로 구성할 수 있습니다.

사용자 지정 개체 인식 – 텍스트를 분석하여 특정 용어와 명사 기반 구문을 분석할 수 있는 사용자 지정 개체 인식 모델(인식기)을 만들 수 있습니다.

자세한 내용은 Amazon Comprehend 사용자 정의을 참조하십시오.

플라이 휠

플라이 휠을 사용하면 장기간 사용자 정의 모델 버전을 학습시키고 관리하는 프로세스를 단순화할 수 있습니다. 플라이 휠은 새 버전의 모델의 학습 및 평가와 관련된 작업을 오케스트레이션 하는 데 도움이 됩니다. 플라이 휠은 사용자 정의 분류 및 사용자 정의 개체 인식을 위한 일반 텍스트 사용자 정의 모델을 지원합니다. 자세한 내용은 플라이 휠을 참조하십시오.

문서 클러스터링 (주제 모델링)

또한 Amazon Comprehend를 사용하여 문서 코퍼스를 검사하여 문서 내의 유사한 키워드를 기반으로 문서를 구성할 수 있습니다. 문서 클러스터링 (주제 모델링) 은 대량의 문서를 단어 빈도에 따라 비슷한 주제 또는 클러스터로 구성하는 데 유용합니다. 자세한 내용은 주제 모델링을 참조하십시오.

예제

다음은 Amazon Comprehend 작업을 애플리케이션에서 사용할 수 있는 방법을 보여주는 예제입니다.

Example 1: 주제에 관한 문서 찾기

Amazon Comprehend 주제 모델링을 사용하여 특정 주제에 대한 문서를 찾습니다. 문서 세트를 스캔하여 논의된 주제를 파악하고 각 주제와 관련된 문서를 찾습니다. Amazon Comprehend가 문서 세트에서 반환해야 하는 주제 수를 지정할 수 있습니다.

Example 2: 고객이 사용자의 상품에 대해 어떻게 생각하는지 알아봅니다.

회사에서 카탈로그를 게시하면 Amazon Comprehend가 상품에 대한 고객의 생각을 알려드립니다. 각 고객 의견을 `DetectSentiment` 운영 팀에 보내면 고객이 제품에 대해 긍정적으로 느끼는지, 부정적인지, 중립적인지 또는 혼동감을 느끼는지 알 수 있습니다.

Example 3: 고객에게 중요한 것이 무엇인지 알아봅니다.

Amazon Comprehend 주제 모델링을 사용하여 고객이 포럼 및 게시판에서 이야기하는 주제를 찾은 다음, 개체 탐지를 사용하여 해당 주제와 관련된 사람, 장소 및 사물을 파악할 수 있습니다. 감성 분석을 사용하여 고객이 주제에 대해 어떻게 생각하는지 파악할 수 있습니다.

이점

Amazon Comprehend를 사용할 때 얻을 수 있는 이점은 다음과 같습니다.

- 강력한 자연어 처리를 앱에 통합 — Amazon Comprehend는 간단한 API로 강력하고 정확한 자연어 처리를 사용할 수 있게 함으로써 애플리케이션에 텍스트 분석 기능을 구축해야 하는 복잡함이 사라집니다. Amazon Comprehend가 제공하는 인사이트를 활용하는 데 텍스트 분석 전문 지식이 필요하지 않습니다.
- 딥 러닝 기반 자연어 처리 — Amazon Comprehend는 딥 러닝 기술을 사용하여 텍스트를 정확하게 분석합니다. 정확도를 높이기 위해 여러 도메인에 걸친 새로운 데이터를 사용하여 모델을 지속적으로 학습시킵니다.
- 확장 가능한 자연어 처리 — Amazon Comprehend를 사용하면 수 백만 개의 문서를 분석하여 문서에 포함된 인사이트를 발견할 수 있습니다.
- 다른 AWS 서비스와 통합 - Amazon Comprehend는 Amazon S3 AWS KMS, 및와 같은 다른 AWS 서비스와 원활하게 작동하도록 설계되었습니다 AWS Lambda. 문서를 Amazon S3에 저장하거나 Firehose를 사용하여 실시간 데이터를 분석합니다. AWS Identity and Access Management (IAM)에 대한 지원을 통해 Amazon Comprehend 작업에 대한 액세스를 쉽게 제어할 수 있습니다. IAM을 사용하면 사용자 및 그룹이 사용자 개발자와 최종 사용자에게 적절하게 액세스할 권한을 부여할 수 있습니다.
- 출력 결과 및 볼륨 데이터 암호화 — Amazon S3에서는 이미 입력 문서를 암호화할 수 있으며 Amazon Comprehend는 이를 더욱 확장합니다. 자체 KMS 키를 사용하여 작업의 출력 결과와 분석 작업을 처리하는 컴퓨팅 인스턴스에 연결된 스토리지 볼륨의 데이터를 암호화할 수 있습니다. 그 결과 보안이 크게 강화됩니다.
- 저렴한 비용 — Amazon Comprehend는 최소 수수료나 사전 약정이 없습니다. 사용자가 분석한 문서와 교육한 사용자 정의 모델에 대한 비용만 지불합니다.

Amazon Comprehend 요금

Amazon Comprehend의 경우 사용자가 사용한 리소스에 대해서만 비용을 지불합니다. 신규 AWS 고객은 Amazon Comprehend를 무료로 시작할 수 있습니다. 자세한 내용은 AWS 무료 사용 티어를 참조하세요.

실시간 또는 비동기 분석 작업 실행에는 사용 요금이 부과됩니다. 사용자는 사용자 정의 모델을 학습시키는 데 비용을 지불하고 사용자 정의 모델 관리에 대한 비용을 지불합니다. 사용자 정의 모델을 사용한 실시간 요청의 경우, 엔드포인트를 시작한 시점부터 엔드포인트를 삭제할 때까지의 엔드포인트 요금을 지불합니다. 플라이 휠을 사용해도 추가 요금이 부과되지 않습니다. 하지만 플라이 휠 반복을 실행하면 새 모델 버전 학습 및 모델 데이터 저장 표준 요금이 발생합니다.

요금 및 추가 세부 정보는 Amazon Comprehend 요금을 참조하세요.

Amazon Comprehend를 처음 사용하십니까?

Amazon Comprehend를 처음 사용하신다면 다음 섹션을 순서대로 읽어보기를 권장합니다.

1. 작동 방법— 이 섹션에서는 Amazon Comprehend 개념을 소개합니다.
2. 설정— 이 섹션에서는 계정을 생성하고 AWS CLI를 설정합니다.
3. Amazon Comprehend 시작하기— 이 섹션에서는 Amazon Comprehend 분석 작업을 실행합니다.
4. 자습서: Amazon Comprehend 고객 리뷰를 통한 인사이트 분석— 이 섹션에서는 감성 및 개체 분석을 수행하고 결과를 시각화합니다.
5. Amazon Comprehend API 참조 – — Amazon Comprehend 작업에 대한 참조 문서입니다.

AWS 는 Amazon Comprehend 서비스에 대해 알아보기 위한 다음 리소스를 제공합니다.

- AWS 기계 학습 블로그에 Amazon Comprehend에 대한 유용한 기사가 있습니다.
- Amazon Comprehend 리소스는 Amazon Comprehend에 대한 유용한 동영상 및 자습서를 제공합니다.

작동 방법

Amazon Comprehend는 사전 학습된 모델을 사용하여 문서 또는 문서 세트에 대한 인사이트를 수집합니다. 대량의 텍스트를 기반으로 이 모델을 지속적으로 학습시키기 때문에 학습 데이터를 제공할 필요가 없습니다.

Amazon Comprehend를 사용하여 사용자 지정 분류 및 사용자 지정 개체 인식을 위한 고유의 사용자 지정 모델을 구축할 수 있습니다. 플라이 휠을 사용하여 사용자 지정 모델을 관리할 수 있습니다.

Amazon Comprehend는 내장 모델을 사용하여 주제 모델링을 제공합니다. 주제 모델링은 문서 모음을 검사하고 문서 내의 유사한 키워드를 기반으로 문서를 구성합니다.

Amazon Comprehend는 동기 및 비동기 문서 처리 모드를 제공합니다. 문서 1개 또는 최대 25개의 문서 배치를 처리하려면 동기 모드를 사용하십시오. 비동기 작업을 사용하면 더 많은 문서를 처리할 수 있습니다.

Amazon Comprehend는 AWS Key Management Service (AWS KMS)와 함께 작동하여 데이터에 대한 향상된 암호화를 제공합니다. 자세한 내용은 Amazon Comprehend에서의 KMS 암호화 단원을 참조하십시오.

주요 개념

- 인사이트
- Amazon Comprehend 사용자 정의
- 주제 모델링
- 문서 처리 모드

인사이트

Amazon Comprehend는 문서 또는 문서 세트를 분석하여 이에 대한 인사이트를 수집할 수 있습니다. Amazon Comprehend가 문서에 대해 개발하는 몇 가지 인사이트는 다음과 같습니다.

- 개체 – Amazon Comprehend는 문서에 식별된 사람, 장소, 위치 등의 개체 목록을 반환합니다.
- 이벤트 – Amazon Comprehend는 특정 유형의 이벤트 및 관련 세부 정보를 감지합니다.
- 핵심 문구 – Amazon Comprehend는 문서에 나타나는 핵심 문구를 추출합니다. 예를 들어 농구 경기에 관한 문서에는 팀 이름, 경기장 이름, 최종 점수를 보낼 수 있습니다.

- 개인 식별 정보(PII) – Amazon Comprehend는 문서를 분석하여 주소, 은행 계좌 번호, 전화번호와 같이 개인을 식별하는 개인 정보를 감지합니다.
- 지배적 언어 – Amazon Comprehend는 문서에서 주로 사용되는 언어를 식별합니다. Amazon Comprehend는 100개의 언어를 식별할 수 있습니다.
- 감성 – Amazon Comprehend는 문서의 지배적인 감성을 판단합니다. 감성은 긍정적, 중립적, 부정적 또는 혼합적일 수 있습니다.
- 표적 감성 – Amazon Comprehend는 문서에 언급된 특정 개체의 감성을 파악합니다. 각 멘션의 감성은 긍정적, 중립적, 부정적 또는 혼합적일 수 있습니다.
- 구문 분석 – Amazon Comprehend는 문서의 각 단어를 분석하여 단어의 품사를 결정합니다. 예를 들어 “It is raining today in Seattle”이라는 문장에서 “it”은 대명사로, “raining”은 동사로, “Seattle”은 고유명사로 식별됩니다.

개체

개체는 사람, 장소, 상품 등 실제 개체의 고유한 이름에 대한 텍스트 참조이며 동시에 날짜 및 수량과 같은 측정값에 대한 정확한 참조입니다.

예를 들어 “존이 2012년에 1313 모킹버드 레인으로 이사했다.”라는 텍스트에서 ‘존’은 `PERSON`으로, ‘1313 모킹버드 레인’은 `LOCATION`로, ‘2012"는 `DATE`로 인식될 수 있습니다.

또한 각 개체에는 Amazon Comprehend가 개체 유형을 올바르게 감지했다는 신뢰도 수준을 나타내는 점수도 있습니다. 점수가 낮은 개체를 필터링하여 잘못된 감지를 사용할 위험을 줄일 수 있습니다.

다음 표는 리소스 유형의 목록입니다.

유형	설명
상업_품목	브랜드 제품
날짜	전체 날짜(예: 2017년 11월 25일), 요일(화요일), 월(5월) 또는 시간(오전 8:30)
이벤트	축제, 콘서트, 선거 등과 같은 행사
위치	국가, 도시, 호수, 건물 등 특정 위치
조직	정부, 회사, 종교, 스포츠 팀 등과 같은 대규모 조직

유형	설명
기타	다른 개체 범주에 속하지 않는 개체
개인	개인, 집단, 별명, 가상 인물
수량	정량화된 금액(예: 통화, 백분율, 숫자, 바이트 등)
제목	영화, 책, 노래 등과 같은 모든 창작물 또는 창작 작품에 부여되는 공식 명칭

Amazon Comprehend에서 지원하는 모든 기본 언어를 사용하여 개체 감지 작업을 수행할 수 있습니다. 여기에는 사전 정의된(사용자 지정이 아닌) 개체 감지만 포함됩니다. 모든 문서는 동일한 언어로 작성되어야 합니다.

다음 API 작업 중 하나를 사용하여 문서 또는 문서 집합의 개체를 감지할 수 있습니다.

- DetectEntities
- BatchDetectEntities
- StartEntitiesDetectionJob

이 작업은 문서의 각 개체에 대해 하나씩 API Entity 객체 목록을 반환합니다. `BatchDetectEntities` 작업은 배치의 각 문서에 대해 하나의 목록으로 구성된 `Entity` 개체 목록을 반환합니다. `StartEntitiesDetectionJob` 작업은 작업의 각 문서에 대한 `Entity` 개체 목록이 포함된 파일을 생성하는 비동기 작업을 시작합니다.

다음 예제는 `DetectEntities` 작업의 응답입니다.

```
{
    "Entities": [
        {
            "Text": "today",
            "Score": 0.97,
            "Type": "DATE",
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
```

```
            "Score": 0.95,
            "Type": "LOCATION",
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ],
    "LanguageCode": "en"
}
```

이벤트

이벤트 감지를 사용하여 텍스트 문서에서 특정 유형의 이벤트 및 관련 개체를 분석할 수 있습니다. Amazon Comprehend는 비동기 분석 작업을 사용하여 대규모 문서 컬렉션에서 이벤트를 감지할 수 있도록 지원합니다. 예제 이벤트 분석 작업을 비롯한 이벤트에 대한 자세한 내용은 Amazon Comprehend Events 출시 발표를 참조하세요.

개체

Amazon Comprehend는 입력 텍스트에서 감지된 이벤트와 관련된 개체 목록을 추출합니다. 개체는 사람, 장소 또는 위치와 같은 실제 개체일 수 있으며 측정값, 날짜, 또는 수량과 같은 개념일 수도 있습니다. 개체가 발생할 때마다 입력 텍스트의 개체에 대한 텍스트 참조인 멘션으로 식별됩니다. 각 고유 개체에 대해 모든 멘션이 목록으로 그룹화됩니다. 이 목록은 입력 텍스트에서 개체가 발생한 위치에 대한 세부 정보를 제공합니다. Amazon Comprehend는 지원되는 이벤트 유형과 관련된 개체만 감지합니다.

지원되는 이벤트 유형과 연결된 각 개체는 다음과 같은 관련 세부 정보와 함께 반환됩니다.

- 멘션: 입력 텍스트에서 동일한 개체가 발생하는 각 항목에 대한 세부 정보.
  - 시작오프셋: 입력 텍스트에서 멘션이 시작되는 위치를 나타내는 문자 오프셋(첫 번째 문자는 위치 0에 있음).
  - 종료오프셋: 입력 텍스트에서 멘션이 끝나는 위치를 나타내는 문자 오프셋.
  - 점수: 개체 유형의 정확성에 대한 Amazon Comprehend의 신뢰도 수준.
  - 그룹점수: 해당 멘션이 동일한 개체에 대한 다른 멘션과 올바르게 그룹화되어 있다는 Amazon Comprehend의 신뢰도 수준.
  - 텍스트: 개체의 텍스트.
  - 유형: 개체의 유형. 지원되는 모든 개체 유형에 대해서는 개체 유형을(를) 참조하세요.

이벤트

Amazon Comprehend는 입력 텍스트에서 감지한 이벤트(지원되는 이벤트 유형)의 목록을 반환합니다. 각 이벤트는 다음과 같은 관련 세부 정보와 함께 반환됩니다.

- 유형: 이벤트 유형. 지원되는 모든 이벤트 유형에 대해서는 이벤트 유형을(를) 참조하세요.
- 인수: 감지된 이벤트와 관련된 인수 목록. 인수는 감지된 이벤트와 관련된 개체로 구성됩니다. 인수의 역할은 누가, 어디서, 언제 무엇을 했는지와 같은 관계를 설명하는 것입니다.
  - 개체인덱스: Amazon Comprehend가 이 분석을 위해 반환한 개체 목록에서 하나의 개체를 식별하는 인덱스 값.
  - 역할: 이 인수의 개체가 이벤트와 어떻게 관련되는지를 설명하는 인수 유형. 지원되는 모든 인수 유형은 인수 유형을(를) 참조하세요.
  - 점수: 역할 감지의 정확성에 대한 Amazon Comprehend의 신뢰도 수준.
- 트리거: 감지된 이벤트의 트리거 목록. 트리거는 이벤트 발생을 나타내는 단일 단어 또는 문구입니다.
  - 시작오프셋: 입력 텍스트에서 트리거가 시작되는 위치를 나타내는 문자 오프셋(첫 번째 문자는 위치 0에 있음).
  - 종료오프셋: 입력 텍스트에서 트리거가 끝나는 위치를 나타내는 문자 오프셋.
  - 점수: 감지의 정확성에 대한 Amazon Comprehend의 신뢰도 수준.
  - 텍스트: 트리거의 텍스트.
  - 그룹점수: 해당 트리거가 동일한 이벤트에 대한 다른 트리거와 올바르게 그룹화되어 있다는 Amazon Comprehend의 신뢰도 수준.
  - 유형: 이 트리거가 나타내는 이벤트 유형.

이벤트 감지 결과 형식

이벤트 감지 작업이 완료되면 Amazon Comprehend는 작업을 시작할 때 지정한 Amazon S3 출력 위치에 분석 결과를 기록합니다.

출력은 다음 형식으로 감지된 각 이벤트에 대한 세부 정보를 제공합니다.

```
{
   "Entities": [
     {
       "Mentions": [
         {
```

```
          "BeginOffset": number,
          "EndOffset": number,
          "Score": number,
          "GroupScore": number,
          "Text": "string",
          "Type": "string"
        }, ...
      ]
    }, ...
  ],
  "Events": [
    {
      "Type": "string",
      "Arguments": [
        {
          "EntityIndex": number,
          "Role": "string",
          "Score": number
        }, ...
      ],
      "Triggers": [
        {
          "BeginOffset": number,
          "EndOffset": number,
          "Score": number,
          "Text": "string",
          "GroupScore": number,
          "Type": "string"
        }, ...
      ]
    }, ...
  ]
}
```

지원되는 개체, 이벤트 및 인수 유형

개체 유형

유형	설명
날짜	구체적이든 일반적이든 관계 없이 날짜 또는 시간에 대한 모든 언급.

유형	설명
시설	건물, 공항, 고속도로, 교량 및 기타 영구적인 인공 구조물 및 부동산 개선입니다.
위치	거리, 도시, 주, 국가, 수역 또는 지리적 좌표와 같은 물리적 위치.
통화 가치	미국 또는 다른 통화로 표시된 물건의 가치. 값은 구체적이거나 대략적일 수 있습니다.
조직	확립된 조직 구조로 정의되는 회사 및 기타 사용자 그룹.
개인	개인 또는 가상 인물의 이름 또는 별명.
개인_직함	개인을 설명하는 모든 직함. 일반적으로 고용 범주(예: CEO) 또는 경칭(예: Mr.)
수량	숫자 또는 값 및 측정 단위.
주식_코드	주식 시세 기호(예: AMZN, 국제 증권 식별 번호(ISIN), 통일 증권 식별 절차 위원회(CUSIP) 또는 증권 거래소 일일 공식 목록(SEDOL).

이벤트 유형

유형	설명
파산	미결제 채무를 상환할 수 없는 개인 또는 회사와 관련된 법적 소송
고용	직원이 고용, 해고, 퇴직하거나 기타 고용 상태가 변경될 때 발생합니다.
기업_인수	회사가 다른 회사의 주식 또는 물리적 자산의 대부분 또는 전부를 소유하여 해당 회사를 장악할 때 발생합니다.

유형	설명
투자_일반	개인이나 회사가 미래의 소득 또는 가치 상승을 기대할 수 있는 자산을 구매할 때 발생합니다.
기업_합병	두 개 이상의 회사가 결합하여 새로운 법인을 만들 때 발생합니다.
IPO	신주 발행을 통해 민간 기업의 주식을 일반에 공개하는 기업 공개(IPO).
권리_발행	기존 주주에게 기존 보유 지분에 비례하여 추가 주식을 매입할 수 있는 권리 그룹(청약 신주인수권증이라고 함).
2차_공모	회사 주주가 증권을 제공하는 것.
선반_공모	발행자가 증권의 새로운 발행을 등록하고, 증권을 다시 등록하거나 벌금을 부과하지 않고 일정 기간 동안 발행의 일부를 매각할 수 있도록 허용하는 증권거래위원회(SEC) 조항입니다. 선반 등록이라고도 합니다.
주식_공개매수	회사의 주주 지분 일부 또는 전부를 매입하겠다는 제안.
주식_분할	회사 이사회가 현재 주주에게 더 많은 주식을 발행하여 발행 주식 수를 늘릴 때 발생합니다. 이 이벤트는 주식 역분할에도 적용됩니다.

인수 유형

파산 관련 인수 유형

인수 유형	설명
신청인	파산을 신청한 사람 또는 회사.
날짜	파산 날짜 또는 시간.

인수 유형	설명
장소	파산이 발생한 장소(또는 가장 가까운 곳)의 위치 또는 시설.

고용 관련 인수 유형

유형	설명
피고용인	회사에 고용된 사람.
피고용인_직함	피고용인의 직함.
고용인	피고용인을 고용한 개인 또는 회사.
시작_날짜	고용 시작 날짜 또는 시간.
종료_날짜	고용 종료 날짜 또는 시간.

기업_인수, 투자_일반 관련 인수 유형

유형	설명
총액	거래와 연결된 금전적 가치.
투자 대상	투자와 관련된 개인 또는 회사.
투자자	자산에 투자하는 개인 또는 회사.
날짜	인수 또는 투자 날짜 또는 시간.
장소	인수 또는 투자가 이루어진 장소(또는 가장 가까운 곳).

기업_합병 관련 인수 유형

유형	설명
날짜	합병 날짜 또는 시간.

유형	설명
신규_회사	합병으로 인해 생긴 새로운 법인.
참가자	합병에 관여한 회사.

IPO, 권리_발행, 2차_공모, 선반_공모, 주식_공개매수 관련 인수 유형

유형	설명
만료_날짜	공모 만료 날짜 또는 시간.
투자자	자산에 투자하는 개인 또는 회사.
피청약자	공모 제안을 받는 개인 또는 회사.
공모_금액	공모와 관련된 금전적 가치.
공모_날짜	공모 날짜 또는 시간.
청약자	공모를 시작한 사람 또는 회사.
청약자_총_가치	공모와 관련된 총 금전적 가치.
기록_날짜	공모 기록 날짜 또는 시간.
매각_대리인	공모 매각을 촉진하는 개인 또는 회사.
주가	주가와 관련된 금전적 가치.
주식_수	공모와 관련된 주식 수.
채권인수인	공모 인수와 관련된 회사.

주식_분할 관련 인수 유형

유형	설명
회사	주식 분할의 주식을 발행하는 회사.

유형	설명
날짜	주식 분할 날짜 또는 시간.
분할_율	주식 분할 전 현재 주식 수 대비 증가된 신규 발행 주식 수의 비율.

핵심 문구

핵심 문구는 특정 사물을 설명하는 명사구가 포함된 문자열입니다. 일반적으로 명사와 이를 구분하는 수식어로 구성됩니다. 예를 들어 '날'은 명사이고, '아름다운 날'은 관사('a')와 형용사('아름다운')가 포함된 명사구입니다. 각 핵심 문구에는 Amazon Comprehend가 해당 문자열이 명사구라는 신뢰도를 나타내는 점수가 포함되어 있습니다. 점수를 사용하여 애플리케이션에 대한 감지 신뢰도가 충분히 높은지 확인할 수 있습니다.

Amazon Comprehend에서 지원하는 모든 기본 언어를 사용하여 핵심 문구 감지 작업을 수행할 수 있습니다. 모든 문서는 동일한 언어로 작성되어야 합니다.

다음 작업 중 하나를 사용하여 문서 또는 문서 집합에서 주요 문구를 감지할 수 있습니다.

- DetectKeyPhrases
- BatchDetectKeyPhrases
- StartKeyPhrasesDetectionJob

이 작업은 문서의 각 핵심 문구당 하나씩 핵심 문구 객체 목록을 반환합니다. `BatchDetectKeyPhrases` 작업은 배치의 각 문서에 대해 하나씩 `KeyPhrase` 객체 목록을 반환합니다. `StartKeyPhrasesDetectionJob` 작업은 작업의 각 문서에 대한 `KeyPhrase` 개체 목록이 포함된 파일을 생성하는 비동기 작업을 시작합니다.

다음 예제는 `DetectKeyPhrases` 작업의 응답입니다.

```
{
    "LanguageCode": "en",
    "KeyPhrases": [
        {
            "Text": "today",
            "Score": 0.89,
            "BeginOffset": 14,
```

```
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
            "Score": 0.91,
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ]
}
```

지배적 언어

Amazon Comprehend를 사용하여 텍스트를 검사하여 지배적 언어를 확인할 수 있습니다. Amazon Comprehend는 RFC 5646의 식별자를 사용하여 언어를 식별합니다. 2자리 ISO 639-1 식별자가 있고 필요한 경우 리전별 하위 태그가 있다면 해당 식별자를 사용합니다. 그렇지 않으면 ISO 639-2 3자리 코드를 사용합니다.

RFC 5646에 대한 자세한 내용은 IETF Tools 웹 사이트의 언어 식별을 위한 태그를 참조하세요.

응답에는 문서에서 특정 언어가 지배적인 언어라는 Amazon Comprehend의 신뢰 수준을 나타내는 점수가 포함됩니다. 각 점수는 다른 점수와 무관합니다. 이 점수가 특정 언어가 문서에서 특정 비율을 차지한다는 것을 의미하지는 않습니다.

책과 같이 긴 문서에 여러 언어가 포함된 경우, 긴 문서를 작은 조각으로 나누고 개별 문서에 대해 `DetectDominantLanguage` 작업을 실행할 수 있습니다. 그런 다음 결과를 집계하여 긴 문서에서 각 언어의 비율을 확인할 수 있습니다.

Amazon Comprehend 언어 감지에는 다음과 같은 제한 사항이 있습니다.

- 음성 언어 감지는 지원하지 않습니다. 예를 들어 “arigato”를 일본어로, “nihao”를 중국어로 감지하지 못합니다.
- 인도네시아어와 말레이시아어 혹은 보스니아어, 크로아티아어, 세르비아어와 같이 가까운 언어 쌍을 구분하기 어려울 수 있습니다.
- 최상의 결과를 얻으려면 20자 이상의 입력 텍스트를 제공하십시오.

Amazon Comprehend는 다음 언어를 감지합니다.

코드	Language
af	아프리칸스어
am	암하라어
ar	아랍어
as	아삼어
az	아제르바이잔어
ba	바쉬르어
be	벨라루스어
bn	벵골어
bs	보스니아어
bg	불가리아어
ca	카탈루냐어
ceb	세부아노어
cs	체코어
cv	추바시어
cy	웨일스어
da	덴마크어
de	독일어
el	그리스어
en	영어
eo	에스페란토어

코드	Language
et	에스토니아어
eu	바스크어
fa	페르시아어
fi	핀란드어
fr	프랑스어
gd	스코틀랜드 게일어
ga	아일랜드어
gl	갈리시아어
gu	구자라트어
ht	아이티어
he	히브리어
ha	하우사어
hi	힌디어
hr	크로아티아어
hu	헝가리어
hy	아르메니아어
ilo	일로코어
id	인도네시아어
is	아이슬란드어
it	이탈리아어

코드	Language
jv	자바어
ja	일본어
kn	칸나다어
ka	조지아어
kk	카자흐스탄어
km	중부 크메르어
ky	키르기즈어
ko	한국어
ku	쿠르드어
lo	라오스어
la	라틴어
lv	라트비아어
lt	리투아니아어
lb	룩셈부르크어
ml	말라얄람어
mt	몰타어
mr	마라티어
mk	마케도니아어
mg	마다가스카르어
mn	몽골어

코드	Language
ms	말레이어
my	버마어
ne	네팔어
new	네와리어
nl	네덜란드어
no	노르웨이어
or	오리야어
om	오로모어
pa	펀자브어
pl	폴란드어
pt	포르투갈어
ps	푸시토어
qu	케추아어
ro	루마니아어
ru	러시아어
sa	산스크리트어
si	신할라어
sk	슬로바키아어
sl	슬로베니아어
sd	신디어

코드	Language
so	소말리아어
es	스페인어
sq	알바니아어
sr	세르비아어
su	순다어
sw	스와힐리어
sv	스웨덴어
ta	타밀어
tt	타타르어
te	텔루구어
tg	타지크어
tl	타갈로그어
th	태국어
tk	투르크멘어
tr	터키어
ug	위구르어
uk	우크라이나어
ur	우르두어
uz	우즈벡어
vi	베트남어

코드	Language
yi	이디시어
yo	요루바어
zh	중국어 간체
zh-TW	중국어 번체

다음 작업 중 하나를 사용하여 문서 또는 문서 집합에서 지배적 언어를 감지할 수 있습니다.

- DetectDominantLanguage
- BatchDetectDominantLanguage
- StartDominantLanguageDetectionJob

`DetectDominantLanguage` 작업은 DominantLanguage 개체를 반환합니다. `BatchDetectDominantLanguage` 작업은 배치의 각 문서에 대해 하나씩 `DominantLanguage` 객체 목록을 반환합니다. `StartDominantLanguageDetectionJob` 작업은 작업의 문서마다 하나씩 `DominantLanguage` 객체 목록이 포함된 파일을 생성하는 비동기 작업을 시작합니다.

다음 예제는 `DetectDominantLanguage` 작업의 응답입니다.

```
{
    "Languages": [
        {
            "LanguageCode": "en",
            "Score": 0.9793661236763
        }
    ]
}
```

감성

Amazon Comprehend를 사용하여 UTF-8 인코딩 텍스트 문서의 콘텐츠 감성을 파악할 수 있습니다. 예를 들어, 감성 분석을 사용하여 블로그 게시물에 달린 댓글의 감성을 파악하여 독자가 게시물을 좋아했는지 확인할 수 있습니다.

Amazon Comprehend에서 지원하는 모든 기본 언어로 문서에 대한 감성을 파악할 수 있습니다. 한 작업의 모든 문서는 동일한 언어로 작성되어야 합니다.

감성이 결정되면 다음 값을 반환합니다.

- 긍정적 — 텍스트는 전반적으로 긍정적인 감성을 표현합니다.
- 부정적 — 텍스트는 전반적으로 부정적인 감성을 표현합니다.
- 혼합 — 텍스트는 긍정적인 감성과 부정적인 감성을 모두 표현합니다.
- 중립 — 텍스트는 긍정적이거나 부정적인 감성을 표현하지 않습니다.

다음 API 작업 중 하나를 사용하여 문서 또는 문서 집합의 감성을 감지할 수 있습니다.

- DetectSentiment
- BatchDetectSentiment
- StartSentimentDetectionJob

이 작업은 텍스트에서 가장 가능성이 높은 감성과 각 감성의 점수를 반환합니다. 점수는 감성이 올바르게 감지되었을 가능성을 나타냅니다. 예를 들어, 아래 예제에서는 텍스트에 `Positive` 감성이 있을 확률이 95%입니다. 텍스트에 `Negative` 감성이 있을 확률은 1% 미만입니다. `SentimentScore`를 사용하여 감지의 정확도가 애플리케이션의 요구 사항을 충족하는지 확인할 수 있습니다.

이 `DetectSentiment` 작업은 감지된 감성이 포함된 객체와 감성점수 객체를 반환합니다. 이 `BatchDetectSentiment` 작업은 배치의 각 문서에 대해 하나씩 감성 및 `SentimentScore` 객체 목록을 반환합니다. `StartSentimentDetectionJob` 작업은 작업의 문서마다 하나씩 감성 및 `SentimentScore` 객체 목록이 포함된 파일을 생성하는 비동기 작업을 시작합니다.

다음 예제는 `DetectSentiment` 작업의 응답입니다.

```
{
"SentimentScore": {
        "Mixed": 0.030585512690246105,
        "Positive": 0.94992071056365967,
        "Neutral": 0.0141543131828308,
        "Negative": 0.00893945890665054
    },
    "Sentiment": "POSITIVE",
    "LanguageCode": "en"
}
```

대상 감성

대상 감성은 입력 문서에 있는 특정 개체(예: 브랜드 또는 제품)와 관련된 감성을 세밀하게 이해할 수 있게 합니다.

대상 감성과 감성의 차이는 출력 데이터의 세분화 수준입니다. 감성 분석은 각 입력 문서에 대한 지배적인 감성을 결정하지만 추가 분석을 위한 데이터는 제공하지 않습니다. 대상 감성 분석은 각 입력 문서의 특정 개체에 대한 개체 수준의 감성을 결정합니다. 결과 데이터를 분석하여 긍정적이거나 부정적인 피드백을 받는 특정 제품 및 서비스를 확인할 수 있습니다.

예를 들어, 일련의 레스토랑 리뷰에서 고객은 “타코가 맛있었고 직원들도 친절했습니다”라는 리뷰를 제공합니다. 이 리뷰를 분석한 결과 다음과 같은 결과가 나옵니다.

- 감성 분석은 각 레스토랑 리뷰의 전반적인 감성이 긍정적인지, 부정적인지, 중립적인지, 또는 복합적인 것인지 여부를 결정합니다. 이 예시에서는 전반적인 감성이 긍정적입니다.
- 대상 감성 분석은 고객이 리뷰에서 언급한 레스토랑의 항목 및 속성에 대한 감성을 결정합니다. 이 예시에서 고객은 “타코”와 “직원”에 대해 긍정적인 의견을 남겼습니다.

대상 감성은 각 분석 작업에 대해 다음과 같은 결과를 제공합니다.

- 문서에 언급된 개체의 ID.
- 각 개체 멘션에 대한 개체 유형 분류.
- 각 개체 멘션에 대한 감성 및 감성 점수.
- 단일 항목에 해당하는 멘션 그룹(공동 참조 그룹).

콘솔 또는 API를 사용하여 대상 감성 분석을 실행할 수 있습니다. 콘솔과 API는 대상 감성에 대한 실시간 분석 및 비동기 분석을 지원합니다.

Amazon Comprehend는 영어로 작성된 문서에 대한 대상 감성을 지원합니다.

자습서를 포함하여 대상 감성에 대한 자세한 내용은 AWS 기계 학습 블로그의 Amazon Comprehend 대상 감성을 사용하여 텍스트로 세분화된 감성 추출을 참조하세요.

주제

- 개체 유형
- 공동 참조 그룹
- 출력 파일 구성

- 콘솔을 사용한 실시간 분석
- 대상 감성 출력 예제

개체 유형

대상 감성은 다음 개체 유형을 식별합니다. 개체가 다른 범주에 속하지 않는 경우 개체 유형을 기타로 지정합니다. 출력 파일에 언급된 각 개체에는 `"Type": "PERSON"` 같은 개체 유형이 포함됩니다.

개체 유형 정의

개체 유형	정의
개인	개인, 사람 그룹, 별명, 가상 인물, 동물 이름 등을 예로 들 수 있습니다.
위치	국가, 도시, 주, 주소, 지질 구조, 수역, 자연 명소, 천문학적 위치 등의 지리적 위치입니다.
조직	예로는 정부, 기업, 스포츠 팀, 종교 등이 있습니다.
시설	건물, 공항, 고속도로, 교량 및 기타 영구적인 인공 구조물 및 부동산 개선입니다.
브랜드	특정 상업용 품목 또는 제품 라인의 조직, 그룹 또는 생산자입니다.
상업_품목	차량 및 단 하나의 품목만 생산된 대형 상품을 포함한 모든 비제네릭 구매 또는 구매 가능 품목입니다.
영화	영화 또는 TV 프로그램입니다. 개체는 전체 이름, 닉네임 또는 자막일 수 있습니다.
음악	노래 전체 또는 일부입니다. 또한 앨범이나 앤솔로지와 같은 개별 음악 창작물의 컬렉션도 포함됩니다.
도서	전문적으로 출판되거나 자체 출판된 책입니다.
소프트웨어	공식적으로 출시된 소프트웨어 제품입니다.
게임	비디오 게임, 보드 게임, 일반 게임 또는 스포츠와 같은 게임입니다.
개인_호칭	공식 직함 및 경칭(예: 사장, 박사, 의사)입니다.

개체 유형	정의
이벤트	예로는 축제, 콘서트, 선거, 전쟁, 컨퍼런스, 홍보 행사 등이 있습니다.
날짜	구체적이든, 일반적이든, 절대적이든, 상대적이든 관계없이 날짜 또는 시간에 대한 모든 참조입니다.
수량	모든 측정값과 단위(통화, 백분율, 숫자, 바이트 등)입니다.
속성	제품의 “품질”, 휴대폰의 “가격” 또는 CPU의 “속도”와 같은 개체의 속성, 특징 또는 특성입니다.
기타	다른 범주에 속하지 않는 개체입니다.

공동 참조 그룹

대상 감성은 각 입력 문서에서 상호 참조 그룹을 식별합니다. 공동 참조 그룹은 문서 내 하나의 실제 개체에 해당하는 멘션 그룹입니다.

Example

다음 고객 리뷰 예제에서 “spa”는 `FACILITY` 개체 유형을 가진 개체입니다. 개체에는 대명사(“it”)로 두 개의 추가 멘션이 있습니다.

출력 파일 구성

대상 감성 분석 작업은 JSON 텍스트 출력 파일을 생성합니다. 파일에는 각 입력 문서에 대해 하나의 JSON 객체가 포함되어 있습니다. 각 JSON 객체는 다음 필드를 포함합니다.

- 개체 — 문서에 있는 개체의 배열입니다.
- 파일 — 입력 문서의 파일 이름입니다.
- 줄 — 입력 파일이 한 줄에 한 문서인 경우, 개체에는 파일에 있는 문서의 줄 번호가 포함됩니다.

> **Note**
>
> 대상 감성이 입력 텍스트의 어떤 항목도 식별하지 못하는 경우 개체 결과로 빈 배열을 반환합니다.

다음 예제는 3줄의 입력이 포함된 입력 파일의 개체를 보여줍니다. 입력 형식은 ONE_DOC_PER_LINE이므로 각 입력 줄은 문서입니다.

```
{ "Entities":[
    {entityA},
    {entityB},
    {entityC}
    ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 0
}
{ "Entities": [
    {entityD},
    {entityE}
  ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 1
}
{ "Entities": [
    {entityF},
    {entityG}
    ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 2
}
```

개체 배열의 개체에는 문서에서 감지된 개체 멘션의 논리적 그룹(상호 참조 그룹이라고 함)이 포함됩니다. 각 개체의 전체 구조는 다음과 같습니다.

```
{"DescriptiveMentionIndex": [0],
  "Mentions": [
     {mentionD},
     {mentionE}
    ]
}
```

개체에는 다음과 같은 필드가 있습니다.

- 멘션 — 문서에 있는 개체에 대한 멘션 배열입니다. 배열은 상호 참조 그룹을 나타냅니다. 예제는 the section called “공동 참조 그룹” 단원을 참조하세요. 멘션 배열의 멘션 순서는 문서 내 위치(오프셋)의 순서입니다. 각 멘션에는 해당 멘션의 감성 점수와 그룹 점수가 포함됩니다. 그룹 점수는 이러한 멘션이 동일한 개체에 속한다는 신뢰도를 나타냅니다.
- DescriptiveMentionIndex — 개체 그룹에 가장 적합한 이름을 제공하는 멘션 배열에 있는 하나 이상의 인덱스입니다. 예를 들어, 개체에는 텍스트 값이 "ABC Hotel," “ABC Hotel,” 및 “it”인 멘션이 세 개 있을 수 있습니다. 가장 좋은 이름은 DescriptiveMentionIndex 값이 [0,1]인 “ABC Hotel”입니다.

각 멘션에는 다음 필드가 포함됩니다.

- 시작오프셋 — 멘션이 시작되는 문서 텍스트의 오프셋입니다.
- 종료오프셋 — 문서 텍스트에서 멘션이 끝나는 지점의 오프셋입니다.
- 그룹점수 — 그룹에 언급된 모든 개체가 동일한 개체와 관련되어 있다는 신뢰도입니다.
- 텍스트 — 개체를 식별하는 문서 내의 텍스트입니다.
- 유형 – 개체의 유형입니다. Amazon Comprehend는 다양한 개체 유형을 지원합니다.
- 점수 — 개체가 관련성이 있다는 신뢰도를 모델링합니다. 값 범위는 0에서 1까지입니다. 여기서 1은 가장 높은 신뢰도입니다.
- 멘션감성 — 멘션에 대한 감성 및 감성 점수를 포함합니다.
- 감성 — 멘션의 감성입니다. 값에는 긍정적, 중립적, 부정적, 혼합이 포함됩니다.
- 감성점수 — 가능한 각 감성에 대한 모델 신뢰도를 제공합니다. 값 범위는 0에서 1까지입니다. 여기서 1은 가장 높은 신뢰도입니다.

감성 값의 의미는 다음과 같습니다.

- 긍정적 — 개체 멘션이 긍정적인 감성을 표현합니다.
- 부정적 — 개체 멘션이 부정적인 감성을 표현합니다.

- 혼합 — 개체 멘션이 긍정적인 감성과 부정적인 감성을 모두 표현합니다.
- 중립 — 개체 멘션이 긍정적이거나 부정적인 감성을 표현하지 않습니다.

다음 예제에서는 입력 문서에서 개체의 멘션이 하나뿐이므로 DescriptiveMentionIndex는 0(멘션 배열의 첫 번째 멘션)이 됩니다. 식별된 개체는 이름이 "I"인 PERSON입니다. 감성 점수는 중립입니다.


```
{"Entities":[
  {
    "DescriptiveMentionIndex": [0],
    "Mentions": [
      {
       "BeginOffset": 0,
       "EndOffset": 1,
       "Score": 0.999997,
       "GroupScore": 1,
       "Text": "I",
       "Type": "PERSON",
       "MentionSentiment": {
         "Sentiment": "NEUTRAL",
         "SentimentScore": {
           "Mixed": 0,
           "Negative": 0,
           "Neutral": 1,
           "Positive": 0
         }
       }
     }
   ]
  }
 ],
 "File": "Input.txt",
 "Line": 0
}
```


콘솔을 사용한 실시간 분석

Amazon Comprehend 콘솔을 사용하여 실시간으로 the section called "대상 감성"을 실행할 수 있습니다. 샘플 텍스트를 사용하거나 직접 입력한 텍스트를 입력 텍스트 상자에 붙여넣은 다음 분석을 선택합니다.

인사이트 패널의 콘솔에는 대상 감성 분석의 세 가지 보기가 표시됩니다.

- 분석된 텍스트 — 분석된 텍스트를 표시하고 각 항목에 밑줄을 긋습니다. 밑줄 색상은 분석에서 개체에 할당한 감성 값(긍정적, 중립적, 부정적 또는 혼합)을 나타냅니다. 콘솔에서 분석되는 텍스트 상자 오른쪽 위 모서리에 색상 매핑을 표시합니다. 개체 위에 커서를 놓으면 콘솔에 해당 개체에 대한 분석 값(개체 유형, 감성 점수)이 포함된 팝업 패널이 표시됩니다.
- 결과 — 텍스트에서 식별된 각 개체 멘션의 행이 포함된 표를 표시합니다. 표에는 각 개체에 대해 개체 및 개체 점수가 표시됩니다. 이 행에는 기본 감성과 각 감성 값의 점수도 포함됩니다. the section called “공동 참조 그룹”으로 알려진 동일한 개체에 대한 멘션이 여러 개 있는 경우 표에는 이러한 멘션이 기본 개체와 관련된 축소 가능한 행 집합으로 표시됩니다.

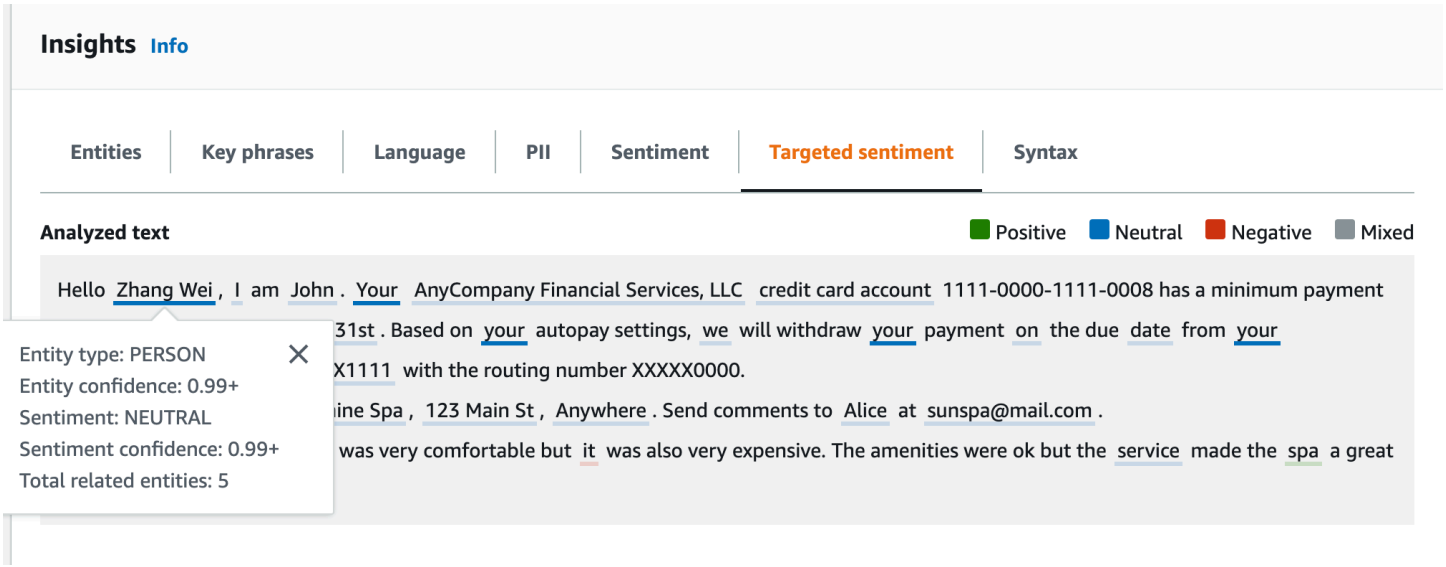
  결과 표의 개체 행을 마우스로 가리키면 콘솔이 분석된 텍스트 패널에서 해당 개체 멘션을 강조 표시합니다.
- 애플리케이션 통합 — API 요청의 파라미터 값과 API 응답에서 반환된 JSON 객체의 구조를 표시합니다. JSON 객체의 필드에 대한 설명은 the section called “출력 파일 구성”을 참조하세요.

콘솔 실시간 분석 예제

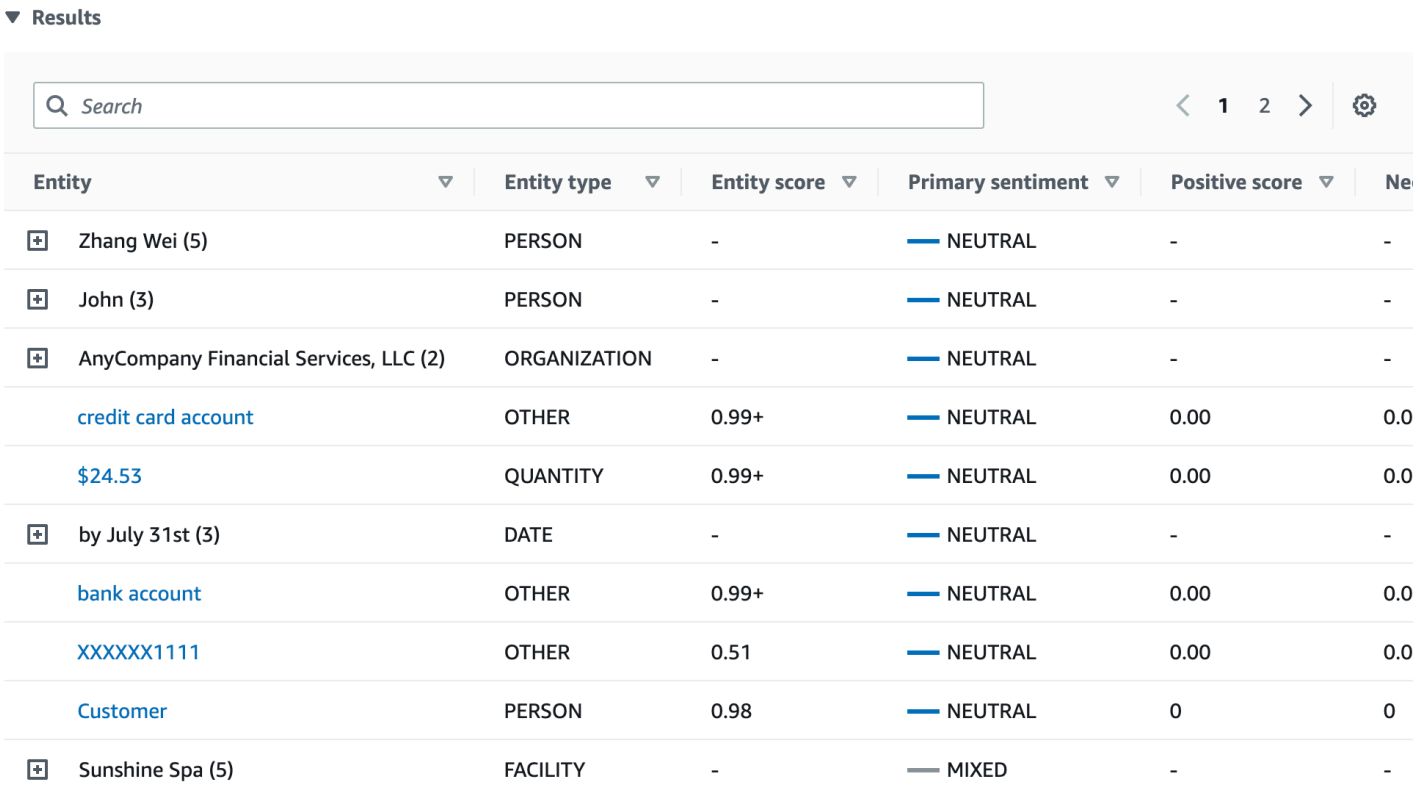
이 예제에서는 콘솔에서 제공하는 기본 입력 텍스트인 다음 텍스트를 입력으로 사용합니다.

```
Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC credit card account
 1111-0000-1111-0008 has a minimum payment
  of $24.53 that is due by July 31st. Based on your autopay settings, we will withdraw
 your payment on the due date from your
  bank account number XXXXXX1111 with the routing number XXXXX0000.
  Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to Alice at
 sunspa@mail.com.
  I enjoyed visiting the spa. It was very comfortable but it was also very expensive.
 The amenities were ok but the service made
  the spa a great experience.
```

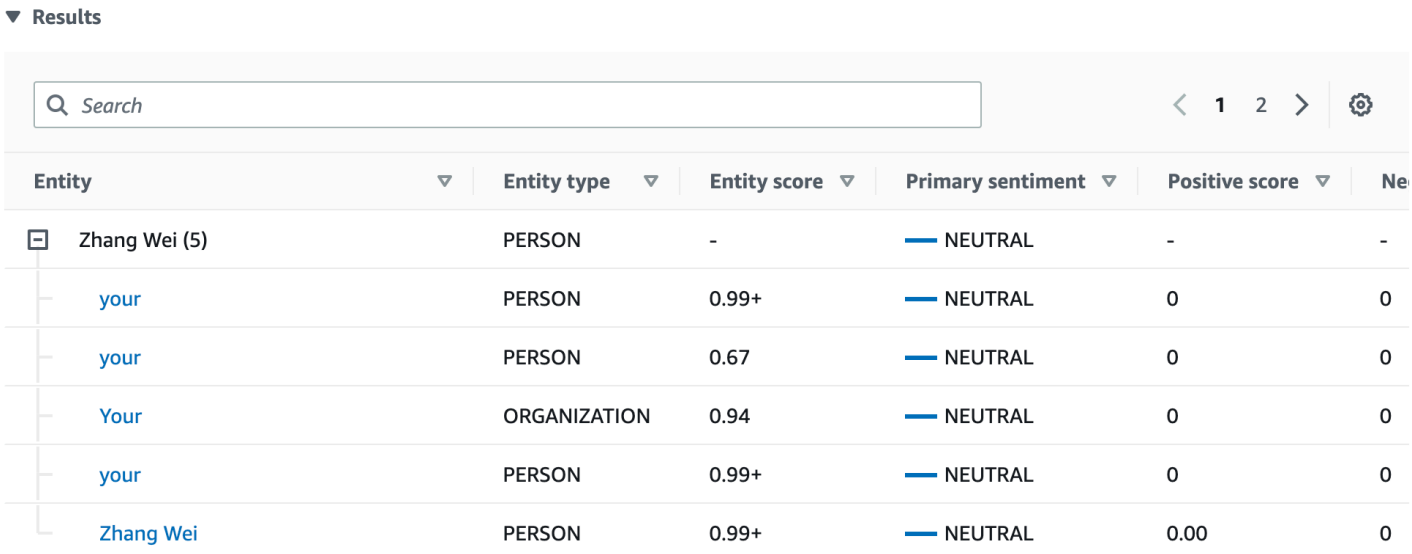
분석된 텍스트 패널에는 이 예제에 대한 다음 출력이 표시됩니다. 텍스트를 마우스로 `Zhang Wei` 텍스트를 가리키면 이 개체에 대한 팝업 패널이 표시됩니다.



결과 표는 개체 점수, 기본 감성, 각 감성 점수를 포함하여 각 항목에 대한 추가 세부 정보를 제공합니다.



이 예제에서 대상 감성 분석은 입력 텍스트의 your라는 각 멘션이 Zhang Wei라는 개인 개체를 언급하는 것임을 인식합니다. 콘솔은 이러한 멘션을 기본 개체와 관련된 접을 수 있는 행 집합으로 표시합니다.



애플리케이션 통합 패널에는 DetectTargetedSentiment API가 생성하는 JSON 객체가 표시됩니다. 자세한 예는 다음 섹션을 참조하세요.

대상 감성 출력 예제

다음 예제에서는 대상 감성 분석 작업의 출력 파일을 보여줍니다. 입력 파일은 세 개의 간단한 문서로 구성되어 있습니다.

```
The burger was very flavorful and the burger bun was excellent. However, customer
 service was slow.
My burger was good, and it was warm. The burger had plenty of toppings.
The burger was cooked perfectly but it was cold. The service was OK.
```

이 입력 파일의 대상 감성 분석은 다음과 같은 결과를 생성합니다.

```
{"Entities":[
  {
    "DescriptiveMentionIndex": [
      0
    ],
    "Mentions": [
      {
        "BeginOffset": 4,
        "EndOffset": 10,
        "Score": 0.999991,
        "GroupScore": 1,
        "Text": "burger",
```

```
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 0,
              "Positive": 1
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 38,
          "EndOffset": 44,
          "Score": 1,
          "GroupScore": 1,
          "Text": "burger",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0.000005,
              "Negative": 0.000005,
              "Neutral": 0.999591,
              "Positive": 0.000398
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 45,
```

```
          "EndOffset": 48,
          "Score": 0.961575,
          "GroupScore": 1,
          "Text": "bun",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0.000327,
              "Negative": 0.000286,
              "Neutral": 0.050269,
              "Positive": 0.949118
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 73,
          "EndOffset": 89,
          "Score": 0.999988,
          "GroupScore": 1,
          "Text": "customer service",
          "Type": "ATTRIBUTE",
          "MentionSentiment": {
            "Sentiment": "NEGATIVE",
            "SentimentScore": {
              "Mixed": 0.000001,
              "Negative": 0.999976,
              "Neutral": 0.000017,
              "Positive": 0.000006
            }
          }
        }
      ]
    }
  ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 0
```

```
}
{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 0,
          "EndOffset": 2,
          "Score": 0.99995,
          "GroupScore": 1,
          "Text": "My",
          "Type": "PERSON",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 1,
              "Positive": 0
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0,
        2
      ],
      "Mentions": [
        {
          "BeginOffset": 3,
          "EndOffset": 9,
          "Score": 0.999999,
          "GroupScore": 1,
          "Text": "burger",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0.000002,
```

```
            "Negative": 0.000001,
            "Neutral": 0.000003,
            "Positive": 0.999994
          }
        }
      },
      {
        "BeginOffset": 24,
        "EndOffset": 26,
        "Score": 0.999756,
        "GroupScore": 0.999314,
        "Text": "it",
        "Type": "OTHER",
        "MentionSentiment": {
          "Sentiment": "POSITIVE",
          "SentimentScore": {
            "Mixed": 0,
            "Negative": 0.000003,
            "Neutral": 0.000006,
            "Positive": 0.999991
          }
        }
      },
      {
        "BeginOffset": 41,
        "EndOffset": 47,
        "Score": 1,
        "GroupScore": 0.531342,
        "Text": "burger",
        "Type": "OTHER",
        "MentionSentiment": {
          "Sentiment": "POSITIVE",
          "SentimentScore": {
            "Mixed": 0.000215,
            "Negative": 0.000094,
            "Neutral": 0.00008,
            "Positive": 0.999611
          }
        }
      }
    ]
  },
  {
    "DescriptiveMentionIndex": [
```

```
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 52,
          "EndOffset": 58,
          "Score": 0.965462,
          "GroupScore": 1,
          "Text": "plenty",
          "Type": "QUANTITY",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 1,
              "Positive": 0
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 62,
          "EndOffset": 70,
          "Score": 0.998353,
          "GroupScore": 1,
          "Text": "toppings",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 0.999964,
              "Positive": 0.000036
            }
          }
        }
```

```
      ]
    }
  ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 1
}
{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 4,
          "EndOffset": 10,
          "Score": 1,
          "GroupScore": 1,
          "Text": "burger",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0.001515,
              "Negative": 0.000822,
              "Neutral": 0.000243,
              "Positive": 0.99742
            }
          }
        },
        {
          "BeginOffset": 36,
          "EndOffset": 38,
          "Score": 0.999843,
          "GroupScore": 0.999661,
          "Text": "it",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "NEGATIVE",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0.999996,
              "Neutral": 0.000004,
              "Positive": 0
```

```
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 53,
          "EndOffset": 60,
          "Score": 1,
          "GroupScore": 1,
          "Text": "service",
          "Type": "ATTRIBUTE",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0.000033,
              "Negative": 0.000089,
              "Neutral": 0.993325,
              "Positive": 0.006553
            }
          }
        }
      ]
    }
  ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 2
}
  }
```

구문 분석

구문 분석을 사용하여 문서의 단어를 파싱하고 문서 내 각 단어의 품사 또는 구문 함수를 반환합니다. 문서에서 명사, 동사, 형용사 등을 식별할 수 있습니다. 이 정보를 사용하여 문서 내용을 더 잘 이해하고 문서 내 단어 간의 관계를 이해할 수 있습니다.

예를 들어, 문서에서 명사를 찾은 다음 해당 명사와 관련된 동사를 찾을 수 있습니다. “할머니가 소파를 옮기셨어요”와 같은 문장에서는 “할머니”와 “소파”라는 명사와 “옮기셨어요”라는 동사를 볼 수 있습니

다. 이 정보를 사용하여 텍스트에서 관심이 있는 단어 조합을 분석하는 애플리케이션을 만들 수 있습니다.

분석을 시작하기 위해 Amazon Comprehend는 원본 텍스트를 구문 분석하여 텍스트에서 개별 단어를 찾습니다. 텍스트를 구문 분석한 후 각 단어에 원본 텍스트에서 차지하는 품사가 할당됩니다.

Amazon Comprehend는 다음과 같은 품사를 식별할 수 있습니다.

토큰	품사
ADJ	형용사 일반적으로 명사를 변형시키는 단어입니다.
ADP	부치사 전치사 또는 후치사 구의 머리말입니다.
ADV	부사 일반적으로 동사를 변형시키는 단어입니다. 형용사 및 기타 부사를 수정할 수도 있습니다.
AUX	조동사 동사구의 동사에 수반되는 기능어입니다.
CCONJ	조정 접속사 조정 접속사는 한 문장의 단어, 구 또는 절을 서로 종속시키지 않고 연결합니다.
CONJ	접속사 접속사는 문장의 단어, 구 또는 절을 연결합니다.

토큰	품사
DET	한정사 특정 명사구를 지정하는 관사 및 기타 단어입니다.
INTJ	감탄사 감탄사 또는 감탄사의 일부로 사용되는 단어입니다.
NOUN	명사 사람, 장소, 사물, 동물 또는 아이디어를 지칭하는 단어입니다.
NUM	숫자 숫자를 표현하는 단어(일반적으로 한정사, 형용사 또는 대명사)입니다.
O	기타 품사 범주를 지정할 수 없는 단어입니다.
PART	불변화사 다른 단어나 구와 연관되어 의미를 부여하는 기능어입니다.
PRON	대명사 명사나 명사구를 대체하는 단어입니다.

토큰	품사
PROPN	고유 명사 특정 개인, 장소 또는 사물의 이름을 나타내는 명사입니다.
PUNCT	문장 부호 텍스트를 구분하는 알파벳이 아닌 문자입니다.
SCONJ	종속 접속사 종속 조항을 문장에 연결하는 접속사입니다. 종속 접속사의 예로는 “because”가 있습니다.
SYM	Symbol 달러 기호($) 또는 수학 기호와 같은 단어 모양의 엔터티입니다.
VERB	동사 사건과 행동을 나타내는 단어입니다.

품사에 대한 자세한 내용은 범용 종속성 웹사이트의 범용 POS 태그를 참조하세요.

작업은 텍스트에서 해당 단어가 나타내는 품사와 단어를 식별하는 토큰을 반환합니다. 각 토큰은 소스 텍스트에 있는 단어를 나타냅니다. 소스에서 단어의 위치, 텍스트에서 해당 단어가 차지하는 품사, 품사가 올바르게 식별되었다는 Amazon Comprehend의 신뢰성, 원본 텍스트에서 파싱된 단어를 제공합니다.

다음은 구문 토큰 목록의 구조입니다. 문서의 각 단어에 대해 하나의 구문 토큰이 생성됩니다.

```
{
   "SyntaxTokens": [
```

```
      {
         "BeginOffset": number,
         "EndOffset": number,
         "PartOfSpeech": {
            "Score": number,
            "Tag": "string"
         },
         "Text": "string",
         "TokenId": number
      }
   ]
}
```

각 토큰은 다음 정보를 제공합니다.

- `BeginOffset` 및 `EndOffset` —입력 텍스트에서 단어의 위치를 제공합니다.
- `PartOfSpeech`—두 가지 정보를 제공합니다. 하나는 품사를 식별하는 `Tag`이고 다른 하나는 품사가 올바르게 식별되었다는 Amazon Comprehend Syntax의 신뢰도를 나타내는 `Score`입니다.
- `Text`—식별된 단어를 제공합니다.
- `TokenId`—토큰의 식별자를 제공합니다. 식별자는 토큰 목록에서 토큰의 위치입니다.

Amazon Comprehend 사용자 정의

기계 교육 기반 NLP 솔루션을 구축하는 데 필요한 기술이 없어도 특정 요구 사항에 맞게 Amazon Comprehend를 사용자 정의할 수 있습니다. Comprehend 사용자 지정은 자동 기계 학습, 즉 AutoML을 사용해 사용자 대신 사용자가 제공한 학습 데이터를 통해 사용자 지정 NLP 모델을 구축합니다.

입력 문서 처리 – Amazon Comprehend는 사용자 지정 분류 및 사용자 지정 개체 인식을 위한 원스텝 문서 처리를 지원합니다. 예를 들어, 일반 텍스트 문서와 반정형 문서(예: PDF 문서, Microsoft Word 문서, 이미지)를 혼합하여 사용자 정의 분석 작업에 입력할 수 있습니다. 자세한 내용은 문서 처리를 참조하십시오.

사용자 정의 분류 — 사용자 정의 분류 모델(분류기)을 생성하여 문서를 고유한 범주로 구성할 수 있습니다. 각 분류 레이블에 해당 레이블을 가장 잘 나타내는 문서 세트를 제공하여 분류기를 학습시키십시오. 일단 학습되면 레이블이 지정되지 않은 여러 문서 세트에 분류기를 사용할 수 있습니다. 콘솔을 사용하여 코드 없는 경험을 하거나 최신 AWS SDK를 설치할 수 있습니다. 자세한 내용은 사용자 지정 분류를 참조하십시오.

사용자 지정 개체 인식 – 텍스트를 분석하여 특정 용어와 명사 기반 구문을 분석할 수 있는 사용자 지정 개체 인식 모델(인식기)을 만들 수 있습니다. 정책 번호나 고객 에스컬레이션을 암시하는 문구와 같은 용어를 추출하도록 인식기를 학습시킬 수 있습니다. 모델을 학습시키려면 개체 목록과 해당 개체를 포함하는 문서 세트를 제공해야 합니다. 모델을 학습시킨 후에는 모델을 대상으로 분석 작업을 제출하여 사용자 지정 개체를 추출할 수 있습니다. 자세한 내용은 사용자 지정 개체 인식을 참조하십시오.

주제 모델링

Amazon Comprehend를 사용하여 문서 컬렉션의 내용을 검토하여 공통 주제를 결정할 수 있습니다. 예를 들어, Amazon Comprehend에 뉴스 기사 모음을 제공하면, 이는 스포츠, 정치 또는 엔터테인먼트와 같은 주제를 결정합니다. 문서의 텍스트에는 주석을 달 필요가 없습니다.

Amazon Comprehend는 잠재 디리클레 할당 기반 학습 모델을 사용하여 문서 집합의 주제를 결정합니다. 각 문서를 검토하여 단어의 문맥과 의미를 파악합니다. 전체 문서 집합에서 같은 문맥에 자주 속하는 단어 집합이 주제를 구성합니다.

단어는 문서에서 해당 주제가 얼마나 널리 사용되는지, 그리고 주제가 해당 단어와 얼마나 유사한지에 따라 문서의 주제와 연관됩니다. 특정 문서의 주제 분포에 따라 동일한 단어가 여러 문서의 다른 주제에 연결될 수 있습니다.

예를 들어, “포도당”이라는 단어는 주로 스포츠에 대해 다루는 기사에서는 “스포츠”라는 주제에 할당할 수 있고, 동일한 단어를 “의학”에 대한 기사에서는 “의학”이라는 주제에 할당할 수 있습니다.

주제와 관련된 각 단어에는 해당 단어가 주제를 정의하는 데 얼마나 도움이 되는지를 나타내는 가중치가 부여됩니다. 가중치는 전체 문서 세트에서 해당 단어가 주제의 다른 단어와 비교하여 해당 주제에서 나타나는 횟수를 나타냅니다.

가장 정확한 결과를 얻으려면 Amazon Comprehend에 사용할 수 있는 가장 큰 코퍼스를 제공해야 합니다. 최상의 결과를 얻으려면,

- 각 주제 모델링 작업에는 최소 1,000개의 문서를 사용해야 합니다.
- 각 문서는 3문장 이상이어야 합니다.
- 문서가 대부분 숫자 데이터로 구성된 경우 코퍼스에서 제거해야 합니다.

주제 모델링은 비동기식 프로세스입니다. StartTopicsDetectionJob 작업을 사용하여 Amazon S3 버킷에서 Amazon Comprehend에 문서 목록을 제출합니다. Amazon S3 버킷으로 응답이 전송됩니다. 입력 버킷과 출력 버킷을 모두 구성할 수 있습니다. ListTopicsDetectionJobs 작업을 사용하여 제출한 주제 모델링 작업의 목록을 가져오고 DescribeTopicsDetectionJob 작업을 사용하여 작업에 대한 정보를 봅니다. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에

관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까?를 참조하세요.

문서는 UTF-8 형식 텍스트 파일이어야 합니다. 문서를 두 가지 방식으로 제출할 수 있습니다. 다음 표에 옵션이 나와 있습니다.

형식	설명
파일당 문서 하나	각 파일에는 입력 문서가 한 개씩 들어 있습니다. 이 방법은 대용량 문서 모음에 가장 적합합니다.
라인당 문서 하나	단일 파일을 입력합니다. 파일의 각 줄은 문서로 간주됩니다. 소셜 미디어 게시물과 같은 짧은 문서에 가장 적합합니다. 각 라인은 줄 바꿈 (LF, \n), 캐리지 리턴 (CR,\ r) 또는 둘 다 (CRLF,\ r\ n) 로 끝나야 합니다. 유니코드 줄 구분자(u+2028)는 줄을 끝내는 데 사용할 수 없습니다.

자세한 정보는 InputDataConfig 데이터 유형을 참조하세요.

Amazon Comprehend는 문서 집합을 처리한 후, 두 개의 파일 `topic-terms.csv` 및 `doc-topics.csv`을 포함하는 압축된 아카이브를 반환합니다. 출력 파일에 대한 자세한 내용은 OutputDataConfig를 참조하세요.

첫 번째 출력 파일(`topic-terms.csv`)은 컬렉션의 주제 목록입니다. 각 주제에 대해 목록에는 기본적으로 주제별 상위 용어가 가중치에 따라 포함됩니다. 예를 들어, Amazon Comprehend에 신문 기사 컬렉션을 제공하면 컬렉션의 처음 두 주제를 설명하기 위해 다음을 반환할 수 있습니다.

주제	Term	가중치
000	팀	0.118533
000	게임	0.106072
000	플레이어	0.031625

주제	Term	가중치
000	시즌	0.023633
000	플레이	0.021118
000	야드	0.024454
000	코치	0.016012
000	게임	0.016191
000	풋볼	0.015049
000	쿼터백	0.014239
001	컵	0.205236
001	음식	0.040686
001	분	0.036062
001	추가	0.029697
001	테이블스푼	0.028789
001	기름	0.021254
001	후추	0.022205
001	티스푼	0.020040
001	와인	0.016588
001	설탕	0.015101

가중치는 특정 주제의 단어에 대한 확률 분포를 나타냅니다. Amazon Comprehend는 각 주제에 대해 상위 10개 단어만 반환하므로 가중치 합계는 1.0이 되지 않습니다. 한 주제에 포함된 단어가 10개 미만인 경우, 드물긴 하지만 가중치 합계는 1.0이 됩니다.

모든 주제에서의 발생 빈도를 살펴봄으로써 구별력에 따라 단어를 정렬합니다. 일반적으로 이 값은 가중치와 동일하지만 표의 “play” 및 “yard”라는 단어와 같이 일부 경우에는 가중치와 다른 순서로 표시됩니다.

반환할 주제 수를 지정할 수 있습니다. 예를 들어, Amazon Comprehend에 25개의 주제를 반환하도록 요청하면 컬렉션에서 가장 중요한 25개의 주제를 반환합니다. Amazon Comprehend는 컬렉션에서 최대 100개의 주제를 감지할 수 있습니다. 도메인에 대한 지식을 바탕으로 주제 수를 선택합니다. 정확한 수치를 찾으려면 몇 번의 실험이 필요할 수 있습니다.

두 번째 `doc-topics.csv` 파일에는 주제와 관련된 문서 및 해당 주제와 관련된 문서 비율이 나열되어 있습니다. `ONE_DOC_PER_FILE`를 지정한 경우 문서는 파일 이름으로 식별됩니다. `ONE_DOC_PER_LINE`를 지정한 경우 문서는 파일 이름과 파일 내에서 인덱스가 0인 줄 번호로 식별됩니다. 예를 들어, Amazon Comprehend는 파일당 문서 1개를 포함하여 제출한 문서 컬렉션에 대해 다음을 반환할 수 있습니다.

문서	주제	비율
sample-doc1	000	0.999330137
sample-doc2	000	0.998532187
sample-doc3	000	0.998384574
...		
sample-docN	000	3.57E-04

Amazon Comprehend는 Open database license (ODbL) v1.0에 따라 여기에서 제공되는 Lemmatization Lists Dataset by MBM의 정보를 활용합니다.

문서 처리 모드

Amazon Comprehend는 세 가지 문서 처리 모드를 지원합니다. 처리해야 하는 문서 수와 결과를 즉시 확인해야 하는 시간에 따라 모드를 선택할 수 있습니다.

- 단일 문서 동기 — 단일 문서로 Amazon Comprehend를 호출하면 동기식 응답을 받아 애플리케이션(또는 콘솔)으로 즉시 전송합니다.

- 다중 문서 동기 — 최대 25개의 문서 컬렉션이 포함된 Amazon Comprehend API를 호출하고 동기식 응답을 받습니다.
- 비동기 배치 — 대규모 문서 컬렉션의 경우 문서를 Amazon S3 버킷에 넣고 비동기 작업(콘솔 또는 API 작업 사용)을 시작하여 문서를 분석합니다. Amazon Comprehend는 요청에서 지정하는 S3 버킷/폴더에 분석 결과를 저장합니다.

주제

단일 문서 처리

단일 문서 작업은 문서 분석 결과를 애플리케이션에 직접 반환하는 동기 작업입니다. 한 번에 문서 하나에서 작동하는 대화형 애플리케이션을 생성할 때는 단일 문서 동기 작업을 사용하십시오.

동기 API 작업 사용에 대한 자세한 내용은 내장 모델을 사용한 실시간 분석(콘솔) 및 API를 사용한 실시간 분석을 참조하세요.

다중 문서 동기 처리

처리하려는 문서가 여러 개 있는 경우 `Batch*` API 작업을 사용하여 한 번에 둘 이상의 문서를 Amazon Comprehend로 보낼 수 있습니다. 각 요청에서 최대 25개의 문서를 전송할 수 있습니다. Amazon Comprehend는 요청의 각 문서에 대해 하나씩 응답 목록을 다시 보냅니다. 이러한 작업을 통해 이루어진 요청은 동기적으로 이루어집니다. 애플리케이션은 작업을 호출한 다음 서비스의 응답을 기다립니다.

`Batch*` 작업을 사용하는 것은 요청의 각 문서에 대해 단일 문서 API를 호출하는 것과 동일합니다. 이러한 API를 사용하면 애플리케이션 성능이 향상될 수 있습니다.

각 API의 입력은 처리할 문서가 포함된 JSON 구조입니다. `BatchDetectDominantLanguage`를 제외한 모든 작업에 대해서는 입력 언어를 설정해야 합니다. 각 요청에 대해 입력 언어를 하나만 설정할 수 있습니다. 예를 들어, 다음은 `BatchDetectEntities` 작업에 대한 입력입니다. 여기에는 두 개의 문서가 포함되어 있으며 영어로 되어 있습니다.

```
{
   "LanguageCode": "en",
```

```
   "TextList": [
      "I have been living in Seattle for almost 4 years",
      "It is raining today in Seattle"
   ]
}
```

Batch* 작업의 응답에는 ResultList 및 ErrorList 등 두 개의 목록이 있습니다. ResultList에는 성공적으로 처리된 각 문서에 대해 하나의 레코드가 들어 있습니다. 요청의 각 문서에 대한 결과는 문서에서 단일 문서 작업을 실행했을 때 얻게 되는 결과와 동일합니다. 각 문서의 결과에는 입력 파일의 문서 순서에 따라 색인이 지정됩니다. BatchDetectEntities 작업의 응답은 다음과 같습니다.

```
{
   "ResultList"  : [
      {
         "Index": 0,
         "Entities": [
            {
               "Text": "Seattle",
               "Score": 0.95,
               "Type": "LOCATION",
               "BeginOffset": 22,
               "EndOffset": 29
            },
            {
               "Text": "almost 4 years",
               "Score": 0.89,
               "Type": "QUANTITY",
               "BeginOffset": 34,
               "EndOffset": 48
            }
         ]
      },
      {
         "Index": 1,
         "Entities": [
            {
              "Text": "today",
              "Score": 0.87,
              "Type": "DATE",
              "BeginOffset": 14,
              "EndOffset": 19
            },
            {
```

```
                "Text": "Seattle",
                "Score": 0.96,
                "Type": "LOCATION",
                "BeginOffset": 23,
                "EndOffset": 30
            }
         ]
      }
   ],
   "ErrorList": []
}
```

요청에서 오류가 발생하는 경우 응답에는 오류가 포함된 문서를 식별하는 ErrorList가 포함됩니다. 문서는 입력 목록의 색인으로 식별됩니다. 예를 들어, BatchDetectLanguage 작업에 대한 다음 입력에는 처리할 수 없는 문서가 포함되어 있습니다.

```
{
   "TextList": [
     "hello friend",
     "$$$$$$",
     "hola amigo"
   ]
}
```

Amazon Comprehend의 응답에는 오류가 포함된 문서를 식별하는 오류 목록이 포함되어 있습니다.

```
{
    "ResultList": [
        {
          "Index": 0,
          "Languages":[
            {
              "LanguageCode":"en",
              "Score": 0.99
            }
          ]
        },
        {
          "Index": 2
          "Languages":[
            {
              "LanguageCode":"es",
```

```
            "Score": 0.82
          }
        ]
      }
  ],
  "ErrorList": [
    {
      "Index": 1,
      "ErrorCode": "InternalServerException",
      "ErrorMessage": "Unexpected Server Error. Please try again."
    }
  ]
}
```

배치 API 작업에 대한 자세한 내용은 실시간 배치 API를 참조하세요.

비동기 일괄 처리

대용량 문서 및 대규모 문서 컬렉션을 분석하려면 Amazon Comprehend 비동기 작업을 사용하십시오.

문서 컬렉션을 분석하려면 일반적으로 다음 단계를 수행합니다.

1. Amazon S3 버킷에 문서를 저장합니다.
2. 하나 이상의 분석 작업을 시작하여 문서를 분석합니다.
3. 분석 작업 진행 상황을 모니터링합니다.
4. 작업이 완료되면 S3 버킷에서 분석 결과를 검색합니다.

비동기 API 작업 사용에 대한 자세한 내용은 콘솔을 사용한 분석 작업 실행(콘솔) 및 API를 사용한 비동기 분석 작업을 참조하세요.

Amazon Comprehend에서 지원되는 언어

Amazon Comprehend는 다양한 기능을 위해 다양한 언어를 지원합니다. 지원되는 언어와 이를 지원하는 기능은 다음 표에서 확인할 수 있습니다.

주제

지원되는 언어

Amazon Comprehend는 하나 이상의 기능(지배적 언어 감지 기능은 제외)에 대해 다음 언어를 지원합니다.

코드	Language
de	독일어
en	영어
es	스페인어
it	이탈리아어
pt	포르투갈어
fr	프랑스어
ja	일본어
ko	한국어
hi	힌디어
ar	아랍어
zh	중국어 간체
zh-TW	중국어 번체

> **Note**
>
> Amazon Comprehend는 RFC 5646의 식별자를 사용하여 언어를 식별합니다. 2자리 ISO 639-1 식별자가 있고 필요한 경우 리전별 하위 태그가 있다면 해당 식별자를 사용합니다. 그렇지 않으면 ISO 639-2 3자리 코드를 사용합니다.
> RFC 5646에 대한 자세한 내용은 IETF Tools 웹 사이트의 언어 식별을 위한 태그를 참조하세요.

Amazon Comprehend 기능이 지원하는 언어

Feature	지원되는 언어
지배적 언어	지배적 언어 참조.
개체	지원되는 모든 언어
핵심 문구	지원되는 모든 언어
PII 개체 감지	영어 및 스페인어.
PII 개체에 레이블 지정	영어 및 스페인어.
감성	지원되는 모든 언어
대상 감성	영어
구문 분석	독일어(de), 영어(en), 스페인어(es), 프랑스어(fr), 이탈리아어(it), 포르투갈어(pt)
주제 모델링	사용하는 언어에 좌우되지 않습니다. 중국어, 일본어, 한국어와 같은 문자 기반 언어는 지원하지 않습니다.
사용자 지정 분류	일반 텍스트 모델은 독일어(de), 영어(en), 스페인어(es), 프랑스어(fr), 이탈리아어(it), 포르투갈어(pt) 등의 언어를 지원합니다.

Feature	지원되는 언어
	기본 문서 모델은 영어 문서만 지원합니다.
사용자 지정 개체 인식	독일어(de), 영어(en), 스페인어(es), 프랑스어(fr), 이탈리아어(it), 포르투갈어(pt) PDF 및 Word의 사용자 지정 개체 인식은 영어 문서만 지원합니다.

설정

Amazon Comprehend 첫 사용 전에 다음 태스크를 완료하십시오.

태스크 설정

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- AWS Command Line Interface (AWS CLI) 설정
- 프로그래밍 방식 액세스 권한 부여

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

AWS Command Line Interface (AWS CLI) 설정

시작하기 연습의 단계를 수행하는 AWS CLI 데가 필요하지 않습니다. 다만 이 설명서의 일부 다른 연습에서는 사용합니다. 원하는 경우이 단계를 건너뛰고 로 이동하여 AWS CLI 나중에를 Amazon Comprehend 시작하기설정할 수 있습니다.

를 설치하고 구성하려면 AWS CLI

1. 를 설치합니다 AWS CLI. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 주제를 참조하세요.

   최신 버전의 설치 또는 업데이트 AWS Command Line Interface
2. AWS CLI구성. 관련 지침은 AWS Command Line Interface 사용 설명서에서 다음 주제를 참조하세요.

   AWS Command Line Interface구성

프로그래밍 방식 액세스 권한 부여

사용자는 AWS 외부에서와 상호 작용하려는 경우 프로그래밍 방식으로 액세스해야 합니다 AWS Management Console. 프로그래밍 방식 액세스를 부여하는 방법은에 액세스하는 사용자 유형에 따라 다릅니다 AWS.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center 참조하세요.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		• AWS SDKs, 도구 및 AWS APIs 경우 SDK 및 도구 참조 안내서의 IAM Identity Center 인증을 참조하세요. AWS SDKs
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용의 지침을 따릅니다.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 사용 AWS Command Line Interface 설명서의 IAM 사용자 자격 증명을 사용하여 인증을 참조하세요. • AWS SDKs 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용하여 인증을 참조하세요. AWS SDKs • AWS APIs 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

Amazon Comprehend 시작하기

다음 연습에서는 Amazon Comprehend 콘솔을 사용하여 비동기 개체 감지 작업을 생성하고 실행합니다. 이 연습에서는 Amazon Simple Storage Service(S3)를 잘 알고 있다고 가정합니다. 더 간단한 예는 내장 모델을 사용한 실시간 분석을 참조하세요.

개체 감지 작업을 만들려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 분석 작업을 선택한 다음 작업 생성을 선택합니다.
3. 작업 설정에서 작업에 이름을 지정합니다. 이 이름은 리전과 계정 내에서 고유한 이름이어야 합니다.
4. 분석 유형에서 개체를 선택합니다.
5. 언어에서 입력 문서의 언어를 선택합니다.
6. 입력 데이터의 데이터 소스에 대해 예제 문서를 선택합니다. 콘솔이 S3 위치를 공개 샘플이 포함된 폴더로 설정합니다.
7. 출력 데이터의 S3 위치에서 출력 파일의 URL 또는 폴더 위치를 Amazon S3에 붙여넣습니다.
8. 액세스 권한 섹션에서 IAM 역할 생성을 선택합니다. 콘솔이 Amazon Comprehend가 입력 및 출력 버킷에 액세스할 수 있도록 적절한 권한을 가진 새 IAM 역할을 생성합니다.
9. 양식 작성을 마치고 나면 작업 생성을 선택하여 주제 탐지 작업을 생성하여 시작합니다.

   새 작업이 작업 목록에 나타나고 상태 필드에 작업 상태가 표시됩니다. 이 필드는 처리 중인 작업은 `IN_PROGRESS`, 성공적으로 완료된 작업은 `COMPLETED`, 오류가 있는 작업을 `FAILED`으로 표시합니다.
10. 작업을 선택하여 작업 세부 정보 패널을 엽니다.
11. 출력의 출력 데이터 위치에서 링크를 선택하여 Amazon S3 콘솔을 엽니다.
12. Amazon S3 콘솔에서 다운로드를 선택하고 `output.tar.gz` 파일을 저장합니다.
13. 파일의 압축을 풀고 Json 파일로 저장합니다.
14. 개체 유형에 대한 설명과 감지된 각 개체의 필드는 the section called “개체”을 참조하세요.

Amazon Comprehend 콘솔 사용 분석

Amazon Comprehend 콘솔을 사용하여 실시간으로 문서를 분석하거나 비동기 분석 작업을 실행할 수 있습니다.

내장 모델을 통한 실시간 분석을 사용하여 개체를 인식하고, 핵심 문구를 추출하고, 기본 언어를 감지하고, PII를 감지하고, 감성을 파악하고, 대상 감성을 분석하고, 구문을 분석할 수 있습니다.

내장된 모델을 사용하여 분석 작업을 실행하여 개체, 이벤트, 구문, 기본 언어, 감성, 표적 감성, 개인 식별 정보(PII)와 같은 인사이트를 찾을 수 있습니다. 주제 모델링 작업도 실행할 수 있습니다.

또한 콘솔은 사용자 지정 모델을 사용한 실시간 및 비동기 분석을 지원합니다. 자세한 정보는 사용자 지정 분류 및 사용자 지정 개체 인식을 참조하세요.

주제

- 내장 모델을 사용한 실시간 분석
- 콘솔을 사용한 분석 작업 실행

내장 모델을 사용한 실시간 분석

Amazon Comprehend 콘솔을 사용하여 UTF-8 인코딩 텍스트 문서를 실시간 분석 할 수 있습니다. 문서는 영어이거나 Amazon Comprehend에서 지원하는 다른 언어 중 하나일 수 있습니다. 결과는 콘솔에 표시되므로 분석을 검토할 수 있습니다.

문서 분석을 시작하려면에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.

샘플 텍스트를 사용자 고유 텍스트로 바꾼 다음 분석을 선택하여 텍스트를 분석할 수 있습니다. 분석 중인 텍스트 아래에 있는 결과 창에는 텍스트에 대한 추가 정보가 표시됩니다.

내장 모델을 사용하여 실시간 분석을 실행합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 실시간 분석을 선택합니다.
3. 입력 유형에서분석유형으로 내장을 선택합니다.
4. 분석하려는 텍스트를 입력합니다.

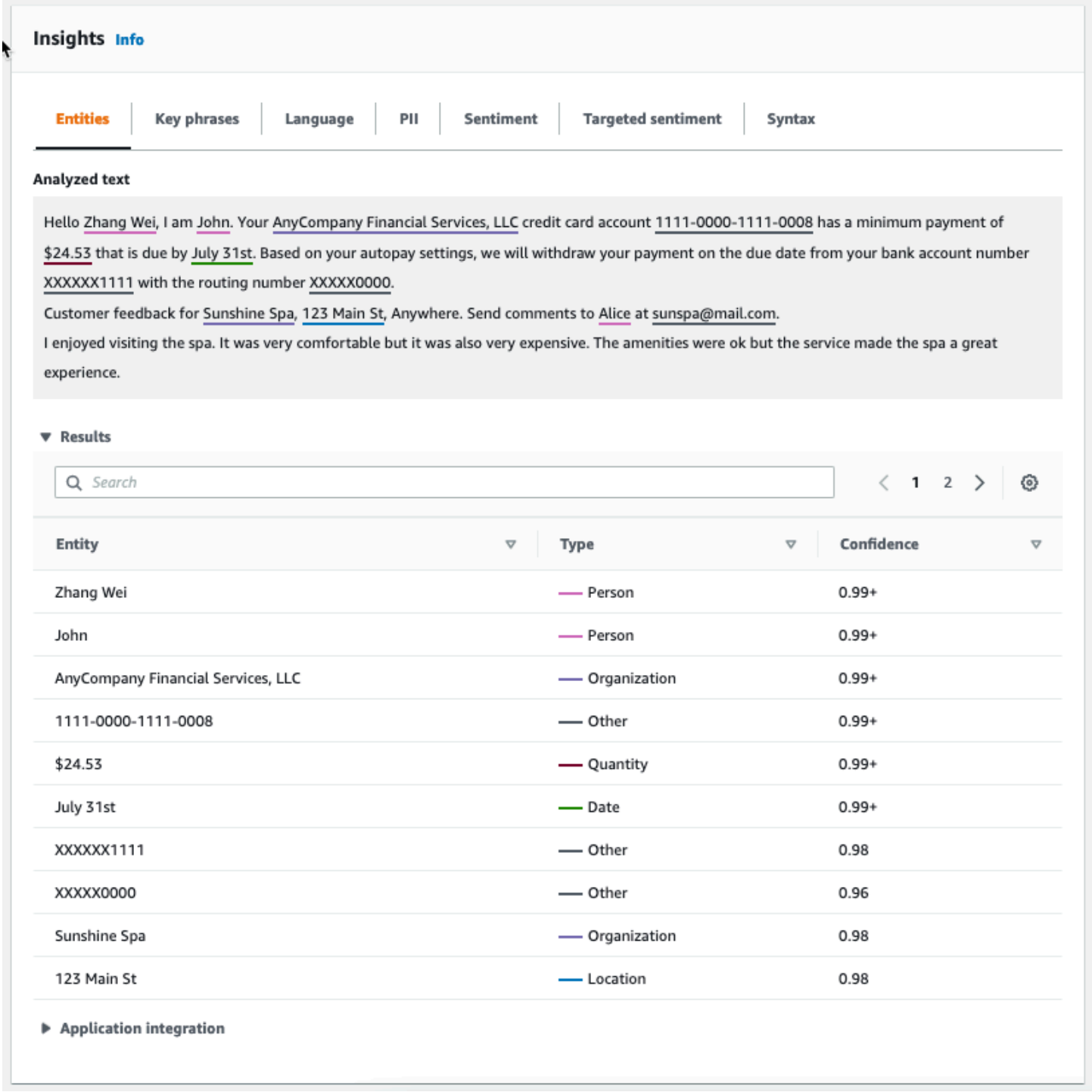
5. 분석을 선택합니다. 콘솔의 인사이트 패널에 텍스트 분석 결과가 표시됩니다. 인사이트 패널에는 각 인사이트 유형에 대한 탭이 있습니다. 다음 섹션에서는 인사이트 유형의 결과에 대해 설명합니다.

주제

- 개체
- 핵심 문구
- Language
- 개인 식별 정보(PII)
- 감성
- 대상 감성
- 구문

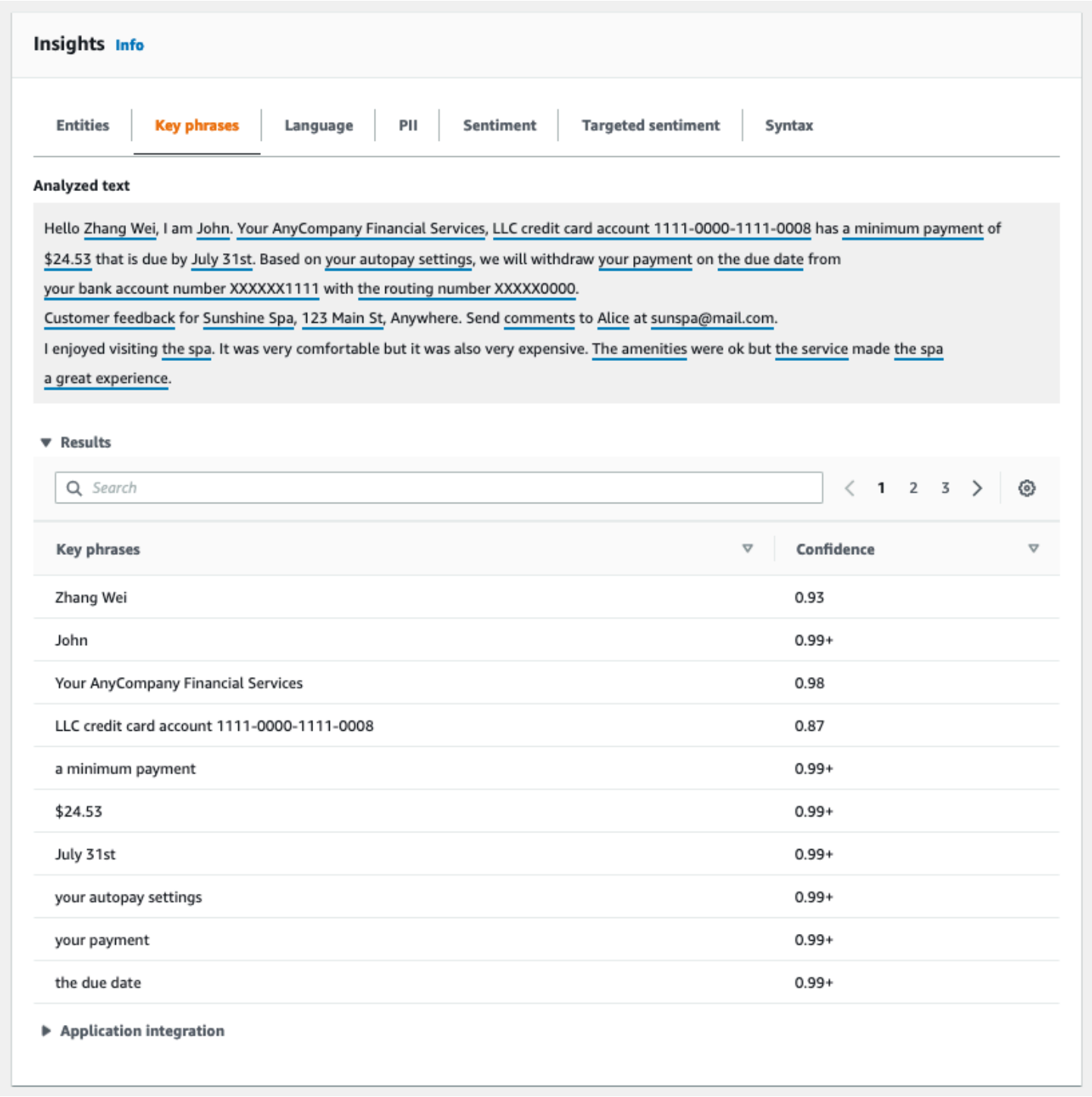
개체

개체 탭에는 Amazon Comprehend가 입력 텍스트에서 감지한 각 개체, 범주 및 신뢰 수준이 나타납니다. 결과는 조직, 위치, 날짜 및 사람과 같은 다양한 개체 유형을 나타내도록 색상으로 구분되어 있습니다. 자세한 내용은 개체 단원을 참조하십시오.



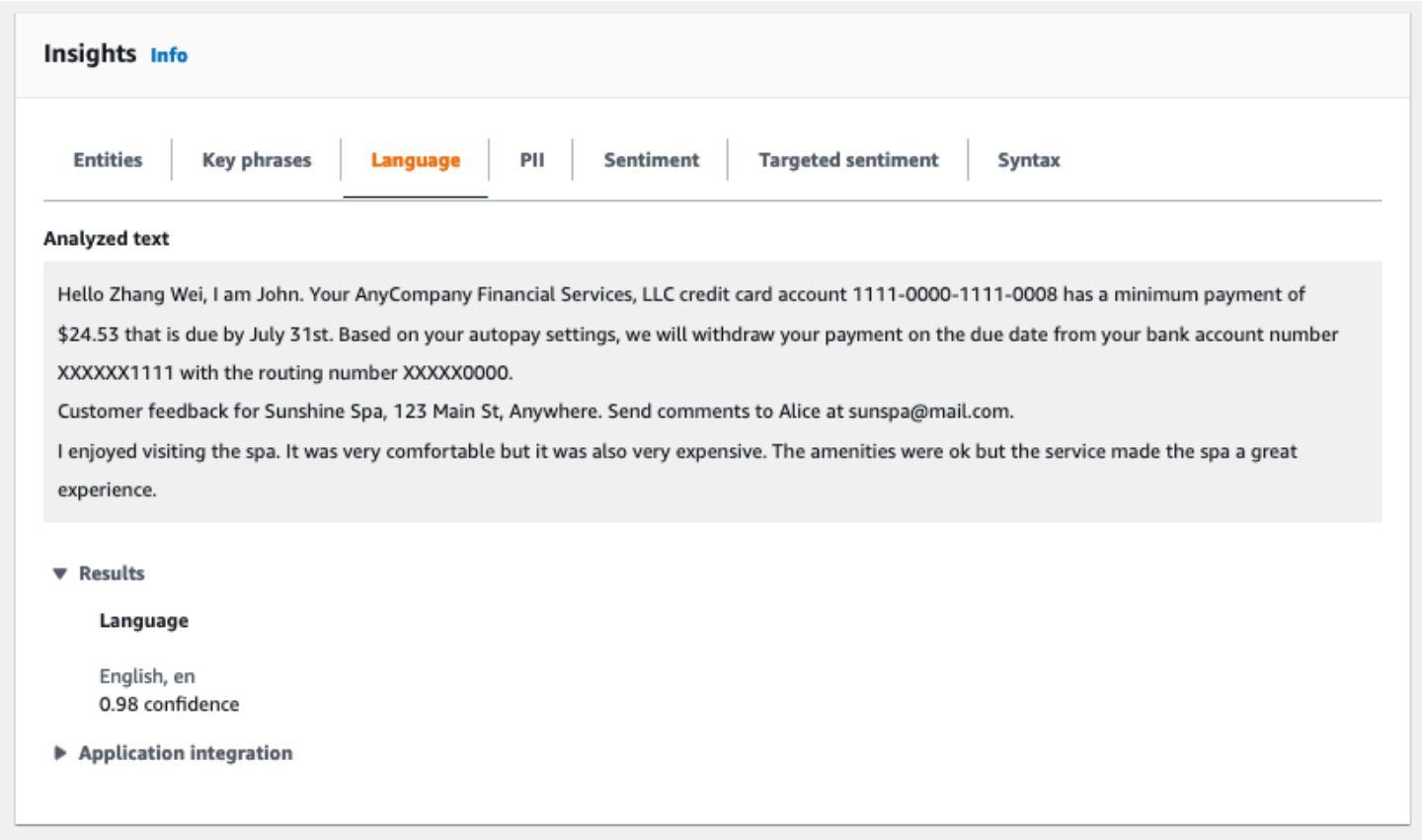
핵심 문구

핵심 문구 탭에는 Amazon Comprehend가 입력 텍스트에서 감지한 핵심 명사구와 관련 신뢰도 수준이 나열됩니다. 자세한 내용은 핵심 문구 단원을 참조하십시오.



Language

언어 탭에는 텍스트의 주요 언어와 주요 언어를 올바르게 감지했다는 Amazon Comprehend의 신뢰도 수준이 표시됩니다. Amazon Comprehend는 100개 언어를 인식할 수 있습니다. 자세한 내용은 지배적 언어 단원을 참조하십시오.



개인 식별 정보(PII)

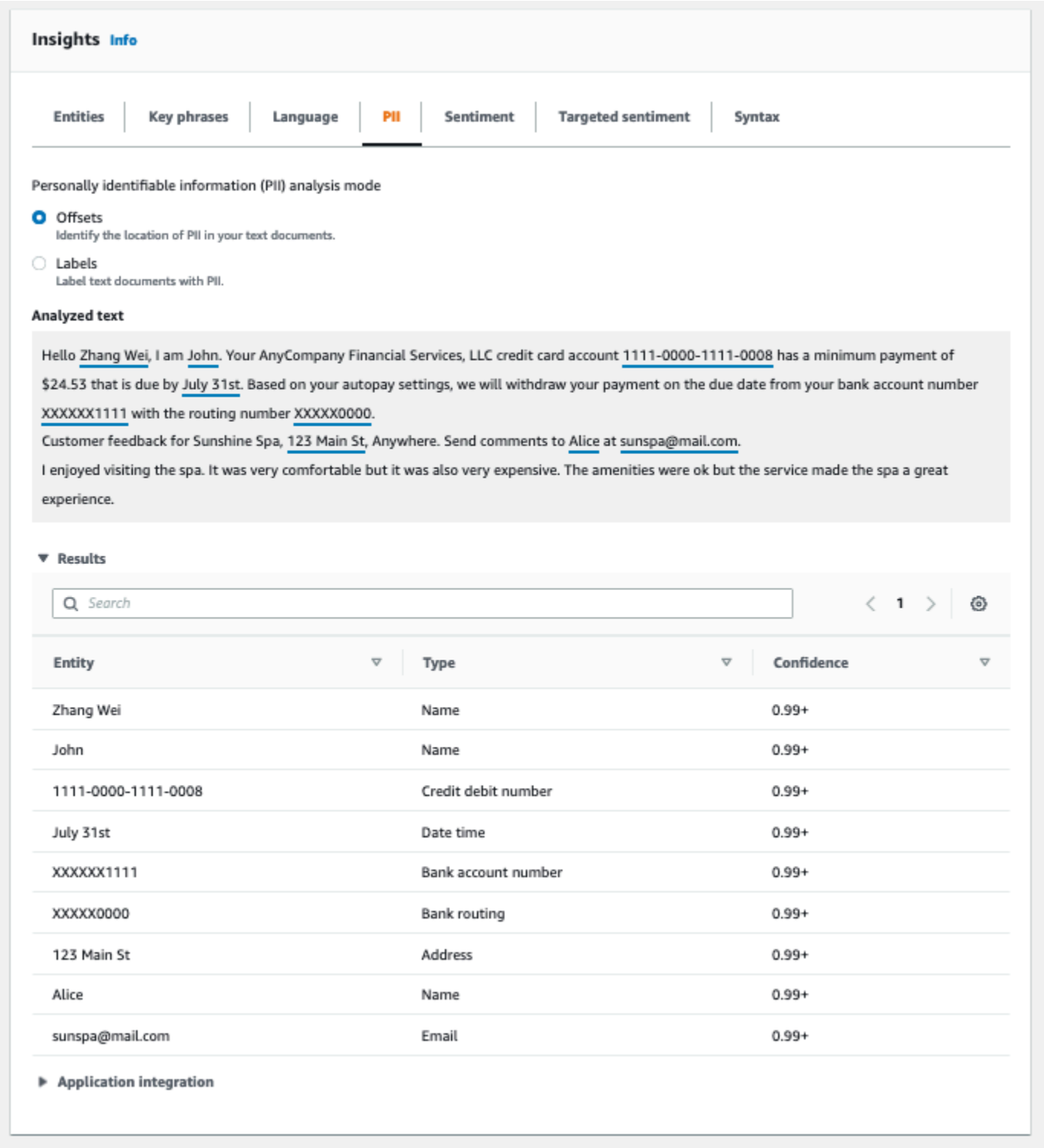
PII 탭에는 개인 식별 정보 (PII) 가 포함된 입력 텍스트의 항목이 나열됩니다. PII 개체는 주소, 은행 계좌 번호, 전화번호 등 개인을 식별하는 데 사용될 수 있는 개인 데이터에 대한 텍스트 참조입니다. 자세한 내용은 PII 개체 감지 단원을 참조하십시오.

PII 탭은 다으므두 가지 분석 모드를 제공합니다.

- 오프셋
- 레이블

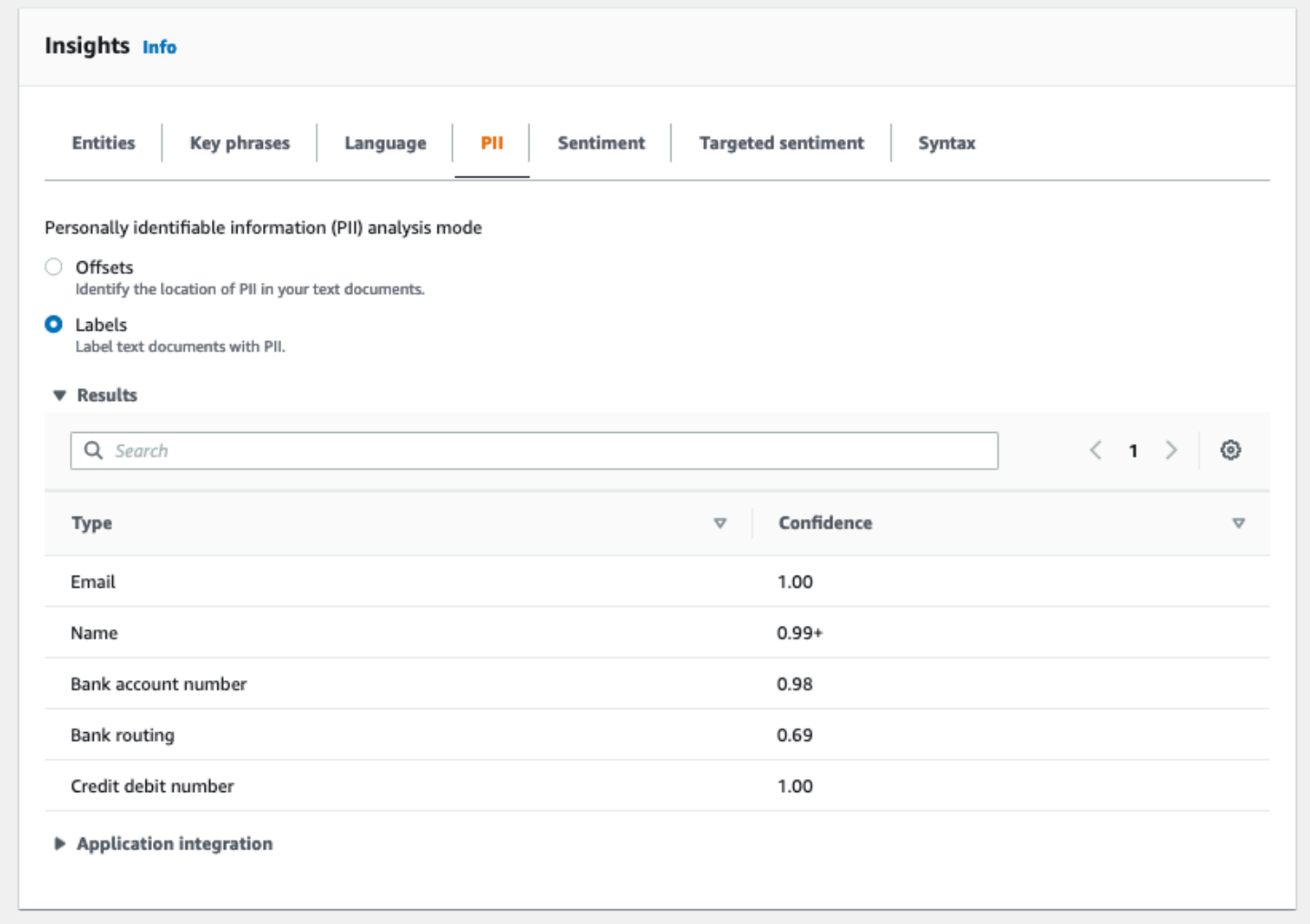
오프셋

오프셋 분석 모드는 텍스트 문서에서 PII의 위치를 식별합니다. 자세한 내용은 PII 개체 찾기 단원을 참조하십시오.



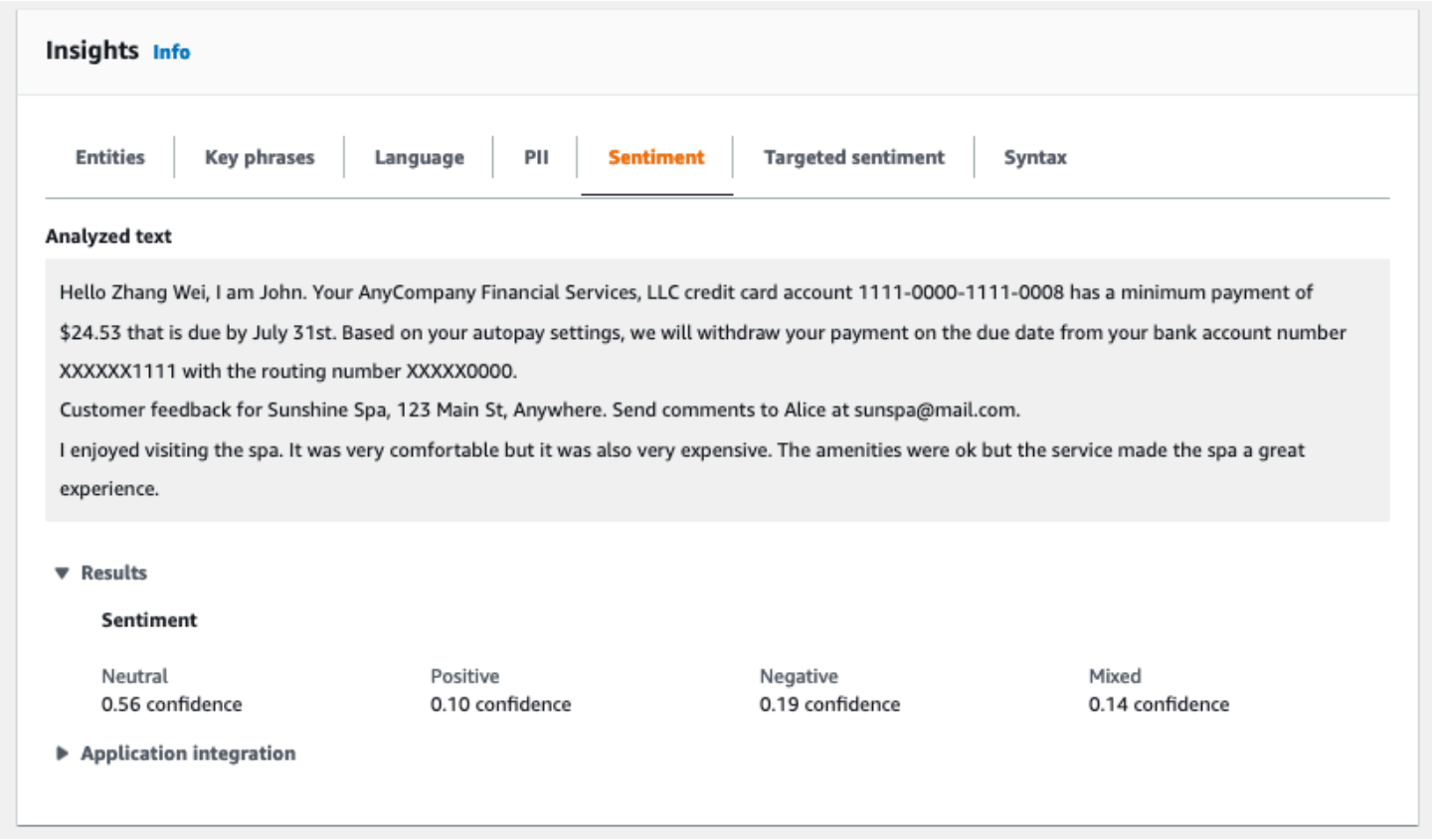
레이블

레이블 분석 모드는 텍스트 문서에 PII가 있는지 확인하고 식별된 PII 개체 유형의 레이블을 반환합니다. 자세한 내용은 PII 개체에 레이블 지정 단원을 참조하십시오.



감성

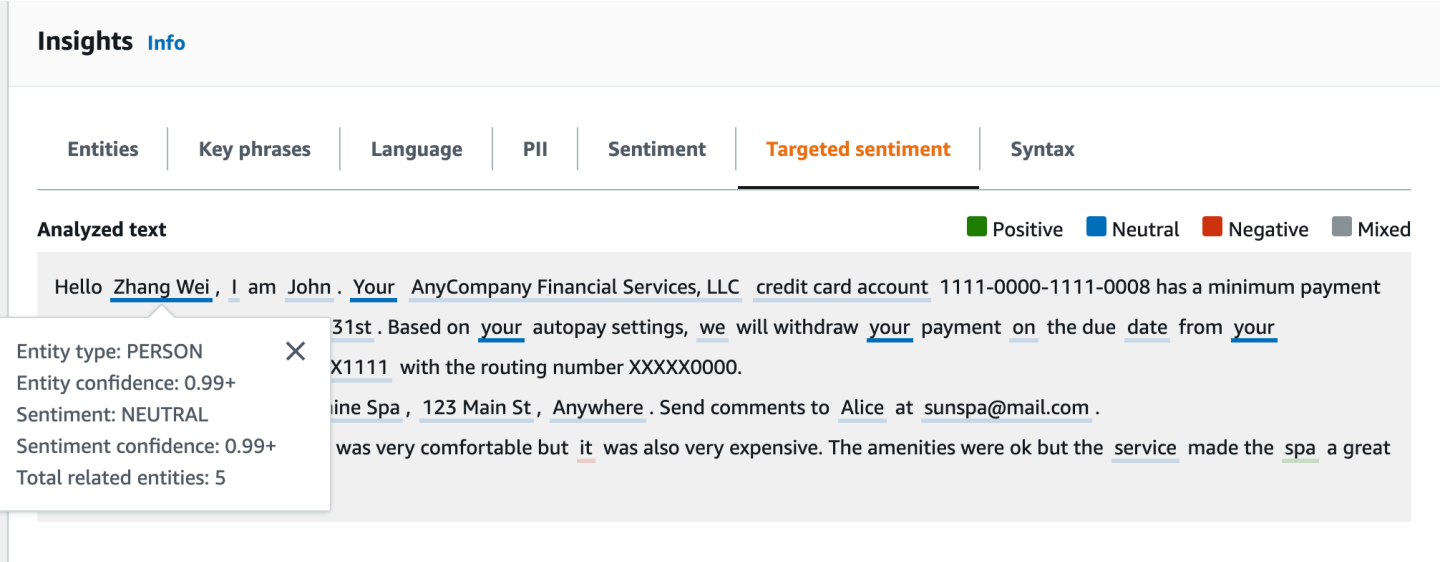
감성 탭에는 텍스트의 지배적인 감성이 표시됩니다. 감성은 중립, 긍정, 부정 또는 혼합으로 평가될 수 있습니다. 이 경우 각 감성에는 신뢰 등급이 있으며, 이는 해당 심리가 우세한 것으로 Amazon Comprehend의 추정치를 제공합니다. 자세한 내용은 감성 단원을 참조하십시오.



대상 감성

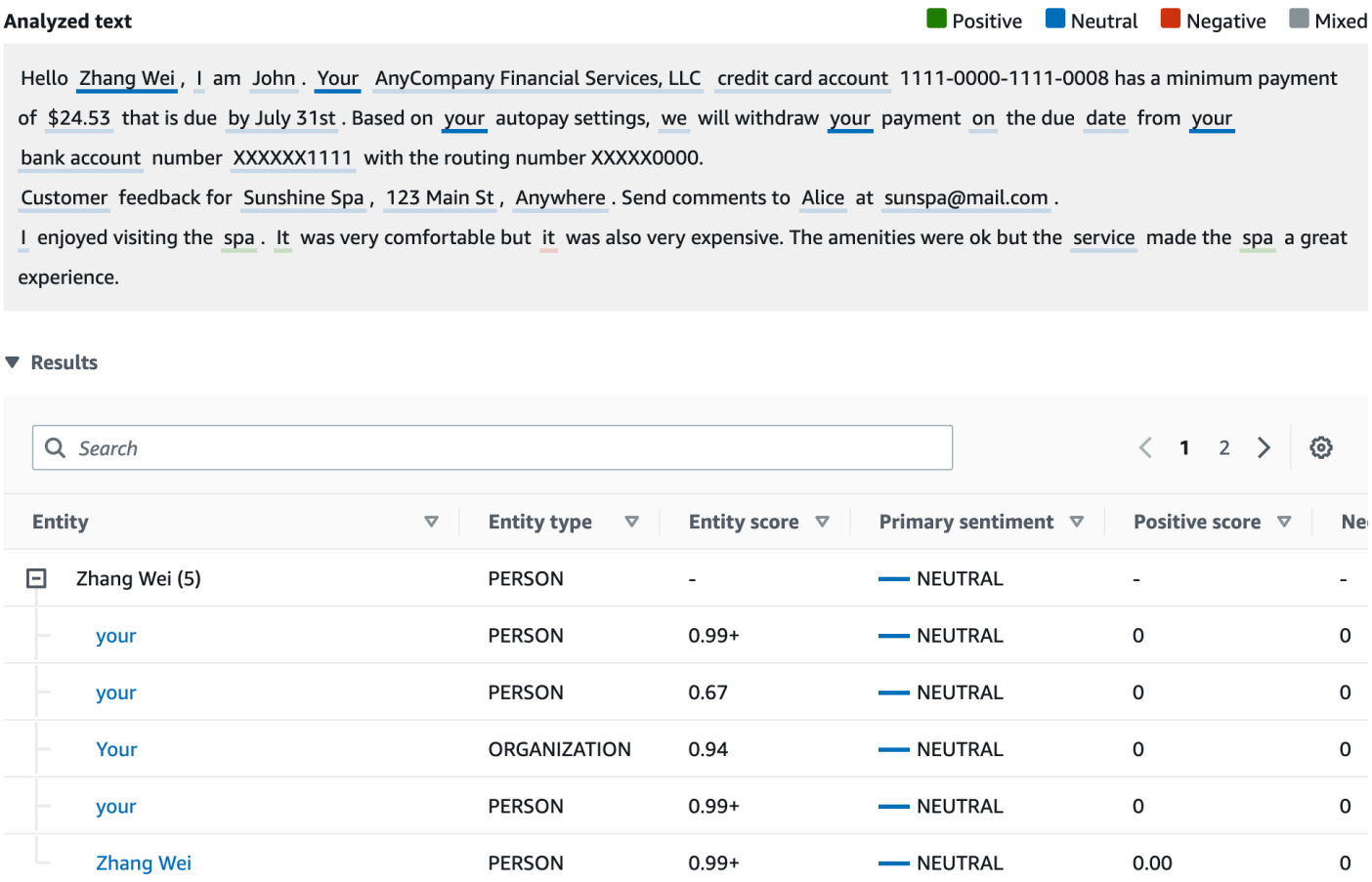
대상 감성 분석은 본문에 언급된 개체에 대해 표현된 감정을 식별합니다. Amazon Comprehend는 신뢰도 등급 및 기타 정보와 함께 개체에 대한 각 언급에 감성 등급을 할당합니다. 감성 등급은 중립, 긍정, 부정 또는 혼합일 수 있습니다.

분석된 텍스트 패널에서 콘솔은 분석된 각 항목에 밑줄을 표시합니다. 밑줄이 그어진 텍스트의 색상은 개체의 전반적인 감정을 나타냅니다. 개체 위에 커서를 올려 놓으면 콘솔은 팝업 창에 추가 정보를 표시합니다.



결과 테이블은 각 개체에 대한 추가 세부 정보를 제공합니다. 동일한 개체에 대해 상호 참조 그룹이라고 하는 멘션이 여러 개 있는 경우 테이블에는 이러한 멘션이 기본 개체와 관련된 축소 가능한 행 집합으로 표시됩니다.

다음 예제에서 개체는 Zhang Wei라는 사람입니다. 대상 감성 분석은 귀하에 대한 각 언급이 동일한 사람에 대한 언급이라고 인식합니다. 콘솔에는 이러한 표현이 기본 개체의 하위 항목으로 표시됩니다.

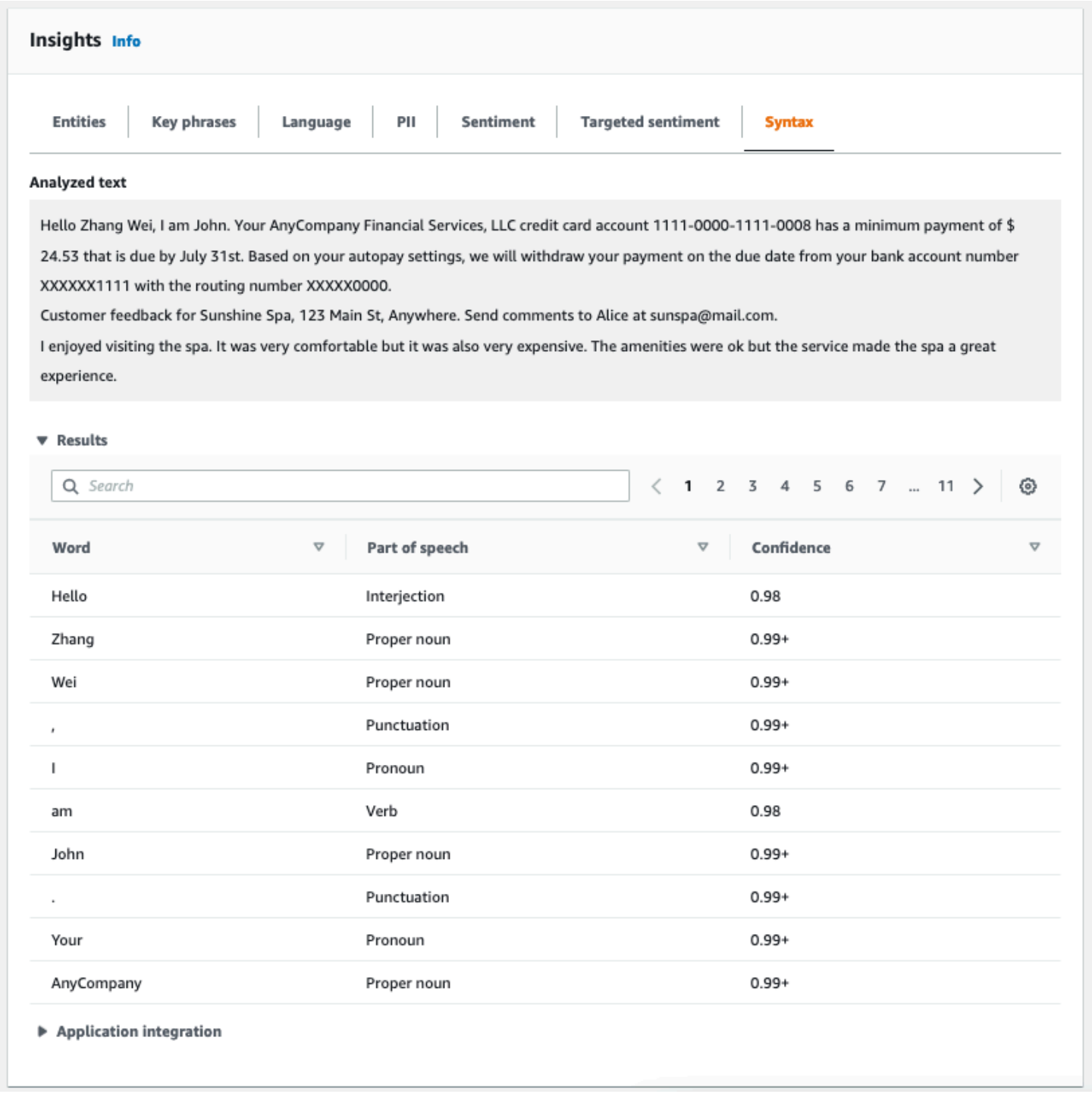


분석 중인 텍스트에 개체 유형 대상 감성이 없으면 대상 감성 분석 결과 필드가 비어 있습니다.

콘솔을 사용한 표적 감성 실시간 분석 사용 방법에 대한 자세한 내용은 콘솔을 사용한 실시간 분석을 참조하세요.

구문

구문 탭에는 텍스트의 각 요소에 대한 분류와 해당 품사 및 관련 신뢰도 점수가 표시됩니다. 자세한 내용은 구문 분석 단원을 참조하십시오.



콘솔을 사용한 분석 작업 실행

Amazon Comprehend 콘솔을 사용하여 비동기 분석 작업을 생성하고 관리할 수 있습니다. 작업은 Amazon S3에 저장된 문서를 분석하여 이벤트, 문구, 기본 언어, 감성 또는 개인 식별 정보 (PII) 와 같은 항목을 찾습니다.

분석 작업을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 분석 작업을 선택한 다음 작업 생성을 선택합니다.
3. 작업 설정에서 분석 작업에 고유한 이름을 지정합니다.
4. 분석 유형에서 내장 분석 유형 중 하나를 선택합니다.

   기본 언어 또는 주제 모델링을 선택하는 경우 다음 단계를 건너뛸 수 있습니다.
5. 선택한 분석 유형에 따라 콘솔에는 다음 추가 필드 중 하나 이상이 표시됩니다.

   - 기본 언어 및 주제 모델링을 제외한 모든 기본 제공 분석 유형에는 언어가 필요합니다.

     입력 문서의 언어를 선택합니다.
   - 이벤트 분석 유형에는 대상 이벤트 유형이 필요합니다.

     입력 문서에서 탐지할 이벤트 유형을 선택합니다. 지원되는 이벤트 유형에 대한 자세한 내용은 이벤트 유형을 참조하세요.
   - PII 분석 유형에는 PII 탐지 설정이 필요합니다.

     출력 모드를 선택합니다. PII 탐지 설정에 대한 자세한 정보는 PII 개체 감지을 참조하세요.
6. 입력 데이터에서 입력 문서의 Amazon S3 내 위치를 지정합니다.

   - 자체 문서를 분석하려면 내 문서를 선택하고 S3 탐색을 선택하여 파일이 들어 있는 버킷 또는 폴더의 경로를 제공합니다.
   - Amazon Comprehend에서 제공하는 샘플을 분석하려면 예제 문서를 선택합니다. 이 경우 Amazon Comprehend는에서 관리하는 버킷을 사용 AWS하며 위치를 지정하지 않습니다.
7. (선택 사항) 입력 형식의 경우 입력 파일에 대해 다음 형식 중 하나를 지정합니다.

   - 파일당 문서 하나 - 각 파일에는 입력 문서 하나가 들어 있습니다. 이 방법은 대용량 문서 모음에 가장 적합합니다.
   - 라인당 하나의 문서 - 하나 이상의 파일이 입력됩니다. 파일의 각 라인은 문서로 간주됩니다. 소셜 미디어 게시물과 같은 짧은 문서에 가장 적합합니다. 각 라인은 줄 바꿈 (LF, \n), 캐리지 리턴 (CR,\ r) 또는 둘 다 (CRLF,\ r\ n) 로 끝나야 합니다. UTF-8 줄 구분자(u+2028)를 사용하여 줄을 끝낼 수는 없습니다.
8. 출력 데이터에서 S3 검색을 선택합니다. Amazon Comprehend가 분석을 통해 생성한 출력 데이터를 기록할 Amazon S3 버킷 또는 폴더를 선택합니다.

9. (선택) 작업의 출력 결과를 암호화하려면 암호화를 선택합니다. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.
   - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID에서 키 별칭 또는 ID를 선택합니다.
   - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ID 아래에 키 별칭 또는 ID의 ARN을 입력합니다.

   > **Note**
   >
   > KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 (키 관리 서비스 (KMS)를 참조하세요.

10. 액세스 권한에는 다음의 IAM 역할을 제공하십시오.
    - 입력 문서의 Amazon S3 위치에 대한 읽기 권한 부여.
    - 입력 문서의 Amazon S3 위치에 대한 쓰기 권한 부여.
    - 또한 `comprehend.amazonaws.com` 서비스 주체가 역할을 수임하고 권한을 획득하는 것을 허용하는 신뢰 정책을 포함시킵니다.

    이러한 권한에 대한 IAM 역할과 적절한 신뢰 정책이 없으면 IAM 역할 생성을 선택하여 새로 만드십시오.
11. 양식 작성을 마치고 나면 작업 생성을 선택하여 주제 탐지 작업을 생성하여 시작합니다.

새 작업이 작업 목록에 나타나고 상태 필드에 작업 상태가 표시됩니다. 이 필드는 처리 중인 작업은 `IN_PROGRESS`, 성공적으로 완료된 작업은 `COMPLETED`, 오류가 있는 작업을 `FAILED`으로 표시합니다. 작업을 클릭하여 오류 메시지를 비롯하여 상세한 작업 정보를 볼 수 있습니다.

작업이 완료되면 Amazon Comprehend는 작업에 대해 지정한 출력 Amazon S3 위치에 분석 결과를 저장합니다. 각 인사이트 유형에 대한 분석 결과 설명은 인사이트을 참조하세요.

Amazon Comprehend API 사용

Amazon Comprehend API는 실시간(동기) 분석을 수행하는 작업과 비동기 분석 작업을 시작하고 관리하는 작업을 지원합니다.

Amazon Comprehend API 작업을 직접 사용하거나, CLI 또는 SDK 중 하나를 사용할 수 있습니다. 이 장의 예제에서는 CLI, Python SDK, 자바 SDK를 사용합니다.

AWS CLI 및 Python 예제를 실행하려면를 설치해야 합니다 AWS CLI. 자세한 내용은 AWS Command Line Interface (AWS CLI) 설정 단원을 참조하십시오.

Java 예제를 실행하려면 AWS SDK for Java를 설치해야 합니다. SDK for Java 설치에 대한 지침은 AWS SDK for Java 설정을 참조하세요.

주제

- AWS SDK에서 Amazon Comprehend 사용
- API를 사용한 실시간 분석
- API를 사용한 비동기 분석 작업

AWS SDK에서 Amazon Comprehend 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 많은 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제

SDK 설명서	코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	Tools for PowerShell 코드 예시
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

가용성 예제

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

API를 사용한 실시간 분석

다음 예제에서는 AWS CLI및 AWS .NET, Java 및 Python용 SDKs를 사용하여 실시간 분석에 Amazon Comprehend API를 사용하는 방법을 보여줍니다. 이러한 예제를 통해 Amazon Comprehend 작업에 대해 알아보고 자체 애플리케이션의 구성 요소로 사용할 수 있습니다.

이 단원의 .NET 예제는 AWS SDK for .NET을 사용합니다. 를 사용하여 .NETAWS Toolkit for Visual Studio을 사용하여 AWS 애플리케이션을 개발할 수 있습니다. 여기에는 애플리케이션을 배포하고 서비스를 관리하는 데 유용한 템플릿과 AWS 탐색기가 포함되어 있습니다. .NET 개발자의 관점은 .NET 개발자 안내서를 AWS참조하세요. AWS

주제

- 지배적 언어 감지
- 명명된 개체 감지

- 핵심 문구 감지
- 감성 확인
- 대상 감성에 대한 실시간 분석
- 구문 감지
- 실시간 배치 API

지배적 언어 감지

텍스트에 사용되는 지배적 언어를 확인하려면 DetectDominantLanguage 작업을 사용합니다. 최대 25개 문서에서 지배적 언어를 일괄 감지하려면 BatchDetectDominantLanguage 작업을 사용합니다. 자세한 내용은 실시간 배치 API 단원을 참조하십시오.

주제

- 사용 AWS Command Line Interface
- AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

사용 AWS Command Line Interface

다음 예제는 AWS CLI로 `DetectDominantLanguage` 작업을 사용하는 방법을 보여 줍니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend detect-dominant-language \
    --region region \
    --text "It is raining today in Seattle."
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "Languages": [
        {
            "LanguageCode": "en",
            "Score": 0.9793661236763
        }
    ]
}
```

AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

지배적 언어를 결정하는 방법에 대한 SDK 예제는 AWS SDK 또는 CLI와 `DetectDominantLanguage` 함께 사용를 참조하세요.

명명된 개체 감지

문서에서 명명된 개체를 확인하려면 DetectEntities 작업을 사용합니다. 최대 25개 문서에서 개체를 일괄 감지하려면 BatchDetectEntities 작업을 사용합니다. 자세한 내용은 실시간 배치 API 단원을 참조하십시오.

주제

- 사용 AWS Command Line Interface
- AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

사용 AWS Command Line Interface

다음 예제는 AWS CLI로 `DetectEntities` 작업을 사용하는 방법을 보여 줍니다. 입력 텍스트의 언어를 지정해야 합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend detect-entities \
    --region region \
    --language-code "en" \
    --text "It is raining today in Seattle."
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "Entities": [
        {
            "Text": "today",
            "Score": 0.97,
            "Type": "DATE",
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
```

```
            "Text": "Seattle",
            "Score": 0.95,
            "Type": "LOCATION",
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ],
    "LanguageCode": "en"
}
```

AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

지배적 언어를 결정하는 방법에 대한 SDK 예제는 AWS SDK 또는 CLI와 `DetectEntities` 함께 사용를 참조하세요.

핵심 문구 감지

텍스트에 사용되는 핵심 명사구를 확인하려면 DetectKeyPhrases 작업을 사용합니다. 최대 25개 문서에서 핵심 명사구를 일괄 감지하려면 BatchDetectKeyPhrases 작업을 사용합니다. 자세한 내용은 실시간 배치 API 단원을 참조하십시오.

주제

- 사용 AWS Command Line Interface
- AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

사용 AWS Command Line Interface

다음 예제는 AWS CLI로 `DetectKeyPhrases` 작업을 사용하는 방법을 보여 줍니다. 입력 텍스트의 언어를 지정해야 합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend detect-key-phrases \
    --region region \
    --language-code "en" \
    --text "It is raining today in Seattle."
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "LanguageCode": "en",
    "KeyPhrases": [
        {
            "Text": "today",
            "Score": 0.89,
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
            "Score": 0.91,
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ]
}
```

AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

핵심 문구를 감지하는 SDK 예제는 AWS SDK 또는 CLI와 `DetectKeyPhrases` 함께 사용를 참조하세요.

감성 확인

Amazon Comprehend는 감성 분석을 위한 다음과 같은 API 작업을 제공합니다.

- DetectSentiment — 문서의 전반적인 감정적 감성을 결정합니다.
- BatchDetectSentition — 최대 25개 문서의 전반적인 감성을 일괄 확인합니다. 자세한 정보는 실시간 배치 API를 참조하세요.
- StartSentimentDetectionJob – 문서 모음에 대해 비동기식 감성 감지 작업을 시작합니다.
- ListSentimentDetectionJobs – 제출한 감성 감지 작업의 목록을 반환합니다.
- DescribeSentimentDetectionJob – 지정된 감성 감지 작업과 관련된 속성(상태 포함)을 가져옵니다.
- StopSentimentDetectionJob – 진행 중인 지정된 감성 작업을 중지합니다.

주제

- 사용 AWS Command Line Interface
- AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

사용 AWS Command Line Interface

다음 예제는 AWS CLI로 `DetectSentiment` 작업을 사용하는 방법을 보여 줍니다. 이 예제는 입력 텍스트의 언어를 지정합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend detect-sentiment \
    --region region \
    --language-code "en" \
    --text "It is raining today in Seattle."
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "SentimentScore": {
        "Mixed": 0.014585512690246105,
        "Positive": 0.31592071056365967,
        "Neutral": 0.5985543131828308,
        "Negative": 0.07093945890665054
    },
    "Sentiment": "NEUTRAL",
    "LanguageCode": "en"
}
```

AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

입력 텍스트의 감성을 결정하는 SDK 예제는 AWS SDK 또는 CLI와 `DetectSentiment` 함께 사용를 참조하세요.

대상 감성에 대한 실시간 분석

Amazon Comprehend는 대상 감성 실시간 분석을 위한 다음과 같은 API 작업을 제공합니다.

- DetectTargetedSentiment — 문서에 언급된 개체의 감성을 분석합니다.
- BatchDetectTargetedSentiment — 최대 25개 문서에 대한 대상 감성을 일괄 분석합니다. 자세한 정보는 실시간 배치 API를 참조하십시오.

분석 중인 텍스트에 대상 감성 개체 유형이 없는 경우 API는 빈 개체 배열을 반환합니다.

사용 AWS Command Line Interface

다음 예제는 AWS CLI로 `DetectTargetedSentiment` 작업을 사용하는 방법을 보여 줍니다. 이 예제는 입력 텍스트의 언어를 지정합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend detect-targeted-sentiment \
    --region region \
    --language-code "en" \
    --text "The burger was cooked perfectly but it was cold. The service was OK."
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
"Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 4,
          "EndOffset": 10,
          "Score": 1,
          "GroupScore": 1,
          "Text": "burger",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0.001515,
              "Negative": 0.000822,
              "Neutral": 0.000243,
              "Positive": 0.99742
            }
          }
        },
        {
          "BeginOffset": 36,
          "EndOffset": 38,
          "Score": 0.999843,
```

```
          "GroupScore": 0.999661,
          "Text": "it",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "NEGATIVE",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0.999996,
              "Neutral": 0.000004,
              "Positive": 0
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 53,
          "EndOffset": 60,
          "Score": 1,
          "GroupScore": 1,
          "Text": "service",
          "Type": "ATTRIBUTE",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0.000033,
              "Negative": 0.000089,
              "Neutral": 0.993325,
              "Positive": 0.006553
            }
          }
        }
      ]
    }
  ]
}
```

구문 감지

텍스트 구문 분석을 통해 개별 단어를 추출하고 각 단어의 품사를 결정하려면 DetectSyntax 작업을 사용합니다. 최대 25개 문서에서 구문 분석을 일괄 수행하려면 BatchDetectSyntax 작업을 사용합니다. 자세한 내용은 실시간 배치 API 단원을 참조하십시오.

주제

- 사용 AWS Command Line Interface.
- AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

사용 AWS Command Line Interface.

다음 예제는 AWS CLI로 `DetectSyntax` 작업을 사용하는 방법을 보여 줍니다. 이 예제는 입력 텍스트의 언어를 지정합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend detect-syntax \
   --region region \
   --language-code "en" \
   --text "It is raining today in Seattle."
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "SyntaxTokens": [
        {
            "Text": "It",
            "EndOffset": 2,
            "BeginOffset": 0,
            "PartOfSpeech": {
                "Tag": "PRON",
                "Score": 0.8389829397201538
            },
            "TokenId": 1
        },
        {
            "Text": "is",
            "EndOffset": 5,
```

```
            "BeginOffset": 3,
            "PartOfSpeech": {
                "Tag": "AUX",
                "Score": 0.9189288020133972
            },
            "TokenId": 2
        },
        {
            "Text": "raining",
            "EndOffset": 13,
            "BeginOffset": 6,
            "PartOfSpeech": {
                "Tag": "VERB",
                "Score": 0.9977611303329468
            },
            "TokenId": 3
        },
        {
            "Text": "today",
            "EndOffset": 19,
            "BeginOffset": 14,
            "PartOfSpeech": {
                "Tag": "NOUN",
                "Score": 0.9993606209754944
            },
            "TokenId": 4
        },
        {
            "Text": "in",
            "EndOffset": 22,
            "BeginOffset": 20,
            "PartOfSpeech": {
                "Tag": "ADP",
                "Score": 0.9999061822891235
            },
            "TokenId": 5
        },
        {
            "Text": "Seattle",
            "EndOffset": 30,
            "BeginOffset": 23,
            "PartOfSpeech": {
                "Tag": "PROPN",
                "Score": 0.9940338730812073
```

```
            },
            "TokenId": 6
        },
        {
            "Text": ".",
            "EndOffset": 31,
            "BeginOffset": 30,
            "PartOfSpeech": {
                "Tag": "PUNCT",
                "Score": 0.9999997615814209
            },
            "TokenId": 7
        }
    ]
}
```

AWS SDK for Java, SDK for Python 또는 사용 SDK for .NET

입력 텍스트의 구문을 감지하는 SDK 예제는 AWS SDK 또는 CLI와 `DetectSyntax` 함께 사용를 참조하세요.

실시간 배치 API

최대 25개 문서의 배치를 전송하려면 Amazon Comprehend 실시간 배치 작업을 사용할 수 있습니다. 배치 호출 작업은 요청의 각 문서에 대해 단일 문서 API를 호출하는 것과 동일합니다. 배치 API를 사용하면 애플리케이션 성능이 향상될 수 있습니다. 자세한 내용은 다중 문서 동기 처리 단원을 참조하십시오.

주제

- 를 사용한 배치 처리 AWS CLI
- 를 사용한 배치 처리 AWS SDK for .NET

를 사용한 배치 처리 AWS CLI

이 예제에서는 AWS Command Line Interface로 배치 API 작업을 사용하는 방법을 보여줍니다. `BatchDetectDominantLanguage`를 제외한 모든 작업은 입력으로 `process.json`이라는 다음과 같은 JSON 파일을 사용합니다. 해당 작업에는 `LanguageCode` 개체가 포함되지 않습니다.

JSON 파일("$$$$$$$$")의 세 번째 문서가 배치 처리 중에 오류를 일으킬 것입니다. 작업이 BatchItemError를 응답에 포함하도록 이 문서를 포함시킨 것입니다.

```
{
   "LanguageCode": "en",
   "TextList": [
      "I have been living in Seattle for almost 4 years",
      "It is raining today in Seattle",
      "$$$$$$$$"
   ]
}
```

Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

주제

- 배치(AWS CLI)를 사용한 지배적 언어 감지
- 배치(AWS CLI)를 사용한 개체 감지
- 배치(AWS CLI)를 사용한 핵심 문구 감지
- 배치(AWS CLI)를 사용한 감성 감지

배치(AWS CLI)를 사용한 지배적 언어 감지

BatchDetectDominantLanguage 작업은 각 문서의 지배적 언어를 일괄 결정합니다. Amazon Comprehend에서 감지할 수 있는 언어 목록은 지배적 언어을 참조하세요. 다음 AWS CLI 명령은 BatchDetectDominantLanguage 작업을 호출합니다.

```
aws comprehend batch-detect-dominant-language \
    --endpoint endpoint \
    --region region \
    --cli-input-json file://path to input file/process.json
```

다음은 BatchDetectDominantLanguage 작업의 응답입니다.

```
{
    "ResultList": [
        {
          "Index": 0,
          "Languages":[
            {
              "LanguageCode":"en",
              "Score": 0.99
```

```
            }
          ]
        },
        {
          "Index": 1
          "Languages":[
            {
              "LanguageCode":"en",
              "Score": 0.82
            }
          ]
        }
    ],
    "ErrorList": [
      {
        "Index": 2,
        "ErrorCode": "InternalServerException",
        "ErrorMessage": "Unexpected Server Error. Please try again."
      }
    ]
}
```

배치(AWS CLI)를 사용한 개체 감지

BatchDetectEntities 작업을 사용하여 문서 배치에 있는 개체를 감지할 수 있습니다. 개체 태그 지정에 대한 자세한 내용은 개체을 참조하세요. 다음 AWS CLI 명령은 BatchDetectEntities 작업을 호출합니다.

```
aws comprehend batch-detect-entities \
    --endpoint endpoint \
    --region region \
    --cli-input-json file://path to input file/process.json
```

배치(AWS CLI)를 사용한 핵심 문구 감지

BatchDetectKeyPhrases 작업은 문서 배치의 핵심 명사구를 반환합니다. 다음 AWS CLI 명령은 BatchDetectKeyNounPhrases 작업을 호출합니다.

```
aws comprehend batch-detect-key-phrases
    --endpoint endpoint
    --region region
    --cli-input-json file://path to input file/process.json
```

배치(AWS CLI)를 사용한 감성 감지

BatchDetectSentiment 작업을 사용하여 문서 배치의 전반적인 감성을 감지합니다. 다음 AWS CLI 명령은 BatchDetectSentiment 작업을 호출합니다.

```
aws comprehend batch-detect-sentiment \
    --endpoint endpoint \
    --region region \
    --cli-input-json file://path to input file/process.json
```

를 사용한 배치 처리 AWS SDK for .NET

다음 샘플 프로그램은 SDK for .NET로 BatchDetectEntities 작업을 사용하는 방법을 보여줍니다. 서버의 응답에는 성공적으로 처리된 각 문서에 대한 BatchDetectEntitiesItemResult 객체가 포함됩니다. 문서를 처리할 때 오류가 발생한 경우 응답의 오류 목록에 기록이 있습니다. 이 예제에서는 오류가 있는 각 문서를 가져와서 재전송합니다.

이 단원의 .NET 예제는 AWS SDK for .NET을 사용합니다. 를 사용하여 .NETAWS Toolkit for Visual Studio을 사용하여 AWS 애플리케이션을 개발할 수 있습니다. 여기에는 애플리케이션을 배포하고 서비스를 관리하는 데 유용한 템플릿과 AWS 탐색기가 포함되어 있습니다. .NET 개발자의 관점은 .NET 개발자 안내서를 AWS참조하세요. AWS

```
using System;
using System.Collections.Generic;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend
{
    class Program
    {
        // Helper method for printing properties
        static private void PrintEntity(Entity entity)
        {
            Console.WriteLine("     Text: {0}, Type: {1}, Score: {2}, BeginOffset: {3}
 EndOffset: {4}",
                entity.Text, entity.Type, entity.Score, entity.BeginOffset,
 entity.EndOffset);
        }

        static void Main(string[] args)
        {
```

```
            AmazonComprehendClient comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            List<String> textList = new List<String>()
            {
                { "I love Seattle" },
                { "Today is Sunday" },
                { "Tomorrow is Monday" },
                { "I love Seattle" }
            };

            // Call detectEntities API
            Console.WriteLine("Calling BatchDetectEntities");
            BatchDetectEntitiesRequest batchDetectEntitiesRequest = new
BatchDetectEntitiesRequest()
            {
                TextList = textList,
                LanguageCode = "en"
            };
            BatchDetectEntitiesResponse batchDetectEntitiesResponse =
comprehendClient.BatchDetectEntities(batchDetectEntitiesRequest);

            foreach (BatchDetectEntitiesItemResult item in
batchDetectEntitiesResponse.ResultList)
            {
                Console.WriteLine("Entities in {0}:", textList[item.Index]);
                foreach (Entity entity in item.Entities)
                    PrintEntity(entity);
            }

            // check if we need to retry failed requests
            if (batchDetectEntitiesResponse.ErrorList.Count != 0)
            {
                Console.WriteLine("Retrying Failed Requests");
                List<String> textToRetry = new List<String>();
                foreach(BatchItemError errorItem in
batchDetectEntitiesResponse.ErrorList)
                    textToRetry.Add(textList[errorItem.Index]);

                batchDetectEntitiesRequest = new BatchDetectEntitiesRequest()
                {
                    TextList = textToRetry,
                    LanguageCode = "en"
                };
```

```
                batchDetectEntitiesResponse =
 comprehendClient.BatchDetectEntities(batchDetectEntitiesRequest);

                foreach(BatchDetectEntitiesItemResult item in
 batchDetectEntitiesResponse.ResultList)
                {
                    Console.WriteLine("Entities in {0}:", textList[item.Index]);
                    foreach (Entity entity in item.Entities)
                        PrintEntity(entity);
                }
            }
            Console.WriteLine("End of DetectEntities");
        }
    }
}
```

API를 사용한 비동기 분석 작업

다음 예제는 Amazon Comprehend 비동기 API를 사용하여 AWS CLI를 사용한 분석 작업을 생성하고 관리합니다.

주제

- Amazon Comprehend 인사이트에 사용되는 비동기 분석
- 대상 감성에 대한 비동기 분석
- 이벤트 감지를 위한 비동기 분석
- 주제 모델링을 위한 비동기 분석

Amazon Comprehend 인사이트에 사용되는 비동기 분석

다음 섹션에서는 Amazon Comprehend API를 사용하여 Amazon Comprehend 인사이트를 분석하기 위한 비동기 작업을 실행합니다.

주제

- 사전 조건
- 분석 시작
- 모니터링 분석 작업
- 분석 결과 가져오기

사전 조건

문서는 UTF-8 형식 또는 파일이어야 합니다. 문서를 두 가지 형식으로 제출할 수 있습니다. 사용하는 형식은 다음 표의 설명대로 분석하려는 문서의 유형에 따라 달라집니다.

설명	형식
각 파일에는 입력 문서가 한 개씩 들어 있습니다. 이 방법은 대용량 문서 모음에 가장 적합합니다.	파일당 문서 하나
입력은 하나 이상의 파일입니다. 파일의 각 라인은 문서로 간주됩니다. 소셜 미디어 게시물과 같은 짧은 문서에 가장 적합합니다. 각 라인은 줄 바꿈 (LF, \n), 캐리지 리턴 (CR, \ r) 또는 둘 다 (CRLF,\ r\ n) 로 끝나야 합니다. UTF-8 줄 구분자(u+2028)를 사용하여 줄을 끝낼 수는 없습니다.	라인당 문서 하나

분석 작업을 시작할 때 입력 데이터의 S3 위치를 지정해야 합니다. URI는 호출 중인 API 엔드포인트와 동일한 AWS 리전에 있어야 합니다. URI는 단일 파일을 가리킬 수도 있고 데이터 파일 컬렉션의 접두사일 수도 있습니다. 자세한 정보는 InputDataConfig 데이터 유형을 참조하세요.

Amazon Comprehend에 문서 컬렉션 및 출력 파일을 포함하는 Amazon S3 버킷에 대한 액세스 권한을 부여해야 합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.

분석 시작

분석 작업을 제출하려면 Amazon Comprehend 콘솔 또는 적절한 `Start*` 작업을 사용하십시오.

- StartDominantLanguageDetectionJob — 컬렉션의 각 문서에서 지배적 언어를 감지하는 작업을 시작합니다. 문서의 지배적 언어에 대한 자세한 정보는 지배적 언어을 참조하세요.
- StartEntitiesDetectionJob — 컬렉션의 각 문서에서 개체를 감지하는 작업을 시작합니다. 개체 태그 지정에 대한 자세한 내용은 개체을 참조하세요.
- StartKeyPhrasesDetectionJob — 컬렉션의 각 문서에서 핵심 문구를 감지하는 작업을 시작합니다. 키 이름에 대한 자세한 내용은 핵심 문구을 참조하세요.

- StartPiiEntitiesDetectionJob — 컬렉션의 각 문서에서 개인 식별 정보(PII)를 감지하는 작업을 시작합니다. ACL에 대한 자세한 내용은 PII 개체 감지을 참조하세요.
- StartSentimentDetectionJob — 컬렉션의 각 문서에서 감성을 감지하는 작업을 시작합니다. 감성에 대한 자세한 내용은 감성을 참조하세요.

모니터링 분석 작업

`Start*` 작업은 작업 진행 상황을 모니터링하는 데 사용할 수 있는 ID를 반환합니다.

API를 사용하여 진행 상황을 모니터링하려면 개별 작업의 진행 상황을 모니터링할지, 아니면 여러 작업의 진행 상황을 모니터링할지에 따라 두 작업 중 하나를 사용합니다.

개별 분석 작업의 진행 상황을 모니터링하려면 `Describe*` 작업을 사용하십시오. `Start*` 작업에서 반환된 작업 ID를 제공해야 합니다. `Describe*` 작업의 응답에는 작업 상태가 표시된 `JobStatus` 필드가 포함됩니다.

여러 분석 작업의 진행 상황을 모니터링하려면 `List*` 작업을 사용하십시오. `List*` 작업은 Amazon Comprehend에 제출한 작업 목록을 반환합니다. 응답에는 각 작업에 대한 작업 상태를 알려주는 `JobStatus` 필드가 포함됩니다.

상태 필드가 `COMPLETED` 또는 `FAILED`로 설정된 경우 작업 처리가 완료된 것입니다.

개별 작업의 상태를 확인하려면 수행 중인 분석에 `Describe*` 작업을 사용하십시오.

- DescribeDominantLanguageDetectionJob
- DescribeEntitiesDetectionJob
- DescribeKeyPhrasesDetectionJob
- DescribePiiEntitiesDetectionJob
- DescribeSentimentDetectionJob

여러 작업의 상태를 확인하려면 수행 중인 분석에 `List*` 작업을 사용하십시오.

- ListDominantLanguageDetectionJobs
- ListEntitiesDetectionJobs
- ListKeyPhrasesDetectionJobs
- ListPiiEntitiesDetectionJobs

- ListSentimentDetectionJobs

결과를 특정 기준과 일치하는 작업으로 제한하려면 `List*` 작업의 `Filter` 파라미터를 사용합니다. 작업 이름, 작업 상태 또는 작업이 제출된 날짜와 시간을 기준으로 결과를 필터링할 수 있습니다. 자세한 내용은 Amazon Comprehend API 참조에서 각 `List*` 작업의 `Filter` 파라미터를 참조하세요.

분석 결과 가져오기

분석 작업을 완료한 후 `Describe*` 작업을 사용하여 결과의 위치를 가져옵니다. 작업 상태가 `COMPLETED`인 경우 응답에는 출력 파일의 Amazon S3 위치가 있는 필드가 포함된 `OutputDataConfig` 필드가 포함됩니다. 파일 `output.tar.gz`는 분석 결과를 포함하는 압축된 아카이브입니다.

배치 번역 작업의 상태가 `FAILED`인 경우 작업 응답에는 작업이 완료되지 않은 이유를 설명하는 필드가 `Message` 포함되어 있습니다.

개별 작업의 상태를 확인하려면 적절한 `Describe*` 작업을 사용하십시오.

- DescribeDominantLanguageDetectionJob
- DescribeEntitiesDetectionJob
- DescribeKeyPhrasesDetectionJob
- DescribeSentimentDetectionJob

결과는 문서당 하나의 JSON 구조를 갖는 단일 파일로 반환됩니다. 각 응답 파일에는 상태 필드가 `FAILED`(으)로 설정된 모든 작업에 대한 오류 메시지도 포함됩니다.

다음 각 섹션은 두 입력 형식의 출력 예시를 보여줍니다.

지배적 언어 감지 결과 가져오기

다음은 지배적 언어를 감지한 분석의 출력 파일 예제입니다. 입력 형식은 한 줄에 한 문서입니다. 자세한 내용은 DetectDominantLanguage을 참조하세요.

```
{"File": "0_doc", "Languages": [{"LanguageCode": "en", "Score": 0.9514502286911011},
 {"LanguageCode": "de", "Score": 0.02374090999364853}, {"LanguageCode": "nl", "Score":
 0.003208699868991971}, "Line": 0}
{"File": "1_doc", "Languages": [{"LanguageCode": "en", "Score": 0.9822712540626526},
 {"LanguageCode": "de", "Score": 0.002621392020955682}, {"LanguageCode": "es", "Score":
 0.002386554144322872}], "Line": 1}
```

다음은 입력 형식이 파일당 문서 하나인 분석 결과의 예시입니다.

```
{"File": "small_doc", "Languages": [{"LanguageCode": "en", "Score":
 0.9728053212165833}, {"LanguageCode": "de", "Score": 0.007670710328966379},
 {"LanguageCode": "es", "Score": 0.002847236813977360 7}]}
{"File": "huge_doc", "Languages": [{"LanguageCode": "en", "Score": 0.984955906867981},
 {"LanguageCode": "de", "Score": 0.0026436643674969673}, {"LanguageCode": "fr",
 "Score": 0.0014206881169229746}]}
```

개체 감지 결과 가져오기

다음은 문서에서 개체를 감지한 분석 결과 파일의 예시입니다. 입력 형식은 한 줄에 한 문서입니다. 자세한 정보는 DetectEntities 작업을 참조하세요. 출력에는 두 개의 오류 메시지가 포함됩니다. 하나는 너무 긴 문서이고, 다른 하나는 UTF-8 형식이 아닌 문서에 대한 것입니다.

```
{"File": "50_docs", "Line": 0, "Entities": [{"BeginOffset": 0, "EndOffset": 22,
 "Score": 0.9763959646224976, "Text": "Cluj-NapocaCluj-Napoca", "Type": "LOCATION"}"]}
{"File": "50_docs", "Line": 1, "Entities": [{"BeginOffset": 11, "EndOffset": 15,
 "Score": 0.9615424871444702, "Text": "Maat", "Type": "PERSON"}}]}
{"File": "50_docs", "Line": 2, "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage":
 "Document size exceeds maximum size limit 102400 bytes."}
{"File": "50_docs", "Line": 3, "ErrorCode": "UNSUPPORTED_ENCODING", "ErrorMessage":
 "Document is not in UTF-8 format and all subsequent lines are ignored."}
```

다음은 입력 형식이 파일당 문서 하나인 분석으로부터의 출력 결과의 예입니다. 출력에는 두 개의 오류 메시지가 포함됩니다. 하나는 너무 긴 문서이고, 다른 하나는 UTF-8 형식이 아닌 문서에 대한 것입니다.

```
{"File": "non_utf8.txt", "ErrorCode": "UNSUPPORTED_ENCODING", "ErrorMessage": "Document
 is not in UTF-8 format and all subsequent line are ignored."}
{"File": "small_doc", "Entities": [{"BeginOffset": 0, "EndOffset": 4, "Score":
 0.645766019821167, "Text": "Maat", "Type": "PERSON"}]}
{"File": "huge_doc", "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage": "Document
 size exceeds size limit 102400 bytes."}
```

핵심 문구 감지 결과 가져오기

다음은 문서에서 핵심 문구를 감지한 분석 결과 파일의 예시입니다. 입력 형식은 한 줄에 한 문서입니다. 자세한 내용은 DetectKeyPhrases를 참조하세요.

```
{"File": "50_docs", "KeyPhrases": [{"BeginOffset": 0, "EndOffset": 22, "Score":
 0.8948641419410706, "Text": "Cluj-NapocaCluj-Napoca"}, {"BeginOffset": 45,
 "EndOffset": 49, "Score": 0.9989854693412781, "Text": "Cluj"}], "Line": 0}
```

다음은 입력 형식이 파일당 문서 하나인 분석 결과의 예시입니다.

```
{"File": "1_doc", "KeyPhrases": [{"BeginOffset": 0, "EndOffset": 22, "Score":
 0.8948641419410706, "Text": "Cluj-NapocaCluj-Napoca"}, {"BeginOffset": 45,
 "EndOffset": 49, "Score": 0.9989854693412781, "Text": "Cluj"}]}
```

개인 식별 정보(PII) 감지 결과 가져오기

다음은 문서에서 PII 개체를 감지한 분석 작업의 출력 파일 예제입니다. 입력 형식은 한 줄에 한 문서입니다.

```
{"Entities":[{"Type":"NAME","BeginOffset":40,"EndOffset":69,"Score":0.999995},
{"Type":"ADDRESS","BeginOffset":247,"EndOffset":253,"Score":0.998828},
{"Type":"BANK_ACCOUNT_NUMBER","BeginOffset":406,"EndOffset":411,"Score":0.693283}],"File":"doc.
{"Entities":[{"Type":"SSN","BeginOffset":1114,"EndOffset":1124,"Score":0.999999},
{"Type":"EMAIL","BeginOffset":3742,"EndOffset":3775,"Score":0.999993},
{"Type":"PIN","BeginOffset":4098,"EndOffset":4102,"Score":0.999995}],"File":"doc.txt","Line":1}
```

다음은 입력 형식이 파일당 문서 하나인 분석으로부터의 출력 결과의 예입니다.

```
{"Entities":[{"Type":"NAME","BeginOffset":40,"EndOffset":69,"Score":0.999995},
{"Type":"ADDRESS","BeginOffset":247,"EndOffset":253,"Score":0.998828},
{"Type":"BANK_ROUTING","BeginOffset":279,"EndOffset":289,"Score":0.999999}],"File":"doc.txt"}
```

감성 감지 결과 가져오기

다음은 문서에 표현된 감성을 감지한 분석의 결과 파일 예제입니다. 문서 하나가 너무 길기 때문에 오류 메시지가 포함됩니다. 입력 형식은 한 줄에 한 문서입니다. 자세한 내용은 DetectSentiment 작업을 참조하세요.

```
{"File": "50_docs", "Line": 0, "Sentiment": "NEUTRAL", "SentimentScore": {"Mixed":
 0.002734508365392685, "Negative": 0.008935936726629734, "Neutral": 0.9841893315315247,
 "Positive": 0.004140198230743408}}
{"File": "50_docs", "Line": 1, "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage":
 "Document size is exceeded maximum size limit 5120 bytes."}
```

```
{"File": "50_docs", "Line": 2, "Sentiment": "NEUTRAL", "SentimentScore":
 {"Mixed": 0.0023119584657251835, "Negative": 0.0029857370536774397, "Neutral":
 0.9866572022438049, "Positive": 0.00804515462368726 7}}
```

다음은 입력 형식이 파일당 문서 하나인 분석 결과의 예시입니다.

```
{"File": "small_doc", "Sentiment": "NEUTRAL", "SentimentScore": {"Mixed":
 0.0023450672160834074, "Negative": 0.0009663937962614, "Neutral": 0.9795311689376831,
 "Positive": 0.017157377675175667}}
{"File": "huge_doc", "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage": "Document
 size is exceeds the limit of 5120 bytes."}
```

대상 감성에 대한 비동기 분석

대상 감성의 실시간 분석에 대한 자세한 내용은 the section called “대상 감성에 대한 실시간 분석”를 참조하세요.

Amazon Comprehend는 비동기 대상 감성 분석을 시작하고 관리하기 위한 다음 API 작업을 제공합니다.

- StartTargetedSentimentDetectionJob – 문서 모음에 대해 비동기 대상 감정 감지 작업 시작을 시작합니다.
- ListTargetedSentimentDetectionJobs – 제출한 대상 감성 감지 작업의 목록을 반환합니다.
- DescribeTargetedSentimentDetectionJob – 지정된 대상 감성 감지 작업과 관련된 속성(상태 포함)을 가져옵니다.
- StopTargetedSentimentDetectionJob – 진행 중인 지정된 대상 감성 작업을 중지합니다.

주제

시작하기 전에

시작하기 전에 다음 사항을 확인해야 합니다.

- 입력 및 출력 버킷 - 입력 및 출력에 사용할 Amazon S3 버킷을 식별합니다. 버킷은 사용자가 호출하는 API와 동일한 리전에 있어야 합니다.

- IAM 서비스 역할 - 입력 및 출력 버킷에 액세스하려면 권한이 있는 IAM 서비스 역할이 필요합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.

AWS CLI를 사용한 대상 감성 분석.

다음 예제는 AWS CLI로 `StartTargetedSentimentDetectionJob` 작업을 사용하는 방법을 보여줍니다. 이 예제는 입력 텍스트의 언어를 지정합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend start-targeted-sentiment-detection-job \
       --job-name "job name" \
       --language-code "en" \
       --cli-input-json file://path to JSON input file
```

`cli-input-json` 파라미터에는 다음 예시에 표시된 대로 요청 데이터가 포함된 JSON 파일의 경로를 제공해야 합니다.

```
{
    "InputDataConfig": {
        "S3Uri": "s3://input bucket/input path",
        "InputFormat": "ONE_DOC_PER_FILE"
    },
    "OutputDataConfig": {
        "S3Uri": "s3://output bucket/output path"
    },
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role"
}
```

작업 시작 요청이 성공하면 다음과 같은 응답을 받게 됩니다.

```
{
    "JobStatus": "SUBMITTED",
    "JobArn": "job ARN"
    "JobId": "job ID"
}
```

이벤트 감지를 위한 비동기 분석

주제

문서 세트의 이벤트를 감지하려면 StartEventsDetectionJob을 사용하여 비동기 작업을 시작하십시오.

시작하기 전에

시작하기 전에 다음 사항을 확인해야 합니다.

- 입력 및 출력 버킷 - 입력 및 출력에 사용할 Amazon S3 버킷을 식별합니다. 버킷은 사용자가 호출하는 API와 동일한 리전에 있어야 합니다.
- IAM 서비스 역할 - 입력 및 출력 버킷에 액세스하려면 권한이 있는 IAM 서비스 역할이 필요합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한 단원을 참조하십시오.

를 사용하여 이벤트 감지 AWS CLI

다음 예제에서는에서 StartEventsDetectionJob 작업을 사용하는 방법을 보여줍니다. AWS CLI

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend start-events-detection-job \
  --region region \
  --job-name job name \
  --cli-input-json file://path to JSON input file
```

`cli-input-json` 파라미터에는 다음 예시에 표시된 대로 요청 데이터가 포함된 JSON 파일의 경로를 제공해야 합니다.

```
{
  "InputDataConfig": {
      "S3Uri": "s3://input bucket/input path",
```

```
        "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
        "S3Uri": "s3://output bucket/output path"
    },
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role"
    "LanguageCode": "en",
    "TargetEventTypes": [
        "BANKRUPTCY",
        "EMPLOYMENT",
        "CORPORATE_ACQUISITION",
        "INVESTMENT_GENERAL",
        "CORPORATE_MERGER",
        "IPO",
        "RIGHTS_ISSUE",
        "SECONDARY_OFFERING",
        "SHELF_OFFERING",
        "TENDER_OFFERING",
        "STOCK_SPLIT"
    ]
}
```

이벤트 감지 작업 시작 요청이 성공하면 다음과 같은 응답을 받게 됩니다.

```
{
    "JobStatus": "SUBMITTED",
    "JobId": "job ID"
}
```

를 사용하여 이벤트 나열 AWS CLI

제출한 이벤트 감지 작업의 목록을 보려면 ListEventsDetectionJobs 작업을 사용하십시오. 이 목록에는 사용한 입력 및 출력 위치와 각 감지 작업의 상태에 대한 정보가 포함됩니다. 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend list-events-detection-jobs --region region
```

응답은 다음과 비슷한 JSON 형식으로 받게 됩니다.

```
{
```

```
 "EventsDetectionJobPropertiesList": [
    {
       "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role",
       "EndTime": timestamp,
       "InputDataConfig": {
          "InputFormat": "ONE_DOC_PER_LINE",
          "S3Uri": "s3://input bucket/input path"
       },
       "JobId": "job ID",
       "JobName": "job name",
       "JobStatus": "COMPLETED",
       "LanguageCode": "en",
       "Message": "message",
       "OutputDataConfig": {
          "S3Uri": "s3://output bucket/ouput path"
       },
       "SubmitTime": timestamp,
       "TargetEventTypes": [
         "BANKRUPTCY",
         "EMPLOYMENT",
         "CORPORATE_ACQUISITION",
         "INVESTMENT_GENERAL",
         "CORPORATE_MERGER",
         "IPO",
         "RIGHTS_ISSUE",
         "SECONDARY_OFFERING",
         "SHELF_OFFERING",
         "TENDER_OFFERING",
         "STOCK_SPLIT"
  ]
    }
 ],
 "NextToken": "next token"
}
```

를 사용하여 이벤트 설명 AWS CLI

DescribeEventsDetectionJob 작업을 사용하여 기존 작업의 상태를 얻을 수 있습니다. 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend describe-events-detection-job \
  --region region \
```

```
  --job-id job ID
```

응답은 다음과 같은 JSON 형식으로 받게 됩니다.

```
{
 "EventsDetectionJobProperties": {
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role",
    "EndTime": timestamp,
    "InputDataConfig": {
       "InputFormat": "ONE_DOC_PER_LINE",
       "S3Uri": "S3Uri": "s3://input bucket/input path"
    },
    "JobId": "job ID",
    "JobName": "job name",
    "JobStatus": "job status",
    "LanguageCode": "en",
    "Message": "message",
    "OutputDataConfig": {
       "S3Uri": "s3://output bucket/output path"
    },
    "SubmitTime": timestamp,
    "TargetEventTypes": [
      "BANKRUPTCY",
      "EMPLOYMENT",
      "CORPORATE_ACQUISITION",
      "INVESTMENT_GENERAL",
      "CORPORATE_MERGER",
      "IPO",
      "RIGHTS_ISSUE",
      "SECONDARY_OFFERING",
      "SHELF_OFFERING",
      "TENDER_OFFERING",
      "STOCK_SPLIT"
  ]
 }
}
```

이벤트 감지 결과 가져오기

다음은 문서에서 이벤트를 감지한 분석 작업의 출력 파일 예제입니다. 입력 형식은 한 줄에 한 문서입니다.

```
{"Entities": [{"Mentions": [{"BeginOffset": 12, "EndOffset": 27, "GroupScore":
 1.0, "Score": 0.916355, "Text": "over a year ago", "Type": "DATE"}]}, {"Mentions":
 [{"BeginOffset": 33, "EndOffset": 39, "GroupScore": 1.0, "Score": 0.996603,
 "Text": "Amazon", "Type": "ORGANIZATION"}]}, {"Mentions": [{"BeginOffset":
 66, "EndOffset": 77, "GroupScore": 1.0, "Score": 0.999283, "Text": "Whole
 Foods", "Type": "ORGANIZATION"}]}], "Events": [{"Arguments": [{"EntityIndex":
 2, "Role": "INVESTEE", "Score": 0.999283}, {"EntityIndex": 0, "Role": "DATE",
 "Score": 0.916355}, {"EntityIndex": 1, "Role": "INVESTOR", "Score": 0.996603}],
 "Triggers": [{"BeginOffset": 373, "EndOffset": 380, "GroupScore": 0.999984,
 "Score": 0.999955, "Text": "acquire", "Type": "CORPORATE_ACQUISITION"}], "Type":
 "CORPORATE_ACQUISITION"}, {"Arguments": [{"EntityIndex": 2, "Role": "PARTICIPANT",
 "Score": 0.999283}], "Triggers": [{"BeginOffset": 115, "EndOffset": 123, "GroupScore":
 1.0, "Score": 0.999967, "Text": "combined", "Type": "CORPORATE_MERGER"}], "Type":
 "CORPORATE_MERGER"}], "File": "doc.txt", "Line": 0}
```


이벤트 출력 파일 구조와 지원되는 이벤트 유형에 대한 자세한 내용은 이벤트을 참조하세요.

주제 모델링을 위한 비동기 분석

문서 세트의 주제를 결정하려면 StartTopicsDetectionJob을 사용하여 비동기 작업을 시작하십시오. 영어 또는 스페인어로 작성된 문서의 주제를 모니터링할 수 있습니다.

주제

- 시작하기 전에
- 사용 AWS Command Line Interface
- SDK for Python 또는 사용 SDK for .NET

시작하기 전에

시작하기 전에 다음 사항을 확인해야 합니다.

- 입력 및 출력 버킷 - 입력 및 출력에 사용할 Amazon S3 버킷을 식별합니다. 버킷은 사용자가 호출하는 API와 동일한 리전에 있어야 합니다.
- IAM 서비스 역할 - 입력 및 출력 버킷에 액세스하려면 권한이 있는 IAM 서비스 역할이 필요합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한 단원을 참조하십시오.

사용 AWS Command Line Interface

다음 예제에서는에서 StartTopicsDetectionJob 작업을 사용하는 방법을 보여줍니다. AWS CLI

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend start-topics-detection-job \
                --number-of-topics topics to return \
                --job-name "job name" \
                --region region \
                --cli-input-json file://path to JSON input file
```

cli-input-json 파라미터에는 다음 예시에 표시된 대로 요청 데이터가 포함된 JSON 파일의 경로를 제공해야 합니다.

```
{
    "InputDataConfig": {
        "S3Uri": "s3://input bucket/input path",
        "InputFormat": "ONE_DOC_PER_FILE"
    },
    "OutputDataConfig": {
        "S3Uri": "s3://output bucket/output path"
    },
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role"
}
```

주제 감지 작업 시작 요청이 성공하면 다음과 같은 응답을 받게 됩니다.

```
{
    "JobStatus": "SUBMITTED",
    "JobId": "job ID"
}
```

제출한 주제 감지 작업의 목록을 보려면 ListTopicsDetectionJobs 작업을 사용하십시오. 이 목록에는 사용한 입력 및 출력 위치와 각 감지 작업의 상태에 대한 정보가 포함됩니다. 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend list-topics-detection-jobs \-- region
```

응답은 다음과 비슷한 JSON 형식으로 받게 됩니다.

```
{
    "TopicsDetectionJobPropertiesList": [
        {
            "InputDataConfig": {
                "S3Uri": "s3://input bucket/input path",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "NumberOfTopics": topics to return,
            "JobId": "job ID",
            "JobStatus": "COMPLETED",
            "JobName": "job name",
            "SubmitTime": timestamp,
            "OutputDataConfig": {
                "S3Uri": "s3://output bucket/output path"
            },
            "EndTime": timestamp
        },
        {
            "InputDataConfig": {
                "S3Uri": "s3://input bucket/input path",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "NumberOfTopics": topics to return,
            "JobId": "job ID",
            "JobStatus": "RUNNING",
            "JobName": "job name",
            "SubmitTime": timestamp,
            "OutputDataConfig": {
                "S3Uri": "s3://output bucket/output path"
            }
        }
    ]
}
```

DescribeTopicsDetectionJob 작업을 사용하여 기존 작업의 상태를 가져올 수 있습니다. 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend describe-topics-detection-job --job-id job ID
```

응답은 다음과 같은 JSON 형식으로 받게 됩니다.

```
{
    "TopicsDetectionJobProperties": {
        "InputDataConfig": {
            "S3Uri": "s3://input bucket/input path",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "NumberOfTopics": topics to return,
        "JobId": "job ID",
        "JobStatus": "COMPLETED",
        "JobName": "job name",
        "SubmitTime": timestamp,
        "OutputDataConfig": {
            "S3Uri": "s3://output bucket/ouput path"
        },
        "EndTime": timestamp
    }
}
```

SDK for Python 또는 사용 SDK for .NET

주제 모델링 작업을 시작하는 방법에 대한 SDK 예제는 AWS SDK 또는 CLI와 `StartTopicsDetectionJob` 함께 사용를 참조하세요.

신뢰와 안전

사용자는 온라인 애플리케이션(예: P2P 채팅 및 포럼 토론), 웹 사이트에 게시된 의견, 생성형 AI 애플리케이션(생성형 AI 모델의 입력 프롬프트 및 출력)을 통해 대량의 텍스트 콘텐츠를 생성합니다. Amazon Comprehend 신뢰와 안전 기능은 이 콘텐츠를 조정하여 사용자에게 안전하고 포괄적인 환경을 제공하는 데 도움이 될 수 있습니다.

Amazon Comprehend 신뢰와 안전 기능을 사용하면 다음과 같은 이점이 있습니다.

- 신속한 조정: 온라인 플랫폼에 부적절한 콘텐츠가 없도록 대량의 텍스트를 빠르고 정확하게 조정합니다.
- 사용자 지정 가능: API 응답의 중재 임계값을 애플리케이션 요구 사항에 맞게 사용자 지정할 수 있습니다.
- 사용하기 쉬움: LangChain 통합을 통해 또는 AWS CLI SDKs를 사용하여 신뢰 및 안전 기능을 구성합니다.

Amazon Comprehend 신뢰와 안전은 콘텐츠 조정과 관련된 다음과 같은 측면을 다룹니다.

- Toxicity detection – 유해하거나 불쾌하거나 부적절할 수 있는 콘텐츠를 감지합니다. 증오심 표현, 위협 또는 학대 등이 그 예입니다.
- Intent classification – 명시적 또는 묵시적인 악의적 의도가 있는 콘텐츠를 감지합니다. 차별적이거나 불법적인 콘텐츠, 의학적, 법적, 정치적, 논란의 여지가 있는, 개인적 또는 재정적 주제에 대해 표현하거나 조언을 요청하는 콘텐츠를 예로 들 수 있습니다.
- Privacy protection – 사용자가 실수로 개인 식별 정보(PII)가 드러나는 콘텐츠를 제공할 수 있습니다. Amazon Comprehend PII는 PII를 감지하고 수정하는 기능을 제공합니다.

주제

유해성 감지

Amazon Comprehend 유해성 감지는 텍스트 기반 상호 작용에서 유해 콘텐츠를 실시간으로 감지하는 기능을 제공합니다. 유해성 감지를 사용하여 온라인 플랫폼에서 P2P 대화를 조정하거나, 생성형 AI의 입력 및 출력을 모니터링할 수 있습니다.

유해성 감지는 다음과 같은 범주의 불쾌한 콘텐츠를 감지합니다.

그래픽

그래픽 스피치는 시각적으로 묘사적이고 상세하며 불쾌할 정도로 생생한 이미지를 사용합니다. 이러한 언어는 받는 사람에 대한 모욕, 불쾌감 또는 피해를 증폭시키기 위해 장황하게 표현되는 경우가 많습니다.

괴롭힘_또는_학대

의도와 상관없이 말하는 사람과 듣는 사람 사이에 파괴적인 권력 역학을 강요하는 말은 받는 사람의 심리적 안녕에 영향을 주거나 사람을 대상화합니다.

헤이트_스피치

인종, 민족, 성 정체성, 종교, 성적 취향, 능력, 출신 국가 또는 기타 정체성 집단을 근거로 특정 개인이나 집단을 비판, 모욕, 비인간화하는 언어.

모욕

비하하거나, 굴욕하거나, 조롱하거나, 모욕하거나, 얕보는 언어가 포함된 말.

욕설

무례하거나 저속하거나 불쾌감을 주는 단어, 문구 또는 두문자어가 포함된 연설은 욕설로 간주됩니다.

성적

신체 부위, 신체적 특징 또는 성별을 직간접적으로 언급하여 성적 관심, 활동 또는 흥분을 나타내는 언어.

폭력_또는_위협

개인이나 집단에 대해 고통, 부상 또는 적대감을 주려는 위협을 포함하는 언어.

유해성

위의 모든 범주에서 유해하다고 간주될 수 있는 단어, 구문 또는 두문자어가 포함된 언어.

API를 사용한 유해 콘텐츠 감지

텍스트에서 유해 콘텐츠를 감지하려면 동기식 DetectToxicContent 작업을 사용하십시오. 이 작업은 입력으로 제공받은 텍스트 문자열 목록을 분석합니다. API 응답에는 입력 목록의 크기와 일치하는 결과 목록이 포함됩니다.

현재 유해 콘텐츠 감지는 영어만 지원합니다. 입력 텍스트의 경우 최대 10개의 텍스트 문자열 목록을 제공할 수 있습니다. 각 문자열의 최대 크기는 1KB입니다.

유해 콘텐츠 감지는 각 입력 문자열에 대해 목록당 한 항목씩 분석 결과 목록을 반환합니다. 항목에는 텍스트 문자열에서 식별된 유해 콘텐츠 유형 목록과 각 콘텐츠 유형에 대한 신뢰도 점수가 포함됩니다. 항목에는 문자열에 대한 유해성 점수도 포함됩니다.

다음 예제는 AWS CLI 및 Python을 이용한 `DetectToxicContent` 작업을 사용하는 방법을 보여 줍니다.

AWS CLI

AWS CLI에서 다음 명령을 사용하여 유해 콘텐츠를 감지할 수 있습니다.

```
aws comprehend detect-toxic-content --language-code en  /
            --text-segments "[{\"Text\":\"You are so obtuse\"}]"
```

는 다음 결과로 AWS CLI 응답합니다. 텍스트 세그먼트는 `INSULT` 카테고리에서 높은 신뢰도 점수를 받아 유해성 점수도 높습니다.

```
{
   "ResultList": [
      {
         "Labels": [
                {
                    "Name": "PROFANITY",
                    "Score": 0.0006000000284984708
                },
                {
                    "Name": "HATE_SPEECH",
                    "Score": 0.00930000003427267
                },
                {
                    "Name": "INSULT",
                    "Score": 0.9204999804496765
                },
```

```
                {
                    "Name": "GRAPHIC",
                    "Score": 9.999999747378752e-05
                },
                {
                    "Name": "HARASSMENT_OR_ABUSE",
                    "Score": 0.0052999998442828655
                },
                {
                    "Name": "SEXUAL",
                    "Score": 0.01549999974668026
                },
                {
                    "Name": "VIOLENCE_OR_THREAT",
                    "Score": 0.007799999788403511
                }
            ],
            "Toxicity": 0.7192999720573425
        }
    ]
}
```

text-segments 파라미터에 다음 형식을 사용하여 최대 10개의 텍스트 문자열을 입력할 수 있습니다.

```
    --text-segments "[{\"Text\":\"text string 1\"},
                     {\"Text\":\"text string2\"},
                     {\"Text\":\"text string3\"}]"
```

는 다음 결과로 AWS CLI 응답합니다.

```
{
    "ResultList": [
        {
            "Labels": [ (truncated) ],
                "Toxicity": 0.3192999720573425
        },
        {
            "Labels": [ (truncated) ],
                "Toxicity": 0.1192999720573425
        },
```

```
      {
         "Labels": [ (truncated) ],
            "Toxicity": 0.0192999720573425
      }
   ]
}
```

Python (Boto)

다음 예제는 Python을 사용하여 유해 콘텐츠를 감지하는 방법을 보여줍니다.

```
import boto3
client = boto3.client(
    service_name='comprehend',
    region_name=region) # For example, 'us-west-2'

response = client.detect_toxic_content(
    LanguageCode='en',
    TextSegments=[{'Text': 'You are so obtuse'}]
)
print("Response: %s\n" % response)
```

신속한 안전성 분류

Amazon Comprehend는 대규모 언어 모델(LLM) 또는 기타 생성형 AI 모델의 일반 텍스트 입력 프롬프트를 분류하기 위해 사전 훈련된 이진 분류기를 제공합니다.

프롬프트 안전성 분류기는 입력 프롬프트를 분석하고 프롬프트가 안전한지, 안전하지 않은지에 대한 신뢰도 점수를 할당합니다.

안전하지 않은 프롬프트는 개인 정보 또는 사적인 정보를 요청하거나, 불쾌하거나 불법적인 콘텐츠를 생성하거나, 의료, 법률, 정치 또는 금융 주제에 대한 조언을 요청하는 등 악의적인 의도를 표현하는 입력 프롬프트입니다.

API를 사용한 신속한 안전성 분류

텍스트 문자열에 대해 신속한 안전성 분류를 실행하려면 동기식 ClassifyDocument 작업을 사용하십시오. 입력에는 일반 영어 텍스트 문자열을 제공해야 합니다. 문자열 최대 크기는 10KB입니다.

응답에는 각 클래스의 신뢰도 점수와 함께 두 개의 등급(SAFE 및 UNSAFE)이 포함됩니다. 점수의 값 범위는 0에서 1까지입니다. 여기서 1은 가장 높은 신뢰도입니다.

다음 예제에서는 AWS CLI 및 Python에서 프롬프트 안전 분류를 사용하는 방법을 보여줍니다.

AWS CLI

다음 예제는 AWS CLI로 프롬프트 안전성 분류기를 사용하는 방법을 보여줍니다.

```
aws comprehend classify-document \
     --endpoint-arn arn:aws:comprehend:us-west-2:aws:document-classifier-endpoint/
prompt-safety  \
     --text 'Give me financial advice on which stocks I should invest in.'
```

는 다음 출력으로 AWS CLI 응답합니다.

```
{
    "Classes": [
        {
            "Score": 0.6312999725341797,
            "Name": "UNSAFE_PROMPT"
        },
        {
            "Score": 0.3686999976634979,
            "Name": "SAFE_PROMPT"
        }
    ]
}
```

> **Note**
>
> `classify-document` 명령을 사용할 때 `--endpoint-arn` 파라미터에 대해 AWS CLI 구성 AWS 리전 과 동일한를 사용하는 ARN을 전달해야 합니다. 를 구성하려면 `aws configure` 명령을 AWS CLI실행합니다. 이 예제에서 엔드포인트 ARN의 리전 코드는 `us-west-2`입니다. 다음 리전 중 하나에서 프롬프트 안전성 분류기를 사용할 수 있습니다.
>
> - us-east-1
> - us-west-2
> - eu-west-1
> - ap-southeast-2

Python (Boto)

다음 예제는 Python에서 프롬프트 안전성 분류기를 사용하는 방법을 보여줍니다.

```
import boto3
client = boto3.client(service_name='comprehend', region_name='us-west-2')

response = client.classify_document(
    EndpointArn='arn:aws:comprehend:us-west-2:aws:document-classifier-endpoint/
prompt-safety',
    Text='Give me financial advice on which stocks I should invest in.'
)
print("Response: %s\n" % response)
```

> **Note**
>
> `classify_document` 메서드를 사용할 때는 `EndpointArn` 인수에 boto3 SDK 클라이언트와 동일한 AWS 리전 을 사용하는 ARN을 전달해야 합니다. 이 예제에서는 클라이언트와 엔드포인트 ARN이 모두 `us-west-2`을 사용합니다. 다음 리전 중 하나에서 프롬프트 안전성 분류기를 사용할 수 있습니다.
>
> - us-east-1
> - us-west-2
> - eu-west-1
> - ap-southeast-2

PII 감지 및 수정

Amazon Comprehend 콘솔 또는 APIs를 사용하여 영어 또는 스페인어 텍스트 문서에서 개인 식별 정보(PII)를 감지할 수 있습니다. PII는 개인을 식별할 수 있는 개인 데이터에 대한 텍스트 참조입니다. PII의 예로는 주소, 은행 계좌 번호, 전화번호 등이 있습니다.

텍스트에서 PII 엔터티를 검색하거나 삭제할 수 있습니다. PII 엔터티를 감지하기 위해 실시간 분석 또는 비동기 배치 작업을 사용할 수 있습니다. 텍스트의 PII 개체를 수정하려면 비동기 일괄 작업을 시작합니다.

자세한 내용은 개인 식별 정보(PII) 를 참조하십시오.

개인 식별 정보(PII)

Amazon Comprehend 콘솔 또는 APIs를 사용하여 영어 또는 스페인어 텍스트 문서에서 개인 식별 정보(PII)를 감지할 수 있습니다. PII는 개인을 식별할 수 있는 개인 데이터에 대한 텍스트 참조입니다. PII의 예로는 주소, 은행 계좌 번호, 전화번호 등이 있습니다.

PII 감지를 통해 PII 개체를 찾거나 텍스트에서 PII 개체를 수정할 수 있습니다. PII 엔터티를 감지하기 위해 실시간 분석 또는 비동기 배치 작업을 사용할 수 있습니다. 텍스트의 PII 개체를 수정하려면 비동기 일괄 작업을 시작합니다.

개인 식별 정보(PII)를 위한 Amazon S3 객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에서 문서를 검색하는 방법을 구성할 수 있습니다. PII가 포함된 문서에 대한 액세스를 제어하고 문서에서 개인 식별 정보를 수정할 수 있습니다. 자세한 내용은 개인 식별 정보(PII)를 위한 Amazon S3 객체 Lambda 액세스 포인트 사용을 참조하십시오.

주제

- PII 개체 감지
- PII 개체에 레이블 지정
- PII 실시간 분석(콘솔)
- PII 비동기 분석 작업(콘솔)
- PII 실시간 분석(API)
- PII 비동기 분석 작업(API)

PII 개체 감지

Amazon Comprehend를 사용하여 영어 또는 스페인어 텍스트 문서에서 PII 엔터티를 감지할 수 있습니다. PII 개체는 특정 유형의 개인 식별 정보(PII) 입니다. PII 감지를 사용하여 PII 개체를 찾거나 텍스트에서 PII 개체를 수정합니다.

주제

- PII 개체 찾기
- PII 개체 수정
- PII 범용 개체 유형
- 국가별 PII 개체 유형

PII 개체 찾기

텍스트에서 PII 개체를 찾으려면 실시간 분석을 사용하여 단일 문서를 빠르게 분석할 수 있습니다. 문서 모음에 대해 비동기 일괄 작업을 시작할 수도 있습니다.

콘솔 또는 API를 사용하여 단일 문서를 실시간으로 분석할 수 있습니다. 입력 텍스트에는 최대 100킬로바이트의 UTF-8 인코딩 문자가 포함될 수 있습니다.

예를 들어 다음 입력 텍스트를 제출하여 PII 개체를 찾을 수 있습니다.

Paulo Santos님, 안녕하세요? 귀하의 신용카드 계좌 1111-0000-1111-0000의 최신 명세서가 123 Any Street, Seattle, WA 98109로 우편 발송되었습니다.

출력물에는 “Paul Santos”라는 NAME유형, “1111-0000-1111-0000"라는 CREDIT_DEBIT_NUMBER유형 및 “123 Any Street, Seattle, WA 98109"는 ADDRESS 유형이라는 정보가 포함됩니다.

Amazon Comprehend는 감지된 PII 개체 리스트를 각 PII 개체에 대한 다음 정보와 함께 반환합니다.

- 감지된 텍스트 범위가 감지된 개체 유형일 확률을 추정하는 점수.
- PII 개체 유형.
- 개체의 시작과 끝을 나타내는 문자 오프셋으로 지정되는 문서 내 PII 개체의 위치.

예를 들어, 앞서 언급한 입력 텍스트는 다음과 같은 응답을 생성합니다.

```
{
    "Entities": [
        {
            "Score": 0.9999669790267944,
            "Type": "NAME",
            "BeginOffset": 6,
            "EndOffset": 18
        },
        {
            "Score": 0.8905550241470337,
            "Type": "CREDIT_DEBIT_NUMBER",
            "BeginOffset": 69,
            "EndOffset": 88
        },
        {
            "Score": 0.9999889731407166,
            "Type": "ADDRESS",
            "BeginOffset": 103,
```

```
            "EndOffset": 138
        }
    ]
}
```

PII 개체 수정

텍스트의 PII 개체을 수정하려면 콘솔 또는 API를 사용하여 비동기 일괄 작업을 시작할 수 있습니다. Amazon Comprehend는 각 PII 개체에 대한 수정 사항이 포함된 입력 텍스트의 복사본을 반환합니다.

예를 들어 다음 입력 텍스트를 제출하여 PII 개체를 수정할 수 있습니다.

Paulo Santos님, 안녕하세요? 귀하의 신용카드 계좌 1111-0000-1111-0000의 최신 명세서가 123 Any Street, Seattle, WA 98109로 우편 발송되었습니다.

출력파일은 다음 텍스트를 포함합니다.

***** ******님, 안녕하세요? 귀하의 신용카드 계좌 ******************의 최신 명세서가 *** *** ******* ******** ** *****로 우편 발송되었습니다.

PII 범용 개체 유형

이메일 주소 및 신용카드 번호와 같은 일부 PII 개체 유형은 범용입니다(개별 국가에 국한되지 않음). Amazon Comprehend는 다음과 같은 유형의 범용 PII 개체를 감지합니다.

ADDRESS

실제 주소(예: “100 Main Street, Anytown, USA" 또는 "Suite #12, Building 123") 주소에는 거리, 건물, 위치, 도시, 주, 국가, 카운티, 우편번호, 구역, 타운 등의 정보가 포함될 수 있습니다.

AGE

개인의 연령(수량 및 시간 단위 포함). 예를 들어, Amazon Comprehend는 “저는 40세입니다”라는 문구에서 “40세”를 연령으로 인식합니다.

AWS_ACCESS_KEY

보안 액세스 키와 연결된 고유 식별자입니다. 액세스 키 ID와 보안 액세스 키를 사용하여 프로그래밍 방식의 AWS 요청에 암호화 방식으로 서명합니다.

AWS_SECRET_KEY

액세스 키와 관련된 고유 식별자. 액세스 키 ID와 보안 액세스 키를 사용하여 프로그래밍 방식의 AWS 요청에 암호화 방식으로 서명합니다.

CREDIT_DEBIT_CVV

비자, 마스터카드, 디스커버 신용카드 및 직불카드에 있는 3자리 카드 인증 코드(CVV). 아메리칸 익스프레스 신용카드나 직불카드의 경우 CVV는 4자리 숫자 코드입니다.

CREDIT_DEBIT_EXPIRY

신용카드 또는 직불카드 만료 날짜 이 숫자는 보통 네 자리 숫자이며, 월/년 또는 MM/YY 형식으로 지정되는 경우가 많습니다. Amazon Comprehend는 01/21, 01/2021, 및 Jan 2021과 같은 만료 날짜를 인식합니다.

CREDIT_DEBIT_NUMBER

신용카드 또는 직불카드 번호 이 번호의 길이는 13~16자리까지 다양합니다. 하지만 Amazon Comprehend는 마지막 4자리만 있는 경우에도 신용카드 또는 직불카드 번호를 인식합니다.

DATE_TIME

날짜에는 년, 월, 일, 요일 또는 시각이 포함될 수 있습니다. 예를 들어, Amazon Comprehend는 “2020년 1월 19일” 또는 “오전 11시”를 날짜로 인식합니다. Amazon Comprehend는 일부 날짜, 날짜 범위 및 날짜 간격을 인식합니다. 또한 “1990년대”와 같은 십년 단위를 인식할 것입니다.

DRIVER_ID

개인이 공공 도로에서 한 대 이상의 자동차를 운전할 수 있도록 허가하는 공식 문서인 운전면허증에 부여되는 번호입니다. 운전면허증 번호는 영숫자로 구성됩니다.

EMAIL

이메일 주소(예: marymajor@email.com).

INTERNATIONAL_BANK_ACCOUNT_NUMBER

국제 은행 계좌 번호의 형식은 국가별로 다릅니다. www.iban.com/structure를 참조하세요.

IP_ADDRESS

IPv4 주소(예: 198.51.100.0)

차량_번호판

차량 번호판은 차량이 등록된 주 또는 국가에서 발급합니다. 승용차의 형식은 일반적으로 대문자와 숫자로 구성된 5~8자리 숫자입니다. 형식은 발급한 주 또는 국가의 위치에 따라 다릅니다.

MAC_ADDRESS

미디어 액세스 제어(MAC) 주소는 네트워크 인터페이스 컨트롤러(NIC)에 할당되는 고유 식별자입니다.

NAME

개인의 이름. 이 개체 유형에는 Dr., Mr., Mrs., Miss 등의 호칭은 포함되지 않습니다. Amazon Comprehend는 조직 또는 주소의 일부인 이름에는 이 개체 유형을 적용하지 않습니다. 예를 들어 Amazon Comprehend는 “아무개 조직(John Doe Organization)”을 하나의 조직으로 인정하고 “아무개 도로(Jane Doe Street)”를 주소로 인식합니다.

비밀번호

비밀번호로 사용되는 영숫자 문자열(예: “*very20special #pass *”).

PHONE

전화번호 이 개체 유형에는 팩스 및 호출기 번호도 포함됩니다.

PIN

은행 계좌에 액세스할 수 있는 4자리 개인 식별 번호(PIN).

SWIFT_CODE

SWIFT 코드는 특정 은행 또는 지점을 지정하는 데 사용되는 은행 식별 코드(BIC) 의 표준 형식입니다. 은행은 이 코드를 국제 전신 송금과 같은 송금에 사용합니다.

SWIFT 코드는 8자 또는 11자로 구성됩니다. 11자리 코드는 특정 지점을 나타내며, 8자리 코드(또는 'XXX'로 끝나는 11자리 코드)는 본점 또는 주요 사무소를 나타냅니다.

URL

웹 주소(예: www.example.com)

USERNAME

계정을 식별하는 사용자 이름(예: 로그인 이름, 화면 이름, 닉네임 또는 핸들).

VEHICLE_IDENTIFICATION_NUMBER

차량 식별 번호(VIN)는 차량을 고유하게 식별합니다. VIN 콘텐츠와 형식은 ISO 3779 사양에 정의되어 있습니다. 각 국가별로 VIN에 대한 특정 코드와 형식을 가지고 있습니다.

국가별 PII 개체 유형

여권 번호 및 기타 정부 발행 ID 번호 등 일부 PII 개체 유형은 국가별로 다릅니다. Amazon Comprehend는 다음과 같은 유형의 국가별 PII 개체를 감지합니다.

CA_HEALTH_NUMBER

캐나다 보건 서비스 번호는 개인이 의료 혜택을 받는 데 필요한 10자리 고유 식별자입니다.

CA_SOCIAL_INSURANCE_NUMBER

캐나다 사회보험 번호(SIN)는 개인이 정부 프로그램 및 혜택을 이용할 때 필요한 9자리 고유 식별자입니다.

SIN은 세 자리 숫자가 세 개의 그룹 형식으로 되어 있습니다(예: 123-456-789). SIN은 Luhn 알고리즘이라는 간단한 숫자 확인 프로세스를 통해 검증할 수 있습니다.

IN_AADHAAR

인도 아드하르는 인도 정부가 인도 거주자에게 발급하는 12자리 고유 식별 번호입니다. Aadhaar 형식에서는 네 번째와 여덟 번째 자리 뒤에 공백이나 하이픈이 있습니다.

IN_NREGA

인도 국가 농촌 고용 보장법(NREGA) 번호는 문자 2개 + 숫자 14개로 구성됩니다.

IN_PERMANENT_ACCOUNT_NUMBER

인도 영구 계좌 번호는 소득세 부서에서 발급하는 10자리 고유 영숫자 번호입니다.

IN_VOTER_NUMBER

인도 유권자 신분증은 3개의 문자와 그에 이은 7개의 숫자로 구성됩니다.

UK_NATIONAL_HEALTH_SERVICE_NUMBER

영국 국민 보건 서비스 번호는 10~17자리 숫자로, 예를 들어 485 777 3456입니다. 현재 시스템에서는 세 번째와 여섯 번째 자리 뒤에 공백을 넣어 10자리 숫자 형식을 지정합니다. 마지막 숫자는 오류 감지 체크섬입니다.

17자리 숫자 형식에서는 10자리와 13자리 뒤에 공백이 있습니다.

UK_NATIONAL_INSURANCE_NUMBER

영국 국민보험번호(NINO)는 개인에게 국민보험(사회보장) 혜택을 제공합니다. 또한 영국 조세 시스템에서도 일부 용도로 사용됩니다.

이 번호는 9자리 길이이며 문자 2개로 시작하고 그 뒤에 숫자 6개와 문자 1개가 옵니다. NINO는 문자 2개 뒤와 두 번째, 네 번째, 여섯 번째 숫자 뒤에 공백이나 대시를 넣어 형식을 지정할 수 있습니다.

UK_UNIQUE_TAXPAYER_REFERENCE_NUMBER

영국 고유 납세자 참조(UTR)는 납세자 또는 사업체를 식별하는 10자리 숫자입니다.

BANK_ACCOUNT_NUMBER

일반적으로 10~12자리 길이의 미국 은행 계좌 번호입니다. 하지만 Amazon Comprehend는 마지막 4자리만 있는 경우에도 신용카드 또는 직불카드 번호를 인식합니다.

BANK_ROUTING

미국 은행 계좌 라우팅 번호 일반적으로 길이는 9자리이지만 Amazon Comprehend는 마지막 4자리만 있는 경우에도 라우팅 번호를 인식합니다.

PASSPORT_NUMBER

미국 여권 번호 여권 번호의 범위는 6~9자의 영숫자입니다.

US_INDIVIDUAL_TAX_IDENTIFICATION_NUMBER

미국 개인 납세자 식별 번호(ITIN)는 “9"로 시작하고 네 번째 자리로 “7" 또는 “8"이 포함된 9자리 숫자입니다. ITIN은 세 번째 및 네 번째 숫자 뒤에 공백이나 대시를 사용하여 형식을 지정할 수 있습니다.

SSN

미국 사회보장번호(SSN)는 미국 시민권자, 영주권자 및 임시 근로 거주자에게 발급되는 9자리 숫자입니다. 하지만 Amazon Comprehend는 마지막 4자리만 있는 경우에도 은행 계좌 번호 번호를 인식합니다.

PII 개체에 레이블 지정

PII 감지를 실행하면 Amazon Comprehend는 식별된 PII 개체 유형의 레이블을 제공합니다. 예를 들어, Amazon Comprehend에 다음과 같은 입력 텍스트를 제출하는 경우:

Paulo Santos님, 안녕하세요? 귀하의 신용카드 계좌 1111-0000-1111-0000의 최신 명세서가 123 Any Street, Seattle, WA 98109로 우편 발송되었습니다.

출력 결과에는 정확도에 대한 신뢰도 점수와 함께 PII 개체 유형을 나타내는 레이블이 포함됩니다. 이 경우 문서 텍스트 “Paul Santos”, “1111-0000-1111-0000" 및 “123 Any Street, Seattle, WA 98109"는 각각 PII 개체 유형으로 NAME, CREDIT_DEBIT_NUMBER 및 ADDRESS 레이블을 생성합니다. 지원되는 개체 유형에 대한 자세한 내용은 PII 범용 개체 유형을 참조하세요.

Amazon Comprehend는 각 레이블에 대해 다음과 같은 정보를 제공합니다.

- PII 개체 유형의 레이블 이름.
- 감지된 텍스트가 PII 개체 유형으로 레이블이 지정될 확률을 추정하는 점수.

위의 입력 텍스트 예제는 다음과 같은 JSON 출력 결과를 생성합니다.

```
{
    "Labels": [
        {
            "Name": "NAME",
            "Score": 0.9149109721183777
        },
        {
            "Name": "CREDIT_DEBIT_NUMBER",
            "Score": 0.5698626637458801
        }
         {
            "Name": "ADDRESS",
            "Score": 0.9951046109199524
        }
    ]
}
```

PII 실시간 분석(콘솔)

콘솔을 사용하여 텍스트 문서의 PII 실시간 감지를 실행할 수 있습니다. 텍스트 최대 크기는 UTF-8 인코딩 문자의 100KB입니다. 콘솔에 결과가 표시되므로 분석을 검토할 수 있습니다.

내장 모델을 사용하여 PII 감지 실시간 분석을 실행합니다.

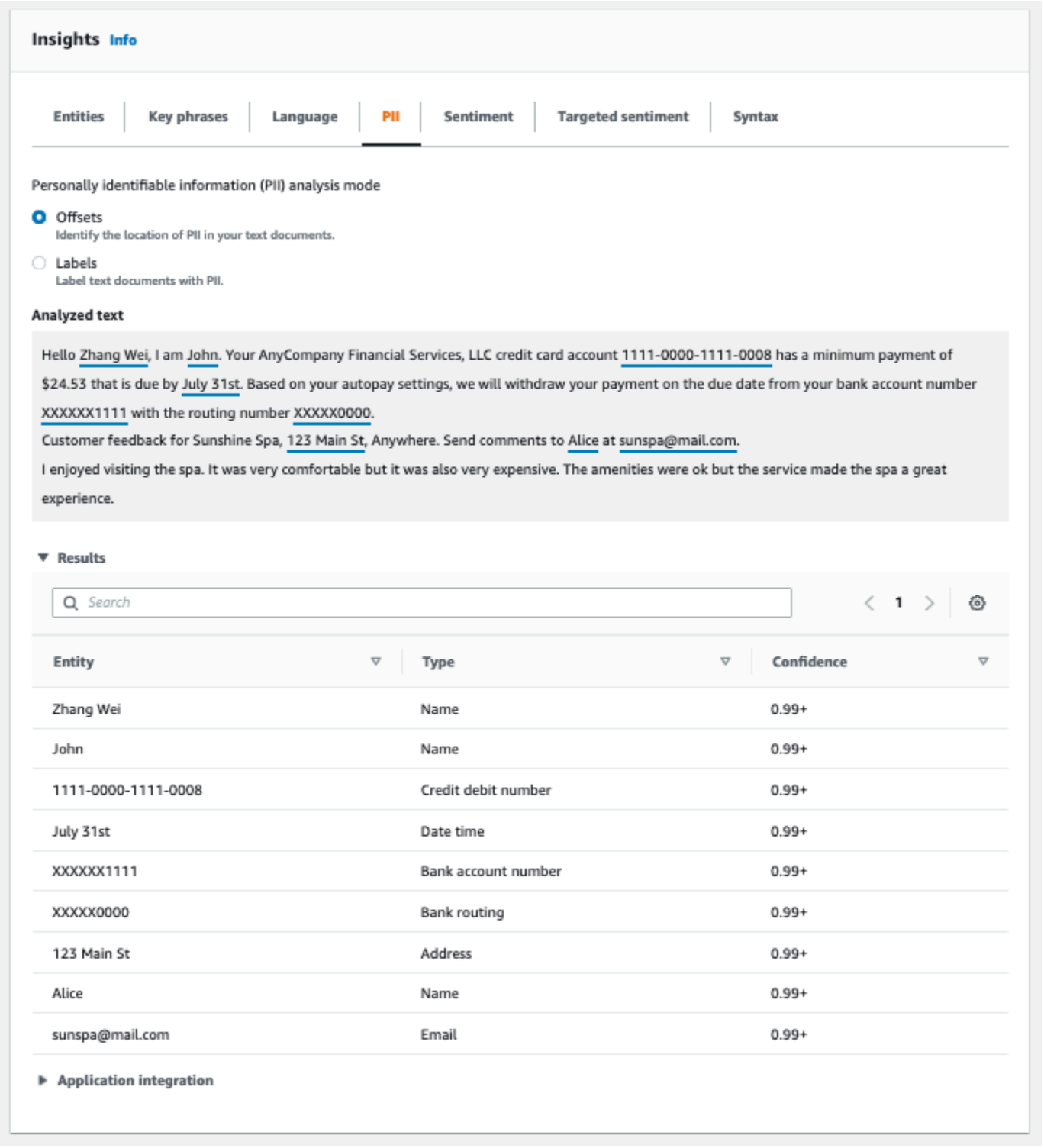
1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 실시간 분석을 선택합니다.
3. 입력 유형에서분석유형으로 내장을 선택합니다.
4. 분석하려는 텍스트를 입력합니다.
5. 분석을 선택합니다. 콘솔의 인사이트 패널에 텍스트 분석 결과가 표시됩니다. PII 탭에는 입력 텍스트에서 감지된 PII 개체들이 나열됩니다.

인사이트 패널의 PII 탭에는 두 가지 분석 모드의 결과가 표시됩니다.

- 오프셋 — 텍스트 문서에서 PII의 위치를 식별합니다.
- 레이블 - 식별된 PII 개체 유형의 레이블을 식별합니다.

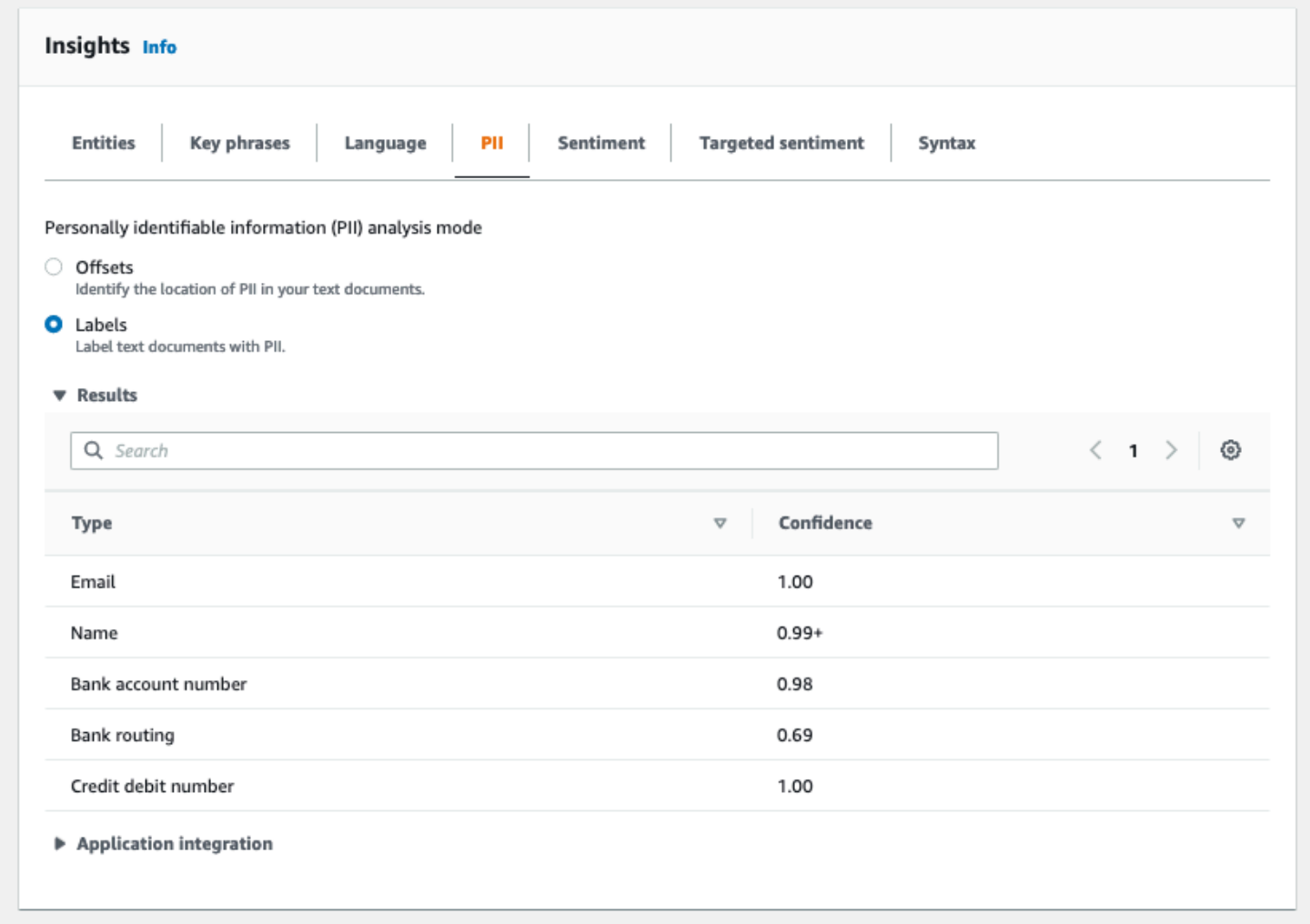
오프셋

오프셋 분석 모드는 텍스트 문서에서 PII의 위치를 식별합니다. 자세한 내용은 PII 개체 찾기을 참조하십시오.



레이블

레이블 분석 모드는 식별된 PII 개체 유형의 레이블을 반환합니다. 자세한 내용은 PII 개체에 레이블 지정을 참조하십시오.



PII 비동기 분석 작업(콘솔)

콘솔을 사용하여 PII 개체를 감지하는 비동기 분석 작업을 생성할 수 있습니다. PII 개체 유형에 대한 자세한 내용은 PII 개체 감지을 참조하세요.

분석 작업을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 분석 작업을 선택한 다음 작업 생성을 선택합니다.
3. 작업 설정에서 분석 작업에 고유한 이름을 지정합니다.
4. 분석 유형에서 개인 식별 정보(PII)를 선택합니다.
5. 언어에서 지원되는 언어(영어 또는 스페인어) 중 하나를 선택합니다.
6. 출력 모드에서 다음 선택 사항 중 하나를 선택합니다.

- 오프셋 — 작업 출력은 각 PII 개체의 위치를 반환합니다.
- 수정 — 작업 출력은 각 PII 개체가 수정된 입력 텍스트의 사본을 반환합니다.

7. (선택) 출력 모드로 수정을 선택하면 수정할 PII 개체 유형을 선택할 수 있습니다.
8. 입력 데이터에서 입력 문서의 Amazon S3 내 위치를 지정합니다.
   - 자체 문서를 분석하려면 내 문서를 선택하고 S3 탐색을 선택하여 파일이 들어 있는 버킷 또는 폴더의 경로를 제공합니다.
   - Amazon Comprehend에서 제공하는 샘플을 분석하려면 예제 문서를 선택합니다. 이 경우 Amazon Comprehend는에서 관리하는 버킷을 사용 AWS하며 위치를 지정하지 않습니다.
9. (선택 사항) 입력 형식의 경우 입력 파일에 대해 다음 형식 중 하나를 지정합니다.
   - 파일당 문서 하나 - 각 파일에는 입력 문서 하나가 들어 있습니다. 이 방법은 대용량 문서 모음에 가장 적합합니다.
   - 라인당 하나의 문서 - 하나 이상의 파일이 입력됩니다. 파일의 각 라인은 문서로 간주됩니다. 소셜 미디어 게시물과 같은 짧은 문서에 가장 적합합니다. 각 라인은 줄 바꿈 (LF, \n), 캐리지 리턴 (CR,\ r) 또는 둘 다 (CRLF,\ r\ n) 로 끝나야 합니다. UTF-8 줄 구분자(u+2028)를 사용하여 줄을 끊낼 수는 없습니다.
10. 출력 데이터에서 S3 검색을 선택합니다. Amazon Comprehend가 분석을 통해 생성한 출력 데이터를 기록할 Amazon S3 버킷 또는 폴더를 선택합니다.
11. (선택) 작업의 출력 결과를 암호화하려면 암호화를 선택합니다. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.
    - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID에서 키 별칭 또는 ID를 선택합니다.
    - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ID 아래에 키 별칭 또는 ID의 ARN을 입력합니다.

    > **Note**
    >
    > KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 (키 관리 서비스 (KMS)를 참조하세요.

12. 액세스 권한에는 다음의 IAM 역할을 제공하십시오.
    - 입력 문서의 Amazon S3 위치에 대한 읽기 권한 부여.
    - 입력 문서의 Amazon S3 위치에 대한 쓰기 권한 부여.

- 또한 comprehend.amazonaws.com 서비스 주체가 역할을 수임하고 권한을 획득하는 것을 허용하는 신뢰 정책을 포함시킵니다.

  이러한 권한에 대한 IAM 역할과 적절한 신뢰 정책이 없으면 IAM 역할 생성을 선택하여 새로 만드십시오.

13. 양식 작성을 마치고 나면 작업 생성을 선택하여 주제 탐지 작업을 생성하여 시작합니다.

새 작업이 작업 목록에 나타나고 상태 필드에 작업 상태가 표시됩니다. 이 필드는 처리 중인 작업은 IN_PROGRESS, 성공적으로 완료된 작업은 COMPLETED, 오류가 있는 작업을 FAILED으로 표시합니다. 작업을 클릭하여 오류 메시지를 비롯하여 상세한 작업 정보를 볼 수 있습니다.

작업이 완료되면 Amazon Comprehend는 작업에 대해 지정한 출력 Amazon S3 위치에 분석 결과를 저장합니다. 분석 결과에 대한 설명은 PII 개체 감지을 참조하세요.

PII 실시간 분석(API)

Amazon Comprehend는 문서의 개인 식별 정보(PII)를 분석하기 위한 실시간 동기 API 작업을 제공합니다.

주제

- PII 실시간 개체(API) 찾기
- PII 실시간 개체에 레이블 지정(API)

PII 실시간 개체(API) 찾기

단일 문서에서 PII를 찾으려면 Amazon Comprehend DetectPIIEntities 작업을 사용하면 됩니다. 입력 텍스트에는 최대 100킬로바이트의 UTF-8 인코딩 문자가 포함될 수 있습니다. 지원되는 언어에는 영어와 스페인어가 포함됩니다.

(CLI)를 사용하여 PII 찾기

다음 예제에서는 AWS CLI로 DetectPiiEntities 작업을 사용합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend detect-pii-entities \
```

```
  --text "Hello Paul Santos. The latest statement for your credit card \
  account 1111-0000-1111-0000 was mailed to 123 Any Street, Seattle, WA \
   98109." \
  --language-code en
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
      "Entities": [
          {
              "Score": 0.9999669790267944,
              "Type": "NAME",
              "BeginOffset": 6,
              "EndOffset": 18
          },
          {
              "Score": 0.8905550241470337,
              "Type": "CREDIT_DEBIT_NUMBER",
              "BeginOffset": 69,
              "EndOffset": 88
          },
          {
              "Score": 0.9999889731407166,
              "Type": "ADDRESS",
              "BeginOffset": 103,
              "EndOffset": 138
          }
      ]
  }
```

PII 실시간 개체에 레이블 지정(API)

실시간 동기 API 작업을 사용하여 식별된 PII 개체 유형의 레이블을 반환할 수 있습니다. 자세한 내용은 PII 개체에 레이블 지정을 참조하십시오.

PII 개체에 레이블 지정(CLI)

다음 예제에서는 AWS CLI로 `ContainsPiiEntities` 작업을 사용합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend contains-pii-entities \
```

```
--text "Hello Paul Santos. The latest statement for your credit card \
account 1111-0000-1111-0000 was mailed to 123 Any Street, Seattle, WA \
 98109." \
--language-code en
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "Labels": [
        {
            "Name": "NAME",
            "Score": 0.9149109721183777
        },
        {
            "Name": "CREDIT_DEBIT_NUMBER",
            "Score": 0.8905550241470337
        }
         {
            "Name": "ADDRESS",
            "Score": 0.9951046109199524
        }
    ]
}
```

PII 비동기 분석 작업(API)

PII 비동기 분석(API)

비동기 API 작업을 사용하여 PII 개체을 찾거나 수정하는 분석 작업을 생성할 수 있습니다. PII 개체 유형에 대한 자세한 내용은 PII 개체 감지을 참조하세요.

주제

- 비동기 작업으로 PII 개체 찾기(API)
- 비동기 작업으로 PII 개체 수정(API)

비동기 작업으로 PII 개체 찾기(API)

비동기 일괄 작업을 실행하여 문서 모음에서 PII를 찾습니다. 이 작업을 실행하려면 Amazon S3에 문서를 업로드하고 PII 개체 감지 작업 시작 요청을 제출하세요.

주제

시작하기 전에

시작하기 전에 다음 사항을 확인해야 합니다.

- 입력 및 출력 버킷 - 입력 파일 및 출력 파일에 사용하려는 Amazon S3 버킷을 식별합니다. 버킷은 사용자가 호출하는 API와 동일한 리전에 있어야 합니다.
- IAM 서비스 역할 - 입력 및 출력 버킷에 액세스하려면 권한이 있는 IAM 서비스 역할이 필요합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.

입력 파라미터

요청 시 다음의 필수적인 파라미터를 포함시킵니다.

- `InputDataConfig` — 요청에 InputDataConfig 정의를 제공합니다. 여기에는 작업에 대한 입력 속성이 포함되어야 합니다. `S3Uri` 파라미터에서 입력 문서의 Amazon S3 위치를 지정합니다.
- `OutputDataConfig` — 요청에 OutputDataConfig 정의를 제공합니다. 여기에는 작업에 대한 입력 속성이 포함되어야 합니다. `S3Uri` 파라미터에서 Amazon Comprehend가 분석 결과를 기록하는 Amazon S3 위치를 지정하십시오.
- `DataAccessRoleArn` - AWS Identity and Access Management 역할의 Amazon 리소스 이름(ARN)을 제공합니다. 이 역할은 Amazon Comprehend에 입력 데이터에 대한 읽기 액세스 권한과 Amazon S3 내 출력 위치에 대한 쓰기 액세스 권한을 부여해야 합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.
- `Mode` - 이 파라미터를 `ONLY_OFFSETS`로 설정합니다. 이 설정을 사용하면 출력 결과는 입력 텍스트에서 각 PII 개체를 찾는 문자 오프셋을 제공합니다. 출력 결과에는 신뢰도 점수와 PII 개체 유형도 포함됩니다.
- `LanguageCode` -이 파라미터를 `en` 또는 로 설정합니다`es`. Amazon Comprehend는 영어 또는 스페인어 텍스트로 PII 감지를 지원합니다.

비동기 작업 메서드

StartPiiEntitiesDetectionJob이 작업 ID를 반환함으로써 작업 진행 상황을 모니터링하고 완료 시 작업 상태를 찾아 확인할 수 있습니다.

분석 작업의 진행 상황을 모니터링하려면 DescribePiiEntitiesDetectionJob 작업에 작업 ID를 제공하십시오. DescribePiiEntitiesDetectionJob의 응답에는 작업의 현재 상태를 나타내는 JobStatus 필드가 포함되어 있습니다. 성공적인 작업은 다음 상태로 전환됩니다.

제출됨 -> 진행 중-> 완료됨

분석 작업이 완료된 후(JobStatus가 완료됨, 실패함 또는 중지됨) 결과 위치를 가져오는 데 DescribePiiEntitiesDetectionJob을 사용합니다. 작업 상태가 COMPLETED인 경우 응답에는 출력 파일의 Amazon S3 위치가 있는 필드가 포함된 OutputDataConfig 필드가 포함됩니다.

Amazon Comprehend 비동기 분석을 위해 따라야 할 단계에 대한 추가 세부 정보는 비동기 일괄 처리을 참조하세요.

출력 파일 형식

출력 파일 이름은 입력 파일과 동일하며 끝에 .out이 추가됩니다. 이 파일에는 분석 결과가 들어 있습니다.

다음은 문서에서 PII 개체를 감지한 분석 작업의 출력 파일 예제입니다. 입력 형식은 한 줄에 한 문서입니다.

```
{
  "Entities": [
    {
      "Type": "NAME",
      "BeginOffset": 40,
      "EndOffset": 69,
      "Score": 0.999995
    },
    {
      "Type": "ADDRESS",
      "BeginOffset": 247,
      "EndOffset": 253,
      "Score": 0.998828
    },
    {
      "Type": "BANK_ACCOUNT_NUMBER",
```

```
      "BeginOffset": 406,
      "EndOffset": 411,
      "Score": 0.693283
    }
  ],
  "File": "doc.txt",
  "Line": 0
},
{
  "Entities": [
    {
      "Type": "SSN",
      "BeginOffset": 1114,
      "EndOffset": 1124,
      "Score": 0.999999
    },
    {
      "Type": "EMAIL",
      "BeginOffset": 3742,
      "EndOffset": 3775,
      "Score": 0.999993
    },
    {
      "Type": "PIN",
      "BeginOffset": 4098,
      "EndOffset": 4102,
      "Score": 0.999995
    }
  ],
  "File": "doc.txt",
  "Line": 1
 }
```

다음은 입력 형식이 파일당 문서 하나인 분석으로부터의 출력 결과의 예입니다.

```
{
  "Entities": [
    {
      "Type": "NAME",
      "BeginOffset": 40,
      "EndOffset": 69,
      "Score": 0.999995
    },
```

```
    {
      "Type": "ADDRESS",
      "BeginOffset": 247,
      "EndOffset": 253,
      "Score": 0.998828
    },
    {
      "Type": "BANK_ROUTING",
      "BeginOffset": 279,
      "EndOffset": 289,
      "Score": 0.999999
    }
  ],
  "File": "doc.txt"
}
```

를 사용한 비동기 분석 AWS Command Line Interface

다음 예제에서는 AWS CLI로 `StartPiiEntitiesDetectionJob` 작업을 사용합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend start-pii-entities-detection-job \
    --region region \
    --job-name job name \
    --cli-input-json file://path to JSON input file
```

`cli-input-json` 파라미터에는 다음 예시에 표시된 대로 요청 데이터가 포함된 JSON 파일의 경로를 제공해야 합니다.

```
{
  "InputDataConfig": {
      "S3Uri": "s3://input bucket/input path",
      "InputFormat": "ONE_DOC_PER_LINE"
  },
  "OutputDataConfig": {
      "S3Uri": "s3://output bucket/output path"
  },
  "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role"
  "LanguageCode": "en",
  "Mode": "ONLY_OFFSETS"
```

```
}
```

이벤트 감지 작업을 시작하는 요청이 받아들여지면 다음과 유사한 응답을 받게 됩니다.

```
{
  "JobId": "5d2fbe6e...e2c"
  "JobArn":  "arn:aws:comprehend:us-west-2:123456789012:pii-entities-detection-
job/5d2fbe6e...e2c"
  "JobStatus": "SUBMITTED",
}
```

DescribeEventsDetectionJob 작업을 사용하여 기존 작업의 상태를 얻을 수 있습니다. 이벤트 감지 작업을 시작하는 요청이 받아들여지면 다음과 유사한 응답을 받게 됩니다.

```
aws comprehend describe-pii-entities-detection-job \
    --region region \
    --job-id job ID
```

작업이 성공적으로 완료되면 다음과 비슷한 응답을 받게 됩니다.

```
{
    "PiiEntitiesDetectionJobProperties": {
  "JobId": "5d2fbe6e...e2c"
  "JobArn":  "arn:aws:comprehend:us-west-2:123456789012:pii-entities-detection-
job/5d2fbe6e...e2c"
  "JobName": "piiCLItest3",
  "JobStatus": "COMPLETED",
  "SubmitTime": "2022-05-05T14:54:06.169000-07:00",
  "EndTime": "2022-05-05T15:00:17.007000-07:00",
  "InputDataConfig": {
       (identical to the input data that you provided with the request)
    }
}
```

비동기 작업으로 PII 개체 수정(API)

텍스트의 PII 개체를 수정하려면 비동기 일괄 작업을 시작합니다. 이 작업을 실행하려면 Amazon S3에 문서를 업로드하고 StartPiiEntitiesDetectionJob 요청을 제출하세요.

주제

시작하기 전에

시작하기 전에 다음 사항을 확인해야 합니다.

- 입력 및 출력 버킷 - 입력 파일 및 출력 파일에 사용하려는 Amazon S3 버킷을 식별합니다. 버킷은 사용자가 호출하는 API와 동일한 리전에 있어야 합니다.
- IAM 서비스 역할 - 입력 및 출력 버킷에 액세스하려면 권한이 있는 IAM 서비스 역할이 필요합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.

입력 파라미터

요청 시 다음의 필수적인 파라미터를 포함시킵니다.

- `InputDataConfig` — 요청에 InputDataConfig 정의를 제공합니다. 여기에는 작업에 대한 입력 속성이 포함되어야 합니다. `S3Uri` 파라미터에서 입력 문서의 Amazon S3 위치를 지정합니다.
- `OutputDataConfig` — 요청에 OutputDataConfig 정의를 제공합니다. 여기에는 작업에 대한 입력 속성이 포함되어야 합니다. `S3Uri` 파라미터에서 Amazon Comprehend가 분석 결과를 기록하는 Amazon S3 위치를 지정하십시오.
- `DataAccessRoleArn` – AWS Identity and Access Management 역할의 Amazon 리소스 이름(ARN)을 제공하십시오. 이 역할은 Amazon Comprehend에 입력 데이터에 대한 읽기 액세스 권한과 Amazon S3 내 출력 위치에 대한 쓰기 액세스 권한을 부여해야 합니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.
- `Mode` - 이 파라미터를 `ONLY_REDACTION`로 설정합니다. 이 설정을 사용하면 Amazon Comprehend는 입력 문서의 사본을 Amazon S3의 출력 위치에 기록합니다. 이 사본에서는 각 PII 개체가 수정됩니다.
- `RedactionConfig` — 요청에 RedActionConfig 정의를 제공합니다. 여기에는 수정을 위한 구성 파라미터가 포함됩니다. 수정할 PII의 유형을 지정하고 각 PII 개체를 해당 유형의 이름 또는 원하는 문자로 대체할지 여부를 지정합니다.
  - `PiiEntityTypes` 배열에서 수정할 PII 개체 유형을 지정합니다. 모든 개체 유형을 수정하려면 배열 값을 `["ALL"]`로 설정합니다.

  - 각 PII 개체를 해당 유형으로 바꾸려면 MaskMode 파라미터를 REPLACE_WITH_PII_ENTITY_TYPE로 설정합니다. 예를 들어 이 설정을 사용하면 PII 개체 “아무개(Jane Doe)”가 “[이름]”으로 대체됩니다.
  - 각 PII 개체의 문자를 귀하가 선택하는 문자로 바꾸려면 MaskMode 파라미터를 MASK로 설정하고 MaskCharacter 파라미터를 대체 문자로 설정합니다. 문자는 한 개만 제공하십시오. 유효한 문자는 !, #, $, %, &, *, 및 @입니다. 예를 들어 이 설정을 사용하면 PII 개체 “아무개(Jane Doe)”를 “*** ***"로 바꿀 수 있습니다.
- LanguageCode -이 파라미터를 en 또는 로 설정합니다es. Amazon Comprehend는 영어 또는 스페인어 텍스트로 PII 감지를 지원합니다.

출력 파일 형식

다음 예제는 PII를 수정하는 분석 작업의 입력 및 출력 파일을 보여줍니다. 입력 형식은 한 줄에 한 문서입니다.

```
{
Managing Your Accounts Primary Branch Canton John Doe Phone Number 443-573-4800 123
 Main StreetBaltimore, MD 21224
Online Banking HowardBank.com  Telephone 1-877-527-2703 Bank 3301 Boston Street,
 Baltimore, MD 21224
```

이 입력 파일을 수정하는 분석 작업은 다음과 같은 출력 파일을 생성합니다.

```
{
Managing Your Accounts Primary Branch ****** ******** Phone Number ************
 ********************************
Online Banking **************  Telephone ************** Bank
 ************************************
 }
```

를 사용한 PII 수정 AWS Command Line Interface

다음 예제에서는 AWS CLI로 StartPiiEntitiesDetectionJob 작업을 사용합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend start-pii-entities-detection-job \
```

```
    --region region \
    --job-name job name \
    --cli-input-json file://path to JSON input file
```

cli-input-json 파라미터에는 다음 예시에 표시된 대로 요청 데이터가 포함된 JSON 파일의 경로를 제공해야 합니다.

```
{
    "InputDataConfig": {
        "S3Uri": "s3://input bucket/input path",
        "InputFormat": "ONE_DOC_PER_LINE"
    },
    "OutputDataConfig": {
        "S3Uri": "s3://output bucket/output path"
    },
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role"
    "LanguageCode": "en",
    "Mode": "ONLY_REDACTION"
    "RedactionConfig": {
        "MaskCharacter": "*",
        "MaskMode": "MASK",
        "PiiEntityTypes": ["ALL"]
    }
}
```

이벤트 감지 작업을 시작하는 요청이 받아들여지면 다음과 유사한 응답을 받게 됩니다.

```
{
  "JobId": "7c4fbe6e...e5b"
  "JobArn":  "arn:aws:comprehend:us-west-2:123456789012:pii-entities-detection-
job/7c4fbe6e...e5b"
  "JobStatus": "SUBMITTED",
}
```

DescribeEventsDetectionJob 작업을 사용하여 기존 작업의 상태를 얻을 수 있습니다.

```
aws comprehend describe-pii-entities-detection-job \
    --region region \
    --job-id job ID
```

작업이 성공적으로 완료되면 다음과 비슷한 응답을 받게 됩니다.

```
{
  "PiiEntitiesDetectionJobProperties": {
     "JobId": "7c4fbe6e...e5b"
     "JobArn":  "arn:aws:comprehend:us-west-2:123456789012:pii-entities-detection-
job/7c4fbe6e...e5b"
     "JobName": "piiCLIredtest1",
     "JobStatus": "COMPLETED",
     "SubmitTime": "2022-05-05T14:54:06.169000-07:00",
     "EndTime": "2022-05-05T15:00:17.007000-07:00",
     "InputDataConfig": {
        (identical to the input data that you provided with the request)
  }
}
```

문서 처리

Amazon Comprehend는 사용자 지정 분류 및 사용자 지정 개체 인식을 위한 원스텝 문서 처리를 지원합니다. 예를 들어, 일반 텍스트 문서와 반정형 문서(예: PDF 문서, Microsoft Word 문서, 이미지)를 혼합하여 사용자 정의 분석 작업에 입력할 수 있습니다.

텍스트 추출이 필요한 입력 파일의 경우 Amazon Comprehend는 분석을 실행하기 전에 자동으로 텍스트 추출을 수행합니다. 텍스트 콘텐츠를 추출하기 위해 Amazon Comprehend는 기본 반정형 문서에는 내부 구문 분석기를 사용하고 이미지 및 스캔 문서에는 Amazon Textract API를 사용합니다.

Amazon Comprehend 문서 처리는 아시아 태평양(도쿄) 및 AWS GovCloud(미국 서부)가 사용자 지정 분류를 위한 일반 텍스트 모델만 지원하는 것을 지원되는 리전제외하고 각 Amazon Comprehend에서 사용할 수 있습니다.

다음 주제에서는 Amazon Comprehend가 사용자 지정 분석을 위해 지원하는 입력 문서 유형에 대한 세부 정보를 제공합니다.

주제

실시간 사용자 지정 분석을 위한 입력

사용자 지정 모델을 사용한 실시간 분석은 단일 문서를 입력으로 사용합니다. 다음 주제는 사용할 수 있는 입력 문서 유형에 대해 설명합니다.

주제

일반 텍스트 문서

입력 문서를 UTF-8 형식의 텍스트로 제공합니다.

반정형 문서

반정형 문서에는 기본 PDF 문서와 Word 문서가 포함됩니다.

기본적으로 실시간 사용자 지정 분석은 Amazon Comprehend 파서를 사용하여 Word 파일 및 디지털 PDF 파일에서 텍스트를 추출합니다. PDF 파일의 경우 이 기본 설정을 재정의하고 Amazon Textract를 사용하여 텍스트를 추출할 수 있습니다. 텍스트 추출 옵션을 설정하는 참조.

이미지 파일 및 스캔한 PDF 파일

지원되는 이미지 유형에는 JPEG, PNG 및 TIFF가 포함됩니다.

기본적으로 사용자 지정 개체 인식은 Amazon Textract `DetectDocumentText` API 작업을 사용하여 이미지 파일 및 스캔한 PDF 파일에서 텍스트를 추출합니다. 이 기본 설정을 재정의하여 `AnalyzeDocument` API 작업을 대신 사용할 수 있습니다. 텍스트 추출 옵션을 설정하는을 참조하세요.

Amazon Textract 출력

Amazon Textract `DetectDocumentText` API 또는 `AnalyzeDocument` API의 JSON 출력을 사용자 지정 분류 및 사용자 지정 개체 인식을 위한 실시간 API 작업에 대한 입력으로 제공할 수 있습니다. Amazon Comprehend에서는 실시간 API 작업에 대해 이 입력 유형을 지원하지만 콘솔에서는 지원하지 않습니다.

실시간 분석을 위한 최대 문서 크기

모든 입력 문서 유형의 경우 입력 파일의 최대 크기는 1페이지이며 10,000자를 넘지 않아야 합니다.

다음 표는 입력 문서의 최대 파일 크기를 보여줍니다.

파일 유형	최대 크기(API)	최대 크기(콘솔)
UTF-8 텍스트 문서	10KB	10KB
PDF 문서	10MB	5MB

파일 유형	최대 크기(API)	최대 크기(콘솔)
Word 문서	10MB	1MB
이미지 파일	10MB	5MB
Textract 출력 파일	1MB	해당 사항 없음

반정형 문서의 오류

ClassifyDocument 또는 DetectEntities API 작업에서 반정형 문서 또는 이미지 파일에서 텍스트를 추출하는 동안 문서 수준 또는 페이지 수준의 오류를 경험할 수 있습니다.

페이지 수준 오류

ClassifyDocument 또는 DetectEntities API 작업에서 입력 문서의 페이지를 처리하는 동안 오류가 발생하는 경우, API 응답에는 각 오류에 대한 오류 목록 항목이 포함됩니다.

오류 목록 항목의 `ErrorCode`에는 다음 값 중 하나가 포함됩니다.

- TEXTRACT_BAD_PAGE – Amazon Textract이 페이지를 읽을 수 없습니다. Amazon Textract의 페이지 제한에 대한 자세한 내용은 Amazon Textract의 페이지 할당량을 참조하세요.
- TEXTRACT_PROVISIONED_THROUGHPUT_EXCEEDED – 요청 수가 처리량 한도를 초과했습니다. Amazon Textract의 처리량에 대한 자세한 내용은 Amazon Textract의 기본 처리량을 참조하세요.
- PAGE_CHARACTERS_EXCEEDED – 페이지에 텍스트 문자가 너무 많습니다(최대 10,000자).
- PAGE_SIZE_EXCEEDED – 최대 페이지 크기는 10MB입니다.
- INTERNAL_SERVER_ERROR – 요청 중 서비스 문제가 발생했습니다. API 요청을 다시 시도하세요.

문서 수준 오류

ClassifyDocument 또는 DetectEntities API 작업이 입력 문서에서 문서 수준 오류를 감지하는 경우 API는 `InvalidRequestException` 오류 응답을 반환합니다.

응답에서 Reason필드에 `INVALID_DOCUMENT` 값이 포함됩니다.

Detail 필드에는 다음 값 중 하나가 포함될 수 있습니다.

- DOCUMENT_SIZE_EXCEEDED – 문서 크기가 너무 큽니다. 파일 크기를 확인하고 요청을 다시 제출하세요.
- UNSUPPORTED_DOC_TYPE – 지원되지 않는 문서 유형입니다. 파일 유형을 확인하고 요청을 다시 제출하세요.
- PAGE_LIMIT_EXCEEDED – 문서에 페이지가 너무 많습니다. 파일의 페이지 수를 확인하고 요청을 다시 제출하세요.
- TEXTRACT_ACCESS_DENIED_EXCEPTION – Amazon Textract에 대한 액세스가 거부되었습니다. 계정에 Amazon Textract DetectDocumentText 및 AnalyzeDocument API 작업을 사용할 권한이 있는지 확인하고 요청을 다시 제출하세요.

비동기 사용자 지정 분석을 위한 입력

사용자 지정 비동기 분석 작업에 여러 문서를 입력할 수 있습니다. 다음 주제는 사용할 수 있는 입력 문서 유형에 대해 설명합니다. 최대 파일 크기는 입력 문서의 유형에 따라 달라집니다.

주제

- 일반 텍스트 문서
- 반정형 문서
- 이미지 파일 및 스캔한 PDF 파일
- Amazon Textract 출력 JSON 파일

일반 텍스트 문서

모든 일반 텍스트 입력 문서를 UTF-8 형식의 텍스트로 제공합니다. 다음 표에는 최대 파일 크기 및 기타 지침이 나열되어 있습니다.

> **Note**
>
> 이러한 제한은 모든 입력 파일이 일반 텍스트인 경우에 적용됩니다.

설명	할당량/지침
파일 형식당 문서 1개에 대한 최대 파일 크기(사용자 지정 분류)	1바이트–10MB

설명	할당량/지침
문서 크기(사용자 지정 개체 인식)	1바이트–1MB
최대 파일 수, 파일당 문서 하나	1,000,000
최대 줄 수, 한 줄에 문서 한 개(요청 중인 모든 파일의 경우)	1,000,000
문서 코퍼스 크기(일반 텍스트의 모든 문서 합산)	1바이트–5GB

반정형 문서

반정형 문서에는 기본 PDF 문서와 Word 문서가 포함됩니다.

다음 표에는 최대 파일 크기 및 기타 지침이 나열되어 있습니다.

설명	할당량/지침
문서 크기(PDF)	1바이트–50MB
문서 크기(Docx)	1바이트–5MB
최대 파일 수	500
PDF 또는 Docx 파일의 최대 페이지 수	100
텍스트 추출 후의 문서 코퍼스 크기(일반 텍스트, 모든 파일 합산)	1바이트–5GB

기본적으로 사용자 지정 분석은 Amazon Comprehend 파서를 사용하여 Word 파일 및 디지털 PDF 파일에서 텍스트를 추출합니다. PDF 파일의 경우 이 기본 설정을 재정의하고 Amazon Textract를 사용하여 텍스트를 추출할 수 있습니다. 텍스트 추출 옵션을 설정하는 참조.

이미지 파일 및 스캔한 PDF 파일

사용자 지정 분석은 JPEG, PNG 및 TIFF 이미지를 지원합니다.

다음 표에는 이미지의 최대 파일 크기가 나열되어 있습니다. 스캔한 PDF 파일에는 원본 PDF 파일과 동일한 최대 크기 제한이 적용됩니다.

설명	할당량/지침
이미지 크기(JPG 또는 PNG)	1바이트–10MB
이미지 크기(TIFF)	1바이트–10MB 최대 한 페이지.

이미지에 대한 추가적인 내용은 이미지 모범 사례를 참조하세요.

기본적으로 Amazon Comprehend는 Amazon Textract `DetectDocumentText` API 작업을 사용하여 이미지 파일 및 스캔한 PDF 파일에서 텍스트를 추출합니다. 이 기본 설정을 재정의하여 `AnalyzeDocument` API 작업을 대신 사용할 수 있습니다. 텍스트 추출 옵션을 설정하는을 참조하세요.

Amazon Textract 출력 JSON 파일

사용자 지정 엔터티 인식의 경우(사용자 지정 분류는 아님) Amazon Textract `AnalyzeDocument` API 작업의 출력 파일을 분석 작업에 대한 입력으로 제공할 수 있습니다.

텍스트 추출 옵션을 설정하는

기본적으로 Amazon Comprehend는 입력 파일 유형에 따라 다음 작업을 수행하여 파일에서 텍스트를 추출합니다.

- Word 파일 - Amazon Comprehend 파서가 텍스트를 추출합니다.
- 디지털 PDF 파일 — Amazon Comprehend 파서가 텍스트를 추출합니다.
- 이미지 파일 및 스캔한 PDF 파일 - Amazon Comprehend는 Amazon Textract `DetectDocumentText` API를 사용하여 텍스트를 추출합니다.

이미지 파일 및 PDF 파일의 경우 `DocumentReaderConfig` 파라미터를 사용하여 이러한 기본 추출 작업을 재정의할 수 있습니다. 이 파라미터는 실시간 또는 비동기 사용자 지정 분석을 위해 Amazon Comprehend 콘솔 또는 API를 사용할 때 사용할 수 있습니다.

`DocumentReaderConfig` 파라미터에는 다음과 같은 세 개의 필드가 있습니다.

- DocumentReadMode - `SERVICE_DEFAULT`로 설정하면 Amazon Comprehend가 기본 작업을 수행합니다.

FORCE_DOCUMENT_READ_ACTION으로 설정하면 Amazon Textract를 사용하여 디지털 PDF 파일을 파싱합니다.

- DocumentReadAction - Amazon Comprehend가 텍스트 추출에 Amazon Textract를 사용할 때 사용할 Amazon Textract API(DetectDocumentText 또는 AnalyzeDocument)를 설정합니다.
- FeatureTypes — AnalyzeDocument API 작업을 사용하도록 DocumentReadAction을 설정하는 경우 FeatureTypes(TABLES, FORMS) 중 하나 또는 둘 다를 추가할 수 있습니다. 이러한 기능은 문서의 표 및 양식에 대한 추가 정보를 제공합니다. 이러한 기능에 대한 자세한 내용은 Amazon Textract 문서 분석 응답 객체를 참조하세요.

다음 예제에서는 특정 사용 사례를 위한 DocumentReaderConfig 구성 방법을 보여줍니다.

1. 모든 PDF 파일에 대해 Amazon Textract를 사용합니다.
   a. DocumentReadMode – FORCE_DOCUMENT_READ_ACTION으로 설정합니다.
   b. DocumentReadAction – TEXTRACT_DETECT_DOCUMENT_TEXT으로 설정합니다.
   c. FeatureTypes — 필수는 아닙니다.
2. 모든 PDF 및 이미지 파일에 대해 Amazon Textract AnalyzeDocument API를 사용합니다.
   a. DocumentReadMode – FORCE_DOCUMENT_READ_ACTION으로 설정합니다.
   b. DocumentReadAction – TEXTRACT_ANALYZE_DOCUMENT으로 설정합니다.
   c. FeatureTypes - TABLES 또는 FORMS로 설정하거나 두 기능을 모두 설정합니다.
3. 스캔된 PDF 및 이미지 파일에 대해 Amazon Textract AnalyzeDocument API를 사용합니다.
   a. DocumentReadMode – SERVICE_DEFAULT으로 설정합니다.
   b. DocumentReadAction – TEXTRACT_ANALYZE_DOCUMENT으로 설정합니다.
   c. FeatureTypes - TABLES 또는 FORMS로 설정하거나 두 기능을 모두 설정합니다.

Amazon Textract 옵션에 대한 자세한 내용은 DocumentReaderConfig를 참조하세요.

이미지 모범 사례

사용자 지정 분류 또는 사용자 지정 개체 인식을 위해 이미지 파일을 사용하는 경우 다음 지침을 사용하여 최상의 결과를 얻으십시오.

- 150DPI 이상의 고품질 이미지를 제공하는 것이 가장 바람직합니다.

- 이미지 파일이 지원되는 형식(TIFF, JPEG 또는 PNG) 중 하나를 사용하는 경우 Amazon S3에 업로드하기 전에 파일을 변환하거나 다운샘플링하지 마십시오.

문서의 표에서 텍스트를 추출할 때 최상의 결과를 얻으려면 다음 방법을 따르십시오.

- 문서의 표가 페이지의 주변 요소와 시각적으로 구분되어 있어야 합니다. 예를 들어 표는 이미지나 복잡한 패턴 위에 겹쳐지지 않습니다.
- 표 안의 텍스트는 수직으로 배치되어 있습니다. 예를 들어, 텍스트가 페이지의 다른 텍스트를 기준으로 회전되어 있지 않습니다.

표에서 텍스트를 추출할 때 다음과 같은 경우에 결과가 일치하지 않을 수 있습니다.

- 병합된 표 셀은 여러 열에 걸쳐 있습니다.
- 표에는 같은 표의 다른 부분과 다른 셀, 행 또는 열이 있습니다.

사용자 지정 분류

사용자 지정 분류를 사용하여 문서를 사용자가 정의한 범주(클래스)로 구성할 수 있습니다. 사용자 지정 분류는 두 단계로 이루어져 있습니다. 먼저 사용자가 관심 있어 하는 클래스를 인식하도록 사용자 지정 분류 모델(분류기라고도 함)을 학습시킵니다. 그런 다음 모델을 사용하여 원하는 수의 문서 세트를 분류합니다.

예를 들어 지원 요청의 내용을 분류하여 요청을 적절한 지원 팀에 전달할 수 있습니다. 또는 고객으로부터 받은 이메일을 분류하여 고객 요청 유형에 따라 지침을 제공할 수 있습니다. Amazon Comprehend와 Amazon Transcribe를 결합하여 음성을 텍스트로 변환한 다음 지원 전화 통화에서 오는 요청을 분류할 수 있습니다.

단일 문서에 대해 사용자 지정 분류를 동기적으로(실시간) 실행하거나 비동기 작업을 시작하여 문서 세트를 분류할 수 있습니다. 계정에 각각 다른 데이터를 사용하여 학습된 사용자 지정 분류기를 여러 개 둘 수 있습니다. 사용자 지정 분류는 일반 텍스트, PDF, Word 및 이미지와 같은 다양한 입력 문서 유형을 지원합니다.

분류 작업을 제출할 때는 분석해야 하는 문서 유형에 따라 사용할 분류기 모델을 선택합니다. 예를 들어, 일반 텍스트 문서를 분석하려면 일반 텍스트 문서로 학습시킨 모델을 사용하면 가장 정확한 결과를 얻을 수 있습니다. 반정형 문서(예: PDF, Word, 이미지, Amazon Textract 출력 또는 스캔한 파일)를 분석하려면 기본 문서로 학습시킨 모델을 사용하면 가장 정확한 결과를 얻을 수 있습니다.

주제

분류기 학습 데이터 준비

사용자 지정 분류의 경우 멀티클래스 모드 또는 멀티레이블 모드에서 모델을 학습합니다. 멀티클래스 모드는 단일 클래스를 각 문서와 연결합니다. 멀티레이블 모드는 하나 이상의 클래스를 각 문서와 연결합니다. 입력 파일 형식은 모드마다 다르므로 학습 데이터를 생성하기 전에 사용할 모드를 선택합니다.

> **Note**
>
> Amazon Comprehend 콘솔에서는 멀티클래스 모드를 단일 레이블 모드라고 합니다.

사용자 지정 분류는 일반 텍스트 문서로 학습시키는 모델 및 기본 문서(예: PDF, Word 또는 이미지)로 학습시키는 모델을 지원합니다. 분류기 모델 및 지원되는 문서 유형에 대한 자세한 내용은 학습 분류 모델을 참조하세요.

사용자 지정 분류기 모델을 학습할 데이터를 준비하려면:

1. 이 분류기가 분석할 클래스를 식별합니다. 사용할 모드(멀티클래스 또는 멀티레이블)를 결정합니다.
2. 모델이 일반 텍스트 문서 또는 반정형 문서를 분석하기 위한 모델인지에 따라 분류기 모델 유형을 결정합니다.
3. 각 클래스에 대한 문서 예제를 수집합니다. 최소 학습 요구 사항은 문서 분류를 위한 일반 할당량을 참조하세요.
4. 일반 텍스트 모델에서, 사용할 학습 파일 형식(CSV 파일 또는 증강 매니페스트 파일)을 선택합니다. 네이티브 문서 모델을 학습시키려면 항상 CSV 파일을 사용합니다.

주제

- 분류기 학습 파일 형식
- 멀티클래스 모드
- 멀티레이블 모드

분류기 학습 파일 형식

일반 텍스트 모델의 경우 분류기 훈련 데이터를 CSV 파일 또는 SageMaker AI Ground Truth를 사용하여 생성하는 증강 매니페스트 파일로 제공할 수 있습니다. CSV 파일 또는 증강 매니페스트 파일에는 각 학습 문서의 텍스트와 관련 레이블이 포함됩니다.

네이티브 문서 모델의 경우 분류기 학습 데이터를 CSV 파일로 제공합니다. CSV 파일에는 각 학습 문서의 파일 이름과 관련 레이블이 포함됩니다. 학습 작업을 위해 Amazon S3 입력 폴더에 학습 문서를 포함시킵니다.

CSV 파일

레이블이 지정된 학습 데이터를 UTF-8 인코딩 텍스트로 CSV 파일에 제공합니다. 헤더 행은 포함시키지 않습니다. 파일에 헤더 행을 추가하면 런타임 오류가 발생할 수 있습니다.

CSV 파일의 각 행에 대해 첫 번째 열에는 하나 이상의 클래스 레이블이 포함되며, 클래스 레이블은 임의의 유효한 UTF-8 문자열일 수 있습니다. 의미가 중첩되지 않는 명확한 클래스 이름을 사용하는 것이 좋습니다. 이름에는 공백이 포함될 수 있으며 밑줄이나 하이픈으로 연결된 여러 단어로 구성될 수 있습니다.

행의 값을 구분하는 쉼표 앞이나 뒤의 문자에 공백을 두지 마십시오.

CSV 파일의 정확한 내용은 분류기 모드와 학습 데이터 유형에 따라 달라집니다. 자세한 내용은 멀티클래스 모드 및 멀티레이블 모드를 참조하세요.

증강 매니페스트 파일

증강 매니페스트 파일은 SageMaker AI Ground Truth를 사용하여 생성하는 레이블이 지정된 데이터 세트입니다. Ground Truth는 사용자 또는 사용자가 고용한 작업 인력이 기계 학습 모델을 위한 학습 데이터 세트를 구축하는 데 도움이 되는 데이터 레이블 지정 서비스입니다.

Ground Truth 및 Ground Truth가 생성하는 출력에 대한 자세한 내용은 Amazon SageMaker AI 개발자 안내서의 SageMaker AI Ground Truth를 사용하여 데이터 레이블 지정을 참조하세요. Amazon SageMaker

증강 매니페스트 파일은 JSON 라인 형식입니다. 이 파일에서 각 라인은 학습 문서 및 관련 레이블이 포함된 완전한 JSON 객체입니다. 각 라인의 정확한 내용은 분류기 모드에 따라 다릅니다. 자세한 내용은 멀티클래스 모드 및 멀티레이블 모드를 참조하세요.

Amazon Comprehend에 학습 데이터를 제공할 때는 하나 이상의 레이블 속성 이름을 지정합니다. 지정하는 속성 이름의 수는 증강시킨 매니페스트 파일이 단일 레이블 지정 작업의 출력인지 아니면 체인 레이블 지정 작업의 출력인지에 따라 달라집니다.

파일이 단일 레이블 지정 작업의 출력인 경우 Ground Truth 작업에서 단일 레이블 속성 이름을 지정하십시오.

파일이 체인 레이블 지정 작업의 출력인 경우 체인에 있는 하나 이상의 작업에 대한 레이블 속성 이름을 지정하십시오. 각 레이블 속성 이름은 개별 작업의 주석을 제공합니다. 체인 레이블 지정 작업의 증강 매니페스트 파일에 대해 이러한 속성을 최대 5개까지 지정할 수 있습니다.

체인 레이블 지정 작업에 대한 자세한 내용과 해당 작업이 생성하는 출력의 예는 Amazon SageMaker AI 개발자 안내서의 체인 레이블 지정 작업을 참조하세요.

멀티클래스 모드

멀티클래스 모드에서 분류는 각 문서에 하나의 클래스를 할당합니다. 개별 클래스는 상호 배타적입니다. 예를 들어 영화를 코미디 또는 SF로 분류할 수 있지만 두 가지 모두로 분류할 수는 없습니다.

> **Note**
>
> Amazon Comprehend 콘솔에서는 멀티클래스 모드를 단일 레이블 모드라고 합니다.

주제

일반 텍스트 모델

일반 텍스트 모델을 훈련하려면 레이블이 지정된 훈련 데이터를 CSV 파일 또는 SageMaker AI Ground Truth의 증강 매니페스트 파일로 제공할 수 있습니다.

CSV 파일

CSV 파일을 사용하여 분류기를 학습시키는 방법에 대한 일반적인 정보는 CSV 파일를 참조하세요.

학습 데이터를 2열 CSV 파일로 제공합니다. 각 행의 첫 번째 열에는 클래스 레이블 값이 들어 있습니다. 두 번째 열에는 해당 클래스의 예제 텍스트 문서가 들어 있습니다. 각 행은\n 또는\r\n 문자로 끝나야 합니다.

다음 예제는 세 개의 문서가 있는 CSV 파일을 보여줍니다.

```
CLASS,Text of document 1
CLASS,Text of document 2
CLASS,Text of document 3
```

다음 예제는 이메일 메시지가 스팸인지 여부를 감지하도록 사용자 지정 분류기를 학습시키는 CSV 파일의 한 행을 보여줍니다.

```
SPAM,"Paulo, your $1000 award is waiting for you! Claim it while you still can at
 http://example.com."
```

증강 매니페스트 파일

분류기 학습을 위한 증강 매니페스트 파일 사용에 대한 일반적인 내용은 증강 매니페스트 파일를 참조하세요.

일반 텍스트 문서의 경우 증강 매니페스트 파일의 각 라인은 학습 문서, 단일 클래스 이름 및 Ground Truth의 기타 메타데이터를 포함하는 완전한 JSON 객체입니다. 다음 예제는 스팸 이메일 메시지를 인식하도록 사용자 지정 분류기를 학습하기 위한 증강 매니페스트 파일입니다.

```
{"source":"Document 1 text", "MultiClassJob":0, "MultiClassJob-metadata":
{"confidence":0.62, "job-name":"labeling-job/multiclassjob", "class-name":"not_spam",
 "human-annotated":"yes", "creation-date":"2020-05-21T17:36:45.814354",
 "type":"groundtruth/text-classification"}}
{"source":"Document 2 text", "MultiClassJob":1, "MultiClassJob-metadata":
{"confidence":0.81, "job-name":"labeling-job/multiclassjob", "class-name":"spam",
 "human-annotated":"yes", "creation-date":"2020-05-21T17:37:51.970530",
 "type":"groundtruth/text-classification"}}
{"source":"Document 3 text", "MultiClassJob":1, "MultiClassJob-metadata":
{"confidence":0.81, "job-name":"labeling-job/multiclassjob", "class-name":"spam",
 "human-annotated":"yes", "creation-date":"2020-05-21T17:37:51.970566",
 "type":"groundtruth/text-classification"}}
```

다음 예제는 증강 매니페스트 파일의 JSON 객체 하나를 가독성에 맞게 포맷한 것을 보여줍니다.

```
{
   "source": "Paulo, your $1000 award is waiting for you! Claim it while you still can
 at http://example.com.",
   "MultiClassJob": 0,
   "MultiClassJob-metadata": {
       "confidence": 0.98,
       "job-name": "labeling-job/multiclassjob",
       "class-name": "spam",
       "human-annotated": "yes",
       "creation-date": "2020-05-21T17:36:45.814354",
       "type": "groundtruth/text-classification"
   }
}
```

이 예제에서 `source` 속성은 학습 문서의 텍스트를 제공하고 `MultiClassJob` 속성은 분류 목록에 있는 하나의 클래스에 대한 색인을 할당합니다. `job-name` 속성은 Ground Truth에서 레이블 지정 작업에 대해 정의한 이름입니다.

Amazon Comprehend에서 분류기 학습 작업을 시작할 때 동일한 레이블 지정 작업 이름을 지정합니다.

네이티브 문서 모델

네이티브 문서 모델은 네이티브 문서(예: PDF, DOCX, 이미지)를 사용하여 학습시키는 모델입니다. 학습 데이터를 CSV 파일로 제공합니다.

CSV 파일

CSV 파일을 사용하여 분류기를 학습시키는 방법에 대한 일반적인 정보는 CSV 파일를 참조하세요.

학습 데이터를 3열 CSV 파일로 제공합니다. 각 행의 첫 번째 열에는 클래스 레이블 값이 들어 있습니다. 두 번째 열에는 이 클래스의 예제 문서의 파일 이름이 들어 있습니다. 세 번째 열에는 페이지 번호가 들어 있습니다. 예제 문서가 이미지인 경우 페이지 번호는 선택 사항입니다.

다음 예제는 세 개의 입력 문서를 참조하는 CSV 파일을 보여줍니다.

```
CLASS,input-doc-1.pdf,3
CLASS,input-doc-2.docx,1
CLASS,input-doc-3.png
```

다음 예제는 이메일 메시지가 스팸인지 여부를 감지하도록 사용자 지정 분류기를 학습시키는 CSV 파일의 한 행을 보여줍니다. PDF 파일의 2페이지에 스팸의 예가 나와 있습니다.

```
SPAM,email-content-3.pdf,2
```

멀티레이블 모드

멀티레이블 모드에서 개별 클래스는 상호 배타적이지 않은 서로 다른 범주를 나타냅니다. 멀티레이블 분류는 각 문서에 하나 이상의 클래스를 지정합니다. 예를 들어 한 영화를 다큐멘터리로 분류하고 다른 영화를 공상 과학, 액션, 코미디로 분류할 수 있습니다.

학습의 경우 멀티레이블 모드는 최대 100개의 고유한 클래스를 포함하는 최대 100만 개의 예제를 지원합니다.

주제

- 일반 텍스트 모델

- 네이티브 문서 모델

일반 텍스트 모델

일반 텍스트 모델을 훈련하려면 레이블이 지정된 훈련 데이터를 CSV 파일 또는 SageMaker AI Ground Truth의 증강 매니페스트 파일로 제공할 수 있습니다.

CSV 파일

CSV 파일을 사용하여 분류기를 학습시키는 방법에 대한 일반적인 정보는 CSV 파일를 참조하세요.

학습 데이터를 2열 CSV 파일로 제공합니다. 각 행의 첫 번째 열에는 클래스 레이블 값이 들어 있고 두 번째 열에는 이러한 클래스에 대한 예제 텍스트 문서가 들어 있습니다. 첫 번째 열에 클래스를 두 개 이상 입력하려면 각 클래스 사이에 구분 기호(예: |)를 사용합니다.

```
CLASS,Text of document 1
CLASS,Text of document 2
CLASS|CLASS|CLASS,Text of document 3
```

다음 예제는 영화 요약에서 장르를 감지하도록 사용자 지정 분류기를 학습시키는 CSV 파일의 한 행을 보여줍니다.

```
COMEDY|MYSTERY|SCIENCE_FICTION|TEEN,"A band of misfit teens become unlikely detectives
 when they discover troubling clues about their high school English teacher. Could the
 strange Mrs. Doe be an alien from outer space?"
```

클래스 이름 사이의 기본적인 구분 기호는 파이프(|) 입니다. 하지만 다른 문자를 구분 기호로 사용할 수도 있습니다. 구분 기호는 사용자의 클래스 이름의 모든 문자와 구별되어야 합니다. 예를 들어 클래스가 CLASS_1, CLASS_2, 및 CLASS_3인 경우 밑줄(_)은 클래스 이름의 일부입니다. 따라서 클래스 이름을 구분할 때 밑줄을 구분 기호로 사용하지 마십시오.

증강 매니페스트 파일

분류기 학습을 위한 증강 매니페스트 파일 사용에 대한 일반적인 내용은 증강 매니페스트 파일를 참조하세요.

일반 텍스트 문서의 경우 증강 매니페스트 파일의 각 라인은 완전한 JSON 객체입니다. 여기에는 Ground Truth의 학습 문서, 클래스 이름 및 기타 메타데이터가 포함됩니다. 다음 예제는 영화 요약에서 장르를 감지하도록 사용자 지정 분류기를 학습시키는 데 사용되는 증강 매니페스트 파일입니다.

```
{"source":"Document 1 text", "MultiLabelJob":[0,4], "MultiLabelJob-metadata":{"job-
name":"labeling-job/multilabeljob", "class-map":{"0":"action", "4":"drama"}, "human-
annotated":"yes", "creation-date":"2020-05-21T19:02:21.521882", "confidence-map":
{"0":0.66}, "type":"groundtruth/text-classification-multilabel"}}
{"source":"Document 2 text", "MultiLabelJob":[3,6], "MultiLabelJob-metadata":{"job-
name":"labeling-job/multilabeljob", "class-map":{"3":"comedy", "6":"horror"}, "human-
annotated":"yes", "creation-date":"2020-05-21T19:00:01.291202", "confidence-map":
{"1":0.61,"0":0.61}, "type":"groundtruth/text-classification-multilabel"}}
{"source":"Document 3 text", "MultiLabelJob":[1], "MultiLabelJob-metadata":
{"job-name":"labeling-job/multilabeljob", "class-map":{"1":"action"}, "human-
annotated":"yes", "creation-date":"2020-05-21T18:58:51.662050", "confidence-map":
{"1":0.68}, "type":"groundtruth/text-classification-multilabel"}}
```

다음 예제는 증강 매니페스트 파일의 JSON 객체 하나를 가독성에 맞게 포맷한 것을 보여줍니다.

```
{
      "source": "A band of misfit teens become unlikely detectives when
                  they discover troubling clues about their high school English
 teacher.
                    Could the strange Mrs. Doe be an alien from outer space?",
      "MultiLabelJob": [
          3,
          8,
          10,
          11
      ],
      "MultiLabelJob-metadata": {
          "job-name": "labeling-job/multilabeljob",
          "class-map": {
              "3": "comedy",
              "8": "mystery",
              "10": "science_fiction",
              "11": "teen"
          },
          "human-annotated": "yes",
          "creation-date": "2020-05-21T19:00:01.291202",
          "confidence-map": {
              "3": 0.95,
              "8": 0.77,
              "10": 0.83,
              "11": 0.92
          },
          "type": "groundtruth/text-classification-multilabel"
```

```
    }
}
```

이 예제에서 source 속성은 학습 문서의 텍스트를 제공하고 MultiLabelJob 속성은 분류 목록에 있는 여러 클래스의 색인을 할당합니다. MultiLabelJob 메타데이터의 작업 이름은 Ground Truth에서 레이블 지정 작업에 대해 정의한 이름입니다.

네이티브 문서 모델

네이티브 문서 모델은 네이티브 문서(예: PDF, DOCX, 이미지 파일)를 사용하여 학습시키는 모델입니다. 레이블이 지정된 학습 데이터는 CSV 파일로 제공합니다.

CSV 파일

CSV 파일을 사용하여 분류기를 학습시키는 방법에 대한 일반적인 정보는 CSV 파일를 참조하세요.

학습 데이터를 3열 CSV 파일로 제공합니다. 각 행의 첫 번째 열에는 클래스 레이블 값이 들어 있습니다. 두 번째 열에는 이러한 클래스의 예제 문서의 파일 이름이 들어 있습니다. 세 번째 열에는 페이지 번호가 들어 있습니다. 예제 문서가 이미지인 경우 페이지 번호는 선택 사항입니다.

첫 번째 열에 클래스를 두 개 이상 입력하려면 각 클래스 사이에 구분 기호(예: |)를 사용합니다.

```
CLASS,input-doc-1.pdf,3
CLASS,input-doc-2.docx,1
CLASS|CLASS|CLASS,input-doc-3.png,2
```

다음 예제는 영화 요약에서 장르를 감지하도록 사용자 지정 분류기를 학습시키는 CSV 파일의 한 행을 보여줍니다. PDF 파일의 2페이지에는 코미디/청소년 영화의 예시가 포함되어 있습니다.

```
COMEDY|TEEN,movie-summary-1.pdf,2
```

클래스 이름 사이의 기본적인 구분 기호는 파이프(|) 입니다. 하지만 다른 문자를 구분 기호로 사용할 수도 있습니다. 구분 기호는 사용자의 클래스 이름의 모든 문자와 구별되어야 합니다. 예를 들어 클래스가 CLASS_1, CLASS_2, 및 CLASS_3인 경우 밑줄(_)은 클래스 이름의 일부입니다. 따라서 클래스 이름을 구분할 때 밑줄을 구분 기호로 사용하지 마십시오.

학습 분류 모델

사용자 지정 분류를 위해 모델을 학습시키려면 범주를 정의하고 사용자 지정 모델을 학습시키는 데 필요한 예제 문서를 제공합니다. 멀티클래스 또는 멀티레이블 모드에서 모델을 학습시킵니다. 멀티클래

스 모드는 단일 클래스를 각 문서와 연결합니다. 멀티레이블 모드는 하나 이상의 클래스를 각 문서와 연결합니다.

사용자 지정 분류는 일반 텍스트 모델과 네이티브 문서 모델이라는 두 가지 유형의 분류기 모델을 지원합니다. 일반 텍스트 모델은 텍스트 내용을 기준으로 문서를 분류합니다. 네이티브 문서 모델 또한 텍스트 내용을 기준으로 문서를 분류합니다. 네이티브 문서 모델은 문서 레이아웃에서 보내는 추가 신호를 사용할 수도 있습니다. 네이티브 문서 모델을 네이티브 문서로 학습시켜 모델이 레이아웃 정보를 학습할 수 있도록 합니다.

일반 텍스트 모델은 다음과 같은 특성을 갖습니다.

- UTF-8 인코딩 텍스트 문서를 사용하여 모델을 학습합니다.
- 영어, 스페인어, 독일어, 이탈리아어, 프랑스어 또는 포르투갈어 중 하나의 언어로 된 문서를 사용하여 모델을 학습할 수 있습니다.
- 지정된 분류기의 학습 문서는 모두 같은 언어를 사용해야 합니다.
- 학습 문서는 일반 텍스트이므로 텍스트 추출에 대한 추가 비용은 없습니다.

네이티브 문서 모델은 다음과 같은 특성을 갖습니다.

- 다음과 같은 문서 유형을 포함하는 반정형 문서를 사용하여 모델을 학습시킵니다.
  - 디지털 및 스캔한 PDF 문서.
  - Word 문서(DOCX).
  - 이미지: JPG 파일, PNG 파일 및 단일 페이지 TIFF 파일.
  - Textract API 출력 JSON 파일
- 영어 문서를 사용하여 모델을 학습시킵니다.
- 학습 문서에 스캔한 문서 파일이 포함된 경우 텍스트 추출에 대한 추가 비용이 발생합니다. 자세한 내용은 Amazon Comprehend 요금 페이지를 참조하세요.

두 모델 중 하나를 사용하여 지원되는 모든 문서 유형을 분류할 수 있습니다. 그러나 가장 정확한 결과를 얻으려면 일반 텍스트 모델을 사용하여 일반 텍스트 문서를 분류하고 네이티브 문서 모델을 사용하여 반정형 문서를 분류하는 것이 좋습니다.

주제

- 사용자 지정 분류기 학습(콘솔)
- 사용자 지정 분류기 학습시키기(API)

- 학습 데이터 테스트
- 분류기 학습 출력
- 사용자 지정 분류기 지표

사용자 지정 분류기 학습(콘솔)

콘솔을 사용하여 사용자 지정 분류기를 만들고 학습시킨 다음 사용자 지정 분류기를 사용하여 문서를 분석할 수 있습니다.

사용자 지정 분류기를 학습시키려면 일련의 학습 문서가 필요합니다. 문서 분류기가 인식할 수 있도록 사용자가 원하는 범주로 이러한 문서에 레이블을 지정합니다. 학습 문서 준비에 대한 자세한 내용은 분류기 학습 데이터 준비를 참조하세요.

문서 분류기 모델 생성 및 학습하기

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 사용자 정의을 선택한 다음 사용자 정의 분류를 선택합니다.
3. 새 모델 생성을 선택합니다.
4. 모델 설정에서 분류기의 모델 이름을 입력합니다. 이 이름은 귀하의 계정과 현재의 리전 내에서 고유해야 합니다.

   (선택) 버전 이름을 입력합니다. 이 이름은 귀하의 계정과 현재의 리전 내에서 고유해야 합니다.
5. 학습 문서의 언어를 선택합니다. 분류기가 지원하는 언어를 보려면 학습 분류 모델를 참조하세요.
6. (선택) Amazon Comprehend가 학습 작업을 처리하는 동안 스토리지 볼륨의 데이터를 암호화하려면 분류기 암호화를 선택합니다. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.

   - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID의 키 ID를 선택합니다.
   - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ARN에 키 ID의 ARN을 입력합니다.

   > **Note**
   >
   > KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 AWS Key Management Service (AWS KMS)를 참조하세요.

7. 데이터 사양에서 사용할 학습 모델 유형을 선택합니다.

   - 일반 텍스트 문서: 일반 텍스트 모델을 만들려면 이 옵션을 선택합니다. 일반 텍스트 문서를 사용하여 모델을 학습시킵니다.
   - 네이티브 문서: 네이티브 문서 모델을 만들려면 이 옵션을 선택합니다. 네이티브 문서(PDF, Word, 이미지)를 사용하여 모델을 학습시킵니다.

8. 학습 데이터의 데이터 형식을 선택합니다. 데이터 파일 형식에 대한 자세한 내용은 분류기 학습 파일 형식를 참조하세요.

   - CSV 파일: 학습 데이터가 CSV 파일 형식을 사용하는 경우 이 옵션을 선택합니다.
   - 증강 매니페스트: Ground Truth를 사용하여 학습 데이터용 증강 매니페스트 파일을 만든 경우 이 옵션을 선택합니다. 학습 모델 유형으로 일반 텍스트 문서를 선택한 경우 이 형식을 사용할 수 있습니다.

9. 사용할 분류기 모드를 선택합니다.

   - 단일 레이블 모드: 문서에 할당할 범주가 상호 배타적이며 각 문서에 하나의 레이블을 할당하도록 분류기를 학습시키려는 경우 이 모드를 선택합니다. Amazon Comprehend API에서는 단일 레이블 모드를 멀티클래스 모드라고 합니다.
   - 멀티레이블 모드: 문서에 여러 범주를 동시에 적용할 수 있고 각 문서에 하나 이상의 레이블을 할당하도록 분류기를 학습시키려는 경우 이 모드를 선택합니다.

10. 멀티레이블 모드를 선택하면 레이블의 구분 기호를 선택할 수 있습니다. 학습 문서에 여러 클래스가 있는 경우 이 구분 기호를 사용하여 레이블을 구분할 수 있습니다. 기본 구분 기호는 파이프 문자입니다.

11. (선택) 데이터 형식으로 증강 매니페스트를 선택한 경우 증강 매니페스트 파일을 5개까지 입력할 수 있습니다. 각 증강 매니페스트 파일에는 학습 데이터세트 또는 테스트 데이터세트가 들어 있습니다. 최소 하나의 학습 데이터세트를 제공해야 합니다. 테스트 데이터세트는 선택 사항입니다. 다음 단계에 따라 증강 매니페스트 파일을 구성합니다.

    a. 학습 및 테스트 데이터세트에서 입력 위치 패널을 펼칩니다.

    b. 데이터세트 유형에서 학습 데이터 또는 테스트 데이터를 선택합니다.

    c. SageMaker AI Ground Truth 증강 매니페스트 파일 S3 위치에 매니페스트 파일이 포함된 Amazon S3 버킷의 위치를 입력하거나 S3 찾아보기를 선택하여 탐색합니다. 학습 작업을 위한 액세스 권한에 사용하는 IAM 역할에는 S3 버킷에 대한 읽기 권한이 있어야 합니다.

    d. 속성 이름에는 주석이 포함된 속성의 이름을 입력합니다. 파일에 여러 체인으로 연결된 레이블 작업의 주석이 포함되어 있으면 각 작업에 대한 속성을 추가하십시오.

    e. 다른 입력 위치를 추가하려면 입력 위치 추가를 선택하고 다음 위치를 구성합니다.

12. (선택) CSV 파일을 데이터 형식으로 선택한 경우 다음 단계를 사용하여 학습 데이터 세트와 선택적 테스트 데이터 세트를 구성합니다.

    a. 학습 데이터세트에서 학습 데이터 CSV 파일이 들어 있는 Amazon S3 버킷의 위치를 입력하거나 S3 찾아보기를 선택하여 해당 버킷으로 이동합니다. 학습 작업을 위한 액세스 권한에 사용하는 IAM 역할에는 S3 버킷에 대한 읽기 권한이 있어야 합니다.

    (선택) 학습 모델 유형으로 네이티브 문서를 선택한 경우 학습 예제 파일이 들어 있는 Amazon S3 폴더의 URL도 제공해야 합니다.

    b. 테스트 데이터 세트에서 Amazon Comprehend가 학습된 모델을 테스트할 수 있도록 추가 데이터를 제공할지 여부를 선택합니다.

    - 자동 분할: 자동 분할은 테스트 데이터로 사용하기 위해 학습 데이터의 10%를 자동으로 선택하여 비축합니다.
    - (선택) 고객 제공: Amazon S3에 있는 테스트 데이터 CSV 파일의 URL을 입력합니다. Amazon S3에서 해당 위치로 이동하여 폴더 선택을 선택할 수도 있습니다.

      (선택) 학습 모델 유형으로 네이티브 문서를 선택한 경우 테스트 파일이 포함된 Amazon S3 폴더의 URL도 제공해야 합니다.

13. (선택) 문서 읽기 모드에서 기본 텍스트 추출 작업을 우선 지정할 수 있습니다. 이 옵션은 스캔한 문서의 텍스트 추출에 적용되므로 일반 텍스트 모델에는 필요하지 않습니다. 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하십시오.
14. (일반 텍스트 모델, 선택) 출력 데이터에 혼동행렬과 같은 학습 출력 데이터를 저장할 Amazon S3 버킷의 위치를 입력합니다. 자세한 내용은 혼동행렬를 참조하십시오.

    (선택) 학습 작업의 출력 결과를 암호화하기로 선택한 경우 암호화를 선택합니다. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.

    - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID의 키 별칭을 선택하십시오.
    - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ID 아래에 키 별칭 또는 ID의 ARN을 입력합니다.

15. IAM 역할의 경우 기존 IAM 역할 선택을 선택한 다음 학습 문서가 포함된 S3 버킷에 대한 읽기 권한이 있는 기존 IAM 역할을 선택합니다. 역할에는 `comprehend.amazonaws.com`으로 시작하는 신뢰 정책이 있어야 유효합니다.

    이러한 권한을 가진 IAM 역할이 아직 없는 경우, IAM 역할 생성을 선택하여 역할을 생성하십시오. 이 역할을 부여할 액세스 권한을 선택한 다음 이름 접미사를 선택하여 사용자 계정의 IAM 역할과 이 역할을 구분합니다.

> **Note**
>
> 암호화된 입력 문서의 경우 사용되는 IAM 역할에도 `kms:Decrypt` 권한이 있어야 합니다. 자세한 내용은 KMS 암호화를 사용하는 데 필요한 권한을 참조하십시오.

16. (선택) VPC에서 Amazon Comprehend로 리소스를 시작하려면 VPC 아래에 VPC ID를 입력하거나 드롭다운 목록에서 ID를 선택합니다.

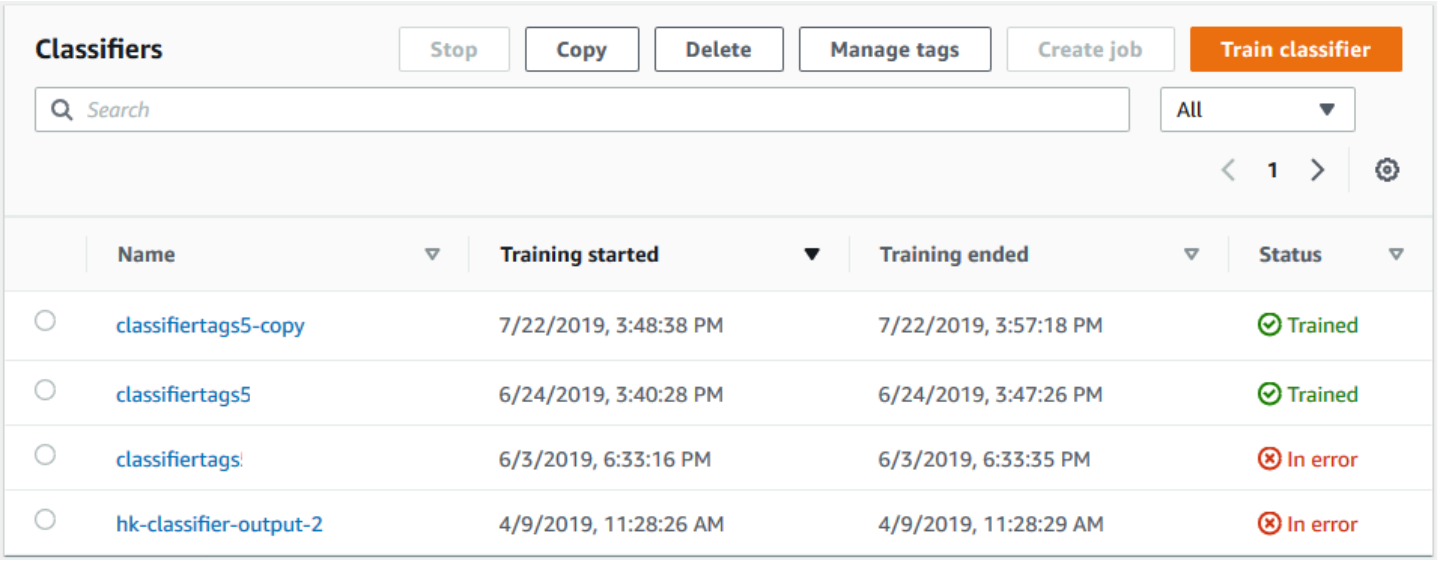
    1. 서브넷에서 서브넷을 선택합니다. 첫 번째 서브넷을 선택한 후 추가 서브넷을 선택할 수 있습니다.
    2. 보안 그룹을 지정한 경우, 보안 그룹에서 사용할 보안 그룹을 선택합니다. 첫 번째 보안 그룹을 선택한 후 추가 보안 그룹을 선택할 수 있습니다.

> **Note**
>
> 분류 작업에 VPC를 사용하는 경우 생성 및 시작 작업에 사용되는 `DataAccessRole`은 입력 문서와 출력 버킷에 액세스하는 VPC에 대한 권한이 있어야 합니다.

17. (선택) 사용자 지정 분류기에 태그를 추가하려면 태그에 키-값 페어를 입력합니다. 태그 추가를 선택합니다. 분류기를 만들기 전에 이 쌍을 제거하려면 태그 제거를 선택합니다. 자세한 내용은 리소스에 태그 지정을 참조하십시오.
18. 생성을 선택합니다.

콘솔에 분류기 페이지가 표시됩니다. 새 분류기가 테이블에 나타나고 `Submitted`으로 상태가 표시됩니다. 분류기가 학습 문서를 처리하기 시작하면 상태가 `Training`으로 바뀝니다. 분류기를 사용할 준비가 되면 상태가 `Trained`또는 `Trained with warnings`으로 변경됩니다. 상태가 `TRAINED_WITH_WARNINGS`인 경우 분류기 학습 출력에서 건너뛴 파일 폴더를 검토하십시오.

Amazon Comprehend에서 생성 또는 학습 중에 오류가 발생한 경우 상태가 `In error`로 변경됩니다. 표에서 분류기 작업을 선택하여 오류 메시지를 포함하여 분류기에 대한 추가 정보를 얻을 수 있습니다.



사용자 지정 분류기 학습시키기(API)

사용자 지정 분류기를 만들고 학습하려면 문서 분류기 만들기 작업을 사용하십시오.

DescribeDocumentClas3 작업을 사용하여 요청 진행 상황을 모니터링할 수 있습니다. `Status` 필드가 `TRAINED`으로 전환된 후 분류기를 사용하여 문서를 분류할 수 있습니다. 상태가 `TRAINED_WITH_WARNINGS`인 경우 분류기 학습 출력에 있는 `CreateDocumentClassifier` 작업에서 건너뛴 파일 폴더를 검토하십시오.

주제

- 를 사용하여 사용자 지정 분류 훈련 AWS Command Line Interface
- AWS SDK for Java 또는 SDK for Python 사용

를 사용하여 사용자 지정 분류 훈련 AWS Command Line Interface

다음 예제는 AWS CLI를 사용하여 `CreateDocumentClassifier` 작업, `DescribeDocumentClassificationJob`작업 및 기타 사용자 지정 분류기 API를 사용하는 방법을 보여줍니다.

Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

`create-document-classifier` 작업을 사용하여 일반 텍스트 사용자 지정 분류기를 만듭니다.

```
aws comprehend create-document-classifier \
```

```
     --region region \
     --document-classifier-name testDelete \
     --language-code en \
     --input-data-config S3Uri=s3://S3Bucket/docclass/file name \
     --data-access-role-arn arn:aws:iam::account number:role/testFlywheelDataAccess
```

네이티브 사용자 지정 분류기를 만들려면 `create-document-classifier` 요청에 다음과 같은 추가 파라미터를 제공합니다.

1. 문서유형: 값을 SEMI_STRUCTURED_DOCUMENT로 설정합니다.
2. 문서: 학습 문서(및 선택적으로 테스트 문서)를 위한 S3 위치.
3. OutputDataConfig: 출력 문서의 S3 위치(및 선택적 KMS 키)를 제공합니다.
4. DocumentReaderConfig: 텍스트 추출 설정을 위한 선택적 필드입니다.

```
aws comprehend create-document-classifier \
     --region region \
     --document-classifier-name testDelete \
     --language-code en \
     --input-data-config
          S3Uri=s3://S3Bucket/docclass/file name \
           DocumentType \
             Documents  \
     --output-data-config S3Uri=s3://S3Bucket/docclass/file name \
     --data-access-role-arn arn:aws:iam::account number:role/testFlywheelDataAccess
```

`DescribeDocumentClassifier` 작업을 사용하여 문서 분류기 ARN으로 사용자 지정 분류기에 대한 정보를 가져옵니다.

```
aws comprehend describe-document-classifier \
     --region region \
     --document-classifier-arn arn:aws:comprehend:region:account number:document-
classifier/file name
```

`DeleteDocumentClassifier` 작업을 사용하여 사용자 지정 분류기를 삭제합니다.

```
aws comprehend delete-document-classifier \
     --region region \
     --document-classifier-arn arn:aws:comprehend:region:account number:document-
classifier/testDelete
```

`ListDocumentClassifiers` 작업을 사용하여 계정의 모든 사용자 지정 분류기를 나열합니다.

```
aws comprehend list-document-classifiers
     --region region
```

AWS SDK for Java 또는 SDK for Python 사용

사용자 지정 분류기를 만들고 학습시키는 방법에 대한 SDK 예제는 AWS SDK 또는 CLI와 `CreateDocumentClassifier` 함께 사용를 참조하세요.

학습 데이터 테스트

모델을 학습시킨 후 Amazon Comprehend는 사용자 지정 분류기 모델을 테스트합니다. 테스트 데이터 세트를 제공하지 않는 경우 Amazon Comprehend는 학습 데이터의 90%를 사용하여 모델을 학습시킵니다. 학습 데이터의 10%를 테스트에 사용할 수 있도록 비축합니다. 테스트 데이터 세트를 제공하는 경우 테스트 데이터에는 학습 데이터 세트의 각 고유 레이블에 대한 예제가 하나 혹은 다수 포함되어야 합니다.

모델을 테스트하면 모델의 정확도를 추정하는 데 사용할 수 있는 지표가 제공됩니다. 콘솔은 콘솔의 분류기 세부 정보 페이지에 있는 분류기 성능 섹션에 지표를 표시합니다. 또한 DescribeDocumentClassifier 작업에서 반환되는 `Metrics` 필드에도 반환됩니다.

다음 예제 학습 데이터에는 DOCUMENTARY, DOCUMENTARY, SCIENCE_FICTION, DOCUMENTARY, ROMANTIC_COMEDY라는 다섯 개의 레이블이 있습니다. DOCUMENTARY, SCIENCE_FICTION, 및 ROMANTIC_COMEDY의 세 가지 클래스가 있습니다.

1열	2열
DOCUMENTARY	문서 텍스트 1
DOCUMENTARY	문서 텍스트 2
SCIENCE_FICTION	문서 텍스트 3
DOCUMENTARY	문서 텍스트 4
ROMANTIC_COMEDY	문서 텍스트 5

자동 분할(Amazon Comprehend가 테스트에 사용하기 위해 학습 데이터의 10%를 비축하는 경우)의 경우, 학습 데이터에 특정 레이블의 제한된 예가 포함되어 있으면 테스트 데이터 세트에 포함된 해당 레이블의 예는 0개일 수 있습니다. 예를 들어 학습 데이터세트에 DOCUMENTARY 클래스 인스턴스 1000개, SCIENCE_FICTION 인스턴스 900개, ROMANTIC_COMEDY 인스턴스 1개가 포함되어 있는 경우 테스트 데이터세트에는 DOCUMENTARY 100개와 SCIENCE_FICTION 인스턴스 90개가 포함되지만 ROMANTIC_COMEDY 인스턴스는 유일한 예제이므로 포함될 수 없습니다.

모델 학습을 마치면 학습 메트릭을 통해 모델이 필요에 맞게 충분히 정확한지 판단하는 데 사용할 수 있는 정보가 제공됩니다.

분류기 학습 출력

Amazon Comprehend는 사용자 지정 분류기 모델 학습을 완료한 후 CreateDocumentClassifier API 요청 또는 이에 상응하는 콘솔 요청에서 지정한 Amazon S3 출력 위치에 출력 파일을 생성합니다.

Amazon Comprehend는 일반 텍스트 모델 또는 네이티브 문서 모델을 학습시킬 때 혼동행렬을 생성합니다. 네이티브 문서 모델을 학습시킬 때 추가 출력 파일을 생성할 수 있습니다.

주제

- 혼동행렬
- 네이티브 문서 모델에 대한 추가 출력

혼동행렬

사용자 지정 분류기 모델을 학습시키면 Amazon Comprehend는 모델이 학습에서 얼마나 잘 수행되었는지에 대한 지표를 제공하는 혼동행렬을 생성합니다. 이 행렬은 모델이 예측한 레이블 행렬을 실제 문서 레이블과 비교하여 보여줍니다. Amazon Comprehend는 학습 데이터의 일부를 사용하여 혼동행렬을 생성합니다.

혼동행렬은 모델 성능을 개선하기 위해 어떤 클래스가 더 많은 데이터를 사용할 수 있는지를 나타냅니다. 정확한 예측의 비율이 높은 클래스는 행렬의 대각선을 따라 나타나는 결과 수가 가장 많습니다. 대각선에 있는 숫자가 더 작은 경우 클래스의 정답 예측 비율이 더 낮습니다. 이 클래스에 더 많은 학습 예제를 추가하고 모델을 다시 학습시킬 수 있습니다. 예를 들어, 레이블 A 샘플의 40%가 레이블 D로 분류되는 경우 레이블 A와 레이블 D에 더 많은 샘플을 추가하면 분류기의 성능이 향상됩니다.

Amazon Comprehend가 분류기 모델을 생성한 후에는 S3 출력 위치의 `confusion_matrix.json` 파일에서 혼동행렬을 사용할 수 있습니다.

혼동행렬의 형식은 분류기를 멀티클래스 모드를 사용하여 학습시켰는지, 멀티레이블 모드를 사용하여 학습시켰는지에 따라 달라집니다.

주제

- 멀티클래스 모드의 혼동행렬
- 멀티레이블 모드의 혼동행렬

멀티클래스 모드의 혼동행렬

멀티클래스 모드에서는 개별 클래스가 상호 배타적이므로 분류는 각 문서에 하나의 레이블을 할당합니다. 예를 들어 동물은 개일 수도 있고 고양일 수도 있지만 동시에 둘 다일 수는 없습니다.

멀티클래스로 학습된 분류기에 대한 혼동행렬의 다음 예를 생각해 보십시오.

```
  A B X Y <-(predicted label)
A 1 2 0 4
B 0 3 0 1
X 0 0 1 0
Y 1 1 1 1
^
|
(actual label)
```

이 경우 모델은 다음과 같이 예측하였습니다.

- 하나의 “A” 레이블은 정확하게 예측되었고, 두 개의 “A” 레이블은 “B” 레이블로 잘못 예측되었으며 네 개의 “A” 레이블은 “Y” 레이블로 잘못 예측되었습니다.
- 세 개의 “B” 레이블은 정확하게 예측되었고 하나의 “B” 레이블은 “Y” 레이블로 잘못 예측되었습니다.
- 하나의 “X”가 정확하게 예측되었습니다.
- 하나의 “Y” 레이블은 정확하게 예측되었고, 하나는 “A” 레이블로 잘못 예측되었으며, 하나는 “B” 레이블로 잘못 예측되었고, 하나는 “X” 레이블로 잘못 예측되었습니다.

행렬의 대각선(A:A, B:B, X:X, Y:Y)은 정확한 예측치를 보여줍니다. 예측 오차는 대각선 밖의 값입니다. 이 경우 행렬은 다음과 같은 예측 오류율을 보여줍니다.

- A 레이블: 86%
- B 레이블: 25%

- X 레이블: 0%
- Y 레이블: 75%

분류기는 혼동행렬을 JSON 형식의 파일로 반환합니다. 다음 JSON 파일은 이전 예제의 행렬을 나타냅니다.

```
{
 "type": "multi_class",
 "confusion_matrix": [
 [1, 2, 0,4],
 [0, 3, 0, 1],
 [0, 0, 1, 0],
 [1, 1, 1, 1]],
 "labels": ["A", "B", "X", "Y"],
 "all_labels": ["A", "B", "X", "Y"]
}
```

멀티레이블 모드의 혼동행렬

멀티레이블 모드에서 분류는 문서에 하나 이상의 클래스를 할당할 수 있습니다. 멀티클래스로 학습된 분류기에 대한 혼동행렬의 다음 예를 생각해 보십시오.

이 예제에서는 Comedy, Action 및 Drama로 지정할 수 있는 3개의 레이블을 사용할 수 있습니다. 멀티레이블 혼동행렬은 각 레이블에 대해 하나의 2x2 행렬을 만듭니다.

```
Comedy                  Action                  Drama
     No Yes                  No Yes                  No Yes   <-(predicted label)

 No  2   1               No  1   1               No  3   0

Yes  0   2              Yes  2   1              Yes  1   1
 ^                       ^                       ^
 |                       |                       |
 |-----------(was this label actually used)--------|
```

이 경우 모델은 Comedy 레이블에 대해 다음을 반환하였습니다.

- Comedy 레이블이 존재할 것으로 정확하게 예측된 사례는 2건. 참 긍정(TP)
- Comedy 레이블이 없을 것으로 정확하게 예측된 사례는 2건. 참 부정(TN)

- Comedy 레이블이 존재할 것으로 잘못 예측된 사례는 0건. 거짓 긍정(FP)
- Comedy 레이블이 없을 것으로 잘못 예측된 사례는 1건. 거짓 부정(FN)

멀티클래스 혼동행렬과 마찬가지로 각 행렬의 대각선은 정확한 예측치를 보여줍니다.

이 경우 모델은 80%의 확률(TP+TN)로 Comedy 레이블을 정확하게 예측하고 20%(FP+FN)의 확률로 레이블을 잘못 예측했습니다.

분류기는 혼동행렬을 JSON 형식의 파일로 반환합니다. 다음 JSON 파일은 이전 예제의 행렬을 나타냅니다.

```
{
"type": "multi_label",
"confusion_matrix": [
 [[2, 1],
 [0, 2]],
 [[1, 1],
 [2, 1]],
 [[3, 0],
 [1, 1]]
],
"labels": ["Comedy", "Action", "Drama"]
"all_labels": ["Comedy", "Action", "Drama"]
}
```

네이티브 문서 모델에 대한 추가 출력

Amazon Comprehend는 네이티브 문서 모델을 학습시킬 때 추가 출력 파일을 생성할 수 있습니다.

Amazon Textract 출력

Amazon Comprehend가 Amazon Textract API를 간접 호출하여 학습 문서의 텍스트를 추출한 경우 Amazon Textract 출력 파일을 S3 출력 위치에 저장합니다. 다음과 같은 디렉터리 구조를 사용합니다.

- 학습 문서:

  `amazon-textract-output/train/<file_name>/<page_num>/textract_output.json`
- 테스트 문서:

  `amazon-textract-output/test/<file_name>/<page_num>/textract_output.json`

귀하가 API 요청에서 테스트 문서를 제공한 경우 Amazon Comprehend가 테스트 폴더를 채웁니다.

문서 주석 실패

실패한 주석이 있는 경우 Amazon Comprehend는 Amazon S3 출력 위치(skipped_documents/ 폴더 내)에 다음 파일을 생성합니다.

- failed_annotations_train.jsonl

  학습 데이터에서 주석이 실패한 경우 파일이 존재합니다.
- failed_annotations_test.jsonl

  요청에 테스트 데이터가 포함되어 있고 테스트 데이터에 주석이 실패한 경우 파일이 존재합니다.

실패한 주석 파일은 다음 형식의 JSONL 파일입니다.

```
{
     "File": "String", "Page": Number, "ErrorCode": "...", "ErrorMessage": "..."}
    {"File": "String", "Page": Number, "ErrorCode": "...", "ErrorMessage": "..."
  }
```

사용자 지정 분류기 지표

Amazon Comprehend는 사용자 지정 분류기의 성능을 추정하는 데 도움이 되는 지표를 제공합니다. Amazon Comprehend는 분류기 학습 작업의 테스트 데이터를 사용하여 지표를 계산합니다. 지표는 학습 중 모델의 성능을 정확하게 나타내므로 유사한 데이터를 분류할 때 모델 성능을 근사치로 계산합니다.

DescribeDocumentClassifier와 같은 API 작업을 사용하여 사용자 지정 분류기에 대한 지표를 검색할 수 있습니다.

> **Note**
>
> 기본 정밀도, 재현율 및 F1 점수 지표에 대한 이해는 지표: 정밀도, 재현율, FScore를 참조하세요. 이러한 지표는 클래스 수준에서 정의됩니다. Amazon Comprehend는 다음에서 설명하는 것처럼 매크로 평균을 사용하여 이러한 지표를 테스트 세트 P, R 및 F1에 결합합니다.

주제

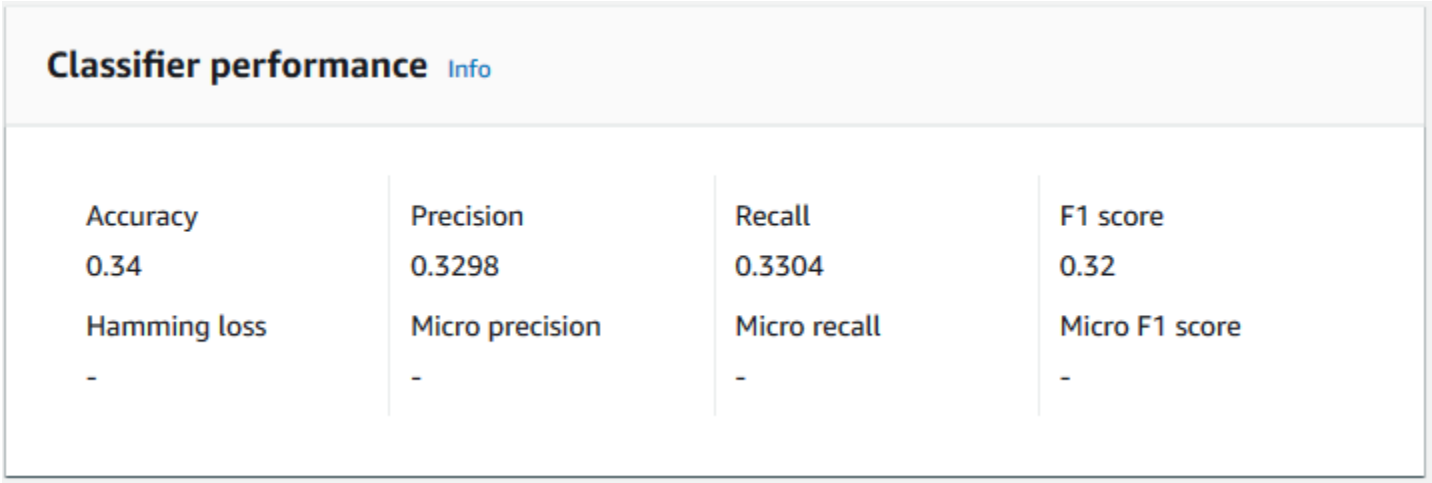
Metrics

Amazon Comprehend는 다음 지표를 지원합니다.

주제

- 정확도
- 정밀도(매크로 정밀도)
- 재현율(매크로 재현율)
- F1 점수(매크로 F1 점수)
- 해밍 손실
- 마이크로 정밀도
- 마이크로 재현율
- 마이크로 F1 점수

분류기의 지표를 보려면 콘솔에서 분류기 세부 정보 페이지를 여십시오.



정확도

정확도는 모델이 정확하게 예측한 테스트 데이터의 레이블 비율을 나타냅니다. 정확도를 계산하려면 테스트 문서에서 정확하게 예측된 레이블 수를 테스트 문서의 총 레이블 수로 나눕니다.

예

실제 레이블	예측 레이블	정확/부정확
1	1	정확
0	1	부정확:
2	3	부정확:
3	3	정확
2	2	정확
1	1	정확
3	3	정확

정확도는 정확한 예측 수를 전체 테스트 표본 수로 나눈 값(= 5/7 = 0.714 또는 71.4%)으로 구성됩니다.

정밀도(매크로 정밀도)

정밀도는 테스트 데이터에서 분류기 결과의 유용성을 측정한 것입니다. 정확하게 분류된 문서 수를 해당 클래스의 총 분류 수로 나눈 값으로 정의됩니다. 정밀도가 높다는 것은 관련성이 없는 결과보다 관련성이 있는 결과를 분류기가 훨씬 더 많이 반환했음을 의미합니다.

이 `Precision` 지표는 매크로 정밀도라고도 합니다.

다음 예제는 테스트 세트의 정밀도 결과를 보여줍니다.

레이블	샘플 크기	레이블 정밀도
레이블_1	400	0.75
레이블_2	300	0.80
레이블_3	30000	0.90
레이블_4	20	0.50
레이블_5	10	0.40

따라서 모델의 정밀도(매크로 정밀도) 측정법은 다음과 같습니다.

```
Macro Precision = (0.75 + 0.80 + 0.90 + 0.50 + 0.40)/5 = 0.67
```

재현율(매크로 재현율)

이는 모델이 예측할 수 있는 텍스트의 올바른 카테고리 비율을 나타냅니다. 이 지표는 사용 가능한 모든 레이블의 재현율 점수 평균 결과입니다. 재현율은 테스트 데이터에 대한 분류기 결과의 완전성을 나타내는 척도입니다.

재현율이 높다는 것은 분류기가 관련 결과를 대부분 반환했음을 의미합니다.

이 `Recall` 지표는 매크로 재현율이라고도 합니다.

다음 예는 테스트 세트의 재현율 결과를 보여줍니다.

레이블	샘플 크기	레이블 재현율
레이블_1	400	0.70
레이블_2	300	0.70
레이블_3	30000	0.98
레이블_4	20	0.80
레이블_5	10	0.10

따라서 모델의 재현율(매크로 재현율) 지표는 다음과 같습니다.

```
Macro Recall = (0.70 + 0.70 + 0.98 + 0.80 + 0.10)/5 = 0.656
```

F1 점수(매크로 F1 점수)

F1 점수는 `Precision` 및 `Recall` 값에서 파생됩니다. 이 속성은 분류기의 전반적인 정확도를 측정합니다. 최고 점수는 1이고, 최저 점수는 0입니다.

Amazon Comprehend는 매크로 F1 점수를 계산합니다. 이 점수는 레이블 F1 점수의 가중치가 적용되지 않은 평균입니다. 다음 테스트 세트를 예로 들어 보겠습니다.

레이블	샘플 크기	레이블 F1 점수
레이블_1	400	0.724
레이블_2	300	0.824
레이블_3	30000	0.94
레이블_4	20	0.62
레이블_5	10	0.16

모델의 F1 점수(매크로 F1 점수)는 다음과 같이 계산됩니다.

```
Macro F1 Score = (0.724 + 0.824 + 0.94 + 0.62 + 0.16)/5 = 0.6536
```

해밍 손실

레이블의 잘못 예측된 부분. 전체 레이블 수 대비 잘못된 레이블의 비율로도 볼 수 있습니다. 점수가 0에 가까울수록 좋습니다.

마이크로 정밀도

원본:

마이크로 정밀도는 모든 정밀도 점수를 합산한 전체 점수를 기반으로 한다는 점을 제외하면 정밀 측정법과 비슷합니다.

마이크로 재현율

마이크로 재현율은 모든 재현율 점수의 전체 점수를 합산하여 계산한다는 점을 제외하면 재현율 지표와 유사합니다.

마이크로 F1 점수

마이크로 F1 점수는 마이크로 정밀도 지표와 마이크로 재현율 지표의 조합입니다.

사용자 지정 분류기의 성능 개선

지표는 분류 작업 중에 사용자 지정 분류기가 어떻게 수행되는지에 대한 인사이트를 제공합니다. 지표가 낮으면 분류 모델이 사용 사례에 효과적이지 않을 수 있습니다. 분류기 성능을 개선할 수 있는 몇 가지 옵션이 있습니다.

1. 학습 데이터에서 범주의 명확한 구분을 정의하는 구체적인 예를 제공하십시오. 예를 들어, 고유한 단어/문장을 사용하여 범주를 나타내는 문서를 제공하십시오.
2. 학습 데이터에 제대로 표현되지 않은 레이블에 대한 데이터를 더 추가하십시오.
3. 카테고리의 편중을 줄이도록 하십시오. 데이터에서 가장 큰 레이블이 가장 작은 레이블에 있는 문서의 10배 이상인 경우 가장 작은 레이블의 문서 수를 늘려 보십시오. 대표성이 높은 클래스와 대표성이 가장 낮은 클래스 간의 왜곡 비율을 많으면 10:1까지 줄여야 합니다. 대표성이 높은 클래스에서 입력 문서를 제거해 볼 수도 있습니다.

실시간 분석 실행

사용자 지정 분류기를 학습시킨 후 실시간 분석을 사용하여 문서를 분류할 수 있습니다. 실시간 분석은 단일 문서를 입력으로 받아 결과를 동기적으로 반환합니다. 사용자 지정 분류는 다양한 문서 유형을 실시간 분석을 위한 입력으로 받아들입니다. 자세한 내용은 실시간 사용자 지정 분석을 위한 입력를 참조하십시오.

이미지 파일 또는 스캔한 PDF 문서를 분석하려는 경우, IAM 정책에서 두 가지 Amazon Textract API 메서드(DetectDocumentText와 AnalyzeDocument)를 사용할 수 있는 권한을 부여해야 합니다. Amazon Comprehend는 텍스트 추출 중에 이러한 메서드를 간접적으로 호출합니다. 정책 예제는 문서 분석 작업을 수행하는 데 필요한 권한을 참조하십시오.

사용자 지정 모델을 사용하여 실시간 분석을 실행할 엔드포인트를 생성해야 합니다.

주제

- 사용자 지정 분류를 위한 실시간 분석(콘솔)
- 사용자 지정 분류를 위한 실시간 분석(API)
- 실시간 분석을 위한 출력

사용자 지정 분류를 위한 실시간 분석(콘솔)

사용자 지정 분류 모델을 사용하여 실시간 분석을 실행하기 위해 Amazon Comprehend 콘솔을 사용할 수 있습니다.

엔드포인트를 생성하여 실시간 분석을 실행합니다. 엔드포인트에는 사용자 지정 모델을 실시간 추론에 사용할 수 있는 관리형 리소스가 포함되어 있습니다.

프로비저닝 엔드포인트 처리량 및 관련 비용에 대한 자세한 내용은 Amazon Comprehend 엔드포인트 사용을 참조하세요.

주제

- 사용자 지정 분류를 위한 엔드포인트 생성
- 실시간 사용자 지정 분류 실행

사용자 지정 분류를 위한 엔드포인트 생성

엔드포인트를 생성하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 엔드포인트를 선택하고 엔드포인트 생성 버튼을 선택합니다. 엔드포인트 생성 화면이 열립니다.
3. 엔드포인트에 이름을 지정합니다. 이 이름은 현재의 리전 및 계정 내에서 고유해야 합니다.
4. 새 엔드포인트를 연결할 사용자 지정 모델을 선택합니다. 드롭다운에서 모델 이름을 기준으로 검색할 수 있습니다.

   > **Note**
   >
   > 엔드포인트를 연결하려면 먼저 모델을 생성해야 합니다. 아직 모델이 없는 경우 학습 분류 모델을 참조하세요.

5. (선택) 엔드포인트에 태그를 추가하려면 태그 에 키-값 페어를 입력하고 태그 추가를 선택합니다. 엔드포인트를 생성하기 전에 이 페어를 제거하려면 태그 제거를 선택합니다
6. 엔드포인트에 할당할 추론 단위(IU) 수를 입력합니다. 각 단위는 초당 최대 2개 문서에 대한 초당 100자의 처리량을 나타냅니다. 엔드포인트 처리량에 대한 자세한 내용은 Amazon Comprehend 엔드포인트 사용을 참조하세요.
7. (선택) 새 엔드포인트를 생성하는 경우 IU 예측기를 사용할 수 있는 옵션이 있습니다. 처리량이나 초당 분석하려는 문자 수에 따라 필요한 추론 단위 수를 파악하기 어려울 수 있습니다. 이 선택적 단계는 요청할 IU 수를 결정하는 데 도움이 될 수 있습니다.
8. 구매 요약에서 시간당, 일별, 월별 예상 엔드포인트 비용을 검토하십시오.

9. 시작 시점부터 삭제할 때까지 계정에 엔드포인트에 대한 요금이 발생한다는 것을 이해하면 확인란을 선택합니다.
10. 엔드포인트 생성을 선택합니다.

실시간 사용자 지정 분류 실행

엔드포인트를 생성한 후에는 사용자 지정 모델을 사용하여 실시간 분석을 실행할 수 있습니다. 콘솔에서 실시간 분석을 실행하는 두 가지 방법이 있습니다. 다음과 같이 텍스트를 입력하거나 파일을 업로드할 수 있습니다.

사용자 지정 모델을 사용하여 실시간 분석을 실행하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 실시간 분석을 선택합니다.
3. 입력 유형에서 분석 유형으로 사용자 지정을 선택합니다.
4. 사용자 지정 모델 유형에서 사용자 지정 분류를 선택합니다.
5. 엔드포인트에서 사용하려는 엔드포인트를 선택합니다. 이 엔드포인트는 특정 사용자 지정 모델에 연결됩니다.
6. 분석을 위한 입력 데이터를 지정하려면 텍스트를 입력하거나 파일을 업로드할 수 있습니다.
   - 텍스트를 입력하려면:
     a. 입력 텍스트를 선택합니다.
     b. 분석할 텍스트를 입력합니다.
   - 파일을 업로드하려면:
     a. 파일 업로드를 선택하고 업로드할 파일 이름을 입력합니다.
     b. (선택) 고급 읽기 작업에서 텍스트 추출을 위한 기본 작업을 우선 지정할 수 있습니다. 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하십시오.

   최상의 결과를 얻으려면 입력 유형을 분류기 모델 유형과 일치시키십시오. 기본 문서를 일반 텍스트 모델에 제출하거나 일반 텍스트를 네이티브 문서 모델에 제출하는 경우 콘솔에 경고가 표시됩니다. 자세한 내용은 학습 분류 모델를 참조하십시오.
7. 분석을 선택합니다. Amazon Comprehend는 사용자 지정 모델을 사용하여 입력 데이터를 분석합니다. Amazon Comprehend는 검색된 클래스를 각 클래스에 대한 신뢰도 평가와 함께 표시합니다.

사용자 지정 분류를 위한 실시간 분석(API)

Amazon Comprehend API를 사용하여 사용자 지정 모델로 실시간 분류를 실행할 수 있습니다. 먼저, 실시간 분석을 실행할 엔드포인트를 생성합니다. 엔드포인트를 생성한 후 실시간 분류를 실행합니다.

이 섹션의 예제에서는 Unix, Linux 및 macOS용 명령 형식을 사용합니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

프로비저닝 엔드포인트 처리량 및 관련 비용에 대한 자세한 내용은 Amazon Comprehend 엔드포인트 사용을 참조하세요.

주제

사용자 지정 분류를 위한 엔드포인트 생성

다음 예제는 AWS CLI를 사용한 CreateEndpoint API 작업을 보여줍니다.

```
aws comprehend create-endpoint \
    --desired-inference-units number of inference units \
    --endpoint-name endpoint name \
    --model-arn arn:aws:comprehend:region:account-id:model/example \
    --tags Key=My1stTag,Value=Value1
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
   "EndpointArn": "Arn"
}
```

실시간 사용자 지정 분류 실행

사용자 지정 분류 모델의 엔드포인트를 만든 후에는 해당 엔드포인트를 사용하여 ClassifyDocument API 작업을 실행합니다. `text` 또는 `bytes` 파라미터를 사용하여 텍스트 입력을 제공할 수 있습니다. `bytes` 파라미터를 사용하여 다른 입력 유형을 입력합니다.

이미지 파일 및 PDF 파일의 경우 `DocumentReaderConfig` 파라미터를 사용하여 기본 텍스트 추출 작업을 재정의할 수 있습니다. 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하세요.

최상의 결과를 얻으려면 입력 유형을 분류기 모델 유형과 일치시키십시오. API 응답은 네이티브 문서를 일반 텍스트 모델에 제출하거나 일반 텍스트 파일을 네이티브 문서 모델에 제출하는 경우의 경고를 포함합니다. 자세한 내용은 학습 분류 모델 단원을 참조하십시오.

사용 AWS Command Line Interface

다음 예제에서는 ClassifyDocument CLI 명령을 사용하는 방법을 보여 줍니다.

를 사용하여 텍스트 분류 AWS CLI

다음 예제에서는 텍스트 블록에 대해 실시간 분류를 실행합니다.

```
aws comprehend classify-document \
     --endpoint-arn arn:aws:comprehend:region:account-id:endpoint/endpoint name \
     --text 'From the Tuesday, April 16th, 1912 edition of The Guardian newspaper: The
 maiden voyage of the White Star liner Titanic,
     the largest ship ever launched ended in disaster. The Titanic started her trip
 from Southampton for New York on Wednesday. Late
     on Sunday night she struck an iceberg off the Grand Banks of Newfoundland. By
 wireless telegraphy she sent out signals of distress,
     and several liners were near enough to catch and respond to the call.'
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "Classes": [
       {
          "Name": "string",
          "Score": 0.9793661236763
       }
    ]
 }
```

를 사용하여 반정형 문서 분류 AWS CLI

PDF, Word 또는 이미지 파일의 사용자 지정 분류를 분석하려면 `bytes` 파라미터의 입력 파일을 사용하여 `classify-document` 명령을 실행합니다.

다음 예제에서는 이미지를 입력 파일로 사용합니다. `fileb` 옵션을 사용하여 이미지 파일 바이트를 base-64로 인코딩합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 이진 대용량 객체를 참조하세요.

이 예제에서는 텍스트 추출 옵션을 설정하기 위해 config.json이라는 이름의 JSON 파일도 전달합니다.

```
$ aws comprehend classify-document \
> --endpoint-arn arn \
> --language-code en \
> --bytes fileb://image1.jpg   \
> --document-reader-config file://config.json
```

config.json 파일에는 다음 내용이 포함되어 있습니다.

```
{
   "DocumentReadMode": "FORCE_DOCUMENT_READ_ACTION",
   "DocumentReadAction": "TEXTRACT_DETECT_DOCUMENT_TEXT"
}
```

Amazon Comprehend가 다음과 같이 응답합니다.

```
{
    "Classes": [
       {
          "Name": "string",
          "Score": 0.9793661236763
       }
    ]
}
```

자세한 내용은 Amazon Comprehend API 참조 내 분류 문서를 참조하세요.

실시간 분석을 위한 출력

텍스트 입력을 위한 출력

텍스트 입력을 위한 출력에는 분류기 분석으로 식별된 클래스 또는 레이블 목록이 포함됩니다. 다음 예제는 2개의 클래스로 이루어진 목록을 보여줍니다.

```
"Classes": [
  {
```

```
    "Name": "abc",
    "Score": 0.2757999897003174,
    "Page": 1
  },
  {
    "Name": "xyz",
    "Score": 0.2721000015735626,
    "Page": 1
  }
]
```

반구조화된 입력을 위한 출력

반정형 입력 문서 또는 텍스트 파일의 경우 출력에 다음과 같은 추가 필드가 포함될 수 있습니다.

- 문서 메타데이터(DocumentMetadata) — 문서에 대한 추출 정보입니다. 메타데이터에는 문서의 페이지 목록과 각 페이지에서 추출한 문자 수가 포함됩니다. 요청에 `Byte` 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 문서 유형(DocumentType) — 입력 문서에 있는 각 페이지의 문서 유형입니다. 요청에 `Byte` 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 오류(Errors) — 입력 문서를 처리하는 동안 시스템에서 감지한 페이지 수준 오류입니다. 시스템에서 오류가 발생하지 않으면 이 필드는 비어 있습니다.
- 경고(Warnings) — 입력 문서를 처리하는 동안 경고가 감지되었습니다. 응답에는 입력 문서 유형과 지정한 엔드포인트와 관련된 모델 유형이 일치하지 않을 경우 경고가 포함됩니다. 시스템에서 경고가 생성되지 않으면 이 필드는 비어 있습니다.

이러한 출력 필드에 대한 자세한 내용은 Amazon Comprehend API 참조의 문서 분류를 참조하세요.

다음 예제에서는 한 페이지 분량의 네이티브 PDF 입력 문서의 출력을 보여줍니다.

```
{
  "Classes": [
      {
          "Name": "123",
          "Score": 0.39570000767707825,
          "Page": 1
      },
      {
          "Name": "abc",
```

```
            "Score": 0.2757999897003174,
            "Page": 1
        },
        {
            "Name": "xyz",
            "Score": 0.2721000015735626,
            "Page": 1
        }
    ],
    "DocumentMetadata": {
        "Pages": 1,
        "ExtractedCharacters": [
            {
                "Page": 1,
                "Count": 2013
            }
        ]
    },
    "DocumentType": [
        {
            "Page": 1,
            "Type": "NATIVE_PDF"
        }
    ]
}
```


비동기 작업 실행

사용자 지정 분류기를 학습시킨 후 비동기 작업을 사용하여 대규모 문서 또는 한 배치 내 여러 문서를 일괄 분석할 수 있습니다.

사용자 지정 분류에는 다양한 입력 문서 유형이 허용됩니다. 자세한 내용은 비동기 사용자 지정 분석을 위한 입력을 참조하십시오.

이미지 파일 또는 스캔한 PDF 문서를 분석하려는 경우, IAM 정책에서 두 가지 Amazon Textract API 메서드(DetectDocumentText와 AnalyzeDocument)를 사용할 수 있는 권한을 부여해야 합니다. Amazon Comprehend는 텍스트 추출 중에 이러한 메서드를 간접적으로 호출합니다. 정책 예제는 문서 분석 작업을 수행하는 데 필요한 권한을 참조하십시오.

일반 텍스트 모델을 사용하여 반정형 문서(이미지, PDF 또는 Docx 파일)를 분류하려면 one `document per file` 입력 형식을 사용합니다. 또한 StartDocumentClassificationJob 요청에 `DocumentReaderConfig` 파라미터를 포함시킵니다.

주제

비동기 분석을 위한 파일 형식

사용자의 모델을 사용하여 비동기 분석을 실행할 때 입력 문서 형식(`One document per line` 또는 `one document per file`)을 선택할 수 있습니다. 사용하는 형식은 다음 표의 설명대로 분석하려는 문서의 유형에 따라 달라집니다.

설명	형식
입력에는 여러 파일이 들어 있습니다. 각 파일에는 입력 문서가 한 개씩 들어 있습니다. 이 형식은 신문 기사나 과학 논문과 같은 대용량 문서 모음에 가장 적합합니다. 또한 네이티브 문서 분류기를 사용하는 반정형 문서(이미지, PDF 또는 Docx 파일)에도 이 형식을 사용하십시오.	파일당 문서 하나
입력은 하나 이상의 파일입니다. 파일의 각 라인은 별도의 입력 문서입니다. 이 형식은 문자 메시지나 소셜 미디어 게시물과 같은 짧은 문서에 가장 적합합니다.	라인당 문서 하나

파일당 문서 하나

`one document per file` 형식을 사용할 경우 각 파일은 하나의 입력 문서를 나타냅니다.

라인당 문서 하나

`One document per line` 형식을 사용하면 각 문서가 별도의 줄에 배치되며 제목은 사용되지 않습니다. 문서의 레이블을 아직 모르기 때문에 레이블은 각 줄에 포함되지 않습니다. 파일의 각 줄(개별 문

서의 끝)은 줄 바꿈(LF,\n), 캐리지 반환(CR,\r) 또는 두 가지 전부(CRLF, \r\n)로 끝나야 합니다. UTF-8 줄 구분자(u+2028)를 사용하여 줄을 끝내지 마십시오.

다음 예제는 입력 파일의 형식을 보여줍니다.

```
Text of document 1 \n
Text of document 2 \n
Text of document 3 \n
Text of document 4 \n
```

어떤 형식이든 텍스트 파일에는 UTF-8 인코딩을 사용하십시오. 이 파일들을 준비한 후 입력 데이터를 위해 사용하는 S3 버킷에 이 파일들을 저장합니다.

분류 작업을 시작할 때 입력 데이터에 이 Amazon S3 위치를 지정합니다. URI는 직접 호출하는 API 엔드포인트와 동일한 리전에 있어야 합니다. URI는 단일 파일을 가리킬 수도 있고(“라인당 문서 하나” 방법을 사용할 때처럼) 데이터 파일 모음의 접두사일 수도 있습니다.

예를 들어 URI `S3://bucketName/prefix`을(를) 사용하는 경우 접두사가 단일 파일이면 Amazon Comprehend는 해당 파일을 입력으로 사용합니다. 접두사로 시작하는 파일이 두 개 이상인 경우 Amazon Comprehend는 이들 모두를 입력으로 사용합니다.

Amazon Comprehend에 문서 모음 및 출력 파일들이 포함된 S3 버킷에 대한 액세스 권한을 부여하십시오. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.

사용자 지정 분류를 위한 분석 작업(콘솔)

사용자 지정 문서 분류기를 만들고 학습시킨 후 콘솔을 사용하여 이 모델로 사용자 지정 분류 작업을 실행할 수 있습니다.

사용자 지정 분류 작업을 생성하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 분석 작업을 선택한 다음 작업 생성을 선택합니다.
3. 분류 작업에 이름을 지정합니다. 이름은 사용자 계정과 현재의 리전에 고유해야 합니다.
4. 분석 유형에서 사용자 지정 분류를 선택합니다.
5. 분류기 선택에서 사용할 사용자 지정 분류기를 선택합니다.

6. (선택) Amazon Comprehend가 작업을 처리하는 동안 사용하는 데이터를 암호화하도록 선택한 경우, 작업 암호화를 선택하십시오. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.

   - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID의 키 ID를 선택합니다.
   - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ARN에 키 ID의 ARN을 입력합니다.

   > **Note**
   >
   > KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 (키 관리 서비스(KMS)를 참조하세요.

7. 입력 데이터에서 입력 문서가 포함된 Amazon S3 버킷의 위치를 입력하거나 S3 찾아보기를 선택하여 해당 버킷으로 이동합니다. 이 버킷은 호출하는 서비스와 동일한 리전에 있어야 합니다. 분류 작업에 대한 액세스 권한에 사용하는 IAM 역할에는 S3 버킷에 대한 읽기 권한이 있어야 합니다.

   모델 학습에서 최고 수준의 정확도를 달성하려면 입력 유형을 분류기 모델 유형과 일치시키십시오. 기본 문서를 일반 텍스트 모델에 제출하거나 일반 텍스트 문서를 네이티브 문서 모델에 제출하는 경우 분류기 작업에서 경고를 반환합니다. 자세한 내용은 학습 분류 모델를 참조하십시오.

8. (선택) 입력 형식으로 입력 문서의 형식을 선택할 수 있습니다. 형식은 파일당 문서 하나, 또는 단일 파일의 라인당 문서 하나일 수 있습니다. 라인당 문서 하나는 텍스트 문서에만 적용됩니다.

9. (선택) 문서 읽기 모드에서 기본 텍스트 추출 작업을 우선 지정할 수 있습니다. 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하십시오.

10. 출력 데이터에서 Amazon Comprehend가 작업의 출력 데이터를 기록해야 하는 Amazon S3 버킷의 위치를 입력하거나 S3 찾아보기를 선택하여 해당 위치로 이동합니다. 이 버킷은 호출하는 서비스와 동일한 리전에 있어야 합니다. 분류 작업에 대한 액세스 권한에 귀하가 사용하는 IAM 역할에는 S3 버킷에 대한 쓰기 권한이 있어야 합니다.

11. (선택) 작업의 출력 결과를 암호화하기로 선택한 경우 암호화를 선택합니다. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.

    - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID에서 키 별칭 또는 ID를 선택합니다.
    - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ID 아래에 키 별칭 또는 ID의 ARN을 입력합니다.

12. (선택) VPC에서 Amazon Comprehend로 리소스를 시작하려면 VPC 아래에 VPC ID를 입력하거나 드롭다운 목록에서 ID를 선택합니다.

1. 서브넷에서 서브넷을 선택합니다. 첫 번째 서브넷을 선택한 후 추가 서브넷을 선택할 수 있습니다.
2. 보안 그룹을 지정한 경우, 보안 그룹에서 사용할 보안 그룹을 선택합니다. 첫 번째 보안 그룹을 선택한 후 추가 보안 그룹을 선택할 수 있습니다.

> **Note**
>
> 분류 작업에 VPC를 사용하는 경우 생성 및 시작 작업에 사용되는 `DataAccessRole`은 출력 버킷에 액세스하는 VPC에 권한을 부여해야 합니다.

13. 작업 생성을 선택하여 문서 분류 작업을 생성합니다.

사용자 지정 분류를 위한 분석 작업(API)

사용자 지정 문서 분류기를 생성하고 학습시킨 후 분류기를 사용하여 분석 작업을 실행할 수 있습니다.

StartDocumentClassificationJob 작업을 사용하면 레이블이 지정되지 않은 문서의 분류를 시작할 수 있습니다. 입력 문서가 포함된 S3 버킷, 출력 문서용 S3 버킷, 사용할 분류기를 지정합니다.

모델 학습에서 최고 수준의 정확도를 달성하려면 입력 유형을 분류기 모델 유형과 일치시키십시오. 기본 문서를 일반 텍스트 모델에 제출하거나 일반 텍스트 문서를 네이티브 문서 모델에 제출하는 경우 분류기 작업에서 경고를 반환합니다. 자세한 내용은 학습 분류 모델를 참조하십시오.

StartDocumentClassificationJob은 비동기식입니다. 작업을 시작한 후에는 DescribeDocumentClassificationJob 작업을 사용하여 작업 진행 상황을 모니터링할 수 있습니다. 응답의 `Status` 필드가 `COMPLETED`으로 표시되면 지정한 위치에서 출력에 액세스할 수 있습니다.

주제

- 사용 AWS Command Line Interface
- AWS SDK for Java 또는 SDK for Python 사용

사용 AWS Command Line Interface

다음 예제는 `StartDocumentClassificationJob` 작업 및 AWS CLI를 사용한 기타 사용자 지정 분류기 API입니다.

다음 예제들은 Unix, Linux, macOS용 명령 형식을 사용합니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

StartDocumentClassificationJob 작업을 사용하여 사용자 지정 분류 작업을 실행합니다.

```
aws comprehend start-document-classification-job \
     --region region \
     --document-classifier-arn arn:aws:comprehend:region:account number:document-
classifier/testDelete \
     --input-data-config S3Uri=s3://S3Bucket/docclass/file
 name,InputFormat=ONE_DOC_PER_LINE \
     --output-data-config S3Uri=s3://S3Bucket/output \
     --data-access-role-arn arn:aws:iam::account number:role/resource name
```

DescribeDocumentClassificationJob 작업을 사용하여 작업 ID로 사용자 지정 분류기에 대한 정보를 가져옵니다.

```
aws comprehend describe-document-classification-job \
     --region region \
     --job-id job id
```

ListDocumentClassificationJobs 작업을 사용하여 계정의 모든 사용자 지정 분류 작업을 나열합니다.

```
aws comprehend list-document-classification-jobs
     --region region
```

AWS SDK for Java 또는 SDK for Python 사용

사용자 지정 분류기 작업을 시작하는 방법에 대한 SDK 예제는 AWS SDK 또는 CLI와 StartDocumentClassificationJob 함께 사용 를 참조하세요.

비동기 분석 작업을 위한 출력

분석 작업이 완료되면 요청에서 지정한 S3 버킷에 결과가 저장됩니다.

텍스트 입력을 위한 출력

텍스트 입력 문서 형식(멀티클래스 또는 멀티레이블)의 경우 작업 출력은 output.tar.gz로 이름이 지정된 단일 파일로 구성됩니다. 이 파일은 텍스트 파일과 출력이 들어 있는 압축된 아카이브 파일입니다.

멀티클래스 출력

멀티클래스 모드에서 학습된 분류기를 사용하면 그 결과에 `classes`가 표시됩니다. 이러한 `classes`는 각각 분류기를 학습시킬 때 범주 세트를 만드는 데 사용되는 클래스입니다.

이러한 출력 필드에 대한 자세한 내용은 Amazon Comprehend API 참조의 문서 분류를 참조하세요.

다음 예제에서는 다음과 같은 함께 상호 배타적인 클래스들을 사용합니다.

```
DOCUMENTARY
SCIENCE_FICTION
ROMANTIC_COMEDY
SERIOUS_DRAMA
OTHER
```

입력 데이터 형식이 라인당 문서 하나인 경우, 출력 파일에는 입력의 각 라인에 라인 하나가 포함됩니다. 각 라인에는 파일 이름, 입력 라인의 0을 기반으로 하는 라인 번호, 문서 내에서 발견되는 단일 혹은 여러 클래스가 포함됩니다. 이는 개별 인스턴스가 올바르게 분류되었다는 Amazon Comprehend의 확신으로 종결됩니다.

예시:

```
{"File": "file1.txt", "Line": "0", "Classes": [{"Name": "Documentary", "Score":
 0.8642}, {"Name": "Other", "Score": 0.0381}, {"Name": "Serious_Drama", "Score":
 0.0372}]}
{"File": "file1.txt", "Line": "1", "Classes": [{"Name": "Science_Fiction", "Score":
 0.5}, {"Name": "Science_Fiction", "Score": 0.0381}, {"Name": "Science_Fiction",
 "Score": 0.0372}]}
{"File": "file2.txt", "Line": "2", "Classes": [{"Name": "Documentary", "Score": 0.1},
 {"Name": "Documentary", "Score": 0.0381}, {"Name": "Documentary", "Score": 0.0372}]}
{"File": "file2.txt", "Line": "3", "Classes": [{"Name": "Serious_Drama", "Score":
 0.3141}, {"Name": "Other", "Score": 0.0381}, {"Name": "Other", "Score": 0.0372}]}
```

입력 데이터 형식이 파일당 문서 하나인 경우 출력 파일에는 문서당 라인 하나가 포함됩니다. 각 라인에는 파일 이름과 문서에 있는 하나 이상의 클래스가 있습니다. 이는 Amazon Comprehend가 개별 인스턴스를 정확하게 분류했다는 확신으로 종결됩니다.

예시:

```
{"File": "file0.txt", "Classes": [{"Name": "Documentary", "Score": 0.8642}, {"Name":
 "Other", "Score": 0.0381}, {"Name": "Serious_Drama", "Score": 0.0372}]}
```

```
{"File": "file1.txt", "Classes": [{"Name": "Science_Fiction", "Score": 0.5}, {"Name":
 "Science_Fiction", "Score": 0.0381}, {"Name": "Science_Fiction", "Score": 0.0372}]}
{"File": "file2.txt", "Classes": [{"Name": "Documentary", "Score": 0.1}, {"Name":
 "Documentary", "Score": 0.0381}, {"Name": "Domentary", "Score": 0.0372}]}
{"File": "file3.txt", "Classes": [{"Name": "Serious_Drama", "Score": 0.3141}, {"Name":
 "Other", "Score": 0.0381}, {"Name": "Other", "Score": 0.0372}]}
```

멀티레이블 출력

멀티레이블 모드에서 학습된 분류기를 사용하면 그 결과에 `labels`가 표시됩니다. 이러한 `labels`는 각각 분류기를 학습시킬 때 범주 세트를 만드는 데 사용되는 레이블입니다.

다음 예제에서는 이러한 고유한 레이블을 사용합니다.

```
SCIENCE_FICTION
ACTION
DRAMA
COMEDY
ROMANCE
```

입력 데이터 형식이 라인당 문서 하나인 경우, 출력 파일에는 입력의 각 라인에 라인 하나가 포함됩니다. 각 라인에는 파일 이름, 입력 라인의 0을 기반으로 하는 라인 번호, 문서 내에서 발견되는 단일 혹은 여러 클래스가 포함됩니다. 이는 개별 인스턴스가 올바르게 분류되었다는 Amazon Comprehend의 확신으로 종결됩니다.

예시:

```
{"File": "file1.txt", "Line": "0", "Labels": [{"Name": "Action", "Score": 0.8642},
 {"Name": "Drama", "Score": 0.650}, {"Name": "Science Fiction", "Score": 0.0372}]}
{"File": "file1.txt", "Line": "1", "Labels": [{"Name": "Comedy", "Score": 0.5},
 {"Name": "Action", "Score": 0.0381}, {"Name": "Drama", "Score": 0.0372}]}
{"File": "file1.txt", "Line": "2", "Labels": [{"Name": "Action", "Score": 0.9934},
 {"Name": "Drama", "Score": 0.0381}, {"Name": "Action", "Score": 0.0372}]}
{"File": "file1.txt", "Line": "3", "Labels": [{"Name": "Romance", "Score": 0.9845},
 {"Name": "Comedy", "Score": 0.8756}, {"Name": "Drama", "Score": 0.7723}, {"Name":
 "Science_Fiction", "Score": 0.6157}]}
```

입력 데이터 형식이 파일당 문서 하나인 경우 출력 파일에는 문서당 라인 하나가 포함됩니다. 각 라인에는 파일 이름과 문서에 있는 하나 이상의 클래스가 있습니다. 이는 Amazon Comprehend가 개별 인스턴스를 정확하게 분류했다는 확신으로 종결됩니다.

예시:

```
{"File": "file0.txt", "Labels": [{"Name": "Action", "Score": 0.8642}, {"Name": "Drama",
 "Score": 0.650}, {"Name": "Science Fiction", "Score": 0.0372}]}
{"File": "file1.txt", "Labels": [{"Name": "Comedy", "Score": 0.5}, {"Name": "Action",
 "Score": 0.0381}, {"Name": "Drama", "Score": 0.0372}]}
{"File": "file2.txt", "Labels": [{"Name": "Action", "Score": 0.9934}, {"Name": "Drama",
 "Score": 0.0381}, {"Name": "Action", "Score": 0.0372}]}
{"File": "file3.txt", "Labels": [{"Name": "Romance", "Score": 0.9845}, {"Name":
 "Comedy", "Score": 0.8756}, {"Name": "Drama", "Score": 0.7723}, {"Name":
 "Science_Fiction", "Score": 0.6157}]}
```

반구조화된 입력 문서를 위한 출력

반구조화된 입력 문서를 위한 출력에 다음과 같은 추가 필드가 포함될 수 있습니다.

- 문서 메타데이터(DocumentMetadata) — 문서에 대한 추출 정보입니다. 메타데이터에는 문서의 페이지 목록과 각 페이지에서 추출한 문자 수가 포함됩니다. 요청에 Byte 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 문서 유형(DocumentType) — 입력 문서에 있는 각 페이지의 문서 유형입니다. 요청에 Byte 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 오류(Errors) — 입력 문서를 처리하는 동안 시스템에서 감지한 페이지 수준 오류입니다. 시스템에서 오류가 발생하지 않으면 이 필드는 비어 있습니다.

이러한 출력 필드에 대한 자세한 내용은 Amazon Comprehend API 참조의 문서 분류를 참조하세요.

다음 예제는 두 페이지로 구성된 스캔한 PDF 파일의 출력을 보여줍니다.

```
[{ #First page output
    "Classes": [
        {
            "Name": "__label__2 ",
            "Score": 0.9993996620178223
        },
        {
            "Name": "__label__3 ",
            "Score": 0.0004330444789957255
        }
    ],
    "DocumentMetadata": {
        "PageNumber": 1,
```

```
        "Pages": 2
    },
    "DocumentType": "ScannedPDF",
    "File": "file.pdf",
    "Version": "VERSION_NUMBER"
},
#Second page output
{
    "Classes": [
        {
            "Name": "__label__2 ",
            "Score": 0.9993996620178223
        },
        {
            "Name": "__label__3 ",
            "Score": 0.0004330444789957255
        }
    ],
    "DocumentMetadata": {
        "PageNumber": 2,
        "Pages": 2
    },
    "DocumentType": "ScannedPDF",
    "File": "file.pdf",
    "Version": "VERSION_NUMBER"
}]
```

사용자 지정 개체 인식

사용자 지정 개체 인식은 사전 설정된 일반 개체 유형에 없는 특정 새 개체 유형을 식별할 수 있도록 지원하여 Amazon Comprehend의 기능을 확장합니다. 즉, 문서를 분석하여 제품 코드 또는 비즈니스별 개체와 같이 필요에 부합하는 개체를 추출할 수 있습니다.

정확한 사용자 지정 개체 인식기를 직접 구축하는 것은 복잡한 프로세스일 수 있습니다. 수동으로 주석을 달아놓은 대규모 학습 문서를 준비하고, 모델 학습에 적합한 알고리즘과 매개변수를 선택해야 하기 때문입니다. Amazon Comprehend는 자동 주석 및 사용자 지정 개체 인식 모델을 생성할 수 있는 모델 개발을 제공하여 이 복잡성을 줄이는 데 도움이 됩니다.

사용자 지정 개체 인식 모델을 만드는 것은 문자열 매칭이나 정규 표현식을 사용하여 문서에서 개체를 추출하는 것보다 더 효과적인 접근 방식입니다. 예를 들어, 문서에서 ENGINEER 이름을 추출할 때는 가능한 모든 이름을 열거하기가 어렵습니다. 또한 맥락이 없으면 ENGINEER 이름과 ANALYST 이름을 구분하기가 어렵습니다. 사용자 지정 개체 인식 모델은 해당 이름이 나타날 가능성이 있는 맥락을 학습할 수 있습니다. 또한 문자열 매칭은 오타가 있거나 새로운 이름 지정 규칙을 따르는 개체를 감지하지 못하지만, 사용자 지정 모델을 사용하면 가능합니다.

사용자 지정 모델을 생성할 수 있는 옵션은 두 가지가 있습니다.

1. 주석 – 모델 학습에 사용할 주석이 달린 개체가 포함된 데이터 세트를 제공합니다.
2. 개체 목록(일반 텍스트만 해당) – 모델 학습에 사용할 개체 목록 및 해당 유형 레이블(예: `PRODUCT_CODES`), 해당 개체가 포함된 주석이 없는 문서 세트를 제공합니다.

주석이 달린 PDF 파일을 사용하여 사용자 지정 개체 인식기를 만들면 사전 처리나 문서 병합 작업 없이 일반 텍스트, 이미지 파일(JPG, PNG, TIFF), PDF 파일, Word 문서 등 다양한 입력 파일 형식에 해당 인식기를 사용할 수 있습니다. Amazon Comprehend는 이미지 파일 또는 Word 문서에 대한 주석 달기를 지원하지 않습니다.

> **Note**
>
> 주석이 달린 PDF 파일을 사용하는 사용자 지정 개체 인식기는 영어 문서만 지원합니다.

한 번에 최대 25개의 사용자 지정 개체에 대해 모델을 학습시킬 수 있습니다. 자세한 정보는 지침 및 할당량 페이지를 참조하세요.

모델을 학습시킨 후에는 이 모델을 사용하여 실시간 개체 감지 및 개체 감지 작업을 수행할 수 있습니다.

주제

개체 인식기 학습 데이터 준비

성공적인 사용자 정의 개체 인식 모델을 학습하려면 모델 트레이너에게 고품질 입력 데이를 제공하는 것이 중요합니다. 좋은 데이터가 없으면 모델은 개체를 올바르게 식별하는 방법을 학습할 수 없습니다.

사용자 정의 개체 인식 모델 학습은 Amazon Comprehend에 데이터를 제공하는 두 가지 방법 중 하나를 선택할 수 있습니다.

- 개체 목록 — Amazon Comprehend가 사용자 정의 개체를 식별하도록 학습할 수 있도록 특정 개체를 나열합니다. 참고: 개체 목록은 일반 텍스트 문서에만 사용할 수 있습니다.
- 주석 — Amazon Comprehend가 개체와 해당 문맥 모두에 대해 학습할 수 있도록 여러 문서에서 개체의 위치를 제공합니다. 이미지 파일, PDF 또는 Word 문서를 분석하기 위한 모델을 만들려면 PDF 주석을 사용하여 인식기를 학습시켜야 합니다.

Amazon Comprehend는 두 경우 모두에 문서의 종류와 개체가 만들어 내는 문맥을 학습하고 문서를 분석할 때 새 개체 탐지를 일반화할 수 있는 인식기를 구축합니다.

사용자 정의 모델을 만들거나 새 버전을 학습시킬 때 테스트 데이터 세트를 제공할 수 있습니다. 테스트 데이터를 제공하지 않는 경우 Amazon Comprehend는 모델을 테스트하기 위해 입력 문서의 10%를 유보합니다. Amazon Comprehend는 나머지 문서를 사용하여 모델을 학습시킵니다.

주석 학습 세트를 위한 테스트 데이터 세트를 제공하는 경우, 테스트 데이터에는 생성 요청에서 지정한 각 항목 유형에 대한 주석이 하나 이상 포함되어야 합니다.

주제

- Annotations

주석을 사용해야 하는 경우와 개체 목록을 사용해야 하는 경우

주석을 만드는 것은 개체 목록을 만드는 것보다 더 많은 작업이 필요하지만 결과 모델은 훨씬 더 정확할 수 있습니다. 개체 목록을 사용하면 더 빠르고 작업 집약도가 줄어들지만 결과의 세분화와 정확도는 떨어집니다. 이는 주석이 Amazon Comprehend가 모델을 학습시킬 때 사용할 수 있는 더 많은 문맥을 제공하기 때문입니다. 이러한 문맥이 없으면 Amazon Comprehend에서 개체를 식별할 때 탐지 오류가 더 많이 발생합니다.

주석 사용에 따른 비용 증가와 워크로드를 피하는 것이 비즈니스적으로 더 합리적인 시나리오가 될 수도 있습니다. 예를 들어 John Johnson이라는 이름은 사용자의 검색에서 중요한 의미를 갖지만 정확히 사용자가 검색하는 그 사람인지 여부는 관련이 없습니다. 또는 개체 목록을 사용할 때 지표가 충분하여 사용자가 원하는 인식기 결과를 제공할 수도 있습니다. 이러한 경우에는 오히려 개체 목록을 사용하는 것이 더 효과적일 수 있습니다.

다음과 같은 경우에는 주석 모드를 사용합니다.

- 이미지 파일, PDF 또는 Word 문서에 대해 추론을 실행하려는 경우 이 시나리오에서는 주석이 달린 PDF 파일을 사용하여 모델을 학습 시키고 이 모델을 사용하여 이미지 파일, PDF 및 Word 문서에 대한 추론 작업을 실행합니다.
- 개체의 의미가 모호하고 상황에 따라 달라질 수 있는 경우 예를 들어 Amazon이라는 용어는 브라질의 강이나 온라인 소매업체인 Amazon.com을 가리킬 수 있습니다. Amazon 같은 사업체를 식별하는 사용자 정의 개체 인식기를 만들 때는 개체 목록 대신 주석을 사용해야 합니다. 이 방법을 사용하면 문맥을 사용하여 개체를 더 잘 찾을 수 있기 때문입니다.
- 주석을 얻기 위한 프로세스를 설정하는 것이 편한 경우 (약간의 노력이 필요할 수 있음).

다음과 같은 경우에는 개체 목록을 사용하는 것이 좋습니다.

- 이미 개체 목록이 있거나 개체의 전체 목록을 작성하는 것이 비교적 쉬운 경우. 개체 목록을 사용하는 경우, 목록이 완벽하거나 최소한 학습용으로 제공하는 문서에 나타날 수 있는 대부분의 유효한 개체가 포함되어 있어야 합니다.
- 주석을 작성하는 것보다 비용이 적게 들기 때문에 처음 사용하는 경우에는 일반적으로 개체 목록을 사용하는 것이 좋습니다. 하지만 학습된 모델은 주석 사용 보다 정확하지 않을 수 있다는 점에 유의해야 합니다.

개체 목록(일반 텍스트만 해당)

개체 목록을 사용하여 모델을 학습하기 위해 두 가지 정보를 제공합니다. 하나는 해당 사용자 지정 개체 유형이 포함된 개체 이름 목록이고 다른 하나는 개체가 표시될 것으로 예상되는 주석이 없는 문서 모음입니다.

개체 목록을 제공하면 Amazon Comprehend는 지능형 알고리즘을 사용하여 문서 내 개체 발생을 감지하여 사용자 지정 개체 인식기 모델을 교육하기 위한 기초로 사용합니다.

개체 목록의 경우 개체 목록에서 개체 유형당 최소 25개의 개체 일치 항목을 제공하십시오.

사용자 지정 개체 인식을 위한 개체 목록에는 다음 열이 있는 쉼표 구분 값(CSV) 파일이 필요합니다.

- 텍스트 — 항목 예제의 텍스트는 함께 제공되는 문서 코퍼스에 표시된 것과 동일합니다.
- 유형 —고객이 정의한 개체 유형입니다. 개체 유형은 대문자와 밑줄로 구분된 문자열이어야 합니다(예: MANAGER 또는 SENIOR_MANAGER). 모델당 최대 25개의 개체 유형을 학습시킬 수 있습니다.

이 `documents.txt` 파일은 네 줄로 구성되어 있습니다.

```
Jo Brown is an engineer in the high tech industry.
John Doe has been a engineer for 14 years.
Emilio Johnson is a judge on the Washington Supreme Court.
Our latest new employee, Jane Smith, has been a manager in the industry for 4 years.
```

개체 목록이 있는 CSV 파일에는 다음과 같은 줄이 있습니다.

```
Text, Type
Jo Brown, ENGINEER
John Doe, ENGINEER
Jane Smith, MANAGER
```

> **Note**
>
> 개체 목록에서 Emilio Johnson의 항목은 ENGINEER 또는 MANAGER 개체를 포함하지 않기 때문에 존재하지 않습니다.

데이터 파일 생성

개체 목록 파일에 문제가 발생할 가능성을 최소화하려면 개체 목록을 적절하게 구성된 CSV 파일에 포함해야 합니다. CSV 파일을 수동으로 구성하려면 다음 조건이 충족되어야 합니다.

- UTF-8 인코딩은 대부분의 경우 기본값으로 사용되더라도 명시적으로 지정해야 합니다.
- 열 이름: Type 및 Text를 포함해야 합니다.

잠재적 문제를 방지하려면 프로그래밍 방식으로 CSV 입력 파일을 생성하는 것이 좋습니다.

다음 예제는 Python을 사용하여 위에 표시된 주석에 대한 CSV를 생성합니다.

```
import csv
with open("./entitylist/entitylist.csv", "w", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow(["Text", "Type"])
    csv_writer.writerow(["Jo Brown", " ENGINEER"])
    csv_writer.writerow(["John Doe", " ENGINEER"])
    csv_writer.writerow(["Jane Smith", " MANAGER"])
```

모범 사례

개체 목록을 사용할 때 최상의 결과를 얻으려면 다음을 포함하여 여러 가지 사항을 고려해야 합니다.

- 목록에 있는 항목의 순서는 모델 학습에 영향을 주지 않습니다.
- 주석이 없는 문서 코퍼스에 언급된 긍정적인 개체 예제의 80~100%를 포함하는 개체 목록 항목을 사용하십시오.
- 일반적인 단어와 구문을 제거하여 문서 코퍼스의 비개체와 일치하는 개체 예제를 사용하지 마십시오. 일치하지 않는 부분이 조금이라도 있어도 결과 모델의 정확성에 큰 영향을 미칠 수 있습니다. 예를 들어, 개체 목록에 있는 것과 같은 단어를 사용하면 찾고 있는 개체와 일치하지 않을 가능성이 높아 정확도에 큰 영향을 미칠 수 있습니다.
- 입력 데이터에 중복이 포함되어서는 안 됩니다. 샘플이 중복되면 테스트 세트가 오염되어 학습 프로세스, 모델 지표 및 행동에 부정적인 영향을 미칠 수 있습니다.
- 실제 사용 사례와 최대한 유사한 문서를 제공하십시오. 토이 데이터나 합성 데이터를 프로덕션 시스템에 사용하지 마십시오. 입력 데이터는 과적합을 방지하고 기본 모델이 실제 예제를 보다 잘 일반화할 수 있도록 최대한 다양해야 합니다.
- 개체 목록은 대소문자를 구분하며 정규 표현식은 현재 지원되지 않습니다. 그러나 학습된 모델은 개체 목록에 제공된 대/소문자와 정확히 일치하지 않더라도 개체를 인식할 수 있는 경우가 많습니다.

- 다른 개체의 하위 문자열(예: “Smith” 및 “Jane Smith”)인 개체가 있는 경우 개체 목록에 둘 다 입력하십시오.

추가 제안 사항은 사용자 지정 개체 인식기 성능 개선에서 확인할 수 있습니다.

Annotations

주석은 사용자 지정 개체 유형을 학습 문서에서 해당 개체가 나타나는 위치와 연결하여 상황에 맞게 개체에 레이블을 지정합니다.

문서와 함께 주석을 제출하면 모델의 정확도를 높일 수 있습니다. 주석을 사용하면 단순히 찾고 있는 개체의 위치를 제공하는 데 그치지 않고 원하는 사용자 지정 개체에 더 정확한 컨텍스트를 제공할 수 있습니다.

예를 들어, 개체 유형이 Judge인 John Johnson이라는 이름을 검색하는 경우 주석을 달면 모델이 찾으려는 사람이 판사임을 알아내는 데 도움이 될 수 있습니다. 컨텍스트를 사용할 수 있는 경우, Amazon Comprehend는 John Johnson이라는 변호사 또는 증인을 찾지 않습니다. Amazon Comprehend는 주석을 제공하지 않고 자체 버전의 주석을 생성하지만 판사만 포함하는 데는 그다지 효과적이지 않습니다. 주석을 직접 작성하면 더 나은 결과를 얻고 사용자 지정 개체를 추출할 때 컨텍스트를 더 잘 활용할 수 있는 모델을 생성하는 데 도움이 될 수 있습니다.

주제

- 최소 주석 수
- 주석 모범 사례
- 일반 텍스트 주석 파일
- PDF 주석 파일
- PDF 파일 주석 달기

최소 주석 수

모델을 학습시키는 데 필요한 최소 입력 문서 및 주석 수는 주석 유형에 따라 다릅니다.

PDF 주석

> 이미지 파일, PDF 또는 Word 문서를 분석하기 위한 모델을 생성하려면 PDF 주석을 사용하여 인식기를 학습하십시오. PDF 주석의 경우 최소 250개의 입력 문서와 항목당 최소 100개의 주석을 제공하십시오.

테스트 데이터세트를 제공하는 경우 테스트 데이터에는 생성 요청에 지정된 각 개체 유형에 대한 주석이 하나 이상 포함되어야 합니다.

일반 텍스트 주석

텍스트 문서 분석을 위한 모델을 만들려면 일반 텍스트 주석을 사용하여 인식기를 학습시킬 수 있습니다.

일반 텍스트 주석의 경우 주석이 달린 입력 문서를 3개 이상 제공하고 항목당 주석을 25개 이상 제공하십시오. 총 50개 미만의 주석을 제공하는 경우, Amazon Comprehend는 모델을 테스트하기 위해 입력 문서의 10% 이상을 예약합니다(학습 요청에서 테스트 데이터 세트를 제공하지 않은 경우). 최소 문서 코퍼스 크기는 5KB라는 점을 잊지 마십시오.

입력에 학습 문서가 몇 개만 포함된 경우 학습 입력 데이터에 개체 중 하나를 언급하는 문서가 너무 적다는 오류가 발생할 수 있습니다. 개체가 언급된 추가 문서와 함께 작업을 다시 제출하세요.

테스트 데이터세트를 제공하는 경우 테스트 데이터에는 생성 요청에 지정된 각 개체 유형에 대한 주석이 하나 이상 포함되어야 합니다.

작은 데이터 세트로 모델을 벤치마킹하는 방법의 예는 Amazon Comprehend가 블로그 사이트에서 사용자 지정 엔터티 인식에 대한 주석 제한을 낮추는 방법을 참조하십시오. AWS

주석 모범 사례

주석을 사용할 때 최상의 결과를 얻으려면 다음을 포함하여 여러 가지 사항을 고려해야 합니다.

- 데이터에 주의를 기울여 주석을 달고 해당 항목에 대한 모든 멘션에 주석을 달았는지 확인하십시오. 주석이 정확하지 않으면 결과가 좋지 않을 수 있습니다.
- 입력 데이터에는 주석을 달려는 PDF의 복제본과 같은 중복된 내용이 포함되어서는 안 됩니다. 샘플이 중복되면 테스트 세트가 오염되어 학습 프로세스, 모델 지표 및 모델 동작에 부정적인 영향을 미칠 수 있습니다.
- 모든 문서에 주석을 달아야 하며, 주석이 없는 문서는 적법한 개체가 없기 때문이지 과실로 인한 것이 아님을 확인하십시오. 예를 들어, “J Doe는 14년 동안 엔지니어로 일했습니다”라는 문서가 있는 경우 “J Doe”와 “John Doe”에 대한 주석도 제공해야 합니다. 그렇게 하지 않으면 모델이 혼동을 하여 모델이 “J Doe”를 ENGINEER로 인식하지 못할 수 있습니다. 이는 동일한 문서 내에서, 또는 문서 간에도 일관되어야 합니다.
- 일반적으로 주석을 많이 달면 더 나은 결과를 얻을 수 있습니다.
- 최소한의 문서와 주석으로 모델을 학습시킬 수 있지만 데이터를 추가하면 일반적으로 모델이 향상됩니다. 모델의 정확도를 높이려면 주석이 달린 데이터의 양을 10% 늘리는 것이 좋습니다. 변경되지

않고 다른 모델 버전에서 테스트할 수 있는 테스트 데이터세트에 대해 추론을 실행할 수 있습니다. 그런 다음 후속 모델 버전의 지표를 비교할 수 있습니다.

- 실제 사용 사례와 최대한 유사한 문서를 제공하십시오. 패턴이 반복되는 합성 데이터는 피해야 합니다. 입력 데이터는 과적합을 방지하고 기본 모델이 실제 예제를 보다 잘 일반화할 수 있도록 최대한 다양해야 합니다.
- 문서는 단어 수 측면에서 다양해야 합니다. 예를 들어, 학습 데이터의 모든 문서가 짧으면 결과 모델이 긴 문서에서 개체를 예측하기 어려울 수 있습니다.
- 사용자 지정 개체를 실제로 감지할 때 사용할 것으로 예상되는 것과 동일한 데이터 분포(추론 시간)를 학습에 적용해 보십시오. 예를 들어, 추론 시 개체가 없는 문서를 보내려는 경우 이 문서도 교육 문서 세트에 포함되어야 합니다.

추가 제안 사항은 사용자 지정 개체 인식기 성능 개선을 참조하세요.

일반 텍스트 주석 파일

일반 텍스트 주석의 경우 주석 목록이 포함된 쉼표로 구분된 값 (CSV) 파일을 생성합니다. 학습 파일 입력 형식이 한 줄에 한 문서인 경우 CSV 파일에는 다음 열이 포함되어야 합니다.

파일	행	오프셋 시작	오프셋 종료	유형
문서가 포함된 파일의 이름입니다. 예를 들어, 문서 파일 중 하나가 `s3://my-S3-bucket/test-files/documents.txt` 에 있는 경우, `File` 열의 값은 `documents.txt` 가 됩니다. 파일 이름의 일부로 파일 확장자(이 경우 `'.txt'`)	개체가 포함된 줄 번호입니다. 입력 형식이 파일당 문서 하나인 경우 이 열을 생략합니다.	개체가 시작되는 위치를 나타내는 입력 텍스트의 문자 오프셋(줄의 시작 부분을 기준으로 함)입니다. 첫 번째 문자는 위치 0에 있습니다.	개체가 끝나는 위치를 나타내는 입력 텍스트의 문자 오프셋입니다.	고객이 정의한 개체 유형. 개체 유형은 대문자로 밑줄로 구분된 문자열이어야 합니다. `MANAGER`, `SENIOR_MANAGER` 또는 `PRODUCT_CODE` 같은 설명형 개체 유형을 사용하는 것이 좋습니다. 모델당 최대 25개의 개체 유형을 학습시킬 수 있습니다.

파일	행	오프셋 시작	오프셋 종료	유형
를 포함해야 합니다.				

학습 파일 입력 형식이 파일당 하나의 문서인 경우, 줄 번호 열을 생략하고 시작 오프셋 및 종료 오프셋 값은 문서 시작부터 개체의 오프셋입니다.

다음 예제는 한 줄에 한 문서에 대한 것입니다. `documents.txt` 파일은 네 줄(행 0, 1, 2, 3)로 구성됩니다.

```
Diego Ramirez is an engineer in the high tech industry.
Emilio Johnson has been an engineer for 14 years.
J Doe is a judge on the Washington Supreme Court.
Our latest new employee, Mateo Jackson, has been a manager in the industry for 4 years.
```

주석 목록이 있는 CSV 파일은 다음과 같습니다.

```
File, Line, Begin Offset, End Offset, Type
documents.txt, 0, 0, 13, ENGINEER
documents.txt, 1, 0, 14, ENGINEER
documents.txt, 3, 25, 38, MANAGER
```

> **Note**
>
> 주석 파일에서 개체를 포함하는 줄 번호는 줄 0으로 시작합니다. 이 예제에서는 `documents.txt`의 줄 2에 개체가 없기 때문에 CSV 파일에는 줄 2에 대한 항목이 없습니다.

데이터 파일 생성

오류 위험을 줄이려면 적절하게 구성된 CSV 파일에 주석을 넣는 것이 중요합니다. CSV 파일을 수동으로 구성하려면 다음 조건이 충족되어야 합니다.

- UTF-8 인코딩은 대부분의 경우 기본값으로 사용되더라도 명시적으로 지정해야 합니다.
- 첫 번째 줄에는 열 머리글: `File`, `Line`(선택 사항), `Begin Offset`, `End Offset`, `Type`이 포함됩니다.

잠재적 문제를 방지하려면 프로그래밍 방식으로 CSV 입력 파일을 생성하는 것이 좋습니다.

다음 예제는 Python을 사용하여 이전에 표시된 주석에 대한 CSV를 생성합니다.

```
import csv
with open("./annotations/annotations.csv", "w", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow(["File", "Line", "Begin Offset", "End Offset", "Type"])
    csv_writer.writerow(["documents.txt", 0, 0, 11, "ENGINEER"])
    csv_writer.writerow(["documents.txt", 1, 0, 5, "ENGINEER"])
    csv_writer.writerow(["documents.txt", 3, 25, 30, "MANAGER"])
```

PDF 주석 파일

PDF 주석의 경우 SageMaker AI Ground Truth를 사용하여 증강 매니페스트 파일에 레이블이 지정된 데이터 세트를 생성합니다. Ground Truth는 사용자(또는 고용한 작업 인력)가 기계 학습 모델에 사용할 학습 데이터 세트 구축에 도움이 되는 데이터 레이블 지정 서비스입니다. Amazon Comprehend는 증강 매니페스트 파일을 사용자 정의 모델의 학습 데이터로 받아들입니다. Amazon Comprehend 콘솔 또는 개체 인식기 생성 API 작업을 사용하여 사용자 지정 개체 인식기를 생성할 때 이러한 파일을 제공할 수 있습니다.

Ground Truth의 기본 제공 작업 유형인 명명된 개체 인식을 사용하여 작업자가 텍스트에서 개체를 식별하도록 하는 레이블 작업을 생성할 수 있습니다. 자세한 내용은 Amazon SageMaker AI 개발자 안내서의 명명된 개체 인식을 참조하세요. Amazon SageMaker Ground Truth에 대한 자세한 내용은 Amazon SageMaker AI Ground Truth를 사용하여 데이터 레이블 지정을 참조하세요.

> **Note**
>
> Ground Truth를 사용하여 중첩된 레이블(둘 이상의 레이블과 연결하는 텍스트)을 정의할 수 있습니다. 하지만 Amazon Comprehend 개체 인식은 중첩 레이블을 지원하지 않습니다.

증강 매니페스트 파일은 JSON 라인 형식입니다. 이 파일에서 각 라인은 학습 문서 및 관련 레이블이 포함된 완전한 JSON 객체입니다. 다음 예제는 텍스트에 언급된 개인의 직업을 감지하도록 개체 인식기를 학습시키는 증강 매니페스트 파일입니다.

```
{"source":"Diego Ramirez is an engineer in the high tech
 industry.","NamedEntityRecognitionDemo":{"annotations":{"entities":
[{"endOffset":13,"startOffset":0,"label":"ENGINEER"}],"labels":
```

```
[{"label":"ENGINEER"}]}},"NamedEntityRecognitionDemo-metadata":
{"entities":[{"confidence":0.92}],"job-name":"labeling-job/
namedentityrecognitiondemo","type":"groundtruth/text-span","creation-
date":"2020-05-14T21:45:27.175903","human-annotated":"yes"}}
{"source":"J Doe is a judge on the Washington Supreme
 Court.","NamedEntityRecognitionDemo":{"annotations":{"entities":
[{"endOffset":5,"startOffset":0,"label":"JUDGE"}],"labels":
[{"label":"JUDGE"}]}},"NamedEntityRecognitionDemo-metadata":
{"entities":[{"confidence":0.72}],"job-name":"labeling-job/
namedentityrecognitiondemo","type":"groundtruth/text-span","creation-
date":"2020-05-14T21:45:27.174910","human-annotated":"yes"}}
{"source":"Our latest new employee, Mateo Jackson, has been a manager in
 the industry for 4 years.","NamedEntityRecognitionDemo":{"annotations":
{"entities":[{"endOffset":38,"startOffset":26,"label":"MANAGER"}],"labels":
[{"label":"MANAGER"}]}},"NamedEntityRecognitionDemo-metadata":
{"entities":[{"confidence":0.91}],"job-name":"labeling-job/
namedentityrecognitiondemo","type":"groundtruth/text-span","creation-
date":"2020-05-14T21:45:27.174035","human-annotated":"yes"}}
```


이 JSON 라인 파일의 각 라인은 완전한 JSON 객체이며, 속성에는 Ground Truth의 문서 텍스트, 주석 및 기타 메타데이터가 포함되어 있습니다. 다음 예제는 증강 매니페스트 파일에 있는 단일 JSON 객체이지만 가독성을 위해 형식을 변경하였습니다.

```
{
  "source": "Diego Ramirez is an engineer in the high tech industry.",
  "NamedEntityRecognitionDemo": {
    "annotations": {
      "entities": [
        {
          "endOffset": 13,
          "startOffset": 0,
          "label": "ENGINEER"
        }
      ],
      "labels": [
        {
          "label": "ENGINEER"
        }
      ]
    }
  },
  "NamedEntityRecognitionDemo-metadata": {
    "entities": [
```

```
      {
        "confidence": 0.92
      }
    ],
    "job-name": "labeling-job/namedentityrecognitiondemo",
    "type": "groundtruth/text-span",
    "creation-date": "2020-05-14T21:45:27.175903",
    "human-annotated": "yes"
  }
}
```

이 예제에서 source 속성은 학습 문서의 텍스트를 제공하고 NamedEntityRecognitionDemo 속성은 텍스트의 개체에 대한 주석을 제공합니다. NamedEntityRecognitionDemo 속성의 이름은 임의적이며 Ground Truth에서 레이블 지정 작업을 정의할 때 사용자가 선택한 이름을 제공합니다.

이 예제에서 NamedEntityRecognitionDemo 속성은 레이블 속성 이름으로 Ground Truth 작업자가 학습 데이터에 할당하는 레이블을 제공하는 속성입니다. Amazon Comprehend에 학습 데이터를 제공할 때는 하나 이상의 레이블 속성 이름을 지정해야 합니다. 지정하는 속성 이름의 수는 증강 매니페스트 파일이 단일 레이블 작업의 출력인지 아니면 연속 레이블 작업 출력인지에 따라 달라집니다.

파일이 단일 레이블 작업 출력인 경우 Ground Truth에서 작업을 생성할 때 사용한 단일 레이블 속성 이름을 지정하십시오.

파일이 체인 레이블 지정 작업의 출력인 경우 체인에 있는 하나 이상의 작업에 대한 레이블 속성 이름을 지정하십시오. 각 레이블 속성 이름은 개별 작업의 주석을 제공합니다. 연속 레이블 작업으로 생성되는 증강 매니페스트 파일에는 이러한 속성을 최대 5개까지 지정할 수 있습니다.

증강 매니페스트 파일에서는 일반적으로 레이블 속성 이름이 키 뒤에 옵니다. source 파일이 연속 작업 출력인 경우 레이블 속성 이름은 여러 개가 있습니다. Amazon Comprehend에 학습 데이터를 제공할 때는 모델과 관련된 주석이 포함된 속성만 제공하십시오. “-metadata”로 끝나는 속성은 지정하지 마십시오.

체인 레이블 지정 작업에 대한 자세한 내용과 출력의 예는 Amazon SageMaker AI 개발자 안내서의 체인 레이블 지정 작업을 참조하세요.

PDF 파일 주석 달기

SageMaker AI Ground Truth에서 훈련 PDFs에 주석을 달려면 먼저 다음 사전 조건을 완료하세요.

- python3.8.x 설치

- jq 설치
- AWS CLI 설치

  us-east-1 리전을 사용하는 경우 Python 환경에 이미 설치되어 있으므로 AWS CLI 설치를 건너뛸 수 있습니다. 이 경우에는 AWS Cloud9에서 Python 3.8을 사용하기 위한 가상 환경을 만듭니다.
- AWS 보안 인증 구성
- 프라이빗 SageMaker AI Ground Truth 인력을 생성하여 주석 지원

  새 프라이빗 작업 인력에서 설치 중에 사용할 작업 팀 이름을 선택하여 기록해 두십시오.

주제

- 환경 설정
- S3 버킷에 파일 업로드
- 주석 작업 생성
- SageMaker AI Ground Truth로 주석 달기

환경 설정

1. Windows를 사용하는 경우 Cygwin을 설치하고, Linux 또는 Mac을 사용하는 경우 이 단계를 건너뛰십시오.
2. GitHub에서 주석 아티팩트를 다운로드합니다. 파일 압축을 풉니다.
3. 터미널 창에서 압축이 풀린 폴더(amazon-comprehend-semi-structured-documents-annotation-tools-main)로 이동합니다.
4. 이 폴더에는 종속성을 설치하고, Python virtualenv를 설정하고, 필요한 리소스를 배포하기 위해 실행하는 `Makefiles` 선택 항목이 포함되어 있습니다. Readme 파일을 검토하여 선택하십시오.
5. 권장 옵션은 단일 명령을 사용하여 모든 종속성을 virtualenv에 설치하고, 템플릿에서 AWS CloudFormation 스택을 빌드하고, 대화형 지침에 AWS 계정 따라 스택을에 배포합니다. 다음 명령 실행:

   ```
   make ready-and-deploy-guided
   ```

   이 명령은 구성 옵션 세트를 제공합니다. AWS 리전 가 올바른지 확인합니다. 다른 모든 필드의 경우 기본값을 그대로 사용하거나 사용자 지정 값을 입력할 수 있습니다. AWS CloudFormation 스택 이름을 수정하는 경우 다음 단계에서 필요에 따라 기록해 둡니다.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

        Looking for config file [samconfig.toml] :  Not found

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [sam-app]:
        AWS Region [us-west-2]:
        Parameter PreHumanLambdaTimeoutInSeconds [300]:
        Parameter ConsolidationLambdaTimeoutInSeconds [300]:
        #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
        Confirm changes before deploy [y/N]:
        #SAM needs permission to be able to create roles to connect to the resources in your template
        Allow SAM CLI IAM role creation [Y/n]:
        Save arguments to configuration file [Y/n]:
        SAM configuration file [samconfig.toml]:
        SAM configuration environment [default]:
```

CloudFormation 스택은 주석 도구에 필요한 AWS Lambda, AWS IAM 역할 및 AWS S3 버킷을 생성하고 관리합니다.

CloudFormation 콘솔의 스택 세부 정보 페이지에서 이러한 각 리소스를 검토할 수 있습니다.

6. 이 명령을 실행하면 배포를 시작하라는 메시지가 표시됩니다. CloudFormation은 지정된 리전에 모든 리소스를 생성합니다.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

        Deploying with following values
        ===============================
        Stack name                   : sam-app
        Region                       : us-west-2
        Confirm changeset            : False
        Deployment s3 bucket         : aws-sam-cli-managed-default-samclisourcebucket-jnw8g1gm4pqh
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {"PreHumanLambdaTimeoutInSeconds": "300", "ConsolidationLambdaTimeoutInSeconds": "300"}
        Signing Profiles             : {}

Initiating deployment
=====================
```

CloudFormation 스택 상태가 생성-완료로 전환되면 리소스를 사용할 준비가 된 것입니다.

S3 버킷에 파일 업로드

설정 섹션에서는 comprehend-semi-structured-documents-${AWS::Region}-${AWS::AccountId}라는 S3 버킷을 생성하는 CloudFormation 스택을 배포했습니다. 이제 원본 PDF 문서를 이 버킷에 업로드합니다.

> **Note**
>
> 이 버킷에는 레이블 제작 작업에 필요한 데이터가 들어 있습니다. Lambda 실행 역할 정책은 Lambda 함수가 이 버킷에 액세스할 수 있는 권한을 부여합니다.

'SemiStructuredDocumentsS3Bucket' 키를 사용하여 CloudFormation Stack 세부 정보에서 S3 버킷 이름을 찾을 수 있습니다.

1. S3 버킷에서 새 폴더를 생성합니다. 이 새 폴더의 이름을 'src'로 지정합니다.
2. PDF 소스 파일을 'src' 폴더에 추가합니다. 이후 단계에서 이 파일에 주석을 달아 인식기를 훈련합니다.
3. (선택 사항) 다음은 로컬 디렉터리에서 S3 버킷으로 소스 문서를 업로드하는 데 사용할 수 있는 AWS CLI 예제입니다.

   ```
   aws s3 cp --recursive local-path-to-your-source-docs s3://deploy-guided/src/
   ```

   또는 리전 및 계정 ID를 사용할 수 있습니다.

   ```
   aws s3 cp --recursive local-path-to-your-source-docs s3://deploy-guided-Region-AccountID/src/
   ```
4. 이제 프라이빗 SageMaker AI Ground Truth 작업 인력이 있고 소스 파일을 S3 버킷, deploy-guided/src/에 업로드했습니다. 주석을 달 준비가 되었습니다.

주석 작업 생성

bin 디렉터리의 comprehend-ssie-annotation-tool-cli.py 스크립트는 SageMaker AI Ground Truth 레이블 지정 작업의 생성을 간소화하는 간단한 래퍼 명령입니다. Python 스크립트는 S3 버킷에서 소스 문서를 읽고 한 줄에 하나의 소스 문서가 포함된, 소스 문서에 상응하는 단일 페이지 매니페스트 파일을 만듭니다. 그러면 스크립트가 레이블링 작업을 생성하는데, 이 작업에는 매니페스트 파일이 입력으로 필요합니다.

Python 스크립트는 설정 섹션에서 구성한 S3 버킷과 CloudFormation 스택을 사용합니다. 스크립트의 필수 입력 파라미터는 다음과 같습니다.

- input-s3-path: S3 버킷에 업로드한 소스 문서에 대한 S3 Uri입니다. 예: `s3://deploy-guided/src/`. 또한 이 경로에 리전 및 계정 ID를 추가할 수 있습니다. 예: `s3://deploy-guided-Region-AccountID/src/`.
- cfn-name: CloudFormation 스택 이름입니다. 스택 이름에 기본값을 사용한 경우 cfn-name은 sam-app입니다.

- work-team-name: SageMaker AI Ground Truth에서 프라이빗 작업 인력을 구축할 때 생성한 작업 인력 이름입니다.
- job-name-prefix: SageMaker AI Ground Truth 레이블 지정 작업의 접두사입니다. 참고로 이 필드는 29자로 제한됩니다. 이 값에는 타임스탬프가 추가됩니다. 예: `my-job-name-20210902T232116`.
- entity-types: 레이블 지정 작업 중에 사용할 개체로, 쉼표로 구분합니다. 이 목록에는 학습 데이터 세트에 주석을 달고 싶은 모든 개체가 포함되어야 합니다. Ground Truth 레이블 지정 작업은 주석자가 PDF 문서의 내용에 레이블을 지정할 수 있도록 이러한 개체만 표시합니다.

스크립트가 지원하는 추가 인수를 보려면 -h 옵션을 사용하여 도움말 내용을 표시하십시오.

- 이전 목록에 설명된 대로 입력 파라미터를 사용하여 다음 스크립트를 실행합니다.

```
python bin/comprehend-ssie-annotation-tool-cli.py \
--input-s3-path s3://deploy-guided-Region-AccountID/src/ \
--cfn-name sam-app \
--work-team-name my-work-team-name \
--region us-east-1 \
--job-name-prefix my-job-name-20210902T232116 \
--entity-types "EntityA, EntityB, EntityC" \
--annotator-metadata "key=info,value=sample,key=Due Date,value=12/12/2021"
```

서버에서 다음과 같은 출력을 생성합니다.

```
Downloaded files to temp local directory /tmp/a1dc0c47-0f8c-42eb-9033-74a988ccc5aa
Deleted downloaded temp files from /tmp/a1dc0c47-0f8c-42eb-9033-74a988ccc5aa
Uploaded input manifest file to s3://comprehend-semi-structured-documents-
us-west-2-123456789012/input-manifest/my-job-name-20220203-labeling-
job-20220203T183118.manifest
Uploaded schema file to s3://comprehend-semi-structured-documents-us-
west-2-123456789012/comprehend-semi-structured-docs-ui-template/my-job-
name-20220203-labeling-job-20220203T183118/ui-template/schema.json
Uploaded template UI to s3://comprehend-semi-structured-documents-us-
west-2-123456789012/comprehend-semi-structured-docs-ui-template/my-job-
name-20220203-labeling-job-20220203T183118/ui-template/template-2021-04-15.liquid
Sagemaker GroundTruth Labeling Job submitted: arn:aws:sagemaker:us-
west-2:123456789012:labeling-job/my-job-name-20220203-labeling-job-20220203t183118
(amazon-comprehend-semi-structured-documents-annotation-tools-main)
 user@3c063014d632 amazon-comprehend-semi-structured-documents-annotation-tools-
main %
```

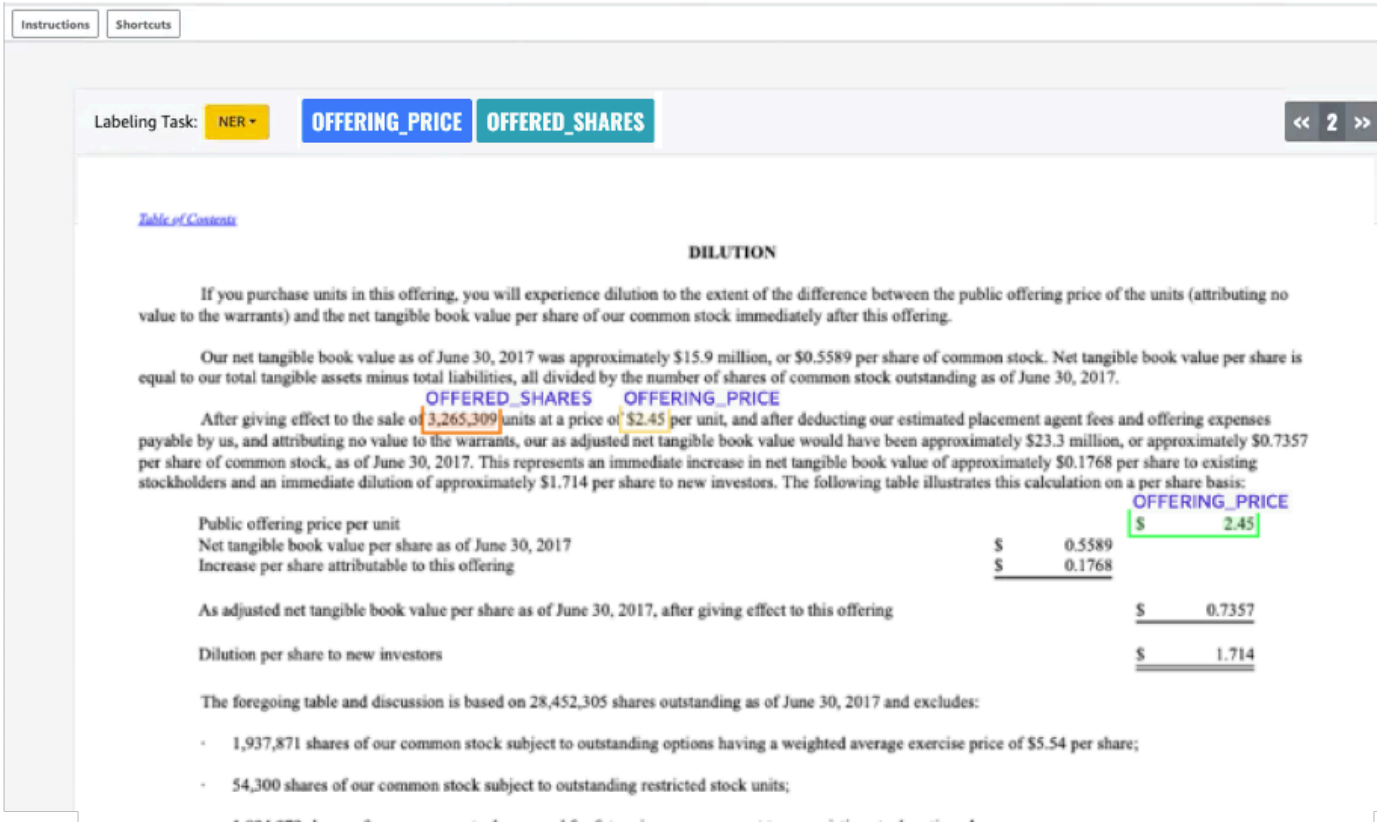
SageMaker AI Ground Truth로 주석 달기

이제 필요한 리소스를 구성하고 레이블 제작 작업을 생성했으므로 레이블 제작 포털에 로그인하여 PDF에 주석을 달 수 있습니다.

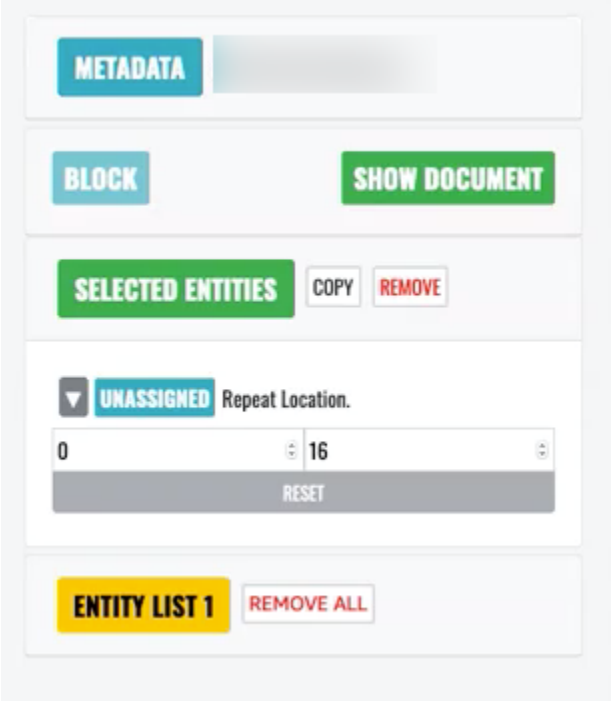
1. Chrome 또는 Firefox 웹 브라우저를 사용하여 SageMaker AI 콘솔에 로그인합니다.
2. 레이블링 인력을 선택하고 비공개를 선택합니다.
3. 프라이빗 작업 인력 요약에서 프라이빗 작업 인력과 함께 만든 레이블 지정 포털 로그인 URL을 선택합니다. 적절한 보안 인증으로 로그인합니다.

   목록에 작업이 없더라도 걱정하지 마십시오. 주석을 달기 위해 업로드한 파일 수에 따라 업데이트하는 데 시간이 걸릴 수 있습니다.
4. 작업을 선택하고 오른쪽 상단에서 작업 시작을 선택하여 주석 화면을 엽니다.

   주석 화면에 문서 중 하나가 열리고, 그 위에는 설정 중에 제공한 개체 유형이 표시됩니다. 개체 유형 오른쪽에는 문서를 탐색하는 데 사용할 수 있는 화살표가 있습니다.



열려 있는 문서에 주석을 답니다. 또한 각 문서에서 주석을 제거하거나 실행 취소하거나 자동으로 태그를 지정할 수 있습니다. 이러한 옵션은 주석 도구의 오른쪽 패널에서 사용할 수 있습니다.



자동 태그를 사용하려면 개체 중 하나의 인스턴스에 주석을 다십시오. 그러면 해당 단어의 다른 모든 인스턴스에 자동으로 해당 개체 유형으로 주석이 추가됩니다.

작업을 마치면 오른쪽 하단에서 제출을 선택하고 탐색 화살표를 사용하여 다음 문서로 이동합니다. 모든 PDF에 주석을 달 때까지 이 과정을 반복합니다.

모든 학습 문서에 주석을 달고 나면 다음 위치의 Amazon S3 버킷에서 JSON 형식의 주석을 찾을 수 있습니다.

```
/output/your labeling job name/annotations/
```

출력 폴더에는 학습 문서 내의 모든 주석을 나열하는 출력 매니페스트 파일도 포함되어 있습니다. 다음 위치에서 출력 매니페스트 파일을 찾을 수 있습니다.

```
/output/your labeling job name/manifests/
```

사용자 지정 개체 인식기 모델 학습

사용자 지정 개체 인식기는 모델을 훈련할 때 포함하는 개체 유형만 식별합니다. 사전 설정된 개체 유형은 자동으로 포함되지 않습니다. 위치, 날짜 또는 사람과 같은 사전 설정된 개체 유형도 식별하려면 해당 개체에 대한 추가 학습 데이터를 제공해야 합니다.

주석이 달린 PDF 파일을 사용하여 사용자 지정 개체 인식기를 만들면 사전 처리나 문서 병합 작업 없이 일반 텍스트, 이미지 파일(JPG, PNG, TIFF), PDF 파일, Word 문서 등 다양한 입력 파일 형식에 해당 인식기를 사용할 수 있습니다. Amazon Comprehend는 이미지 파일 또는 Word 문서에 대한 주석 달기를 지원하지 않습니다.

> **Note**
>
> 주석이 달린 PDF 파일을 사용하는 사용자 지정 개체 인식기는 영어 문서만 지원합니다.

사용자 지정 개체 인식기를 생성한 후에는 DescribeEntityRecognizer 작업을 사용하여 요청 진행 상황을 모니터링할 수 있습니다. `Status` 필드가 `TRAINED`이면 인식기 모델을 사용자 지정 개체 인식에 사용할 수 있습니다.

주제

- 사용자 정의 인식기 학습 (콘솔)
- 사용자 지정 개체 인식기(API) 학습
- 사용자 지정 개체 인식기 지표

사용자 정의 인식기 학습 (콘솔)

Amazon Comprehend 콘솔을 사용하여 사용자 정의 개체 인식기를 생성할 수 있습니다. 이 단원에서는 사용자 정의 개체 인식기 생성과 학습 방법을 보여줍니다.

콘솔을 사용하여 사용자 정의 개체 인식기 만들기 - CSV 형식

사용자 정의 개체 인식기를 만들려면 먼저 모델을 학습시킬 데이터 세트를 제공해야 합니다. 이 데이터 세트를 이용하여 주석이 달린 문서 세트 또는 개체 목록 및 유형 레이블로 구성된 세트와 해당 개체가 포함된 문서 세트를 포함시킵니다. 자세한 정보는 사용자 지정 개체 인식을 참조하십시오.

CSV 파일을 사용하여 사용자 정의 개체 인식기를 학습시키려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 사용자 정의을 선택한 다음 사용자 정의 개체 인식을 선택합니다.
3. 새 모델 생성을 선택합니다.
4. 인식기에 이름을 지정합니다. 이 이름은 리전과 계정 내에서 고유한 이름이어야 합니다.

5. 언어를 선택합니다.
6. 사용자 정의 개체 유형에 인식기가 데이터 세트에서 찾을 수 있도록 하려는 사용자 정의 레이블을 입력합니다.

   개체 유형은 대문자여야 하며, 두 개 이상의 단어로 구성된 경우 밑줄로 단어를 분리해야 합니다.
7. 유형 추가를 선택합니다.
8. 추가 개체 유형을 추가하려면 해당 유형을 입력한 다음 유형 추가를 선택합니다. 추가한 개체 유형 중 하나를 제거하려면 유형 제거를 선택한 다음 목록에서 제거할 개체 유형을 선택합니다. 최대 25개의 개체 유형을 나열할 수 있습니다.
9. 학습 작업을 암호화하려면 인식기 암호화를 선택한 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.

   - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID에서 키 ID를 선택합니다.
   - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ARN에 키 ID의 ARN을 입력합니다.

   > **Note**
   >
   > KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 AWS Key Management Service를 참조하세요.

10. 데이터 사양에서 학습 문서 형식을 선택합니다.

    - CSV 파일 - 학습 문서를 보완하는 CSV 파일입니다. CSV 파일에는 학습된 모델이 탐지할 사용자 정의 개체에 대한 정보가 들어 있습니다. 필요한 파일 형식은 주석 제공인지 아니면 개체 목록 제공인지에 따라 달라집니다.
    - 증강 매니페스트— Amazon SageMaker Ground Truth에서 생성한 레이블이 붙은 데이터 세트입니다. 이 파일은 JSON 라인 형식입니다. 각 라인은 학습 문서와 해당 레이블이 포함된 완전한 JSON 객체입니다. 각 레이블은 학습 문서에 이름이 지정된 개체를 주석에 답니다. 증강 매니페스트 파일은 5개까지 제공할 수 있습니다.

    사용 가능한 형식 및 예제에 대한 자세한 내용은 사용자 지정 개체 인식기 모델 학습를 참조하세요.
11. 학습 유형에서 사용할 학습 유형을 선택합니다.

    - 주석 및 학습 문서 사용하기
    - 개체 목록 및 학습 문서 사용하기

주석을 선택하였다면 Amazon S3에 주석 파일의 URL을 입력합니다. 또한 주석 파일이 있는 Amazon S3의 버킷 또는 폴더로 이동하여 Browse S3를 선택할 수도 있습니다.

개체 목록을 선택하였다면 Amazon S3에 개체 목록의 URL을 입력합니다. 개체 목록이 있는 Amazon S3의 버킷 또는 폴더로 이동한 다음 Browse S3를 선택할 수도 있습니다.

12. Amazon S3의 학습 문서가 들어 있는 입력 데이터 세트의 URL을 입력합니다. 학습 문서가 있는 Amazon S3의 버킷 또는 폴더로 이동하여 폴더 선택을 선택할 수도 있습니다.
13. 테스트 데이터 세트에서 학습된 모델 성능 평가 방법을 선택합니다. 주석 및 개체 목록 학습 유형 모두에 대해 이 작업을 수행할 수 있습니다.

    - 자동 분할: 자동 분할은 제공된 학습 데이터의 10%를 테스트 데이터로 자동으로 사용할 있도록 자동으로 선택합니다.
    - (선택 사항) 고객 제공: 고객 제공을 선택하면 사용자가 정확히 어떤 테스트 데이터를 사용할지 지정할 수 있습니다.

14. 고객 제공 테스트 데이터 세트를 선택하였다면 Amazon S3에 주석 파일의 URL을 입력합니다. 주석 파일이 있는 Amazon S3의 버킷 또는 폴더로 이동한 다음 폴더 선택을 선택할 수 있습니다.
15. IAM 역할 선택 섹션에서 기존 IAM 역할을 선택하거나 새로운 IAM 역할을 생성합니다.

    - 기존 IAM 역할 선택 — 입력 및 출력 Amazon S3 버킷에 액세스할 권한이 있는 IAM 역할이 있으면 이 옵션을 선택합니다.
    - 새 IAM 역할 생성 — Amazon Comprehend가 입력 및 출력 버킷에 액세스할 수 있는 적절한 권한을 가진 새 IAM 역할을 생성하려면 이 옵션을 선택합니다.

    > **Note**
    >
    > 입력 문서가 암호화된 경우 사용된 IAM 역할은 `kms:Decrypt` 권한을 가지고 있어야 합니다. 자세한 내용은 KMS 암호화를 사용하는 데 필요한 권한을 참조하십시오.

16. (선택) VPC에서 Amazon Comprehend로 리소스를 시작하려면 VPC 아래에 VPC ID를 입력하거나 드롭다운 목록에서 ID를 선택합니다.

    1. 서브넷에서 서브넷을 선택합니다. 첫 번째 서브넷을 선택한 후 추가 서브넷을 선택할 수 있습니다.
    2. 보안 그룹을 지정한 경우, 보안 그룹에서 사용할 보안 그룹을 선택합니다. 첫 번째 보안 그룹을 선택한 후 추가 보안 그룹을 선택할 수 있습니다.

> **Note**
>
> 사용자 정의 개체 인식 작업에 VPC를 사용한다면 생성 및 시작 작업에 사용한 `DataAccessRole`은 입력 문서와 출력 버킷에 액세스할 수 있는 VPC 권한을 가지고 있어야 합니다.

17. (선택 사항) 사용자 정의 개체 인식기에 태그를 추가하려면 태그 에 키-값 페어를 입력합니다. 태그 추가를 선택합니다. 인식기 생성 전에 이 페어를 제거하려면 태그 제거를 선택합니다.
18. 학습을 선택합니다.

그러면 새 인식기가 목록에 나타나고 그 상태가 표시됩니다. 처음에는 `Submitted`으로 표시됩니다. 그러면 `Training` 학습 문서를 처리 중인 분류기, `Trained` 사용 준비가 된 분류기, `In error`오류가 있는 분류기를 표시합니다. 작업을 클릭하면 오류 메시지를 포함하여 인식기에 대한 자세한 정보를 얻을 수 있습니다.

콘솔을 사용하여 사용자 정의 개체 인식기 만들기 - 증강 매니페스트

일반 텍스트, PDF 또는 워드 문서를 사용하여 사용자 정의 개체 인식기를 학습시키려면

1. 에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 사용자 정의을 선택한 다음 사용자 정의 개체 인식을 선택합니다.
3. 인식기 학습을 선택합니다.
4. 인식기에 이름을 지정합니다. 이 이름은 리전과 계정 내에서 고유한 이름이어야 합니다.
5. 언어를 선택합니다. 참고: PDF 또는 Word 문서를 학습하는 경우 영어가 지원 언어입니다.
6. 사용자 정의 개체 유형에 인식기가 데이터 세트에서 찾을 수 있도록 하려는 사용자 정의 레이블을 입력합니다.

   개체 유형은 대문자여야 하며, 두 개 이상의 단어로 구성된 경우 밑줄로 단어를 분리해야 합니다.
7. 유형 추가를 선택합니다.
8. 추가 개체 유형을 추가하려면 해당 유형을 입력한 다음 유형 추가를 선택합니다. 추가한 개체 유형 중 하나를 제거하려면 유형 제거를 선택한 다음 목록에서 제거할 개체 유형을 선택합니다. 최대 25개의 개체 유형을 나열할 수 있습니다.
9. 학습 작업을 암호화하려면 인식기 암호화를 선택한 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.

- 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID에서 키 ID를 선택합니다.
- 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ARN에 키 ID의 ARN을 입력합니다.

> **Note**
>
> KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 AWS Key Management Service를 참조하세요.

10. 학습 데이터에서 증강 매니페스트를 데이터 형식으로 선택합니다.

    - 증강 매니페스트 — Amazon SageMaker Ground Truth에서 생성한 레이블이 붙은 데이터 세트입니다. 이 파일은 JSON 라인 형식입니다. 파일의 각 라인은 학습 문서와 해당 레이블이 포함된 완전한 JSON 객체입니다. 각 레이블은 학습 문서에 이름이 지정된 개체를 주석에 답니다. 증강 매니페스트 파일은 5개까지 제공할 수 있습니다. 학습 데이터로 PDF 문서를 사용하는 경우 증강 매니페스트를 선택해야 합니다. 증강 매니페스트 파일은 5개까지 제공할 수 있습니다. 각 파일에 학습 데이터로 사용할 속성을 최대 5개까지 지정할 수 있습니다.

    사용 가능한 형식 및 예제에 대한 자세한 내용은 사용자 지정 개체 인식기 모델 학습를 참조하세요.
11. 학습 모델 유형을 선택합니다.

    일반 텍스트 문서를 선택한 경우 입력 위치 아래에 Amazon SageMaker AIGround Truth 증강 매니페스트 파일의 Amazon S3URL을 입력합니다. Amazon SageMaker AIGround 또한 증강 매니페스트가 있는 Amazon S3의 버킷 또는 폴더로 이동하여 폴더 선택을 선택할 수도 있습니다.
12. 속성 이름에 주석이 포함된 속성의 이름을 입력합니다. 파일에 여러 체인으로 연결된 레이블 작업의 주석이 포함되어 있으면 각 작업에 대한 속성을 추가하십시오. 이 경우 각 속성에는 레이블이 지정된 작업의 주석 세트가 포함됩니다. 참고: 각 파일에는 최대 5개의 속성 이름을 제공할 수 있습니다.
13. 추가 선택.
14. 입력 위치 아래에 PDF, Word 문서를 선택한 경우 Amazon SageMaker AI Ground Truth 증강 매니페스트 파일의 Amazon S3URL을 입력합니다. Amazon SageMaker 또한 증강 매니페스트가 있는 Amazon S3의 버킷 또는 폴더로 이동하여 폴더 선택을 선택할 수도 있습니다.
15. 주석 데이터 파일의 S3 접두사를 입력합니다. 다음은 레이블이 지정된 PDF 문서입니다.
16. 소스 문서의 S3 접두사를 입력합니다. 이는 레이블 제작 작업을 위해 Ground Truth에 제공한 원본 PDF 문서 (데이터 개체) 입니다.

17. 주석이 포함된 속성 이름을 입력합니다. 참고: 각 파일에는 최대 5개의 속성 이름을 제공할 수 있습니다. 사용자가 파일에 지정하지 않은 속성은 모두 무시됩니다.
18. IAM 역할 선택 섹션에서 기존 IAM 역할을 선택하거나 새로운 IAM 역할을 생성합니다.
    - 기존 IAM 역할 선택 — 입력 및 출력 Amazon S3 버킷에 액세스할 권한이 있는 IAM 역할이 있으면 이 옵션을 선택합니다.
    - 새 IAM 역할 생성 — Amazon Comprehend가 입력 및 출력 버킷에 액세스할 수 있는 적절한 권한을 가진 새 IAM 역할을 생성하려면 이 옵션을 선택합니다.

    > **Note**
    >
    > 입력 문서가 암호화된 경우 사용된 IAM 역할은 `kms:Decrypt` 권한을 가지고 있어야 합니다. 자세한 내용은 KMS 암호화를 사용하는 데 필요한 권한을 참조하십시오.

19. (선택) VPC에서 Amazon Comprehend로 리소스를 시작하려면 VPC 아래에 VPC ID를 입력하거나 드롭다운 목록에서 ID를 선택합니다.
    1. 서브넷에서 서브넷을 선택합니다. 첫 번째 서브넷을 선택한 후 추가 서브넷을 선택할 수 있습니다.
    2. 보안 그룹을 지정한 경우, 보안 그룹에서 사용할 보안 그룹을 선택합니다. 첫 번째 보안 그룹을 선택한 후 추가 보안 그룹을 선택할 수 있습니다.

    > **Note**
    >
    > 사용자 정의 개체 인식 작업에 VPC를 사용한다면 생성 및 시작 작업에 사용한 `DataAccessRole`은 입력 문서와 출력 버킷에 액세스할 수 있는 VPC 권한을 가지고 있어야 합니다.

20. (선택 사항) 사용자 정의 개체 인식기에 태그를 추가하려면 태그 에 키-값 페어를 입력합니다. 태그 추가를 선택합니다. 인식기 생성 전에 이 페어를 제거하려면 태그 제거를 선택합니다.
21. 학습을 선택합니다.

그러면 새 인식기가 목록에 나타나고 그 상태가 표시됩니다. 처음에는 `Submitted`으로 표시됩니다. 그러면 `Training` 학습 문서를 처리 중인 분류기, `Trained` 사용 준비가 된 분류기, `In error`오류가

있는 분류기를 표시합니다. 작업을 클릭하면 오류 메시지를 포함하여 인식기에 대한 자세한 정보를 얻을 수 있습니다.

사용자 지정 개체 인식기(API) 학습

사용자 지정 개체 인식 모델을 생성하고 학습하려면 Amazon Comprehend CreateEntityRecognizer API 작업을 사용하십시오.

주제

- 를 사용하여 사용자 지정 개체 인식기 훈련 AWS Command Line Interface
- 를 사용하여 사용자 지정 개체 인식기 훈련 AWS SDK for Java
- Python(Boto3)을 사용한 사용자 지정 개체 인식기 학습

를 사용하여 사용자 지정 개체 인식기 훈련 AWS Command Line Interface

다음 예제는 AWS CLI를 사용하여 `CreateEntityRecognizer` 작업 및 기타 관련 API를 사용하는 방법을 보여 줍니다.

Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

`create-entity-recognizer`CLI 명령을 사용하여 사용자 지정 개체 인식기를 생성합니다. 입력-데이터-구성 파라미터에 대한 자세한 내용은 Amazon Comprehend API 레퍼런스의 CreateEntityRecognizer를 참조하세요.

```
aws comprehend create-entity-recognizer \
     --language-code en \
     --recognizer-name test-6 \
     --data-access-role-arn "arn:aws:iam::account number:role/service-role/
AmazonComprehendServiceRole-role" \
     --input-data-config "EntityTypes=[{Type=PERSON}],Documents={S3Uri=s3://Bucket
 Name/Bucket Path/documents},
                 Annotations={S3Uri=s3://Bucket Name/Bucket Path/annotations}" \
     --region region
```

`list-entity-recognizers` CLI 명령을 사용하여 지역의 모든 개체 인식기를 나열합니다.

```
aws comprehend list-entity-recognizers \
     --region region
```

`describe-entity-recognizer` CLI 명령을 사용하여 사용자 지정 개체 인식기의 작업 상태를 확인합니다.

```
aws comprehend describe-entity-recognizer \
     --entity-recognizer-arn arn:aws:comprehend:region:account number:entity-
recognizer/test-6 \
     --region region
```

를 사용하여 사용자 지정 개체 인식기 훈련 AWS SDK for Java

이 예제에서는 Java를 사용하여 사용자 지정 개체 인식기를 생성하고 모델을 학습합니다.

Java를 사용하는 Amazon Comprehend 예제는 Amazon Comprehend Java 예제를 참조하세요.

Python(Boto3)을 사용한 사용자 지정 개체 인식기 학습

Boto3 SDK 인스턴스화:

```
import boto3
import uuid
comprehend = boto3.client("comprehend", region_name="region")
```

개체 인식기 생성:

```
response = comprehend.create_entity_recognizer(
    RecognizerName="Recognizer-Name-Goes-Here-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "EntityTypes": [
            {
                "Type": "ENTITY_TYPE"
            }
        ],
        "Documents": {
            "S3Uri": "s3://Bucket Name/Bucket Path/documents"
        },
        "Annotations": {
            "S3Uri": "s3://Bucket Name/Bucket Path/annotations"
        }
    }
)
```

```
recognizer_arn = response["EntityRecognizerArn"]
```

모든 인식기 나열:

```
response = comprehend.list_entity_recognizers()
```

인식기가 TRAINED 상태에 도달할 때까지 대기:

```
while True:
    response = comprehend.describe_entity_recognizer(
        EntityRecognizerArn=recognizer_arn
    )

    status = response["EntityRecognizerProperties"]["Status"]
    if "IN_ERROR" == status:
        sys.exit(1)
    if "TRAINED" == status:
        break

    time.sleep(10)
```

사용자 지정 개체 인식기 지표

Amazon Comprehend는 개체 인식기가 작업에 얼마나 잘 작동해야 하는지 추정하는 데 도움이 되는 지표를 제공합니다. 인식기 모델 학습을 기반으로 하므로 학습 중 모델의 성능을 정확하게 나타내긴 하지만 개체 검색 중 API 성능의 근사치에 불과합니다.

학습된 개체 인식기의 메타데이터가 반환될 때마다 지표가 반환됩니다.

Amazon Comprehend는 한 번에 최대 25개 항목에 대한 모델 학습을 지원합니다. 학습된 개체 인식기에서 지표가 반환되면 인식기 전체(글로벌 지표)와 개별 개체(개체 지표) 모두에 대해 점수가 계산됩니다.

세 가지 지표를 모두 글로벌 지표와 개체 지표로 사용할 수 습니다.

- 정밀도

  이는 시스템에서 생성된 개체 중 올바르게 식별되고 올바르게 레이블이 지정된 개체의 비율을 나타냅니다. 이는 모델의 개체 식별이 실제로 올바른 식별이 되는 경우가 얼마나 많은지를 보여줍니다. 이는 총 식별 수치 비율입니다.

즉, 정밀도는 참양성(tp) 및 거짓양성(fp)을 기반으로 하며 정밀도 = tp/(tp + fp)로 계산됩니다.

예를 들어, 문서에 개체의 예가 두 개 있지만 실제로는 하나만 있다고 모델이 예측하는 경우 결과는 참양성 1개와 거짓양성 1개입니다. 이 경우 정밀도 = 1/ (1 + 1)입니다. 모델에서 식별한 두 요소 중 한 요소가 정확하므로 정밀도는 50%입니다.

- 리콜

  이는 문서에 있는 개체 중 시스템에서 올바르게 식별되고 레이블을 지정한 개체의 비율을 나타냅니다. 수학적으로 이것은 올바른 식별의 총 개수(참양성(tp)와 놓친 식별의 총 개수(거짓음성(fn))로 정의됩니다.

  리콜 = tp / (tp + fn)으로 계산됩니다. 예를 들어, 모델이 한 개체를 올바르게 식별했지만 해당 개체가 있는 다른 두 개의 인스턴스를 놓친 경우 결과는 참양성 1개와 거짓음성 2개입니다. 이 경우 리콜은 1/ (1 + 2)입니다. 가능한 세 가지 사례 중 한 개체가 맞기 때문에 리콜은 33.33%입니다.

- F1 점수

  이는 사용자 지정 개체 인식을 위한 모델의 전체 정확도를 측정하는 정밀도 지표와 리콜 지표의 조합입니다. F1 점수는 정밀도 및 리콜 지표의 조화 평균입니다(F1 = 2 * 정밀도 * 리콜/ (정밀도+리콜).

  > **Note**
  >
  > 직관적으로 보면 조화 평균은 단순 평균이나 다른 방법보다 극단에 더 큰 영향을 줍니다(예: `precision` = 0, `recall` = 1은 가능한 모든 기간을 예측하여 간단하게 달성할 수 있습니다. 여기서 단순 평균은 0.5이지만 F1은 0으로 벌칙을 적용합니다).

  위의 예에서는 `precision` = 50%이고 `recall` = 33.33%이므로 F1 = 2 * 0.5 * 0.3333/ (0.5 + 0.3333)입니다. F1 점수는 .3975 또는 39.75% 입니다.

글로벌 및 개별 개체 지표

장소 또는 사람 중 하나인 개체에 대한 다음 문장을 분석하면 글로벌 개체 지표와 개별 개체 지표 간의 관계를 확인할 수 있습니다.

```
John Washington and his friend Smith live in San Francisco, work in San Diego, and own
```

```
    a house in Seattle.
```

이 예제에서 모델은 다음과 같은 예측을 합니다.

```
John Washington = Person
Smith = Place
San Francisco = Place
San Diego = Place
Seattle = Person
```

하지만 예측은 다음과 같았어야 합니다.

```
John Washington = Person
Smith = Person
San Francisco = Place
San Diego = Place
Seattle = Place
```

이에 대한 개별 개체 지표는 다음과 같습니다.

```
entity:  Person
  True positive (TP) = 1 (because John Washington is correctly predicted to be a
    Person).
  False positive (FP) = 1 (because Seattle is incorrectly predicted to be a Person,
    but is actually a Place).
  False negative (FN) = 1 (because Smith is incorrectly predicted to be a Place, but
    is actually a Person).
  Precision = 1 / (1 + 1) = 0.5 or 50%
  Recall = 1 / (1+1) = 0.5 or 50%
  F1 Score = 2 * 0.5 * 0.5 / (0.5 + 0.5) = 0.5 or 50%

entity:  Place
  TP = 2 (because San Francisco and San Diego are each correctly predicted to be a
    Place).
  FP = 1 (because Smith is incorrectly predicted to be a Place, but is actually a
    Person).
  FN = 1 (because Seattle is incorrectly predicted to be a Person, but is actually a
    Place).
  Precision = 2 / (2+1) = 0.6667 or 66.67%
  Recall = 2 / (2+1) = 0.6667 or 66.67%
  F1 Score = 2 * 0.6667 * 0.6667 / (0.6667 + 0.6667) = 0.6667 or  66.67%
```

이에 대한 글로벌 지표는 다음과 같습니다.

글로벌:

```
Global:
  TP = 3 (because John Washington, San Francisco and San Diego are predicted
 correctly.
    This is also the sum of all individual entity TP).
  FP = 2 (because Seattle is predicted as Person and Smith is predicted as Place. This
    is the sum of all individual entity FP).
  FN = 2 (because Seattle is predicted as Person and Smith is predicted as Place. This
    is the sum of all individual FN).
  Global Precision = 3 / (3+2) = 0.6 or 60%
    (Global Precision = Global TP / (Global TP + Global FP))
  Global Recall = 3 / (3+2) = 0.6 or 60%
    (Global Recall = Global TP / (Global TP + Global FN))
  Global F1Score = 2 * 0.6 * 0.6 / (0.6 + 0.6) = 0.6 or 60%
    (Global F1Score = 2 * Global Precision *  Global Recall / (Global Precision +
    Global Recall))
```

사용자 지정 개체 인식기 성능 개선

이러한 지표는 학습된 모델을 사용하여 개체를 식별할 때 학습된 모델이 얼마나 정확하게 수행되는지에 대한 인사이트를 제공합니다. 지표가 예상보다 낮을 경우 지표를 개선하는 데 사용할 수 있는 몇 가지 옵션은 다음과 같습니다.

1. Annotations 또는 개체 목록(일반 텍스트만 해당)을 사용하는지 여부에 따라 해당 설명서의 지침을 준수하여 데이터 품질을 개선하십시오. 데이터를 개선하고 모델을 재학습한 후 더 나은 지표를 발견하면 데이터 품질을 계속 반복하고 개선하여 모델 성능을 개선할 수 있습니다.
2. 개체 목록을 사용하는 경우 주석을 대신 사용하는 것이 좋습니다. 수동 주석을 사용하면 결과를 개선할 수 있는 경우가 많습니다.
3. 데이터 품질 문제가 없는 것이 확실하지만 지표가 여전히 불합리하게 낮은 경우 지원 요청을 제출하세요.

실시간 사용자 지정 인식기 분석 실행

실시간 분석은 작은 문서가 도착하는 즉시 처리하는 애플리케이션에 유용합니다. 예를 들어 소셜 미디어 게시물, 지원 티켓 또는 고객 리뷰에서 사용자 지정 개체를 감지할 수 있습니다.

> **시작하기 전 준비 사항**
>
> 사용자 지정 개체를 감지하려면 먼저 사용자 지정 개체 인식 모델(인식기라고도 함)이 필요합니다. 모델에 대한 자세한 내용은 the section called “인식기 모델 학습”를 참조하세요.

일반 텍스트 주석으로 학습된 인식기는 일반 텍스트 문서에 대한 개체 감지만 지원합니다. PDF 문서 주석으로 학습된 인식기는 일반 텍스트 문서, 이미지, PDF 파일 및 Word 문서에 대한 개체 감지를 지원합니다. 입력 파일에 대한 자세한 내용은 실시간 사용자 지정 분석을 위한 입력를 참조하세요.

이미지 파일 또는 스캔한 PDF 문서를 분석하려는 경우, IAM 정책에서 두 가지 Amazon Textract API 메서드(DetectDocumentText와 AnalyzeDocument)를 사용할 수 있는 권한을 부여해야 합니다. Amazon Comprehend는 텍스트 추출 중에 이러한 메서드를 간접적으로 호출합니다. 예제 정책은 문서 분석 작업을 수행하는 데 필요한 권한을 참조하십시오.

주제

- 사용자 지정 개체 인식을 위한 실시간 분석(콘솔)
- 사용자 지정 개체 인식을 위한 실시간 분석(API)
- 실시간 분석을 위한 출력

사용자 지정 개체 인식을 위한 실시간 분석(콘솔)

Amazon Comprehend 콘솔을 사용하여 사용자 지정 모델로 실시간 분석을 실행할 수 있습니다. 먼저, 실시간 분석을 실행할 엔드포인트를 생성합니다. 엔드포인트를 생성한 후 실시간 분석을 실행합니다.

프로비저닝 엔드포인트 처리량 및 관련 비용에 대한 자세한 내용은 Amazon Comprehend 엔드포인트 사용을 참조하세요.

주제

- 사용자 지정 개체 감지를 위한 엔드포인트 생성
- 실시간 사용자 지정 개체 감지 실행

사용자 지정 개체 감지를 위한 엔드포인트 생성

엔드포인트를 생성하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 엔드포인트를 선택하고 엔드포인트 생성 버튼을 선택합니다. 엔드포인트 생성 화면이 열립니다.
3. 엔드포인트에 이름을 지정합니다. 이 이름은 현재의 리전 및 계정 내에서 고유해야 합니다.
4. 새 엔드포인트를 연결할 사용자 지정 모델을 선택합니다. 드롭다운에서 모델 이름을 기준으로 검색할 수 있습니다.

> **Note**
>
> 엔드포인트를 연결하려면 먼저 모델을 생성해야 합니다. 아직 모델이 없는 경우 사용자 지정 개체 인식기 모델 학습을 참조하세요.

5. (선택) 엔드포인트에 태그를 추가하려면 태그 에 키-값 페어를 입력하고 태그 추가를 선택합니다. 엔드포인트를 생성하기 전에 이 페어를 제거하려면 태그 제거를 선택합니다.
6. 엔드포인트에 할당할 추론 단위(IU) 수를 입력합니다. 각 단위는 초당 최대 2개 문서에 대한 초당 100자의 처리량을 나타냅니다. 엔드포인트 처리량에 대한 자세한 내용은 Amazon Comprehend 엔드포인트 사용을 참조하세요.
7. (선택) 새 엔드포인트를 생성하는 경우 IU 예측기를 사용할 수 있는 옵션이 있습니다. 예측기는 요청할 IU 수를 결정하는 데 도움이 될 수 있습니다. 추론 단위 수는 처리량 또는 초당 분석하려는 문자 수에 따라 달라집니다.
8. 구매 요약에서 시간당, 일별, 월별 예상 엔드포인트 비용을 검토하십시오.
9. 시작 시점부터 삭제할 때까지 계정에 엔드포인트에 대한 요금이 발생한다는 것을 이해하면 확인란을 선택합니다.
10. Create endpoint(엔드포인트 생성)을 선택합니다.

실시간 사용자 지정 개체 감지 실행

사용자 지정 개체 인식기 모델의 엔드포인트를 만든 후 실시간 분석을 실행하여 개별 문서에서 개체를 감지할 수 있습니다.

Amazon Comprehend 콘솔을 사용하여 텍스트의 사용자 지정 개체를 감지하려면 다음 단계를 완료합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 실시간 분석을 선택합니다.
3. 입력 텍스트 섹션에서 분석 유형에 사용자 지정을 선택합니다.
4. 엔드포인트 선택에 대해, 사용하려는 개체 감지 모델과 연결된 엔드포인트를 선택합니다.
5. 분석을 위한 입력 데이터를 지정하려면 텍스트를 입력하거나 파일을 업로드할 수 있습니다.
   - 텍스트를 입력하려면:
     a. 입력 텍스트를 선택합니다.
     b. 분석할 텍스트를 입력합니다.
   - 파일을 업로드하려면:
     a. 파일 업로드를 선택하고 업로드할 파일명을 입력합니다.
     b. (선택) 고급 읽기 작업에서 텍스트 추출을 위한 기본 작업을 우선 지정할 수 있습니다. 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하십시오.
6. 분석을 선택합니다. 콘솔에 분석 결과가 신뢰도 평가와 함께 표시됩니다.

사용자 지정 개체 인식을 위한 실시간 분석(API)

Amazon Comprehend API를 사용하여 사용자 지정 모델로 실시간 분석을 실행할 수 있습니다. 먼저, 실시간 분석을 실행할 엔드포인트를 생성합니다. 엔드포인트를 생성한 후 실시간 분석을 실행합니다.

프로비저닝 엔드포인트 처리량 및 관련 비용에 대한 자세한 내용은 Amazon Comprehend 엔드포인트 사용을 참조하세요.

주제

- 사용자 지정 개체 감지를 위한 엔드포인트 생성
- 실시간 사용자 지정 개체 감지 실행

사용자 지정 개체 감지를 위한 엔드포인트 생성

엔드포인트과 관련한 비용에 대한 자세한 내용은 Amazon Comprehend 엔드포인트 사용을 참조하세요.

를 사용하여 엔드포인트 생성 AWS CLI

를 사용하여 엔드포인트를 생성하려면 `create-endpoint` 명령을 AWS CLI사용합니다.

```
$ aws comprehend create-endpoint \
> --desired-inference-units number of inference units \
> --endpoint-name endpoint name \
> --model-arn arn:aws:comprehend:region:account-id:model/example \
> --tags Key=Key,Value=Value
```

명령이 성공하면 Amazon Comprehend는 엔드포인트 ARN으로 응답합니다.

```
{
   "EndpointArn": "Arn"
}
```

이 명령, 파라미터 인수, 출력에 대한 자세한 내용은 AWS CLI 명령 참조의 `create-endpoint`을 참조하세요.

실시간 사용자 지정 개체 감지 실행

사용자 지정 개체 인식기 모델에 대한 엔드포인트를 생성한 후 해당 엔드포인트를 사용하여 DetectEntities API 작업을 실행합니다. `text` 또는 `bytes` 파라미터를 사용하여 텍스트 입력을 제공할 수 있습니다. `bytes` 파라미터를 사용하여 다른 입력 유형을 입력합니다.

이미지 파일 및 PDF 파일의 경우 `DocumentReaderConfig` 파라미터를 사용하여 기본 텍스트 추출 작업을 재정의할 수 있습니다. 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하세요.

AWS CLI를 사용한 텍스트의 개체 감지

텍스트의 사용자 지정 개체를 감지하려면 `text` 파라미터에서 입력 텍스트를 사용하여 `detect-entities` 명령을 실행합니다.

Example : CLI를 사용하여 입력 텍스트의 개체를 감지합니다

```
$ aws comprehend detect-entities \
> --endpoint-arn arn \
> --language-code en \
> --text  "Andy Jassy is the CEO of Amazon."
```

명령이 성공하면 Amazon Comprehend는 분석으로 응답합니다. Amazon Comprehend가 감지한 각 개체에 대해 개체 유형, 텍스트, 위치 및 신뢰도 점수를 제공합니다.

AWS CLI를 사용한 반정형 문서의 개체 감지

PDF, Word, 또는 이미지 파일의 사용자 지정 개체를 감지하려면 `bytes` 파라미터에서 입력 텍스트를 사용하여 `detect-entities` 명령을 실행합니다.

Example : CLI를 사용하여 이미지 파일의 개체를 감지합니다.

이 예제에서는 이미지 바이트를 base64로 인코딩하는 `fileb` 옵션을 사용하여 이미지 파일을 전달하는 방법을 보여줍니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 바이너리 대형 객체를 참조하세요.

이 예제에서는 텍스트 추출 옵션을 설정하기 위해 `config.json`이라는 이름의 JSON 파일도 전달합니다.

```
$ aws comprehend detect-entities \
> --endpoint-arn arn \
> --language-code en \
> --bytes fileb://image1.jpg   \
> --document-reader-config file://config.json
```

config.json 파일에는 다음 내용이 포함되어 있습니다.

```
{
   "DocumentReadMode": "FORCE_DOCUMENT_READ_ACTION",
   "DocumentReadAction": "TEXTRACT_DETECT_DOCUMENT_TEXT"
}
```

명령 구문에 대한 자세한 내용은 Amazon Comprehend API 참조의 DetectEntities를 참조하세요.

실시간 분석을 위한 출력

텍스트 입력을 위한 출력

`Text`파라미터를 사용하여 텍스트를 입력하면 분석에서 감지한 개체 배열이 출력됩니다. 다음 예는 JUDGE 개체 두 개를 탐지한 분석을 보여줍니다.

```
{
        "Entities":
        [
            {
                "BeginOffset": 0,
                "EndOffset": 22,
                "Score": 0.9763959646224976,
                "Text": "John Johnson",
                "Type": "JUDGE"
            },
            {
                "BeginOffset": 11,
                "EndOffset": 15,
                "Score": 0.9615424871444702,
                "Text": "Thomas Kincaid",
                "Type": "JUDGE"
            }
        ]
    }
```

반구조화된 입력을 위한 출력

반정형 입력 문서 또는 텍스트 파일의 경우 출력에 다음과 같은 추가 필드가 포함될 수 있습니다.

- 문서 메타데이터(DocumentMetadata) — 문서에 대한 추출 정보입니다. 메타데이터에는 문서의 페이지 목록과 각 페이지에서 추출한 문자 수가 포함됩니다. 요청에 `Byte` 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 문서 유형(DocumentType) — 입력 문서에 있는 각 페이지의 문서 유형입니다. 요청에 `Byte` 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 블록 — 입력 문서에 있는 각 텍스트 블록에 대한 정보입니다. 블록은 중첩됩니다. 페이지 블록에는 각 텍스트 라인에 대한 블록이 포함되며, 이 블록에는 각 단어에 대한 블록이 포함됩니다. 요청에 `Byte` 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 블록 참조 — 이 개체의 각 블록에 대한 참조입니다. 요청에 `Byte` 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다. 텍스트 파일에는 이 필드가 없습니다.
- 오류(Errors) — 입력 문서를 처리하는 동안 시스템에서 감지한 페이지 수준 오류입니다. 시스템에서 오류가 발생하지 않으면 이 필드는 비어 있습니다.

이러한 출력 필드에 대한 설명은 Amazon Comprehend API 참조의 DetectEntities를 참조하세요. 레이아웃 요소에 대한 자세한 내용은 Amazon Textract 개발자 사용서의 Amazon Textract 분석 객체를 참조하세요.

다음 예제는 한 페이지의 PDF 스캔 입력 문서의 출력을 보여줍니다.


```
{
    "Entities": [{
        "Score": 0.9984670877456665,
        "Type": "DATE-TIME",
        "Text": "September 4,",
        "BlockReferences": [{
            "BlockId": "42dcaaee-c484-4b5d-9e3f-ae0be928b3e1",
            "BeginOffset": 0,
            "EndOffset": 12,
            "ChildBlocks": [{
                    "ChildBlockId": "6e9cbb43-f8be-4da0-9a4b-ff9a6c350a14",
                    "BeginOffset": 0,
                    "EndOffset": 9
                },
                {
                    "ChildBlockId": "599e0d53-ae9f-491b-a762-459b22c79ff5",
                    "BeginOffset": 0,
                    "EndOffset": 2
                },
                {
                    "ChildBlockId": "599e0d53-ae9f-491b-a762-459b22c79ff5",
                    "BeginOffset": 0,
                    "EndOffset": 2
                }
            ]
        }]
    }],
    "DocumentMetadata": {
        "Pages": 1,
        "ExtractedCharacters": [{
            "Page": 1,
            "Count": 609
        }]
    },
    "DocumentType": [{
        "Page": 1,
        "Type": "SCANNED_PDF"
```

```
    }],
    "Blocks": [{
        "Id": "ee82edf3-28de-4d63-8883-40e2e4938ccb",
        "BlockType": "LINE",
        "Text": "Your Band",
        "Page": 1,
        "Geometry": {
            "BoundingBox": {
                "Height": 0.024125460535287857,
                "Left": 0.11745482683181763,
                "Top": 0.06821706146001816,
                "Width": 0.12074867635965347
            },
            "Polygon": [{
                    "X": 0.11745482683181763,
                    "Y": 0.06821706146001816
                },
                {
                    "X": 0.2382034957408905,
                    "Y": 0.06821706146001816
                },
                {
                    "X": 0.2382034957408905,
                    "Y": 0.09234252572059631
                },
                {
                    "X": 0.11745482683181763,
                    "Y": 0.09234252572059631
                }
            ]
        },
        "Relationships": [{
            "Ids": [
                "b105c561-c8d9-485a-a728-7a5b1a308935",
                "60ecb119-3173-4de2-8c5d-de182a5f86a5"
            ],
            "Type": "CHILD"
        }]
    }]
}
```


다음 예제는 네이티브 PDF 문서 분석에 대한 출력을 보여줍니다.

Example PDF 문서의 사용자 정의 개체 인식 분석 결과 예제


```
{
        "Blocks":
        [
            {
                "BlockType": "LINE",
                "Geometry":
                {
                    "BoundingBox":
                    {
                        "Height": 0.012575757575757575,
                        "Left": 0.0,
                        "Top": 0.0015063131313131314,
                        "Width": 0.02262091503267974
                    },
                    "Polygon":
                    [
                        {
                            "X": 0.0,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.014082070707070706
                        },
                        {
                            "X": 0.0,
                            "Y": 0.014082070707070706
                        }
                    ]
                },
                "Id": "4330efed-6334-4fc4-ba48-e050afa95c8d",
                "Page": 1,
                "Relationships":
                [
                    {
                        "ids":
                        [
                            "f343ce48-583d-4abe-b84b-a232e266450f"
```

```
                    ],
                    "type": "CHILD"
                }
            ],
            "Text": "S-3"
        },
        {
            "BlockType": "WORD",
            "Geometry":
            {
                "BoundingBox":
                {
                    "Height": 0.012575757575757575,
                    "Left": 0.0,
                    "Top": 0.0015063131313131314,
                    "Width": 0.02262091503267974
                },
                "Polygon":
                [
                    {
                        "X": 0.0,
                        "Y": 0.0015063131313131314
                    },
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.0015063131313131314
                    },
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.014082070707070706
                    },
                    {
                        "X": 0.0,
                        "Y": 0.014082070707070706
                    }
                ]
            },
            "Id": "f343ce48-583d-4abe-b84b-a232e266450f",
            "Page": 1,
            "Relationships":
            [],
            "Text": "S-3"
        }
    ],
```

```
        "DocumentMetadata":
        {
            "PageNumber": 1,
            "Pages": 1
        },
        "DocumentType": "NativePDF",
        "Entities":
        [
            {
                "BlockReferences":
                [
                    {
                        "BeginOffset": 25,
                        "BlockId": "4330efed-6334-4fc4-ba48-e050afa95c8d",
                        "ChildBlocks":
                        [
                            {
                                "BeginOffset": 1,
                                "ChildBlockId": "cbba5534-ac69-4bc4-beef-306c659f70a6",
                                "EndOffset": 6
                            }
                        ],
                        "EndOffset": 30
                    }
                ],
                "Score": 0.9998825926329088,
                "Text": "0.001",
                "Type": "OFFERING_PRICE"
            },
            {
                "BlockReferences":
                [
                    {
                        "BeginOffset": 41,
                        "BlockId": "f343ce48-583d-4abe-b84b-a232e266450f",
                        "ChildBlocks":
                        [
                            {
                                "BeginOffset": 0,
                                "ChildBlockId": "292a2e26-21f0-401b-a2bf-03aa4c47f787",
                                "EndOffset": 9
                            }
                        ],
                        "EndOffset": 50
```

```
                }
            ],
            "Score": 0.9809727537330395,
            "Text": "6,097,560",
            "Type": "OFFERED_SHARES"
        }
    ],
    "File": "example.pdf",
    "Version": "2021-04-30"
}
```

사용자 지정 개체 인식을 위한 분석 작업 실행

비동기 분석 작업을 실행하여 하나 이상의 문서 집합에서 사용자 지정 개체를 감지할 수 있습니다.

> **시작하기 전 준비 사항**
>
> 사용자 지정 개체를 감지하려면 먼저 사용자 지정 개체 인식 모델(인식기라고도 함)이 필요합니다. 모델에 대한 자세한 내용은 the section called “인식기 모델 학습”를 참조하세요.

일반 텍스트 주석으로 학습된 인식기는 일반 텍스트 문서에 대한 개체 감지만 지원합니다. PDF 문서 주석으로 학습된 인식기는 일반 텍스트 문서, 이미지, PDF 파일 및 Word 문서에 대한 개체 감지를 지원합니다. 텍스트 파일이 아닌 파일의 경우 Amazon Comprehend는 분석을 실행하기 전에 텍스트 추출을 수행합니다. 입력 파일에 대한 자세한 내용은 비동기 사용자 지정 분석을 위한 입력를 참조하세요.

이미지 파일 또는 스캔한 PDF 문서를 분석하려는 경우, IAM 정책에서 두 가지 Amazon Textract API 메서드(DetectDocumentText와 AnalyzeDocument)를 사용할 수 있는 권한을 부여해야 합니다. Amazon Comprehend는 텍스트 추출 중에 이러한 메서드를 간접적으로 호출합니다. 정책 예제는 문서 분석 작업을 수행하는 데 필요한 권한을 참조하세요.

비동기 분석 작업을 실행하려면 다음과 같은 전체 단계를 수행합니다.

1. Amazon S3 버킷에 문서를 저장합니다.
2. API 또는 콘솔을 사용해 분석 작업을 시작합니다.
3. 분석 작업 진행 상황을 모니터링합니다.
4. 작업이 실행되어 완료된 후 작업을 시작할 때 지정한 S3 버킷에서 분석 결과를 검색합니다.

주제

사용자 지정 개체 감지 작업 시작(콘솔)

콘솔을 사용하여 사용자 지정 개체 인식을 위한 비동기 분석 작업을 시작하고 모니터링할 수 있습니다.

비동기 분석 작업을 시작하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 분석 작업을 선택한 다음 작업 생성을 선택합니다.
3. 분류 작업에 이름을 지정합니다. 이 이름은 계정 및 현재 리전에 고유해야 합니다.
4. 분석 유형에서 사용자 지정 개체 인식을 선택합니다.
5. 인식기 모델에서 사용할 사용자 지정 개체 인식기를 선택합니다.
6. 버전에서 사용할 인식기 버전을 선택합니다.
7. (선택) Amazon Comprehend가 작업을 처리하는 동안 사용하는 데이터를 암호화하도록 선택한 경우, 작업 암호화를 선택하십시오. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.

    - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID의 키 ID를 선택합니다.
    - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ARN에 키 ID의 ARN을 입력합니다.

    > **Note**
    >
    > KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 (키 관리 서비스(KMS)를 참조하세요.

8. 입력 데이터에서 입력 문서가 포함된 Amazon S3 버킷의 위치를 입력하거나 S3 찾아보기를 선택하여 해당 버킷으로 이동합니다. 이 버킷은 호출하는 서비스와 동일한 리전에 있어야 합니다. 분석 작업에 대한 액세스 권한에 사용하는 IAM 역할에는 S3 버킷에 대한 읽기 권한이 있어야 합니다.
9. (선택) 입력 형식으로 입력 문서의 형식을 선택할 수 있습니다. 형식은 파일당 문서 하나, 또는 단일 파일의 라인당 문서 하나일 수 있습니다. 라인당 문서 하나는 텍스트 문서에만 적용됩니다.

10. (선택) 문서 읽기 모드에서 기본 텍스트 추출 작업을 우선 지정할 수 있습니다. 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하십시오.
11. 출력 데이터에서 Amazon Comprehend가 작업의 출력 데이터를 기록해야 하는 Amazon S3 버킷의 위치를 입력하거나 S3 찾아보기를 선택하여 해당 위치로 이동합니다. 이 버킷은 호출하는 서비스와 동일한 리전에 있어야 합니다. 분류 작업에 대한 액세스 권한에 사용하는 IAM 역할에는 S3 버킷에 대한 쓰기 권한이 있어야 합니다.
12. (선택) 작업의 출력 결과를 암호화하기로 선택한 경우 암호화를 선택합니다. 그런 다음 현재 계정과 연결된 KMS 키를 사용할지 아니면 다른 계정의 KMS 키를 사용할지 선택합니다.
    - 현재 계정과 연결된 키를 사용하는 경우 KMS 키 ID에서 키 별칭 또는 ID를 선택합니다.
    - 다른 계정과 연결된 키를 사용하는 경우 KMS 키 ID 아래에 키 별칭 또는 ID의 ARN을 입력합니다.
13. (선택) VPC에서 Amazon Comprehend로 리소스를 시작하려면 VPC 아래에 VPC ID를 입력하거나 드롭다운 목록에서 ID를 선택합니다.
    1. 서브넷에서 서브넷을 선택합니다. 첫 번째 서브넷을 선택한 후 추가 서브넷을 선택할 수 있습니다.
    2. 보안 그룹을 지정한 경우, 보안 그룹에서 사용할 보안 그룹을 선택합니다. 첫 번째 보안 그룹을 선택한 후 추가 보안 그룹을 선택할 수 있습니다.

    > **Note**
    >
    > 분석 작업에 VPC를 사용하는 경우 생성 및 시작 작업에 사용되는 `DataAccessRole`은 출력 버킷에 액세스하는 VPC에 대한 권한이 있어야 합니다.

14. 작업 생성 을 선택하여 개체 인식 작업을 생성합니다.

사용자 지정 개체 감지 작업 시작(API)

API를 사용하여 사용자 지정 개체 인식을 위한 비동기 분석 작업을 시작하고 모니터링할 수 있습니다.

StartEntitiesDetectionJob 작업으로 사용자 지정 개체 감지 작업을 시작하려면 학습된 모델의 Amazon 리소스 이름(ARN)인 EntityRecognizerArn을 제공합니다. 이 ARN은 CreateEntityRecognizer 작업에 대한 응답에서 찾을 수 있습니다.

주제

- 를 사용하여 사용자 지정 개체 감지 AWS Command Line Interface
- AWS SDK for Java를 사용한 사용자 지정 개체 감지
- 를 사용하여 사용자 지정 개체 감지 AWS SDK for Python (Boto3)
- PDF 파일에 대한 API 작업 우선 지정

를 사용하여 사용자 지정 개체 감지 AWS Command Line Interface

다음의 Unix, Linux, macOS용 예제를 사용하십시오. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다. 문서 집합에서 사용자 지정 개체를 감지하려면 다음 요청 구문을 사용하십시오.

```
aws comprehend start-entities-detection-job \
     --entity-recognizer-arn "arn:aws:comprehend:region:account number:entity-
recognizer/test-6" \
     --job-name infer-1 \
     --data-access-role-arn "arn:aws:iam::account number:role/service-role/
AmazonComprehendServiceRole-role" \
     --language-code en \
     --input-data-config "S3Uri=s3://Bucket Name/Bucket Path" \
     --output-data-config "S3Uri=s3://Bucket Name/Bucket Path/" \
     --region region
```

Amazon Comprehend는 `JobID` 및 `JobStatus`로 응답하고 요청에서 지정한 S3 버킷의 작업 출력을 반환합니다.

AWS SDK for Java를 사용한 사용자 지정 개체 감지

Java를 사용하는 Amazon Comprehend 예제는 Amazon Comprehend Java 예제를 참조하세요.

를 사용하여 사용자 지정 개체 감지 AWS SDK for Python (Boto3)

이 예제에서는 사용자 지정 개체 인식기를 생성하고 모델을 학습시킨 다음 AWS SDK for Python (Boto3)를 사용하여 엔터니 인식기 작업에서 실행합니다.

Python용 SDK를 인스턴스화합니다.

```
import boto3
import uuid
comprehend = boto3.client("comprehend", region_name="region")
```

개체 인식기 생성:

```
response = comprehend.create_entity_recognizer(
    RecognizerName="Recognizer-Name-Goes-Here-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "EntityTypes": [
            {
                "Type": "ENTITY_TYPE"
            }
        ],
        "Documents": {
            "S3Uri": "s3://Bucket Name/Bucket Path/documents"
        },
        "Annotations": {
            "S3Uri": "s3://Bucket Name/Bucket Path/annotations"
        }
    }
)
recognizer_arn = response["EntityRecognizerArn"]
```

모든 인식기 나열:

```
response = comprehend.list_entity_recognizers()
```

개체 인식기가 학습됨(TRAINED) 상태에 도달할 때까지 기다립니다:

```
while True:
    response = comprehend.describe_entity_recognizer(
        EntityRecognizerArn=recognizer_arn
    )

    status = response["EntityRecognizerProperties"]["Status"]
    if "IN_ERROR" == status:
        sys.exit(1)
    if "TRAINED" == status:
        break

    time.sleep(10)
```

사용자 지정 개체 감지 작업을 시작합니다:

```
response = comprehend.start_entities_detection_job(
    EntityRecognizerArn=recognizer_arn,
    JobName="Detection-Job-Name-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "InputFormat": "ONE_DOC_PER_LINE",
        "S3Uri": "s3://Bucket Name/Bucket Path/documents"
    },
    OutputDataConfig={
        "S3Uri": "s3://Bucket Name/Bucket Path/output"
    }
)
```

PDF 파일에 대한 API 작업 우선 지정

이미지 파일 및 PDF 파일의 경우 DocumentReaderConfig의 InputDataConfig 파라미터를 사용하여 기본 추출 작업을 재정의할 수 있습니다.

다음 예제에서는 myInputDataConfig.json이라는 이름의 JSON 파일을 정의하여 InputDataConfig 값을 설정합니다. DocumentReadConfig이 모든 PDF 파일에 대해 Amazon Textract DetectDocumentText API를 사용하도록 설정합니다.

Example

```
"InputDataConfig": {
  "S3Uri": s3://Bucket Name/Bucket Path",
  "InputFormat": "ONE_DOC_PER_FILE",
  "DocumentReaderConfig": {
      "DocumentReadAction": "TEXTRACT_DETECT_DOCUMENT_TEXT",
      "DocumentReadMode": "FORCE_DOCUMENT_READ_ACTION"
  }
}
```

StartEntitiesDetectionJob 작업 시 myInputDataConfig.json 파일을 InputDataConfig 파라미터로 지정합니다.

```
  --input-data-config file://myInputDataConfig.json
```

DocumentReaderConfig 파라미터에 대한 자세한 내용은 텍스트 추출 옵션을 설정하는을 참조하세요.

비동기 분석 작업을 위한 출력

분석 작업이 완료되면 요청에서 지정한 S3 버킷에 결과가 저장됩니다.

텍스트 입력을 위한 출력

텍스트 입력 파일의 경우 출력은 각 입력 문서의 항목 목록으로 구성됩니다.

다음 예제에서는 라인 형식 당 하나의 문서 형태로 50_docs 이름의 입력 파일에 대한 출력 문서 2개를 보여줍니다.

```
{
        "File": "50_docs",
        "Line": 0,
        "Entities":
        [
            {
                "BeginOffset": 0,
                "EndOffset": 22,
                "Score": 0.9763959646224976,
                "Text": "John Johnson",
                "Type": "JUDGE"
            }
        ]
    }
    {
        "File": "50_docs",
        "Line": 1,
        "Entities":
        [
            {
                "BeginOffset": 11,
                "EndOffset": 15,
                "Score": 0.9615424871444702,
                "Text": "Thomas Kincaid",
                "Type": "JUDGE"
            }
        ]
    }
```

반구조화된 입력을 위한 출력

반구조화된 입력 문서를 위한 출력에 다음과 같은 추가 필드가 포함될 수 있습니다.

- 문서 메타데이터(DocumentMetadata) — 문서에 대한 추출 정보입니다. 메타데이터에는 문서의 페이지 목록과 각 페이지에서 추출한 문자 수가 포함됩니다. 요청에 Byte 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 문서 유형(DocumentType) — 입력 문서에 있는 각 페이지의 문서 유형입니다. 요청에 Byte 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 블록 — 입력 문서에 있는 각 텍스트 블록에 대한 정보입니다. 블록은 블록 안에 중첩될 수 있습니다. 페이지 블록에는 각 텍스트 라인에 대한 블록이 포함되며, 이 블록에는 각 단어에 대한 블록이 포함됩니다. 요청에 Byte 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다.
- 블록 참조 — 이 개체의 각 블록에 대한 참조입니다. 요청에 Byte 파라미터가 포함된 경우 응답에 이 필드가 표시됩니다. 텍스트 파일에는 이 필드가 없습니다.
- 오류(Errors) — 입력 문서를 처리하는 동안 시스템에서 감지한 페이지 수준 오류입니다. 시스템에서 오류가 발생하지 않으면 이 필드는 비어 있습니다.

이러한 출력 필드에 대한 자세한 내용은 Amazon Comprehend API 참조의 DetectEntities를 참조하세요.

다음 예제에서는 한 페이지 분량의 네이티브 PDF 입력 문서의 출력을 보여줍니다.

Example PDF 문서의 사용자 정의 개체 인식 분석 결과 예제


```
{
        "Blocks":
        [
            {
                "BlockType": "LINE",
                "Geometry":
                {
                    "BoundingBox":
                    {
                        "Height": 0.012575757575757575,
                        "Left": 0.0,
                        "Top": 0.0015063131313131314,
                        "Width": 0.02262091503267974
                    },
                    "Polygon":
                    [
                        {
                            "X": 0.0,
                            "Y": 0.0015063131313131314
                        },
```

```
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.0015063131313131314
                    },
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.014082070707070706
                    },
                    {
                        "X": 0.0,
                        "Y": 0.014082070707070706
                    }
                ]
            },
            "Id": "4330efed-6334-4fc4-ba48-e050afa95c8d",
            "Page": 1,
            "Relationships":
            [
                {
                    "ids":
                    [
                        "f343ce48-583d-4abe-b84b-a232e266450f"
                    ],
                    "type": "CHILD"
                }
            ],
            "Text": "S-3"
        },
        {
            "BlockType": "WORD",
            "Geometry":
            {
                "BoundingBox":
                {
                    "Height": 0.012575757575757575,
                    "Left": 0.0,
                    "Top": 0.0015063131313131314,
                    "Width": 0.02262091503267974
                },
                "Polygon":
                [
                    {
                        "X": 0.0,
                        "Y": 0.0015063131313131314
```

```
                    },
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.0015063131313131314
                    },
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.014082070707070706
                    },
                    {
                        "X": 0.0,
                        "Y": 0.014082070707070706
                    }
                ]
            },
            "Id": "f343ce48-583d-4abe-b84b-a232e266450f",
            "Page": 1,
            "Relationships":
            [],
            "Text": "S-3"
        }
    ],
    "DocumentMetadata":
    {
        "PageNumber": 1,
        "Pages": 1
    },
    "DocumentType": "NativePDF",
    "Entities":
    [
        {
            "BlockReferences":
            [
                {
                    "BeginOffset": 25,
                    "BlockId": "4330efed-6334-4fc4-ba48-e050afa95c8d",
                    "ChildBlocks":
                    [
                        {
                            "BeginOffset": 1,
                            "ChildBlockId": "cbba5534-ac69-4bc4-beef-306c659f70a6",
                            "EndOffset": 6
                        }
                    ],
```

```
                        "EndOffset": 30
                    }
                ],
                "Score": 0.9998825926329088,
                "Text": "0.001",
                "Type": "OFFERING_PRICE"
            },
            {
                "BlockReferences":
                [
                    {
                        "BeginOffset": 41,
                        "BlockId": "f343ce48-583d-4abe-b84b-a232e266450f",
                        "ChildBlocks":
                        [
                            {
                                "BeginOffset": 0,
                                "ChildBlockId": "292a2e26-21f0-401b-a2bf-03aa4c47f787",
                                "EndOffset": 9
                            }
                        ],
                        "EndOffset": 50
                    }
                ],
                "Score": 0.9809727537330395,
                "Text": "6,097,560",
                "Type": "OFFERED_SHARES"
            }
        ],
        "File": "example.pdf",
        "Version": "2021-04-30"
    }
```

사용자 정의 모델 생성 및 관리

Amazon Comprehend에는 인사이트 분석 또는 주제 모델링에 사용할 수 있는 내장 NLP(자연어 처리) 모델이 포함되어 있습니다. Amazon Comprehend를 사용하여 개체 인식 및 문서 분류를 위한 사용자 정의 모델을 생성할 수도 있습니다.

모델 버전 관리를 사용하여 모델 기록을 추적할 수 있습니다. 새로운 모델 버전을 만들고 학습시키면 학습 데이터 세트를 변경할 수 있습니다. Amazon Comprehend는 모델 세부 정보 페이지에 각 모델 버전의 세부 정보(모델 성능 포함)를 표시합니다. 시계열에 따라 학습 데이터 세트 변경에 따른 모델 성능의 변화를 확인할 수 있습니다.

Amazon Comprehend 콘솔 또는 API를 사용하여 모델 버전을 생성할 수 있습니다. 대안으로 Amazon Comprehend는 플라이 휠에 새로운 사용자 정의 모델 버전의 학습 및 평가 관련 작업 간소화 기능을 제공합니다.

사용자 지정 모델을 생성한 후 다른 사용자가 모델 사본을 가져오 AWS 계정 도록 허용하여 다른 사용자와 모델을 공유할 수 있습니다.

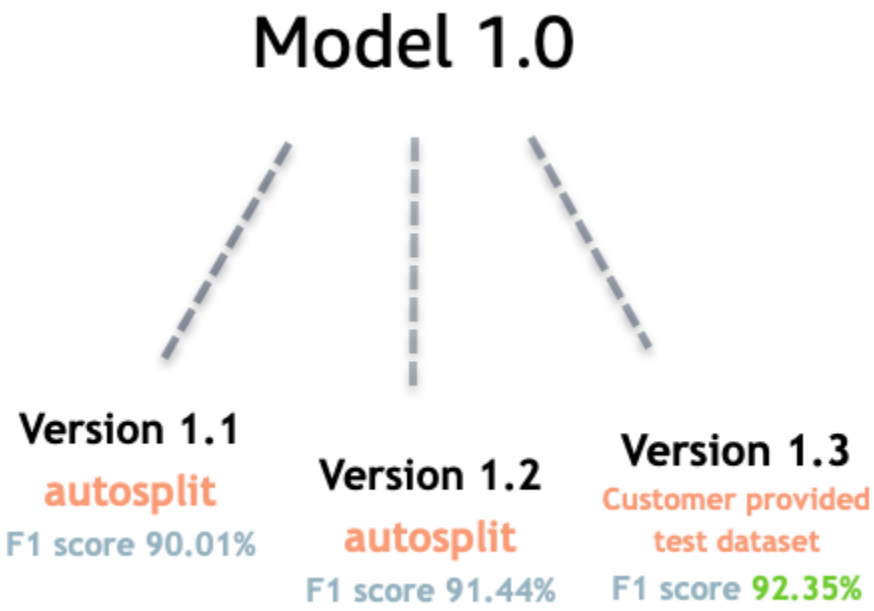
주제

- Amazon Comprehend를 사용한 모델 버전 관리
- 에서 사용자 지정 모델 복사 AWS 계정

Amazon Comprehend를 사용한 모델 버전 관리

인공 지능 및 기계 학습(AI/ML)의 핵심은 빠른 실험입니다. Amazon Comprehend를 사용하면 데이터에 대한 인사이트를 얻는 데 사용할 모델을 학습 및 구축할 수 있습니다. 모델 버전 관리를 사용하면 더 많거나 다른 데이터 세트를 제공하더라도 모델 실행 결과와 관련된 모델링 기록 및 성적을 추적할 수 있습니다. 사용자 정의 분류 모델 또는 사용자 정의 개체 인식 모델을 버전 관리와 함께 사용할 수 있습니다. 시계열에 따른 다양한 버전을 살펴보면 해당 버전이 얼마나 성공적인가에 대한 인사이트를 얻고 성공 상태에 도달하는 데 어떤 파라미터를 사용했는지에 대한 인사이트를 얻을 수 있습니다.

기존 사용자 정의 분류 모델 또는 개체 인식 모델의 새 버전을 학습시킬 때에 모델 세부 정보 페이지에서 새 버전을 생성하기만 하면 됩니다. 그러면 모든 세부 정보가 자동으로 채워집니다. 새 버전 관리는 이전 모델과 동일한 이름(VersionID라고 함)을 사용하지만 생성 과정에서 고유한 버전 이름을 지정해야 합니다. 모델에 새 버전을 추가하면 모델 세부 정보 페이지에서 모든 이전 버전과 그 세부 정보를 한 번에 볼 수 있습니다. 버전 관리를 사용하여 학습 데이터 세트를 변경하면 모델 성능이 어떻게 변하는지 확인할 수 있습니다.



새 사용자 정의 분류기 버전 만들기 (콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 사용자 정의을 선택한 다음 사용자 정의 분류를 선택합니다.
3. 분류자 목록에서 새 버전을 만들 사용자 정의 모델의 이름을 선택합니다. 사용자 지정 모델 세부 정보 페이지가 표시됩니다.
4. 오른쪽 상단에서 새 모델 생성을 선택합니다. 상위 사용자 정의 분류 모델의 세부 정보가 미리 채워진 화면이 열립니다.
5. 버전 이름 아래의 새 버전에 고유한 이름을 추가합니다.
6. 버전 세부 정보에서 새 모델과 관련된 언어 및 레이블 수를 변경할 수 있습니다.
7. 데이터 사양 섹션에서 새 버전에 데이터를 제공하는 방법을 구성합니다. 이전 모델의 문서와 새 문서를 포함하여 전체 데이터를 제공해야 합니다. 분류기 모드 (단일 레이블 또는 다중 레이블), 데이터 형식 (CSV 파일, 증강 매니페스트), 교육 데이터세트, 테스트 데이터세트 (자동 분할 또는 사용자 지정 테스트 데이터 구성) 를 변경할 수 있습니다.
8. (선택 사항) 출력 데이터용 S3 위치를 업데이트합니다.

9. 액세스 권한에서 기존 IAM 역할을 생성하거나 기존 역할을 사용합니다.
10. (선택 사항) VPC 설정 업데이트
11. (선택 사항) 새 버전에 태그를 추가하면 세부 정보를 추적할 수 있습니다.

    사용자 정의 분류기 생성 방법에 대한 자세한 내용은 사용자 정의 분류기 생성을 참조하세요.

새 사용자 정의 개체 인식기 버전 생성 (콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 사용자 정의을 선택한 다음 사용자 정의 개체 인식을 선택합니다.
3. 인식기 모델 목록에서 새 버전 생성에 사용할 인식기의 이름을 선택합니다. 세부 정보 페이지가 표시됩니다.
4. 오른쪽 상단에서 새 버전 학습을 선택합니다. 상위 개체 인식기의 세부 정보가 미리 채워진 화면이 열립니다.
5. 버전 이름 아래의 새 버전에 고유한 이름을 추가합니다.
6. 사용자 정의 개체 유형에서 사용자 데이터 세트에서 인식기로 식별할 사용자 정의 레이블 또는 레이블을 추가하고 유형 추가를 선택합니다. 제공한 주석 또는 개체 목록에서 사용자 정의 개체 유형을 선택합니다. 그러면 인식기는 작업을 실행할 때 포함된 모든 개체 유형을 사용하여 데이터 집합의 개체를 식별합니다. 여러 단어를 사용하는 경우 각 개체 유형은 대문자여야 하며 단어 사이에 밑줄을 넣어야 합니다. 최대 25개의 유형이 허용됩니다.
7. (선택 사항) 작업이 처리되는 동안 저장소 볼륨의 데이터를 암호화하려면 인식기 암호화를 선택합니다.
8. 교육 데이터 섹션에서 주석 및 데이터 형식 세부 정보 (CSV 파일, 증강 매니페스트), 단일 레이블 또는 다중 레이블), 데이터 형식 (CSV, 증강 매니페스트), 교육 데이터세트 및 테스트 데이터 세트 (자동 분할 또는 사용자 지정 테스트 데이터 구성) 를 지정합니다.
9. (선택 사항) 출력 데이터용 S3 위치를 업데이트합니다.
10. 액세스 권한에서 기존 IAM 역할을 생성하거나 기존 역할을 사용합니다.
11. (선택 사항) VPC 설정 업데이트
12. (선택 사항) 새 버전에 태그를 추가하면 세부 정보를 추적할 수 있습니다.

사용자 정의 개체 인식기에 대해 자세히 알아보려면 사용자 정의 개체 인식 및 콘솔을 사용한 사용자 정의 개체 인식기 생성을 참조하세요.

에서 사용자 지정 모델 복사 AWS 계정

Amazon Comprehend 사용자는 2단계 프로세스 AWS 계정 로에서 훈련된 사용자 지정 모델을 복사할 수 있습니다. 먼저 한 AWS 계정 의 사용자(계정 A)가 자신의 계정에 있는 사용자 지정 모델을 공유합니다. 그런 다음 다른 AWS 계정 (계정 B)의 사용자가 모델을 자신의 계정으로 가져옵니다. 계정 B 사용자는 모델을 학습시킬 필요가 없으며, 원본 훈련 데이터나 테스트 데이터를 복사(또는 액세스)할 필요도 없습니다.

계정 A에서 사용자 지정 모델을 공유하기 위해 사용자는 모델 버전에 AWS Identity and Access Management (IAM) 정책을 연결합니다. 이 정책은 사용자 또는 역할과 같은 계정 B의 개체가 자신의 AWS 계정에 있는 Amazon Comprehend로 모델 버전을 가져올 수 있는 권한을 부여합니다. 계정 B 사용자는 원래 모델과 동일한 AWS 리전 으로 모델을 가져와야 합니다.

계정 B에서 모델을 가져오기 위해 이 계정의 사용자는 모델의 Amazon 리소스 이름(ARN)과 같은 필수 세부 정보를 Amazon Comprehend에 제공해야 합니다. 모델을 가져와서이 사용자는 가져온 모델을 복제 AWS 계정 하는 새 사용자 지정 모델을에 생성합니다. 이 모델은 완전히 학습되었으며 문서 분류 또는 명명된 개체 인식과 같은 추론 작업에 사용할 준비가 되어 있습니다.

다음과 같은 경우 사용자 지정 모델을 복사하는 것이 유용합니다.

- 여러를 사용하는 조직에 속합니다 AWS 계정. 예를 들어 조직에 빌드, 단계, 테스트 및 배포와 같은 각 개발 단계에 AWS 계정 대한가 있을 수 있습니다. 또는 데이터 과학 및 엔지니어링과 같은 비즈니스 기능에 AWS 계정 따라 다를 수 있습니다.
- 조직은 Amazon Comprehend에서 사용자 지정 모델을 훈련하고 이를 클라이언트로 제공하는 AWS 파트너와 같은 다른와 협력합니다.

이러한 시나리오에서는 훈련된 사용자 지정 개체 인식기 또는 문서 분류기를 한 개에서 다른 개로 빠르게 복사 AWS 계정 할 수 있습니다. 이러한 방식으로 모델을 복사하는 것이 대체 모델보다 쉽습니다.이 경우 간에 훈련 데이터를 복사 AWS 계정 하여 중복 모델을 훈련할 수 있습니다.

주제

- 사용자 지정 모델을 다른 사용자 지정 모델과 공유 AWS 계정
- 다른에서 사용자 지정 모델 가져오기 AWS 계정

사용자 지정 모델을 다른 사용자 지정 모델과 공유 AWS 계정

Amazon Comprehend를 사용하면 사용자 지정 모델을 다른 사람과 공유하여 다른 사용자가 자신의 AWS 계정으로 이 모델을 가져올 수 있습니다. 사용자가 사용자 지정 모델 중 하나를 가져오면 자신의 계정에 새 사용자 지정 모델을 생성합니다. 이 새 모델은 공유받은 모델과 중복됩니다.

사용자 지정 모델을 공유하려면 다른 사용자가 모델을 가져올 수 있도록 권한을 부여하는 정책을 모델에 연결해야 합니다. 그런 다음 해당 사용자에게 필요한 세부 정보를 제공합니다.

> **Note**
>
> 다른 사용자가 공유한 사용자 지정 모델을 가져올 때는 모델이 포함된 미국 동부(버지니아 북부)와 AWS 리전 같은 동일한 모델을 사용해야 합니다.

주제

시작하기 전 준비 사항

모델을 공유하려면 먼저 사용자 AWS 계정의 Amazon Comprehend에 학습된 사용자 지정 분류기 또는 사용자 지정 개체 인식기가 있어야 합니다. 모델 학습에 대한 자세한 정보는 사용자 지정 분류 또는 사용자 지정 개체 인식을 참조하세요.

필수 권한

IAM 정책문

사용자 지정 모델에 리소스 기반 정책을 추가하려면 먼저 AWS Identity and Access Management (IAM)에 권한이 있어야 합니다. 다음 예시와 같이 모델 정책을 만들고, 가져오고, 삭제할 수 있으려면 사용자, 그룹 또는 역할에 정책이 연결되어 있어야 합니다.

Example 사용자 지정 모델의 리소스 기반 정책을 관리하기 위한 IAM 정책

```
{
  "Effect": "Allow",
```

```
  "Action": [
    "comprehend:PutResourcePolicy",
    "comprehend:DeleteResourcePolicy",
    "comprehend:DescribeResourcePolicy"
  ],
  "Resource": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/foo/
version/*"
}
```

IAM 정책 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하세요. IAM 자격 증명에 정책을 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

AWS KMS 키 정책 설명

암호화된 모델을 공유하는 경우에 대한 권한을 추가해야 할 수 있습니다 AWS KMS. 이 요구 사항은 Amazon Comprehend에서 모델을 암호화하는 데 사용하는 KMS 키 유형에 따라 다릅니다.

AWS 소유 키는 AWS 서비스에서 소유하고 관리합니다. 를 사용하는 경우 AWS 소유 키에 대한 권한을 추가할 필요가 없으며이 섹션을 건너뛸 AWS KMS수 있습니다.

고객 관리형 키 는 사용자가 AWS 계정에서 생성, 소유 및 관리하는 키입니다. 고객 관리형 키를 사용하는 경우 KMS 키 정책에 설명을 추가해야 합니다.

정책 문은 하나 이상의 엔터티(예: 사용자 또는 계정)가 모델을 복호화하는 데 필요한 AWS KMS 작업을 수행할 수 있는 권한을 부여합니다.

조건 키를 사용하면 혼동된 대리자 문제를 방지하는 데 도움이 됩니다. 자세한 내용은 the section called “교차 서비스 혼동된 대리인 방지”를 참조하십시오.

정책에서 다음 조건 키를 사용하여 KMS 키에 액세스하는 개체를 검증합니다. 사용자가 모델을 가져올 때는 소스 모델 버전의 ARN이 조건과 일치하는지 AWS KMS 확인합니다. 정책에 조건을 포함하지 않으면 지정된 보안 주체가 KMS 키를 사용하여 모든 모델 버전을 해독할 수 있습니다.

- aws:SourceArn – 이 조건 키를 `kms:GenerateDataKey` 및 `kms:Decrypt` 작업과 함께 사용합니다.
- kms:EncryptionContext – 이 조건 키를 `kms:GenerateDataKey`, `kms:Decrypt`, `kms:CreateGrant` 작업과 함께 사용합니다.

다음 예제에서 정책은가 소유한 지정된 분류기 모델의 버전 1을 사용할 수 있는 AWS 계정 권한을 부여합니다 AWS 계정 444455556666111122223333.

Example 특정 분류기 모델 버전에 액세스하기 위한 KMS 키 정책

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "AWS":
                "arn:aws:iam::444455556666:root"
      },
      "Action": [
          "kms:Decrypt",
          "kms:GenerateDataKey"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": {
              "aws:SourceArn":
                "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/classifierName/version/1"
          }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
          "AWS":  "arn:aws:iam::444455556666:root"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
           "kms:EncryptionContext:aws:comprehend:arn":
              "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/classifierName/version/1"
        }
      }
    }
  ]
}
```

다음 예제 정책은의 사용자 ExampleUser AWS 계정 444455556666와의 ExampleRole이 Amazon Comprehend 서비스를 통해이 KMS 키에 액세스할 AWS 계정 123456789012 수 있는 권한을 부여합니다.

Example Amazon Comprehend 서비스에 대한 액세스를 허용하는 KMS 키 정책(대안 1).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "arn:aws:iam::444455556666:user/ExampleUser",
                "arn:aws:iam::123456789012:role/ExampleRole"
              ]
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "aws:SourceArn": "arn:aws:comprehend:*"

            }
        }
    },
    {
      "Effect": "Allow",
      "Principal": {
         "AWS": [
                "arn:aws:iam::444455556666:user/ExampleUser",
                "arn:aws:iam::123456789012:role/ExampleRole"
              ]
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
          "StringLike": {
              "kms:EncryptionContext:aws:comprehend:arn": "arn:aws:comprehend:*"
          }
```

```
      }
    }
  ]
}
```

다음 예제 정책은 이전 예제의 대체 구문을 사용하여 Amazon Comprehend 서비스를 통해이 KMS 키에 액세스할 수 있는 권한을 부여합니다 AWS 계정 444455556666.

Example Amazon Comprehend 서비스에 대한 액세스를 허용하는 KMS 키 정책(대안 2).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "AWS": "arn:aws:iam::444455556666:root"
      },
      "Action": [
          "kms:Decrypt",
          "kms:GenerateDataKey",
          "kms:CreateGrant"
      ],
      "Resource": "*",
      "Condition": {
          "StringLike": {
              "kms:EncryptionContext:aws:comprehend:arn": "arn:aws:comprehend:*"
          }
      }
    }
  ]
}
```

자세한 내용은AWS Key Management Service 개발자 안내서의 AWS KMS의 키 정책을 참조하세요.

사용자 지정 모델을 위한 리소스 기반 정책

다른의 Amazon Comprehend 사용자가 계정 AWS 에서 사용자 지정 모델을 AWS 계정 가져오려면 먼저 권한을 부여해야 합니다. 권한을 부여하려면 공유하려는 모델 버전에 리소스 기반 정책을 추가합니다. 리소스 기반 정책은 AWS의 리소스에 연결하는 IAM 정책입니다.

사용자 지정 모델 버전에 리소스 정책을 연결하면 이 정책이 모델 버전에서 `comprehend:ImportModel` 작업을 수행할 수 있는 권한을 사용자, 그룹 또는 역할에 부여합니다.

Example 사용자 지정 모델 버전을 위한 리소스 기반 정책

이 예제는 `Principal` 속성에 승인된 개체를 지정합니다. 리소스 “*”는 정책을 연결하는 특정 모델 버전을 나타냅니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "comprehend:ImportModel",
      "Resource": "*",
      "Principal": {
        "AWS": [
                "arn:aws:iam::111122223333:root",
                "arn:aws:iam::444455556666:user/ExampleUser",
                "arn:aws:iam::123456789012:role/ExampleRole"
         ]
      }
    }
  ]
}
```

사용자 지정 모델에 연결하는 정책의 경우 Amazon Comprehend에서 지원하는 유일한 작업은 `comprehend:ImportModel` 입니다.

차이점에 대한 자세한 내용은 IAM 사용 설명서의 자격 증명 기반 정책 및 리소스 기반 정책을 참조하세요.

1단계: 사용자 지정 모델에 리소스 기반 정책 추가

AWS Management Console AWS CLI또는 Amazon Comprehend API를 사용하여 리소스 기반 정책을 추가할 수 있습니다.

AWS Management Console

AWS Management Console에서 Amazon Comprehend를 사용할 수 있습니다.

리소스 기반 정책을 추가하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.

2. 왼쪽 탐색 메뉴의 사용자 지정에서 사용자 지정 모델이 포함된 페이지를 선택합니다.

   a. 사용자 지정 문서 분류기를 공유하는 경우 사용자 지정 분류를 선택합니다.

   b. 사용자 지정 개체 인식기를 공유하는 경우 사용자 지정 개체 인식을 선택합니다.

3. 모델 목록에서 모델을 선택하면 세부 정보 페이지가 표시됩니다.
4. 버전에서 공유하려는 모델 버전의 이름을 선택합니다.
5. 버전 세부 정보 페이지에서 태그, VPC 및 정책 탭을 선택합니다.
6. 리소스 기반 정책 섹션에서 편집을 선택합니다.
7. 신뢰 정책 편집 페이지에서 다음 작업을 수행합니다.

   a. 정책 이름에는 정책을 만든 후 해당 정책을 쉽게 알아볼 수 있는 이름을 입력합니다.

   b. 권한 부여에서 다음 개체 중 하나 이상을 지정하여 모델을 가져올 수 있도록 승인합니다.

필드	정의 및 예제
서비스 주체	이 모델 버전에 액세스할 수 있는 서비스의 서비스 주체 식별자. 예시: comprehend.amazonaws.com
AWS 계정 IDs	AWS 계정 는이 모델 버전에 액세스할 수 있습니다. 계정에 속한 모든 사용자에게 권한을 부여합니다. 예시: 111122223333, 123456789012
IAM 개체	이 모델 버전에 액세스할 수 있는 사용자 또는 역할을 위한 ARN. 예시: arn:aws:iam::111122223333:user/ExampleUser, arn:aws:iam::444455556666:role/ExampleRole

8. 공유에서 모델 버전의 ARN을 복사하여 사용자의 모델을 가져올 사람과 공유할 수 있습니다. 다른 사용자가 다른에서 사용자 지정 모델을 가져오는 경우 AWS 계정모델 버전 ARN이 필요합니다.
9. 저장(Save)을 선택합니다. Amazon Comprehend는 리소스 기반 정책을 생성하여 모델에 연결합니다.

AWS CLI

를 사용하여 사용자 지정 모델에 리소스 기반 정책을 추가하려면 PutResourcePolicy 명령을 AWS CLI 사용합니다. 이 명령은 다음 파라미터를 사용합니다.

- `resource-arn` – 사용자 지정 모델의 ARN(모델 버전 포함).
- `resource-policy` – 사용자 지정 모델에 연결할 리소스 기반 정책을 정의하는 JSON 파일.

  정책을 인라인 JSON 문자열로 제공할 수도 있습니다. 정책에 유효한 JSON을 제공하려면 속성 이름과 값을 큰 따옴표로 묶습니다. JSON 본문도 큰따옴표로 묶는 경우, 정책 내에 있는 큰따옴표를 이스케이프합니다.
- `policy-revision-id` – Amazon Comprehend가 업데이트하려는 정책에 할당한 개정 ID. 이전 버전이 없는 새 정책을 생성하는 경우에는 이 파라미터를 사용하지 마십시오. Amazon Comprehend가 사용자를 대신하여 개정 ID를 생성합니다.

Example **`put-resource-policy`** 명령을 사용하여 사용자 지정 모델에 리소스 기반 정책을 추가합니다.

이 예제는 PolicyFile.json이라는 JSON 파일에 정책을 정의하고, 이 정책을 모델에 연결합니다. 모델은 mycf1이라는 분류기의 버전 v2입니다.

```
$ aws comprehend put-resource-policy \
> --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/mycf1/version/v2 \
> --resource-policy file://policyFile.json \
> --policy-revision-id revision-id
```

리소스 정책의 JSON 파일에는 다음 내용이 포함되어 있습니다.

- 작업 –이 정책은 지정된 보안 주체에게 `comprehend:ImportModel` 사용 권한을 부여합니다.
- 리소스 – 사용자 지정 모델의 ARN. 리소스 “*”는 `put-resource-policy` 명령에서 지정하는 모델 버전을 나타냅니다.
- 보안 주체 - 정책은 444455556666jane의 AWS 계정 사용자와 AWS 계정 123456789012의 모든 사용자에게 권한을 부여합니다.

```
{
```

```
"Version":"2012-10-17",
 "Statement":[
    {"Sid":"ResourcePolicyForImportModel",
     "Effect":"Allow",
     "Action":["comprehend:ImportModel"],
     "Resource":"*",
     "Principal":
         {"AWS":
            ["arn:aws:iam::444455556666:user/jane",
             "123456789012"]
         }
   }
 ]
}
```

Amazon Comprehend API

Amazon Comprehend API를 사용하여 사용자 지정 모델에 리소스 기반 정책을 추가하려면 PutResourcePolicy API 작업을 사용하십시오.

모델을 생성하는 API 요청에서 사용자 지정 모델에 정책을 추가할 수도 있습니다. 이렇게 하려면 CreateDocumentClassifier 또는 CreateEntityRecognizer 요청을 제출할 때 ModelPolicy 파라미터에 대한 정책 JSON을 제공해야 합니다.

2단계: 다른 사용자가 가져오는 데 필요한 세부 정보를 제공하십시오.

이제 사용자 지정 모델에 리소스 기반 정책을 추가했으므로 다른 Amazon Comprehend 사용자에게 모델을 자신의 AWS 계정으로 가져올 수 있는 권한을 부여했습니다. 하지만 가져오려면 먼저 다음 세부 정보를 제공해야 합니다.

- 모듈 버전의 Amazon 리소스 이름(ARN) 반환.
- 모델이 포함된 AWS 리전 . 모델을 가져오는 사람은 누구나 동일한 AWS 리전 을 사용해야 합니다.
- 모델이 암호화되는지 여부와 암호화되는 경우 사용하는 AWS KMS 키 유형: AWS 소유 키 또는 고객 관리형 키.
- 모델이 고객 관리 키로 암호화된 경우 KMS 키의 ARN을 제공해야 합니다. 모델을 가져오는 사람은 누구나 자신의 AWS 계정에 IAM 서비스 역할의 ARN을 포함해야 합니다. 이 역할은 Amazon Comprehend가 KMS 키를 사용하여 모델을 가져오는 동안 모델을 복호화할 권한을 부여합니다.

다른 사용자가 모델을 가져오는 방법에 대한 자세한 정보는 다른에서 사용자 지정 모델 가져오기 AWS 계정를 참조하세요.

다른에서 사용자 지정 모델 가져오기 AWS 계정

Amazon Comprehend에서는 다른에 있는 사용자 지정 모델을 가져올 수 있습니다 AWS 계정. 모델을 가져오면 계정에 새 사용자 지정 모델이 생성됩니다. 새 사용자 지정 모델은 가져온 모델을 완전히 학습한 복제본입니다.

주제

시작하기 전 준비 사항

다른에서 사용자 지정 모델을 가져오기 전에 모델을 공유한 사람이 다음을 수행하는지 AWS 계정확인합니다.

- 사용자에게 가져오기를 수행할 수 있는 권한 부여: 이 권한은 모델 버전에 연결된 리소스 기반 정책에서 부여됩니다. 자세한 내용은 사용자 지정 모델을 위한 리소스 기반 정책을 참조하십시오.
- 다음 정보 제공:
  - 모듈 버전의 Amazon 리소스 이름(ARN) 반환.
  - 모델이 포함된 AWS 리전 . 가져올 AWS 리전 때 동일한를 사용해야 합니다.
  - 모델이 AWS KMS 키로 암호화되는지 여부와 암호화되는 경우 사용되는 키 유형입니다.

모델이 암호화된 경우 사용되는 KMS 키 유형에 따라 다음과 같은 추가 단계를 수행해야 할 수 있습니다.

- AWS 소유 키— 이 유형의 KMS 키는 AWS에서 소유하고 관리합니다. 모델이 로 암호화된 경우 추가 단계가 필요하지 AWS 소유 키않습니다.
- 고객 관리형 키 -이 유형의 KMS 키는 AWS 고객이 자신의에서 생성, 소유 및 관리합니다 AWS 계정. 모델이 고객 관리 키로 암호화된 경우 모델을 공유한 사람은 다음을 수행해야 합니다.
  - 모델 복호화를 승인합니다. 이 권한은 고객 관리 키에 대한 KMS 키 정책에서 부여됩니다. 자세한 내용은 AWS KMS 키 정책 설명를 참조하십시오.
  - 고객 관리형 키의 ARN을 제공합니다. IAM 서비스 역할을 생성할 때 이 ARN을 사용합니다. 이 역할은 Amazon Comprehend가 KMS 키를 사용하여 모델을 복호화할 권한을 부여합니다.

필수 권한

사용자 지정 모델을 가져오려면 먼저 사용자 또는 관리자가 AWS Identity and Access Management (IAM)에서 필요한 작업을 승인해야 합니다. Amazon Comprehend 사용자는 IAM 정책문을 통해 가져오기 허가를 받아야 합니다. 가져오기 중에 암호화 또는 복호화가 필요한 경우 Amazon Comprehend에 필요한 AWS KMS 키를 사용할 수 있는 권한이 부여되어야 합니다.

IAM 정책문

다음 예제와 같이 사용자, 그룹 또는 역할에는 `ImportModel` 작업을 허용하는 정책이 연결되어 있어야 합니다.

Example 사용자 지정 모델을 가져오기 위한 IAM 정책

```
{
  "Effect": "Allow",
  "Action": [
    "comprehend:ImportModel"
  ],
  "Resource": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/foo/version/*"
}
```

IAM 정책 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하세요. IAM 자격 증명에 정책을 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

AWS KMS 암호화를 위한 IAM 서비스 역할

사용자 지정 모델을 가져올 때 Amazon Comprehend가 다음 경우 중 하나에서 AWS KMS 키를 사용하도록 권한을 부여해야 합니다.

- 고객 관리형 키로 암호화된 사용자 지정 모델을 가져오고 있습니다 AWS KMS. 이 경우 Amazon Comprehend는 KMS 키에 액세스해야 가져오기 중에 모델을 복호화할 수 있습니다.
- 가져오기로 생성한 새 사용자 지정 모델을 암호화하고 고객 관리 키를 사용하려고 합니다. 이 경우 Amazon Comprehend가 새 모델을 암호화할 수 있으려면 KMS 키에 대한 액세스 권한이 필요합니다.

Amazon Comprehend가 이러한 AWS KMS 키를 사용하도록 권한을 부여하려면 IAM 서비스 역할을 생성합니다. 이 유형의 IAM 역할은 AWS 서비스가 사용자를 대신하여 다른 서비스의 리소스에 액세스할

수 있도록 허용합니다. 서비스 역할에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 권한을 위임할 역할 생성을 참조하세요.

Amazon Comprehend 콘솔을 사용하여 가져오면 Amazon Comprehend에서 자동으로 서비스 역할을 생성하도록 할 수 있습니다. 그렇지 않으면 가져오기 전에 IAM에서 서비스 역할을 생성해야 합니다.

다음 예제와 같이 IAM 서비스 역할에는 권한 정책 및 신뢰 정책이 있어야 합니다.

Example 권한 정책

다음 권한 정책은 Amazon Comprehend가 사용자 지정 모델을 암호화하고 해독하는 데 사용하는 AWS KMS 작업을 허용합니다. 이는 두 개의 KMS 키에 대한 액세스 권한을 부여합니다.

- 가져올 모델이 AWS 계정 포함된 KMS 키 하나가에 있습니다. 이는 모델을 암호화하는 데 사용되었으며 Amazon Comprehend는 이를 사용하여 가져오기 중에 모델을 복호화합니다.
- 다른 KMS 키는 모델을 AWS 계정 가져오는에 있습니다. Amazon Comprehend는 이 키를 사용하여 가져오기로 생성된 새 사용자 지정 모델을 암호화합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:CreateGrant"
        ],
        "Resource": [
            "arn:aws:kms:us-west-2:111122223333:key/key-id",
            "arn:aws:kms:us-west-2:444455556666:key/key-id"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDatakey"
        ],
        "Resource": [
            "arn:aws:kms:us-west-2:111122223333:key/key-id",
            "arn:aws:kms:us-west-2:444455556666:key/key-id"
        ],
        "Condition": {
```

```
            "StringEquals": {
                "kms:ViaService": [
                    "s3.us-west-2.amazonaws.com"
                ]
            }
        }
    }
  ]
}
```

Example 신뢰 정책

다음 신뢰 정책은 Amazon Comprehend가 역할을 맡고 권한을 얻는 것을 허용합니다 이를 통해 `comprehend.amazonaws.com` 서비스 주체가 `sts:AssumeRole` 작업을 수행할 수 있습니다. 혼동된 대리인 방지를 위해 하나 이상의 글로벌 조건 컨텍스트 키를 사용하여 권한 범위를 제한합니다. `aws:SourceAccount`의 경우 모델을 가져오는 사용자의 계정 ID를 지정하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "comprehend.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "444455556666"
        }
      }
    }
  ]
}
```

사용자 지정 모델 가져오기

AWS Management Console AWS CLI또는 Amazon Comprehend API를 사용하여 사용자 지정 모델을 가져올 수 있습니다.

AWS Management Console

AWS Management Console에서 Amazon Comprehend를 사용할 수 있습니다.

사용자 지정 모델을 가져오려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 탐색 메뉴의 사용자 지정에서 가져오려는 모델 유형 페이지를 선택합니다.

   a. 사용자 지정 문서 분류기를 가져오는 경우 사용자 지정 분류를 선택합니다.

   b. 사용자 지정 개체 인식기를 가져오는 경우 사용자 지정 개체 인식을 선택합니다.
3. 버전 가져오기를 선택합니다.
4. 모델 버전 가져오기 페이지에서 다음 세부 정보를 입력합니다.

   - 모델 버전 ARN — 가져올 모델 버전의 ARN입니다.
   - 모델 이름 - 가져오기 시 생성되는 새 모델의 사용자 지정 이름입니다.
   - 모델 이름 - 가져오기 시 생성되는 새 모델의 사용자 지정 이름입니다.
5. 모델 암호화에서 가져오기로 생성하는 새 사용자 지정 모델을 암호화하는 데 사용할 KMS 키 유형을 선택합니다.

   - AWS 소유 키 사용 - Amazon Comprehend는 사용자를 대신하여 생성, 관리 및 사용되는 in AWS Key Management Service (AWS KMS) 키를 사용하여 모델을 암호화합니다 AWS.
   - 다른 AWS KMS 키 선택(고급) - Amazon Comprehend는 관리하는 고객 관리형 키를 사용하여 모델을 암호화합니다 AWS KMS.

     이 옵션을 선택하는 경우에 있는 KMS 키를 선택하거나 키 AWS 계정생성을 선택하여 새 AWS KMS 키를 생성합니다.
6. 서비스 액세스 섹션에서 Amazon Comprehend에서 다음 작업에 필요한 모든 AWS KMS 키에 대한 액세스 권한을 부여하십시오.

   - 가져오는 사용자 지정 모델을 복호화합니다.
   - 가져오기로 생성한 새 사용자 지정 모델을 암호화합니다.

   Amazon Comprehend가 KMS 키를 사용하도록 허용하는 IAM 서비스 역할로 액세스 권한을 부여합니다.

   서비스 역할에 대해 다음 중 하나를 수행합니다.

   - 사용하려는 기존 서비스 역할이 있는 경우, 기존 IAM 역할 사용을 선택합니다. 그런 다음 역할 이름에서 해당 역할을 선택합니다.

- Amazon Comprehend에서 역할을 생성하도록 하려면 IAM 역할 생성을 선택합니다.

7. Amazon Comprehend가 역할을 생성하도록 선택한 경우 다음을 수행합니다.

   a. 역할 이름의 경우, 역할 이름 접미사를 입력하면 나중에 역할을 쉽게 알아볼 수 있습니다.

   b. 소스 KMS 키 ARN의 경우, 가져오려는 모델을 암호화하는 데 사용되는 KMS 키의 ARN을 입력합니다. Amazon Comprehend는 가져오기 중에 이 키를 사용하여 모델을 복호화합니다.

8. (선택 사항) 태그 섹션에서 가져오기를 통해 생성하는 새 사용자 지정 모델에 태그를 추가할 수 있습니다. 사용자 지정 모델 태그 지정에 대한 자세한 정보는 새 리소스에 태그 지정을 참조하세요.

9. 확인을 선택합니다.

AWS CLI

AWS CLI에서 명령을 실행하여 Amazon Comprehend를 사용할 수 있습니다.

Example 모델 가져오기 명령

사용자 지정 모델을 가져오려면 `import-model` 명령을 사용하십시오.

```
$ aws comprehend import-model \
> --source-model arn:aws:comprehend:us-west-2:111122223333:document-classifier/foo/
version/bar \
> --model-name importedDocumentClassifier \
> --version-name versionOne \
> --data-access-role-arn arn:aws:iam::444455556666:role/comprehendAccessRole \
> --model-kms-key-id kms-key-id
```

이 예제는 다음 파라미터를 사용합니다.

- `source-model`— 가져올 사용자 지정 모델의 ARN입니다.
- `model-name`— 가져오기 시 생성되는 새 모델의 사용자 지정 이름입니다.
- `version-name`— 가져오기 시 생성되는 새 모델 버전의 사용자 지정 이름입니다.
- `data-access-role-arn` - Amazon Comprehend가 사용자 지정 모델을 암호화하거나 복호화하는 데 필요한 AWS KMS 키를 사용할 수 있도록 허용하는 IAM 서비스 역할의 ARN입니다.
- `model-kms-key-id`— Amazon Comprehend가 이 가져오기로 생성한 사용자 지정 모델을 암호화하는 데 사용하는 KMS 키의 ARN 또는 ID입니다. 이 키는의 AWS KMS 에 있어야 합니다 AWS 계정.

Amazon Comprehend API

Amazon Comprehend API를 통해 사용자 지정 모델을 가져오려면 ImportModel API 작업을 사용하십시오.

플라이 휠

Amazon Comprehend 플라이 휠은 시계열 사용자 정의 모델 개선 프로세스를 간소화합니다. 플라이 휠을 사용하여 새 사용자 정의 모델 버전의 학습 및 평가와 관련된 작업을 오케스트레이션 할 수 있습니다. 플라이 휠은 사용자 정의 분류 및 사용자 정의 개체 인식을 위해 일반 텍스트 사용자 정의 모델을 지원합니다.

주제

플라이 휠 개요

플라이 휠은 사용자 정의 모델의 새 버전에 대한 학습 및 평가를 오케스트레이션하는 Amazon Comprehend 리소스입니다. 플라이 휠을 생성하여 기존의 학습된 모델을 사용하거나 Amazon Comprehend에서 플라이 휠용 새 모델을 생성하고 학습시킬 수 있습니다. 사용자 정의 분류 또는 사용자 정의 개체 인식에 일반 텍스트 사용자 정의 모델과 함께 플라이 휠을 사용하십시오.

Amazon Comprehend 콘솔 또는 API를 사용하여 플라이 휠을 구성하고 관리할 수 있습니다. AWS CloudFormation을 사용하여 플라이 휠을 구성할 수도 있습니다.

플라이 휠을 생성하면 Amazon Comprehend가 사용자 계정에 데이터 레이크를 생성합니다. 데이터 레이크는 모델의 모든 버전에 대한 학습 데이터 및 테스트 데이터 같은 모든 플라이 휠 데이터를 저장하고 관리합니다.

활성 모델 버전을 추론 작업 또는 Amazon Comprehend 엔드포인트에 사용할 플라이 휠 모델 버전으로 설정합니다. 처음에는 플라이 휠에 한 가지 버전의 모델이 포함되어 있습니다. 시계열에 따라 새 모델 버전을 학습시켜 성능이 가장 좋은 버전을 활성 모델 버전으로 선택합니다. 사용자가 플라이 휠

ARN을 지정하여 추론 작업을 실행하면 Amazon Comprehend는 플라이 휠의 활성 모델 버전을 사용하여 작업을 실행합니다.

주기적으로 모델의 레이블이 지정된 새 데이터 (학습 데이터 또는 테스트 데이터)를 얻습니다. 하나 이상의 데이터 세트를 생성하여 플라이 휠에서 새 데이터를 사용할 수 있도록 합니다. 데이터 세트에는 플라이 휠과 관련된 사용자 정의 모델을 학습시키거나 테스트하기 위한 입력 데이터가 포함됩니다. Amazon Comprehend는 입력 데이터를 플라이 휠의 데이터 레이크에 업로드합니다.

새 데이터 세트를 사용자 정의 모델에 통합하려면 플라이 휠 반복을 생성하고 실행합니다. 플라이 휠 반복은 새 데이터 세트를 사용하여 활성 모델 버전을 평가하고 새 모델 버전을 학습시키는 워크플로입니다. 기존 모델 버전과 새 모델 버전의 지표를 기반으로 새 모델 버전을 활성 버전으로 승격할지 여부를 결정할 수 있습니다.

플라이 휠 활성 모델 버전을 사용하여 사용자 정의 분석 (실시간 또는 비동기 작업) 을 실행할 수 있습니다. 플라이 휠 모델을 실시간 분석에 사용하려면 플라이 휠용 엔드포인트를 생성해야 합니다.

플라이 휠을 사용해도 추가 요금이 부과되지 않습니다. 하지만 플라이 휠 반복을 실행하면 새 모델 버전 학습 및 모델 데이터 저장 표준 요금이 발생합니다. 요금에 대한 자세한 내용은 Amazon Comprehend 요금을 참조하세요.

주제

- 플라이 휠 데이터 세트
- 플라이 휠 생성
- 플라이 휠 상태
- 플라이 휠 반복

플라이 휠 데이터 세트

레이블이 지정된 새 데이터를 플라이 휠에 추가하려면 데이터 세트를 만들어야 합니다. 각 데이터 세트를 학습 데이터 또는 테스트 데이터로 구성합니다. 데이터 세트를 특정 플라이 휠 및 사용자 정의 모델과 연결합니다.

데이터 세트를 생성한 후 Amazon Comprehend는 데이터를 플라이 휠의 데이터 레이크에 업로드합니다. 자세한 내용은 플라이 휠 데이터 레이크를 참조하십시오.

플라이 휠 생성

플라이 휠을 생성할 때 플라이 휠을 기존 학습 모델에 연결하거나 플라이 휠을 사용하여 새 모델을 만들 수 있습니다.

기존 모델을 사용하여 플라이 휠을 생성할 때는 활성 모델 버전을 지정합니다. Amazon Comprehend는 모델의 학습 데이터와 테스트 데이터를 플라이 휠의 데이터 레이크에 복사합니다. 모델 학습 및 테스트 데이터가 모델을 생성할 때와 동일한 Amazon S3 위치에 있는지 확인하십시오.

새 모델용 플라이 휠을 생성하려면 플라이 휠을 생성할 때 학습 데이터용 데이터 세트 (및 테스트 데이트용 선택적 데이터 세트) 를 제공해야 합니다. 플라이 휠을 실행하여 첫 번째 플라이 휠 반복을 생성하면 플라이 휠이 새 모델을 학습시킵니다.

사용자 정의 모델을 학습할 때는 모델이 인식할 사용자 정의 레이블 (사용자 정의 분류) 또는 사용자 정의 개체 (사용자 정의 개체 인식) 의 목록을 지정합니다. 사용자 정의 레이블/개체와 관련된 다음과 같은 중요한 사항을 기록해 둡니다.

- 새 모델의 플라이 휠을 생성할 때 플라이 휠 생성 중에 제공하는 레이블/개체 목록이 플라이 휠의 최종 목록이 됩니다.
- 기존 모델에서 플라이 휠을 생성하면 해당 모델과 관련된 레이블/개체 목록이 플라이 휠의 최종 목록이 됩니다.
- 새 데이터 세트를 플라이 휠과 연결하고 해당 데이터 세트에 추가 레이블/개체가 포함된 경우 Amazon Comprehend는 새 레이블/개체를 무시합니다.
- DescribeFlyWheel API 작업을 사용하여 플라이 휠의 레이블/개체 목록을 검토할 수 있습니다.

> **Note**
>
> 사용자 정의 분류의 경우 Amazon Comprehend는 플라이 휠 상태가 활성화 된 후에 레이블 목록을 채웁니다. DescribeFlyWheel API 작업을 호출하기 전에 플라이 휠이 활성화될 때까지 기다리십시오.

플라이 휠 상태

플라이 휠은 다음 상태 사이를 전환합니다.

- 생성 중 - Amazon Comprehend에서 플라이 휠 리소스를 생성하고 있습니다. 플라이 휠에서 `DescribeFlywheel`과 같은 읽기 작업을 수행할 수 있습니다.
- 활성 - 플라이 휠이 활성 상태입니다. 플라이 휠 반복이 진행 중인지 확인하고 반복 상태를 볼 수 있습니다. 플라이 휠에 대한 읽기 작업 및 `DeleteFlywheel`와 `UpdateFlywheel` 같은 작업을 수행할 수 있습니다.
- 업데이트 중 - Amazon Comprehend에서 플라이 휠을 업데이트하고 있습니다. 플라이 휠에서 읽기 작업을 수행할 수 있습니다.

- 삭제 중 - Amazon Comprehend에서 플라이 휠을 삭제하고 있습니다. 플라이 휠에서 읽기 작업을 수행할 수 있습니다.
- 실패 - 플라이 휠 생성 작업이 실패했습니다.

Amazon Comprehend에서 플라이 휠을 삭제한 후에도 플라이 휠 데이터 레이크의 모든 모델 데이터에 계속 액세스할 수 있습니다. Amazon Comprehend는 플라이 휠 리소스를 관리하는 데 필요한 모든 내부 메타데이터를 삭제합니다. Amazon Comprehend는 이 플라이 휠과 관련된 데이터 세트도 삭제합니다 (모델 데이터는 데이터 레이크에 저장됨).

플라이 휠 반복

플라이 휠 모델에 대한 새 학습 또는 테스트 데이터를 얻으면 새 데이터 세트를 하나 이상 생성하여 플라이 휠의 데이터 레이크에 새 데이터를 업로드합니다.

그런 다음 플라이 휠을 실행하여 새 플라이 휠 반복을 생성합니다. 플라이 휠 반복은 새 데이터를 사용하여 현재 활성 모델 버전을 평가하고 그 결과를 데이터 레이크에 저장합니다. 또한 플라이 휠은 새 모델 버전을 생성하고 학습시킵니다.

새 모델이 현재 활성 모델 버전보다 더 나은 성능을 보이면 새 모델 버전을 활성 모델 버전으로 승격할 수 있습니다. 콘솔 또는 UpdateFlyWheel API 작업을 사용하여 활성 모델 버전을 업데이트할 수 있습니다.

플라이 휠 데이터 레이크

플라이 휠을 생성하면 Amazon Comprehend는 모델 버전에 필요한 입력 및 출력 데이터와 같은 모든 플라이 휠 데이터를 포함하는 데이터 레이크를 사용자 계정에 생성합니다.

Amazon Comprehend에서 데이터 레이크 생성 시 지정한 Amazon S3 위치에 데이터 레이크를 생성합니다. 위치를 Amazon S3 버킷으로 지정하거나 Amazon S3 버킷의 새 폴더로 지정할 수 있습니다.

데이터 레이크 폴더 구조

Amazon Comprehend는 데이터 레이크를 생성할 때 Amazon S3 위치에 다음과 같은 폴더 구조를 설정합니다.

> **Warning**
>
> Amazon Comprehend는 데이터 레이크 폴더 구성 및 콘텐츠를 관리합니다. 항상 Amazon Comprehend API 작업을 사용하여 데이터 레이크 폴더를 수정하십시오. 그렇지 않으면 플라이휠이 제대로 작동하지 않을 수 있습니다.

```
Document Pool
Annotations Pool
Staging
Model Datasets
  (data for each version of the model)
  VersionID-1
    Training
    Test
    ModelStats
  VersionID-2
    Training
    Test
    ModelStats
```

모델 버전의 학습 평가를 보려면 다음 단계를 수행하십시오.

1. 데이터 레이크의 루트 수준에서 모델 데이터 세트라는 폴더를 엽니다. 이 폴더에는 모델의 각 버전에 대한 하위 폴더가 있습니다.
2. 원하는 모델 버전의 폴더를 엽니다.
3. ModelStats라는 이름의 폴더를 열어 모델에 대한 통계를 확인합니다.

데이터 레이크 관리

Amazon Comprehend에서 다음 작업을 자동으로 수행하여 데이터 레이크를 관리합니다.

- 데이터 레이크의 폴더 구조를 정의하고 데이터 세트를 적절한 폴더에 집어 넣습니다.
- 모델 학습에 필요한 입력 문서 (예: 텍스트 파일, 주석 파일) 를 관리합니다.
- 모델의 각 버전과 관련된 학습 및 평가 결과 데이터를 관리합니다.
- 데이터 레이크에 저장된 파일의 암호화를 관리합니다.

Amazon Comprehend는 데이터 레이크에 대한 모든 데이터 생성 및 업데이트 작업을 수행합니다. 데이터 레이크의 데이터에 대한 전체 액세스 권한은 그대로 유지됩니다. 예시:

- 데이터 레이크의 콘텐츠에 완전히 액세스할 수 있습니다.
- 플라이 휠을 삭제한 후에도 데이터 레이크는 계속 사용할 수 있습니다.
- 데이터 레이크가 포함된 Amazon S3 버킷의 액세스 로그를 구성할 수 있습니다.
- 데이터에 대한 암호화 키를 제공할 수 있습니다. 플라이 휠을 생성할 때 이 정보를 지정합니다.

다음 모범 사례를 따르는 것이 좋습니다.

- 자체 폴더나 파일을 데이터 레이크에 수작업으로 추가하지 마십시오. 데이터 레이크에 있는 파일을 수정하거나 삭제하지 마십시오.
- 데이터 레이크에 데이터를 추가하거나 수정할 때는 항상 Amazon Comprehend 생성 및 업데이트 작업을 사용하십시오. 예를 들어 `CreateDataset`을 사용하여 모델 버전 학습 또는 테스트를 하고 `StartFlywheelIteration`을 사용하여 모델 버전에 대한 평가 데이터를 생성합니다.
- 데이터 레이크 구조는 시간이 지나면서 발전할 수 있습니다. 데이터 레이크 구조를 명시적으로 사용하는 다운스트림 스크립트나 프로그램을 만들지 마십시오.
- 플라이 휠에 데이터 레이크 위치를 제공할 때는 모든 플라이 휠과 관련된 데이터에 대한 공통 접두사를 만들거나 플라이 휠마다 다른 접두사를 사용하는 것이 좋습니다. 한 플라이 휠의 전체 데이터 레이크 경로를 다른 플라이 휠의 접두사로 사용하지 않는 것이 좋습니다.

IAM 정책 및 권한

다음의 목적으로 플라이휠을 사용하기 위한 정책 및 권한을 구성합니다.

- 사용자의 플라이휠 작업 액세스를 위한 the section called “IAM 사용자 권한 구성”.
- (선택 사항) the section called “AWS KMS 키에 대한 권한 구성” 데이터 레이크.
- Amazon Comprehend에 데이터 레이크에 액세스할 권한을 부여하는 the section called “데이터 액세스 역할 생성”.

IAM 사용자 권한 구성

플라이 휠 기능을 사용하려면 AWS Identity and Access Management (IAM) 자격 증명(사용자, 그룹 및 역할)에 적절한 권한 정책을 추가합니다.

다음 예제는 데이터 세트 생성, 플라이휠 생성 및 관리, 플라이휠 실행에 대한 권한 정책을 보여줍니다.

Example 플라이휠 관리를 위한 IAM 정책

```
{
  "Effect": "Allow",
  "Action": [
    "comprehend:CreateFlywheel",
    "comprehend:DeleteFlywheel",
    "comprehend:UpdateFlywheel",
    "comprehend:ListFlywheels",
    "comprehend:DescribeFlywheel",
    "comprehend:CreateDataset",
    "comprehend:DescribeDataset",
    "comprehend:ListDatasets",
    "comprehend:StartFlywheelIteration",
    "comprehend:DescribeFlywheelIteration",
    "comprehend:ListFlywheelIterationHistory"
  ],
  "Resource": "*"
}
```

Amazon Comprehend에 대한 IAM 정책 생성에 대한 내용은 Amazon Comprehend에서 IAM을 사용하는 방법을 참조하세요.

AWS KMS 키에 대한 권한 구성

데이터 레이크의 데이터에 AWS KMS 키를 사용하는 경우 필요한 권한을 설정합니다. 자세한 내용은 KMS 암호화를 사용하는 데 필요한 권한을 참조하세요.

데이터 액세스 역할 생성

Amazon Comprehend가 데이터 레이크의 플라이휠 데이터에 액세스할 수 있도록 IAM에서 데이터 액세스 역할을 생성합니다. 콘솔을 사용하여 플라이휠을 생성하는 경우 시스템은 이 목적을 위해 선택적으로 새 역할을 생성할 수 있습니다. 자세한 내용은 비동기 작업에 필요한 역할 기반 권한을 참조하십시오.

콘솔을 사용하여 플라이 휠 구성

Amazon Comprehend 콘솔을 사용하여 플라이 휠을 생성, 업데이트 및 삭제할 수 있습니다.

플라이 휠을 생성하면 Amazon Comprehend는 각 모델 버전의 학습 데이터 및 테스트 데이터와 같이 플라이 휠에 필요한 모든 데이터를 보관하기 위해 데이터 레이크를 생성합니다.

플라이 휠을 삭제해도 Amazon Comprehend는 데이터 레이크 또는 플라이 휠과 관련된 모델을 삭제하지 않습니다.

새 플라이 휠을 생성하기 전에 플라이 휠 생성 섹션의 정보를 검토합니다.

주제

- 플라이 휠 생성
- 플라이 휠 업데이트
- 플라이 휠 삭제

플라이 휠 생성

플라이 휠을 생성할 때의 필수 구성 필드는 플라이 휠이 기존 사용자 정의 모델용인지 새 모델용인지에 따라 달라집니다.

플라이 휠을 생성하려면

1. 에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 플라이 휠을 선택합니다.
3. 플라이 휠 테이블에서 새 플라이 휠 생성을 선택합니다.
4. 플라이 휠 이름 에 플라이 휠의 이름을 입력합니다.
5. (선택 사항) 기존 모델용 플라이 휠을 생성하려면 활성 모델 버전에서 필드를 구성하십시오.
   a. 모델의 드롭다운 목록에서 모델을 선택합니다.
   b. 버전의 드롭다운 목록에서 모델 버전을 선택합니다.
6. (선택 사항) 플라이 휠용 새 분류기 모델을 만들려면 사용자 정의 모델 유형에서 사용자 정의 분류를 선택하고 다음 단계에서 파라미터를 구성합니다.
   a. 언어에서 모델의 언어를 선택합니다.
   b. 분류기 모드에서 단일 레이블 모드 또는 다중 레이블 모드를 선택합니다.
   c. 사용자 정의 레이블에서 모델 학습에 사용할 사용자 정의 레이블을 하나 이상 입력합니다. 각 레이블은 입력 학습 데이터에 있는 클래스 중 하나와 일치해야 합니다.

7. (선택 사항) 플라이 휠용 새 개체 인식 모델을 만들려면 사용자 정의 모델 유형에서 사용자 정의 개체 인식을 선택하고 다음 단계에서 파라미터를 구성합니다.

   a. 언어에서 모델의 언어를 선택합니다.

   b. 사용자 정의 개체 유형에서 모델 교육에 사용할 사용자 정의 개체를 최대 25개까지 입력합니다. 각 레이블은 입력 학습 데이터에 있는 개체 유형 중 하나와 일치해야 합니다.

      레이블을 두 개 이상 만들려면 다음 단계를 여러 번 수행하십시오.

      i. 사용자 정의 레이블을 입력합니다. 레이블은 모두 대문자여야 합니다. 밑줄을 레이블의 단어 구분 기호로 사용하십시오.

      ii. 유형 추가를 선택합니다.

      추가한 레이블 중 하나를 제거하려면 레이블 이름 오른쪽에 있는 X를 선택합니다.

8. 볼륨 암호화, 모델 암호화, 데이터 레이크 암호화에 대한 선택 항목을 구성하십시오. 각 항목에 대해 AWS 소유 KMS 키를 사용할지 아니면 사용할 권한이 있는 키를 사용할지 선택합니다.

   - AWS 소유 KMS 키를 사용하는 경우 추가 파라미터가 없습니다.
   - 다른 기존 키를 사용하는 경우 KMS 키 ARN에 키 ID의 ARN을 입력합니다.
   - 새 키를 생성하려면 AWS KMS 키 생성을 선택합니다.

   KMS 키와 관련 암호화의 생성 및 사용에 대한 자세한 내용은 AWS Key Management Service를 참조하세요.

   a. 볼륨 암호화 키를 구성합니다. Amazon Comprehend는이 키를 사용하여 작업이 처리되는 동안 스토리지 볼륨의 데이터를 암호화합니다. AWS 소유 KMS 키를 사용할지 아니면 사용할 권한이 있는 키를 사용할지 선택합니다.

   b. 모델 암호화 키 구성. Amazon Comprehend는 이 키를 사용하여 이 모델 버전의 모델 데이터를 암호화합니다.

9. 데이터 레이크 위치 구성. 자세한 내용은 데이터 레이크 관리를 참조하십시오.

10. (선택 사항) 데이터 레이크 암호화 키를 구성. Amazon Comprehend는 이 키를 사용하여 데이터 레이크에 있는 모든 파일을 암호화합니다.

11. (선택 사항) VPC 설정 구성. VPC에 VPC ID를 입력하거나 드롭다운 목록에서 ID를 선택합니다.

    1. 서브넷에서 서브넷을 선택합니다. 첫 번째 서브넷을 선택한 후 추가 서브넷을 선택할 수 있습니다.

2. 보안 그룹을 지정한 경우, 보안 그룹에서 사용할 보안 그룹을 선택합니다. 첫 번째 보안 그룹을 선택한 후 추가 보안 그룹을 선택할 수 있습니다.

12. 서비스 액세스 권한 구성.

    1. 기존 IAM 역할 사용을 선택하는 경우 드롭다운 목록에서 역할 이름을 선택합니다.
    2. IAM 역할 생성을 선택하면 Amazon Comprehend에서 새 역할을 생성합니다. 콘솔에 Amazon Comprehend가 해당 역할에 대해 구성하는 권한이 표시됩니다. 역할 이름에 해당 역할의 이름을 입력합니다.

13. (선택 사항) 태그 설정 구성. 태그를 추가하려면 태그에 키-값 페어를 입력합니다. 태그 추가를 선택합니다. 플라이 휠을 생성하기 전에 이 페어를 제거하려면 태그 제거를 선택합니다. 자세한 내용은 리소스에 태그 지정을 참조하십시오.

14. 생성을 선택합니다.

플라이 휠 업데이트

플라이 휠 생성 시에만 플라이 휠 이름, 데이터 레이크 위치, 모델 유형 및 모델 구성을 구성할 수 있습니다.

플라이 휠을 업데이트할 때 모델 유형 및 구성 옵션이 현재 모델과 같으면 다른 모델을 지정할 수 있습니다. 새 활성 모델 버전을 구성할 수 있습니다. 또한 암호화 세부 정보, 서비스 액세스 권한 및 VPC 설정을 업데이트할 수 있습니다.

플라이 휠을 업데이트하려면

1. 에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 플라이 휠을 선택합니다.
3. 플라이 휠 테이블에서 업데이트할 플라이 휠을 선택합니다.
4. 활성 모델 버전의 모델 드롭다운 목록에서 모델을 선택하고 모델 버전을 선택합니다.

   양식에는 모델 유형과 모델 구성이 채워져 있습니다.
5. (선택 사항) 볼륨 암호화 및 모델 암호화 설정 구성.
6. (선택 사항) 데이터 레이크 암호화 설정 구성.
7. 서비스 액세스 권한 구성.
8. (선택 사항) VPC 설정 구성.
9. (선택 사항) 태그 설정 구성.

10. 저장을 선택합니다.

플라이 휠 삭제

플라이 휠 삭제

1. 에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 플라이 휠을 선택합니다.
3. 플라이 휠 테이블에서 삭제할 플라이 휠을 선택합니다.
4. Delete(삭제)를 선택합니다.

API를 사용한 플라이 휠 구성

Amazon Comprehend API를 사용하여 플라이 휠을 생성, 업데이트 및 삭제할 수 있습니다.

플라이 휠을 생성하면 Amazon Comprehend는 각 모델 버전의 학습 데이터 및 테스트 데이터와 같이 플라이 휠에 필요한 모든 데이터를 보관하기 위해 데이터 레이크를 생성합니다.

플라이 휠을 삭제해도 Amazon Comprehend는 데이터 레이크 또는 플라이 휠과 관련된 모델을 삭제하지 않습니다.

플라이 휠이 반복 실행하거나 데이터 세트를 생성하는 중이면 플라이 휠 삭제 작업이 실패합니다.

새 플라이 휠을 생성하기 전에 플라이 휠 생성 섹션의 정보를 검토합니다.

기존 모델의 플라이 휠 생성

CreateFlyWheel 작업을 사용하여 기존 모델용 플라이 휠을 만들 수 있습니다.

Example

```
aws comprehend create-flywheel  \
    --flywheel-name "myFlywheel2"  \
    --active-model-arn  "modelArn"  \
    --data-access-role-arn   arn:aws::iam::111122223333:role/testFlywheelDataAccess \
    --data-lake-s3-uri": "https://s3-bucket-endpoint"   \
```

작업이 성공하면 응답에 플라이 휠 ARN이 포함됩니다.

```
{
```

```
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

새 모델용 플라이 휠 생성

CreateFlyWheel 작업을 사용하여 새로운 사용자 지정 분류 모델용 플라이 휠을 만들 수 있습니다.

Example

```
aws comprehend create-flywheel \
    --flywheel-name "myFlywheel2" \
    --data-access-role-arn  arn:aws::iam::111122223333:role/testFlywheelDataAccess \
    --model-type "DOCUMENT_CLASSIFIER" \
    --data-lake-s3-uri  "s3Uri"  \
    --task-config  file://taskConfig.json
```

TaskConfig.json 파일에는 다음 콘텐츠가 포함되어 있습니다:

```
{
    "LanguageCode": "en",
    "DocumentClassificationConfig": {
        "Mode": "MULTI_LABEL",
        "Labels": ["optimism", "anger"]
    }
}
```

API 응답 본문에는 다음 콘텐츠가 포함됩니다.

```
{
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

플라이 휠을 설명합니다.

Amazon Comprehend DescribeFlyWheel작업을 사용하여 플라이 휠에 대한 구성 정보를 검색할 수 있습니다.

```
aws comprehend describe-flywheel \
    --flywheel-arn  "flywheelArn"
```

API 응답 본문에는 다음 콘텐츠가 포함됩니다.

```
{
  "FlywheelProperties": {
      "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/
myTestFlywheel",
      "DataAccessRoleArn": "arn:aws::iam::111122223333:role/Admin",
      "TaskConfig": {
          "LanguageCode": "en",
          "DocumentClassificationConfig": {
              "Mode": "MULTI_LABEL"
          }
      },
      "DataLakeS3Uri": "s3://my-test-datalake/flywheelbasictest/myTestFlywheel/
schemaVersion=1/20220801T014326Z",
      "Status": "ACTIVE",
      "ModelType":  "DOCUMENT_CLASSIFIER",
      "CreationTime": 1659318206.102,
      "LastModifiedTime": 1659318249.05
  }
}
```

플라이 휠 업데이트

UpdateFlyWheel 작업을 사용하여 플라이 휠의 수정 가능한 구성 값을 업데이트 합니다.

일부 구성 필드는 하위 필드가 있는 JSON 구조입니다. 하나 이상의 하위 필드를 업데이트하려면 모든 하위 필드에 값을 제공합니다. Amazon Comprehend는 요청에서 누락된 모든 하위 필드에 대해 값을 null로 설정합니다.

`UpdateFlywheel`요청에 최상위 파라미터를 빠뜨리면 Amazon Comprehend는 플라이 휠에 있는 파라미터 또는 해당 하위 필드의 값을 변경하지 않습니다.

플라이 휠에서 태그를 추가하거나 제거하려면 TagResource 및 UntagResource 작업을 사용합니다.

다음 예와 같이 `ActiveModelArn` 파라미터를 설정하여 모델 버전을 승격할 수 있습니다.

```
aws comprehend update-flywheel \
    --region aws-region \
    --flywheel-arn  "flywheelArn" \
    --active-model-arn  "modelArn" \
```

API 응답 본문에는 다음 콘텐츠가 포함됩니다.

```
{
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

플라이 휠 삭제

Amazon Comprehend DeleteFlyWheel 작업을 사용하여 플라이 휠을 삭제합니다.

```
aws comprehend delete-flywheel \
    --flywheel-arn  "flywheelArn"
```

성공적인 API 응답에는 메시지 본문이 비워 있습니다

플라이 휠 목록

Amazon Comprehend ListFlyWheels 작업을 사용하여 현재 리전의 플라이 휠 목록을 검색할 수 있습니다.

```
aws comprehend list-flywheel \
    --region aws-region \
    --endpoint-url  "uri"
```

API 응답 본문에는 다음 콘텐츠가 포함됩니다.

```
{
    "FlywheelSummaryList": [
        {
            "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/
myTestFlywheel",
            "DataLakeS3Uri": "s3://my-test-datalake/flywheelbasictest/myTestFlywheel/
schemaVersion=1/20220801T014326Z",
            "Status": "ACTIVE",
            ""ModelType":  "DOCUMENT_CLASSIFIER",
            "CreationTime": 1659318206.102,
            "LastModifiedTime": 1659318249.05
        }
    ]
```

```
}
```

데이터 세트 구성

레이블이 지정된 학습 또는 테스트 데이터를 플라이 휠에 추가하려면 Amazon Comprehend 콘솔 또는 API를 사용하여 데이터 세트를 생성합니다.

각 데이터 세트를 학습 데이터 또는 테스트 데이터로 구성합니다. 데이터 세트를 특정 플라이 휠 및 사용자 정의 모델과 연결합니다. 데이터 세트를 생성하면 Amazon Comprehend가 데이터를 플라이 휠의 데이터 레이크에 업로드합니다. 학습 데이터 파일 형식에 대한 자세한 내용은 분류기 학습 데이터 준비 또는 개체 인식기 학습 데이터 준비를 참조하세요.

플라이 휠을 삭제하면 Amazon Comprehend에서 데이터 세트를 삭제합니다. 업로드된 데이터는 데이터 레이크에서 계속 사용할 수 있습니다.

데이터 세트 생성 (콘솔)

데이터세트 생성

1. 에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 플라이 휠을 선택하고 데이터를 추가할 플라이 휠을 선택합니다.
3. 데이터 세트 탭을 선택합니다.
4. 학습 데이터 세트 또는 테스트 데이터 세트 테이블에서 데이터 세트 생성을 선택합니다.
5. 데이터 세트 세부정보에 데이터 세트 이름과 설명 (선택 사항) 을 입력합니다.
6. 데이터 사양에서 데이터 형식과 데이터 세트 유형 구성 필드를 선택합니다.
7. (선택 사항) 입력 형식에서 입력 문서의 형식을 선택합니다.
8. S3의 주석 위치에 주석 파일의 Amazon S3 위치를 입력합니다.
9. S3의 학습 데이터 위치에 문서 파일의 Amazon S3 위치를 입력합니다.
10. 생성(Create)을 선택합니다.

데이터 세트 (API) 생성

CreateDataset 작업을 사용하여 데이터 세트를 생성할 수 있습니다.

Example

```
aws comprehend create-dataset \
```

```
    --flywheel-arn "myFlywheel2" \
    --dataset-name "my-training-dataset"
    --dataset-type "TRAIN"
    --description "my training dataset"
    --cli-input-json file://inputConfig.json
}
```

inputConfig.json 파일에는 다음 내용이 포함되어 있습니다.

```
{
    "DataFormat": "COMPREHEND_CSV",
    "DocumentClassifierInputDataConfig": {
        "S3Uri": "s3://my-comprehend-datasets/multilabel_train.csv"
    }
}
```

데이터 세트에 태그를 추가하거나 제거하려면 TagResource및 UntagResource 작업을 사용합니다.

데이터 세트 설명.

Amazon Comprehend DescribeDataset작업을 사용하여 플라이 휠 구성 정보를 검색합니다.

```
aws comprehend describe-dataset \
    --dataset-arn  "datasetARN"
```

응답에는 다음 콘텐츠가 포함됩니다.

```
{
   "DatasetProperties": {
      "DatasetArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/
myTestFlywheel/dataset/train-dataset",
      "DatasetName": "train-dataset",
      "DatasetType": "TRAIN",
      "DatasetS3Uri": "s3://my-test-datalake/flywheelbasictest/myTestFlywheel/
schemaVersion=1/20220801T014326Z/datasets/train-dataset/20220801T194844Z",
      "Description": "Good Dataset",
      "Status": "COMPLETED",
      "NumberOfDocuments": 90,
      "CreationTime": 1659383324.297
  }
}
```

플라이 휠 반복

플라이 휠 반복을 사용하여 새 모델 버전을 생성 및 관리할 수 있습니다.

주제

반복 워크플로

플라이 휠은 학습시킨 모델 버전으로 시작하거나 초기 데이터 세트를 사용하여 모델 버전을 학습시킵니다.

시계열에 따라 레이블이 지정된 새 데이터를 얻으면 새 모델 버전을 학습시켜 플라이 휠 모델의 성능을 개선합니다. 플라이 휠을 실행하면 새 모델 버전을 학습시키고 평가하는 새로운 반복이 생성됩니다. 성능이 기존 활성 모델 버전보다 우수한 경우 새 모델 버전을 승격시킬 수 있습니다.

플라이 휠 반복 워크플로에는 다음 단계가 포함됩니다.

1. 레이블이 지정된 새 데이터에 대한 데이터 세트를 생성합니다.
2. 플라이 휠을 실행하여 새 반복을 생성합니다. 반복은 다음 단계에 따라 새 모델 버전을 학습시키고 평가합니다.
   a. 새 데이터를 사용하여 활성 모델 버전을 평가합니다.
   b. 새 데이터를 사용하여 새 모델 버전을 학습시킵니다.
   c. 평가 및 훈련 결과를 데이터 레이크에 저장합니다.
   d. 두 모델의 F1 점수를 반환합니다.
3. 반복이 완료되면 기존 활성 모델과 새 모델의 F1 점수를 비교할 수 있습니다.
4. 새 모델 버전의 성능이 우수하면 활성 모델 버전으로 승격시킵니다. 콘솔이나 API를 사용하여 새 모델 버전을 승격시킬 수 있습니다.

반복 관리 (콘솔)

콘솔을 사용하여 새 반복을 시작하고 진행 중인 반복의 상태를 쿼리할 수 있습니다. 또한 완료된 반복의 결과를 볼 수 있습니다.

플라이 휠 반복 시작 (콘솔)

새 반복을 시작하기 전에 하나 이상의 새 학습 또는 테스트 데이터 세트를 만듭니다. 데이터 세트 구성 참조

플라이 휠 반복 시작 (콘솔)

1. 에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 플라이 휠을 선택합니다.
3. 플라이 휠 테이블에서 플라이 휠을 선택합니다.
4. 플라이 휠 실행을 선택합니다.

반복 결과 분석 (콘솔)

플라이 휠 반복을 실행한 후 콘솔은 플라이 휠 반복 테이블에 결과를 표시합니다.

새 모델 버전 홍보 (콘솔)

콘솔의 모델 세부 정보 페이지에서 새 모델 버전을 활성 모델 버전으로 승격시킬 수 있습니다.

플라이 휠 모델 버전을 활성 모델 버전으로 승격 (콘솔)

1. 에 로그인 AWS Management Console 하고 Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 플라이 휠을 선택합니다.
3. 플라이 휠 테이블에서 플라이 휠을 선택합니다.
4. 플라이 휠 세부 정보 테이블에서 플라이 휠 반복테이블에 홍보할 버전을 선택합니다.
5. 활성 모델 만들기를 선택합니다.

반복 관리 (API)

Amazon Comprehend API를 사용하여 새 반복을 시작하고 진행 중인 반복의 상태를 쿼리할 수 있습니다. 또한 완료된 반복의 결과를 볼 수 있습니다.

플라이 휠 반복 시작 (API)

Amazon Comprehend StartFlywheelIteration 작업을 사용하여 플라이 휠 반복을 시작합니다.

```
aws comprehend start-flywheel-iteration \
```

```
    --flywheel-arn  "flywheelArn"
```

응답에는 다음 콘텐츠가 포함됩니다.

```
{
  "FlywheelIterationArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name"
}
```

새 모델 버전 (API) 승격

UpdateFlyWheel 작업을 사용하면 모델 버전을 활성 모델 버전으로 승격시킬 수 있습니다.

ActiveModelArn파라미터가 설정된 UpdateFlywheel 요청을 새 활성 모델 버전의 ARN으로 전송합니다.

```
aws comprehend update-flywheel \
    --active-model-arn  "modelArn" \
```

응답에는 다음 콘텐츠가 포함됩니다.

```
{
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

플라이 휠 반복 결과 (API) 설명

Amazon Comprehend DescribeFlyWheelIteration 작업은 실행이 완료된 후 반복에 대한 정보를 반환합니다.

```
aws comprehend describe-flywheel-iteration \
 --flywheel-arn "flywheelArn" \
 --flywheel-iteration-id  "flywheelIterationId" \
 --region aws-region
```

응답에는 다음 콘텐츠가 포함됩니다.

```
{
    "FlywheelIterationProperties": {
        "FlywheelArn": "flywheelArn",
```

```
        "FlywheelIterationId": "iterationId",
        "CreationTime": <createdAt>,
        "EndTime": <endedAt>,
        "Status": <status>,
        "Message": <message>,
        "EvaluatedModelArn": "modelArn",
        "EvaluatedModelMetrics": {
            "AverageF1Score": <value>,
            "AveragePrecision": <value>,
            "AverageRecall": <value>,
            "AverageAccuracy": <value>
        },
        "TrainedModelArn": "modelArn",
        "TrainedModelMetrics": {
            "AverageF1Score": <value>,
            "AveragePrecision": <value>,
            "AverageRecall": <value>,
            "AverageAccuracy": <value>
        }
    }
}
```

반복 기록 가져오기 (API)

ListFlywheelIterationHistory 작업을 사용하여 반복 기록에 대한 정보를 가져올 수 있습니다.

```
aws comprehend list-flywheel-iteration-history \
 --flywheel-arn "flywheelArn"
```

응답에는 다음 콘텐츠가 포함됩니다.

```
{
    "FlywheelIterationPropertiesList": [
        {
            "FlywheelArn": "<flywheelArn>",
            "FlywheelIterationId": "20220907T214613Z",
            "CreationTime": 1662587173.224,
            "EndTime": 1662592043.02,
            "Status": "<status>",
            "Message": "<message>",
            "EvaluatedModelArn": "modelArn",
            "EvaluatedModelMetrics": {
                "AverageF1Score": 0.8333333333333333,
```

```
                "AveragePrecision": 0.75,
                "AverageRecall": 0.9375,
                "AverageAccuracy": 0.8125
            },
            "TrainedModelArn": "modelArn",
            "TrainedModelMetrics": {
                "AverageF1Score": 0.865497076023392,
                "AveragePrecision": 0.7636363636363637,
                "AverageRecall": 1.0,
                "AverageAccuracy": 0.84375
            }
        }
    ]
}
```

분석에 플라이 휠 사용

플라이 휠의 활성 모델 버전을 사용하여 사용자 정의 분류 또는 개체 인식 분석을 실행할 수 있습니다. 활성 모델 버전을 구성할 수 있습니다. 콘솔 또는 UpdateFlyWheel API 작업을 사용하여 모델의 새 버전을 활성 모델 버전으로 설정할 수 있습니다.

플라이 휠을 사용하려면 분석 작업을 구성할 때 사용자 정의 모델 ARN 대신 플라이 휠 ARN을 지정하십시오. Amazon Comprehend는 플라이 휠의 활성 모델 버전을 사용하여 분석을 실행합니다.

실시간 분석

엔드포인트를 사용하여 실시간 분석을 실행합니다. 엔드포인트를 생성하거나 업데이트할 때 모델 ARN 대신 플라이 휠 ARN을 사용하여 엔드포인트를 구성할 수 있습니다. 실시간 분석을 할 때는 플라이 휠과 관련된 엔드포인트를 선택합니다. Amazon Comprehend는 플라이 휠의 활성 모델 버전을 사용하여 분석을 실행합니다.

UpdateFlyWheel을 사용하여 플라이 휠의 새 활성 모델 버전을 설정하면 엔드포인트가 자동으로 업데이트 되어 새 활성 모델 버전을 사용하기 시작합니다. 엔드포인트가 자동으로 업데이트되지 않도록 하려면 UpdateEndpoint를 사용하여 직접 모델 버전 ARN을 사용하도록 엔드포인트를 구성하십시오. 플라이 휠 활성 모델 버전이 변경되더라도 엔드포인트는 이 모델 버전을 계속 사용합니다.

사용자 정의 분류에는 ClassifyDocument API 작업을 사용하십시오. 사용자 정의 개체 인식에는 DetectEntities API 요청을 사용하세요. EndpointArn 파라미터에 플라이 휠의 엔드포인트를 입력합니다.

또한 콘솔을 사용하여 사용자 정의 분류 또는 사용자 정의 개체 인식을 실시간 분석할 수 있습니다.

비동기식 API

사용자 정의 분류에는 StartDocumentClassificationJob API 요청을 사용하여 비동기 작업을 시작하십시오. DocumentClassifierArn 대신 FlywheelArn 파라미터를 제공하십시오.

사용자 정의 개체 인식에는 StartEntiesDetectionJob API 요청을 사용하십시오. EntityRecognizerArn 대신 FlywheelArn 파라미터를 제공하십시오.

콘솔을 사용하여 사용자 정의 분류 또는 사용자 정의 개체 인식 비동기 분석 작업을 할 수 있습니다. 작업을 생성할 때 인식자 모델 또는 분류자 모델 필드에 플라이 휠 ARN을 입력하십시오.

Amazon Comprehend 엔드포인트 관리

Amazon Comprehend에서는 엔드포인트를 사용하여 사용자 지정 모델을 실시간 분류 또는 개체 감지에 사용할 수 있습니다. 엔드포인트를 생성한 후 비즈니스 요구 사항이 변함에 따라 엔드포인트를 변경할 수 있습니다. 예를 들어 엔드포인트 사용률을 모니터링하고 Auto Scaling을 적용하여 용량에 맞게 엔드포인트 프로비저닝을 자동으로 설정할 수 있습니다. 단일 보기에서 모든 엔드포인트를 관리할 수 있으며, 엔드포인트가 더 이상 필요하지 않을 경우 삭제하여 비용을 절감할 수 있습니다.

엔드포인트를 관리하려면 먼저 엔드포인트를 생성해야 합니다. 자세한 내용은 다음 절차를 참조하세요.

- 사용자 지정 분류를 위한 엔드포인트 생성
- 사용자 지정 개체 감지를 위한 엔드포인트 생성

주제

Amazon Comprehend 엔드포인트 개요

Amazon Comprehend 콘솔의 엔드포인트 페이지는 엔드포인트에 대한 글로벌 보기를 제공합니다. 엔드포인트 개요 페이지에서는 모든 엔드포인트를 한 곳에서 확인하여 엔드포인트 사용량과 실제 리소스 사용량을 비교할 수 있습니다. 엔드포인트 페이지 오른쪽 상단에서 보려는 엔드포인트(모든 엔드포인트, 사용자 지정 분류자 엔드포인트 또는 사용자 지정 개체 엔드포인트)를 지정할 수 있습니다.

이 페이지에서 엔드포인트를 생성, 업데이트, 모니터링 및 삭제할 수 있습니다. 엔드포인트 개요 섹션에서 엔드포인트 목록, 엔드포인트가 호스팅하는 사용자 지정 모델, 생성 시간, 프로비저닝 처리량 및 엔드포인트 상태를 볼 수 있습니다. 엔드포인트 개요 테이블에서 특정 엔드포인트를 선택하면 엔드포인트 세부 정보가 표시됩니다.

또한 AWS 비즈니스 지원 또는 AWS 기업 지원 고객은 엔드포인트에 고유한 Trusted Advisor 검사에 액세스할 수 있습니다. 자세한 내용은 Amazon Comprehend Trusted Advisor 에서 사용를 참조하세요. 검사 및 설명의 전체 목록은 Trusted Advisor 모범 사례를 참조하세요.

엔드포인트 관리에 대한 자세한 내용은 다음 주제를 참조하세요.

- Amazon Comprehend 엔드포인트 사용
- Amazon Comprehend 엔드포인트 모니터링
- Amazon Comprehend 엔드포인트 업데이트
- Amazon Comprehend Trusted Advisor 에서 사용
- Amazon Comprehend 엔드포인트 삭제

> **Important**
>
> 실시간 사용자 지정 분류 비용은 설정한 처리량과 엔드포인트가 활성화된 시간을 기준으로 합니다. 엔드포인트를 더 이상 사용하지 않거나 장기간 사용하지 않는 경우 Auto Scaling 정책을 설정하여 비용을 절감해야 합니다. 또는 엔드포인트를 더 이상 사용하지 않는 경우 엔드포인트를 삭제하여 추가 비용이 발생하지 않도록 할 수 있습니다. 자세한 내용은 엔드포인트를 사용한 Auto Scaling을 참조하십시오.

Amazon Comprehend 엔드포인트 사용

사용자 지정 모델을 사용하여 실시간 분석을 실행할 엔드포인트를 생성합니다. 엔드포인트에는 사용자 지정 모델을 실시간 추론에 사용할 수 있는 관리형 리소스가 포함되어 있습니다.

Amazon Comprehend는 추론 단위(IU)를 사용하여 엔드포인트에 처리량을 할당합니다. IU는 초당 100자의 데이터 처리량을 나타냅니다. 엔드포인트에 최대 10개의 추론 단위를 프로비저닝할 수 있습니다. 엔드포인트를 업데이트하여 엔드포인트 처리량을 늘리거나 줄일 수 있습니다.

입력 문서에 반정형 문서나 이미지 파일이 포함된 경우 초당 100자의 처리량은 입력 파일에서 추출한 문자에 해당합니다. 엔드포인트에 프로비저닝하는 IU 수는 입력 문서의 문자 밀도에 따라 달라집니다.

ClassifyDocument 및 DetectEntities API 응답에는 각 입력 페이지의 문자 수가 포함됩니다. 이 정보를 사용하여 원하는 처리량을 달성하기 위해 프로비저닝할 추론 단위의 수를 추정할 수 있습니다.

실시간 분석을 완료한 후에는 엔드포인트를 삭제하십시오. 활성 상태가 지속되면 엔드포인트에 대한 요금이 계속 청구되기 때문입니다. 추가 실시간 분석을 실행할 준비가 되면 다른 엔드포인트를 생성할 수 있습니다.

엔드포인트 비용에 대한 자세한 내용은 Amazon Comprehend 요금을 참조하세요.

엔드포인트를 생성한 후 Amazon CloudWatch로 엔드포인트를 모니터링하고, 업데이트하여 추론 단위를 변경하거나, 더 이상 필요하지 않을 때는 삭제할 수 있습니다. 자세한 내용은 Amazon Comprehend 엔드포인트 모니터링를 참조하십시오.

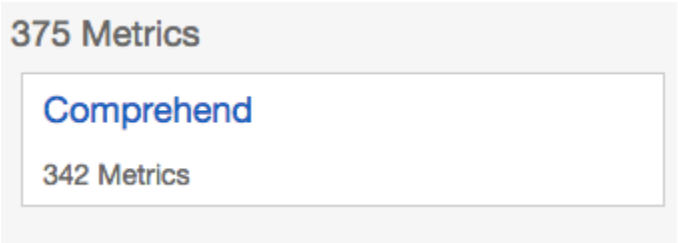
Amazon Comprehend 엔드포인트 모니터링

추론 단위(IUs. 엔드포인트 업데이트에 대한 자세한 내용은 the section called “엔드포인트 업데이트”를 참조하세요.

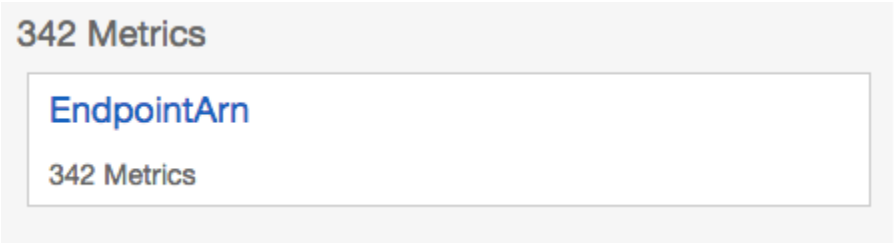
Amazon CloudWatch 콘솔에서 엔드포인트 사용량을 모니터링하여 엔드포인트의 처리량을 가장 잘 조정하는 방법을 결정할 수 있습니다.

CloudWatch로 엔드포인트 사용량 모니터링

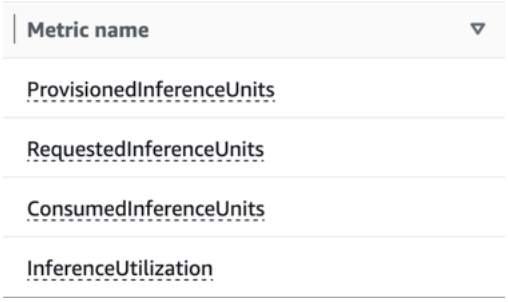
1. 에 로그인 AWS Management Console 하고 CloudWatch 콘솔을 엽니다.
2. 왼쪽에서 지표를 선택하고 모든 지표를 선택합니다.
3. 모든 지표에서 Comprehend를 선택합니다.



4. CloudWatch 콘솔에는 Comprehend 지표의 차원이 표시됩니다. EndpointArn 차원을 선택합니다.



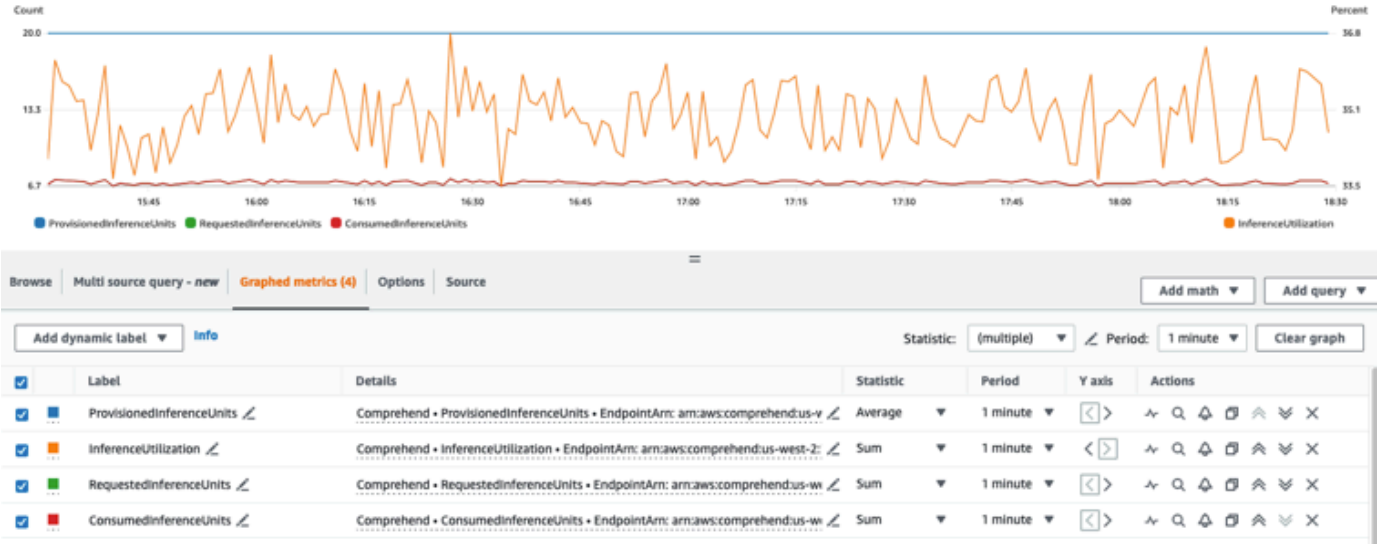
콘솔에는 각 엔드포인트에 대한 ProvisionedInferenceUnits, RequestedInferenceUnits, ConsumedInferenceUnits 및 InferenceUtilization이 표시됩니다.



4개의 지표를 선택하고 그래프로 표시된 지표 탭으로 이동합니다.

5. RequestedInferenceUnits 및 ConsumedInferenceUnits의 통계 열을 Sum으로 설정합니다.
6. InferenceUtilization의 통계 열을 합계로 설정합니다.
7. ProvisionedInferenceUnits의 통계 열을 평균으로 설정합니다.
8. 모든 지표의 기간 열을 1분으로 변경합니다.
9. InferenceUtilization을 선택하고 화살표를 선택하여 별도의 Y축으로 이동합니다.

그래프를 분석할 준비가 되었습니다.



CloudWatch 지표를 기반으로 엔드포인트의 처리량을 자동으로 조정하도록 Auto Scaling을 설정할 수도 있습니다. 엔드포인트에서 Auto Scaling 사용에 대한 자세한 내용은 [엔드포인트를 사용한 Auto Scaling]을 참조하세요.

- ProvisionedInferenceUnits -이 지표는 요청이 수행된 시점의 평균 프로비저닝된 IUs 수를 나타냅니다.

- RequestedInferenceUnits - 처리하도록 전송된 서비스에 제출된 각 요청의 사용량을 기준으로 합니다. 이는 처리되도록 전송된 요청을 제한(ConsumedInferenceUnits) 없이 실제로 처리된 요청과 비교하는 데 도움이 될 수 있습니다. 이 지표의 값은 처리하도록 전송된 문자 수를 1분 동안 1 IU로 처리할 수 있는 문자 수로 나누어 계산됩니다.
- ConsumedInferenceUnits - 성공적으로 처리된(제한되지 않은) 서비스에 제출된 각 요청의 사용량을 기준으로 합니다. 이는 사용량을 프로비저닝된 IU와 비교할 때 유용할 수 있습니다. 이 지표의 값은 처리된 문자 수를 1IU에 대해 1분 동안 처리할 수 있는 문자 수로 나누어 계산합니다.
- InferenceUtilization - 요청별로 내보내집니다. 이 값은 ConsumedInferenceUnits에 정의된 소비된 IU를 ProvisionedInferenceUnits로 나누고 100에 대한 백분율로 변환하여 계산합니다.

> **Note**
>
> 모든 지표는 요청이 성공한 경우에만 내보내집니다. 요청이 제한적이거나 내부 서버 오류 또는 고객 오류로 인해 실패하는 경우 지표가 표시되지 않습니다.

Amazon Comprehend 엔드포인트 업데이트

엔드포인트를 생성한 후에는 필요한 처리량 수준이 변경되거나 필요에 대한 첫 번째 추정치가 변경되는 경우가 많습니다. 이 경우 처리량을 늘리거나 줄이기 위해 엔드포인트를 업데이트해야 할 수 있습니다. 처리량은 엔드포인트를 프로비저닝한 추론 단위 수에 따라 결정됩니다. 각 추론 단위는 초당 최대 2개 문서에 대한 초당 100자의 처리량을 나타냅니다. 엔드포인트와 연결된 모델의 버전을 업데이트해야 할 수도 있습니다. 엔드포인트를 편집할 때 엔드포인트에 대해 다른 버전의 모델을 선택할 수 있습니다.

엔드포인트에 태그를 추가하여 체계적으로 정리하는 것도 유용할 수 있습니다. 엔드포인트를 업데이트하는 동안에도 이 작업을 수행할 수 있습니다. 엔드포인트에 대한 자세한 정보는 리소스에 태그 지정을 참조하세요.

엔드포인트를 업데이트하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 엔드포인트를 선택합니다.
3. 분류기 목록에서 엔드포인트를 업데이트하려는 사용자 지정 모델의 이름을 선택하고 링크를 따라 이동합니다. 모델 세부 정보 페이지가 표시됩니다.

4. 모델 세부 정보 페이지에서 버전 세부 정보를 선택합니다. 엔드포인트 목록이 표시됩니다.
5. 엔드포인트의 엔드포인트 확인란을 선택합니다. 엔드포인트 표의 오른쪽 상단에서 작업 아이콘을 선택합니다.
6. 편집을 선택합니다. 프로비저닝된 IU를 업데이트하고 태그를 편집할 수 있습니다.
7. 변경 내용을 저장합니다.
8. 엔드포인트가 프로비저닝되는 데 사용되는 추론 단위 수를 편집하려면 편집을 선택합니다.
9. 엔드포인트에 할당할 업데이트된 추론 단위 수를 입력합니다. 각 단위는 초당 100자의 처리량을 나타냅니다. 엔드포인트당 최대 10개의 추론 단위를 할당할 수 있습니다.

> **Note**
>
> 엔드포인트 사용 비용은 운영 시간과 처리량(추론 단위 수 기준)을 기반으로 합니다. 따라서 추론 단위 수를 늘리면 운영 비용이 증가합니다. 자세한 내용은 Amazon Comprehend 요금을 참조하세요.

10. 엔드포인트 편집을 선택합니다. 엔드포인트 세부 정보 페이지가 표시됩니다.
11. 페이지 상단의 브레드크럼에서 모델 이름을 선택하여 엔드포인트가 업데이트되고 있는지 확인합니다. 사용자 지정 모델 세부 정보 페이지에서 엔드포인트 목록으로 이동하여 엔드포인트 옆에 업데이트가 표시되는지 확인합니다. 업데이트가 완료되면 준비로 표시됩니다.

    다음 예제에서는 AWS CLI에서 UpdateEndpoint 작업을 사용하는 방법을 보여줍니다.

    다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

    ```
    aws comprehend update-endpoint \
        --desired-inference-units updated number of inference units \
        --desired-model-arn arn:aws:comprehend:region:account-id:model type/model name
     \
        --desired-data-access-role-arn arn:aws:iam:account id:role/role name
        --endpoint-arn arn:aws:comprehend:region:account id:endpoint/endpoint name
    ```

    작업이 성공하면 Amazon Comprehend는 비어있는 HTTP 본문과 함께 HTTP 본문과 함께 HTTP 200 응답으로 응답합니다.
12. 엔드포인트에 연결된 사용자 지정 모델을 편집하려면 사용자 지정 모델 세부 정보 페이지에서 엔드포인트 목록으로 이동합니다.
13. 변경하려는 엔드포인트를 선택하고 편집을 선택합니다.

14. 엔드포인트 설정 페이지에서, 엔드포인트에 따라 분류기 모델 선택 또는 인식기 모델 선택 아래의 드롭다운에서 모델을 검색할 수 있습니다. 원하는 모델을 선택합니다.
15. 버전 선택 아래에서 원하는 모델 버전을 검색할 수 있습니다. 버전을 선택합니다.
16. 엔드포인트 편집을 선택하여 저장합니다.

Amazon Comprehend Trusted Advisor 에서 사용

AWS Trusted Advisor 는 AWS 모범 사례에 따라 리소스를 프로비저닝하는 데 도움이 되는 권장 사항을 제공하는 온라인 도구입니다.

기본 또는 개발자 지원 플랜이 있는 경우 Trusted Advisor 콘솔을 사용하여 서비스 제한 범주의 모든 검사 및 보안 범주의 6개 검사에 액세스할 수 있습니다. 비즈니스 또는 엔터프라이즈 지원 플랜이 있는 경우 Trusted Advisor 콘솔과 AWS Support API를 사용하여 모든 Trusted Advisor 검사에 액세스할 수 있습니다.

Amazon Comprehend는 다음 Trusted Advisor 검사를 지원하여 고객이 실행 가능한 권장 사항을 제공하여 Amazon Comprehend 엔드포인트의 비용과 보안을 최적화할 수 있도록 지원합니다.

Amazon Comprehend 사용률이 낮은 엔드포인트

Amazon Comprehend 활용도가 낮은 엔드포인트 검사는 엔드포인트의 처리량 구성을 평가합니다. 이 검사는 엔드포인트가 실시간 추론 요청에 대해 활성 상태로 사용되지 않을 때 경고합니다. 15일 이상 사용되지 않은 엔드포인트는 사용률이 낮은 것으로 간주됩니다. 모든 엔드포인트는 처리량 세트, 및 엔드포인트가 활성 상태인 기간 모두를 기준으로 요금이 발생합니다. 엔드포인트가 지난 15일 동안 사용되지 않은 경우, 애플리케이션 Autoscaling을 사용하여 리소스에 대한 조정 정책을 정의하는 것이 좋습니다. 지난 30일 동안 사용되지 않았고 Auto Scaling 정책이 정의된 엔드포인트의 경우 비동기 추론을 사용하거나 이를 삭제하는 것이 좋습니다. 이러한 검사 결과는 매일 한 번 자동으로 새로 고쳐지며 Trusted Advisor 콘솔의 CostOptimization 범주에서 확인할 수 있습니다.

모든 엔드포인트의 사용률 상태와 해당 권장 사항을 보려면

1. 에 로그인 AWS Management Console 하고 Trusted Advisor 콘솔을 엽니다.
2. 탐색 창에서 CostOptimization 검사 범주를 선택합니다.
3. 범주 페이지에서 각 검사 범주에 대한 요약을 봅니다.

   - 작업 권장(빨간색) - 검사에 대한 작업을 Trusted Advisor 권장합니다.
   - 조사 권장(노란색) – Trusted Advisor 가 검사 대상이 될 수 있는 문제를 감지합니다.

- 감지된 문제 없음(녹색) -가 검사 문제를 감지 Trusted Advisor 하지 못합니다.
- 제외된 항목(회색) - 제외된 항목(예: 검사에서 무시할 리소스)이 있는 검사의 수입니다.

4. Amazon Comprehend 사용률이 낮은 엔드포인트 검사를 선택하여 검사 설명과 다음 세부 정보를 확인합니다.

   - 알림 기준 - 검사 상태가 변경될 때의 임계값을 설명합니다.
   - 권장 작업 - 이 검사에 권장되는 작업에 대해 설명합니다.
   - 리소스 테이블: 엔드포인트 세부 정보와 권장 사항에 따른 각 상태를 나열한 표입니다.

5. 리소스 테이블에서 엔드포인트가 지난 30일 동안 사용되지 않음이란 경고로 인해 조사 권장 사항으로 플래그가 지정된 경우 Amazon Comprehend 콘솔의 엔드포인트 세부 정보 페이지로 이동할 수 있습니다.

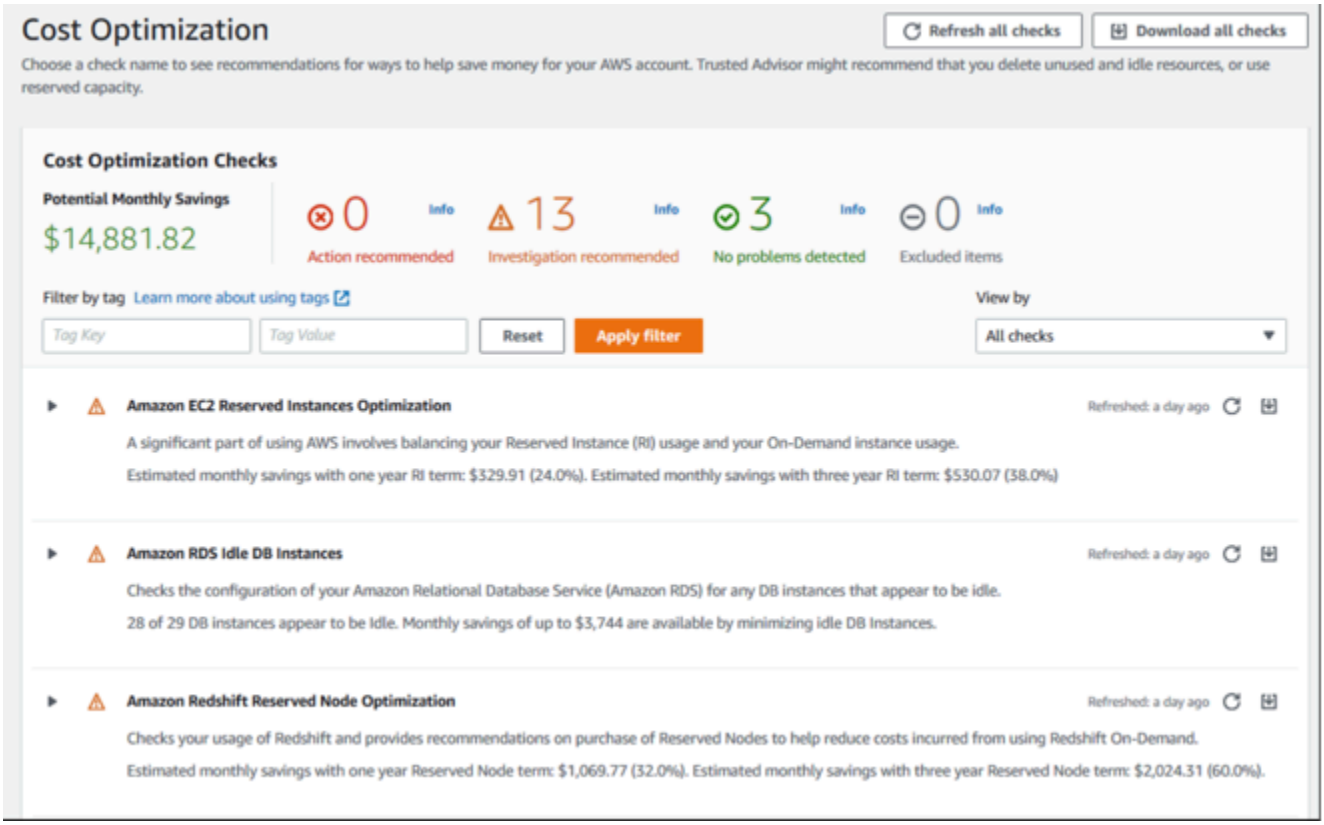
   - 이 엔드포인트를 더 이상 사용하지 않으려면 삭제를 선택하십시오.
   - 삭제를 다시 선택하여 삭제를 확인합니다. 사용자 지정 모델 세부 정보 페이지가 표시됩니다. 삭제한 엔드포인트 옆에 삭제 중이 표시되는지 확인합니다. 엔드포인트가 삭제되면 엔드포인트 목록에서 해당 엔드포인트가 제거됩니다.

6. Trusted Advisor 콘솔의 리소스 테이블에서 엔드포인트가 지난 15일 동안 사용되지 않아 조사 권장 상태로 플래그가 지정되어 있고 AutoScaling이 비활성화된 경우 Amazon Comprehend 콘솔의 엔드포인트 세부 정보 페이지로 이동하여 엔드포인트를 조정할 수 있습니다.

   - 이 엔드포인트에 구성된 처리량을 줄이려면 편집을 클릭하십시오. 엔드포인트에 할당할 업데이트된 추론 단위 수를 입력한 다음 승인할 확인란을 선택한 다음 엔드포인트 편집을 선택합니다. 업데이트가 완료되면 상태가 준비로 표시됩니다.
   - 처리량 구성을 수동으로 조정하는 대신 엔드포인트에서 엔드포인트 프로비저닝을 자동으로 설정하려면 애플리케이션 Autoscaling을 사용하는 것이 좋습니다.

7. Trusted Advisor 콘솔의 리소스 테이블에서 사용된 활성 이유 때문에 엔드포인트에 감지된 문제 없음 상태로 플래그가 지정된 경우 실시간 추론 요청을 실행하는 데 엔드포인트가 적극적으로 사용되고 있으며 권장 작업이 없음을 의미합니다.

다음은 Trusted Advisor 콘솔의 CostOptimization 범주 보기를 보여주는 예시입니다.



Amazon Comprehend 엔드포인트 액세스 위험

Amazon Comprehend 엔드포인트 액세스 위험 검사는 기본 모델이 고객 관리형 키를 사용하여 암호화된 엔드포인트에 대한 AWS Key Management Service (AWS KMS) 키 권한을 평가합니다. 고객 관리형 키가 비활성화된 경우, 또는 Amazon Comprehend에 허용된 권한을 변경하기 위해 키 정책이 변경되면 엔드포인트 가용성에 영향이 있을 수 있습니다. 키가 비활성화된 경우 활성화하는 것이 좋습니다. 키 정책이 변경되었는데 엔드포인트를 계속 사용하려는 경우에는 키 정책을 업데이트하는 것이 좋습니다. 검사 결과는 하루 동안 여러 번 자동으로 새로 고침됩니다. 이 검사는 Trusted Advisor 콘솔의 Fault Tolerance 범주에서 확인할 수 있습니다.

Amazon Comprehend 엔드포인트의 AWS KMS 키 상태를 보려면

1. 에 로그인 AWS Management Console 하고 Trusted Advisor 콘솔을 엽니다.
2. 탐색 창에서 FaultTolerance 검사 범주를 선택합니다.
3. 범주 페이지에서 각 검사 범주에 대한 요약을 봅니다.

   - 작업 권장(빨간색) - 검사에 대한 작업을 Trusted Advisor 권장합니다.
   - 조사 권장(노란색) Trusted Advisor 가 검사 대상이 될 수 있는 문제를 감지합니다.
   - 감지된 문제 없음(녹색) -가 검사 문제를 감지 Trusted Advisor 하지 못합니다.

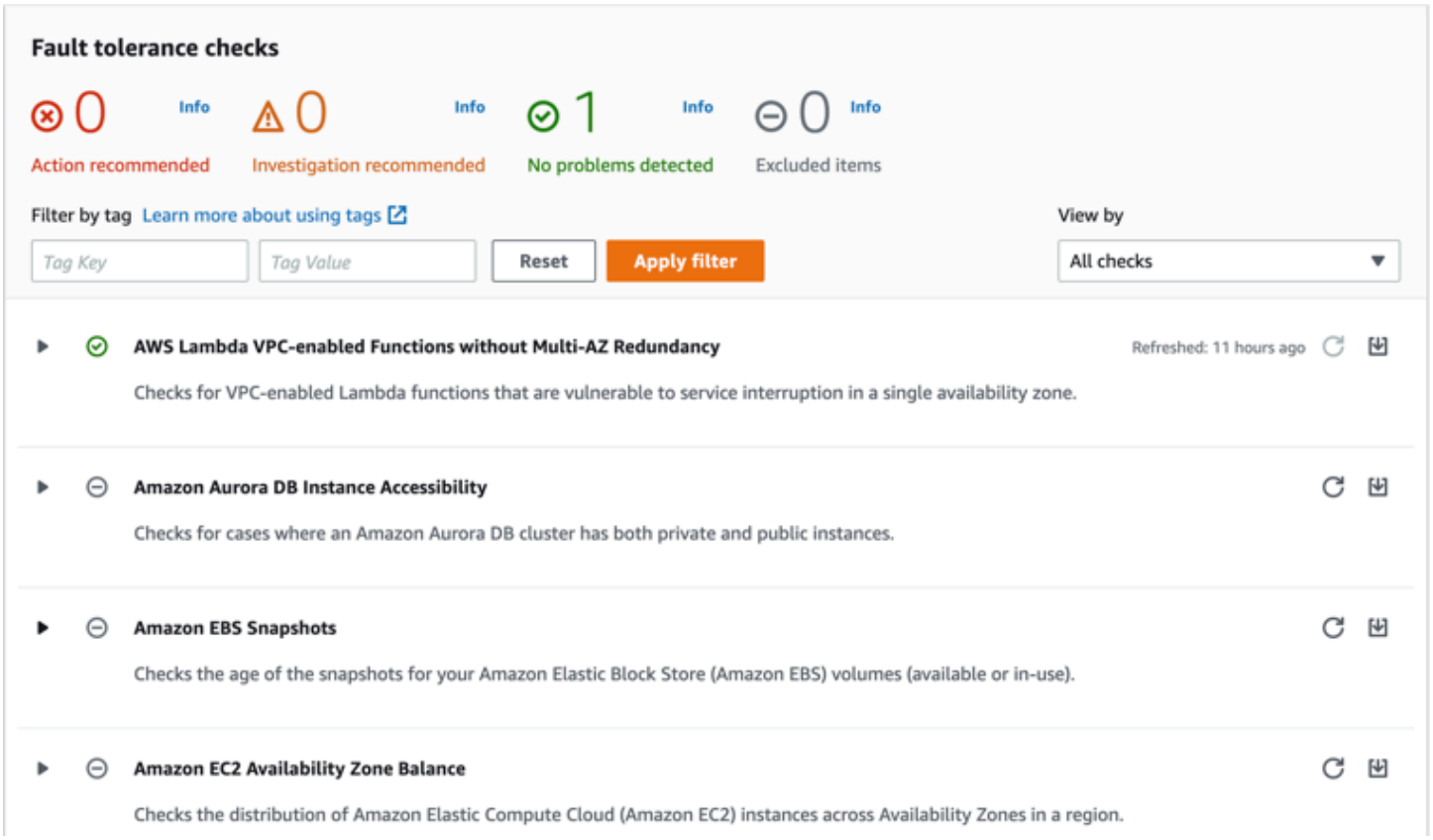
- 제외된 항목(회색) - 제외된 항목(예: 검사에서 무시할 리소스)이 있는 검사의 수입니다.

4. Amazon Comprehend 엔드포인트 액세스 위험 검사를 선택하면 검사 설명 및 다음 세부 정보를 볼 수 있습니다.

   - 알림 기준 - 검사 상태가 변경될 때의 임계값을 설명합니다.
   - 권장 작업 - 이 검사에 권장되는 작업에 대해 설명합니다.
   - 리소스 테이블: KMS로 암호화된 엔드포인트 세부 정보와 권장 작업이 있는지 여부에 따라 각 엔드포인트의 상태를 나열하는 표입니다.

5. 리소스 테이블에서 엔드포인트에 작업 권장 상태로 플래그가 지정된 경우 KMS KeyId 열의 링크를 선택하면 해당 AWS KMS 키 페이지로 리디렉션됩니다.
   - 비활성화된 AWS KMS 키를 활성화하려면 키 작업을 선택하고 활성화를 선택합니다.
   - 키 상태가 활성화됨으로 나열된 경우 키 정책 섹션에서 정책 보기로 전환을 선택하여 키 정책을 업데이트하십시오. 키 정책 문서를 편집하여 Amazon Comprehend에 필요한 권한을 제공한 다음 변경 사항 저장을 선택합니다.

다음은 Trusted Advisor 콘솔의 FaultTolerance 범주 보기의 예입니다.



이러한 검사 및 결과는 AWS Support API의 Trusted Advisor 섹션을 참조하여 볼 수도 있습니다.

CloudWatch를 사용하여 경보를 설정하는 방법에 대한 자세한 내용은 Trusted Advisor CloudWatch를 사용한 경보 생성을 참조하세요. Trusted Advisor 모범 사례 검사의 전체 세트는 AWS Trusted Advisor 모범 사례 체크리스트를 참조하세요.

Amazon Comprehend 엔드포인트 삭제

엔드포인트가 더 이상 필요하지 않으면 비용이 발생하지 않도록 엔드포인트를 삭제해야 합니다. 필요할 때마다 엔드포인트 섹션에서 다른 엔드포인트를 쉽게 만들 수 있습니다.

엔드포인트를 삭제하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 메뉴에서 엔드포인트를 선택합니다.
3. 엔드포인트 테이블에서 삭제하려는 엔드포인트를 찾습니다. 검색을 하거나 모든 엔드포인트를 필터링하여 필요한 엔드포인트를 찾을 수 있습니다.
4. 삭제하려는 엔드포인트의 엔드포인트 확인란을 선택합니다. 엔드포인트 표의 오른쪽 상단에서 작업 아이콘을 선택합니다.
5. Delete(삭제)를 선택합니다.
6. 삭제를 다시 선택하여 삭제를 확인합니다. 엔드포인트 페이지가 표시됩니다. 삭제한 엔드포인트 옆에 삭제 중이 표시되는지 확인합니다. 엔드포인트가 삭제되면 엔드포인트 목록에서 해당 엔드포인트가 제거됩니다.

엔드포인트를 삭제하려면(AWS CLI)

다음 예제에서는 AWS CLI에서 DeleteEndpoint 작업을 사용하는 방법을 보여줍니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws comprehend delete-endpoint \
    --endpoint-arn arn:aws:comprehend:region:account-id endpoint/endpoint name
```

작업이 성공하면 Amazon Comprehend는 비어있는 HTTP 본문과 함께 HTTP 본문과 함께 HTTP 200 응답으로 응답합니다.

엔드포인트를 사용한 Auto Scaling

문서 분류 엔드포인트 및 개체 인식기 엔드포인트에 프로비저닝된 추론 단위 수를 수동으로 조정하는 대신, Auto Scaling을 사용하여 용량에 맞게 엔드포인트 프로비저닝을 자동으로 설정할 수 있습니다.

Auto Scaling을 사용하여 엔드포인트에 프로비저닝된 추론 단위 수를 조정하는 방법은 두 가지입니다.

- 대상 추적: Auto Scaling을 설정하여 사용량에 따라 용량에 맞게 엔드포인트 프로비저닝을 조정합니다.
- 예약된 크기 조정: Auto Scaling을 설정하여 지정된 일정의 용량에 맞게 엔드포인트 프로비저닝을 조정합니다.

Auto Scaling은 AWS Command Line Interface ()를 통해서만 설정할 수 있습니다AWS CLI. Auto Scaling에 대한 자세한 내용은 애플리케이션 Auto Scaling이란?을 참조하세요.

대상 추적

대상 추적을 사용하면 사용량에 따라 용량에 맞게 엔드포인트 프로비저닝을 조정할 수 있습니다. 사용 용량이 프로비저닝된 용량의 목표 백분율 내에 있도록 추론 단위 수가 자동으로 조정됩니다. 대상 추적을 사용하여 문서 분류 엔드포인트 및 개체 인식기 엔드포인트의 일시적인 사용 급증을 수용할 수 있습니다. 자세한 내용은 애플리케이션 Auto Scaling의 대상 추적 조정 정책을 참조하세요.

> **Note**
>
> 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

대상 추적 설정

엔드포인트에 대한 대상 추적을 설정하려면 AWS CLI 명령을 사용하여 확장 가능한 대상을 등록한 다음 조정 정책을 생성합니다. 확장 가능 대상은 추론 단위를 엔드포인트 프로비저닝을 조정하는 데 사용되는 리소스로 정의하고, 조정 정책은 프로비저닝된 용량의 Auto Scaling을 제어하는 지표를 정의합니다.

대상 추적을 설정하려면

1. 확장 가능 대상을 등록합니다. 다음 예에서는 확장 가능 대상을 등록하여 최소 용량은 추론 단위 1개, 최대 용량은 추론 단위 2개로 엔드포인트 프로비저닝을 조정합니다.

문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling register-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits \
    --min-capacity 1 \
    --max-capacity 2
```

개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling register-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --min-capacity 1 \
    --max-capacity 2
```

2. 확장 가능 대상의 등록을 확인하려면 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling describe-scalable-targets \
    --service-namespace comprehend \
    --resource-id endpoint ARN
```

3. 조정 정책을 위한 대상 추적 구성을 생성하고 구성을 `config.json`이라는 파일에 저장합니다. 다음은 InferenceUtilization 지표를 70%로 유지하는 것을 대상으로 하는 문서 분류 엔드포인트에 대한 대상 추적 구성의 예입니다.

```
{
  "TargetValue": 70,
  "CustomizedMetricSpecification": {
    "MetricName": "InferenceUtilization",
    "Namespace": "MyNamespace",
    "Dimensions": [
```

```
      {
        "Name": "EndpointArn",
        "Value": "arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name"
      }
    ],
    "Statistic": "Sum",
    "Unit": "Percent"
  }
}
```

다음은 개체 인식기 엔드포인트의 예입니다.

```
{
  "TargetValue": 70,
  "CustomizedMetricSpecification": {
    "MetricName": "InferenceUtilization",
    "Namespace": "MyNamespace",
    "Dimensions": [
      {
        "Name": "EndpointArn",
        "Value": "arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name"
      }
    ],
    "Statistic": "Sum",
    "Unit": "Percent"
  }
}
```

4. 조정 정책을 생성합니다. 다음 예제에서는 `config.json` 파일에 정의된 대상 추적 구성을 기반으로 조정 정책을 생성합니다.

   문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling put-scaling-policy \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits \
    --policy-name TestPolicy \
    --policy-type TargetTrackingScaling \
```

```
    --target-tracking-scaling-policy-configuration file://config.json
```

개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling put-scaling-policy \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --policy-name TestPolicy \
    --policy-type TargetTrackingScaling \
    --target-tracking-scaling-policy-configuration file://config.json
```

고려 사항

Comprehend 엔드포인트에서 대상 추적을 사용할 때 다음 고려 사항이 적용됩니다.

- 엔드포인트 지표는 성공적인 요청에 대해서만 내보내집니다. 내부 서버 오류 또는 고객 오류로 인해 제한되거나 실패한 요청의 경우 지표가 표시되지 않습니다.
- 데이터 포인트가 누락되면 지원 CloudWatch 경보 상태가 로 변경됩니다`INSUFFICIENT_DATA` . 이 경우 Application Auto Scaling은 엔드포인트를 확장할 수 없습니다.
- 지표 수학은 이러한 제한을 해결하는 데 도움이 될 수 있습니다. 예를 들어, 지표가 보고되지 않을 때 0의 값을 사용하려면 `m1`가 지표인 `FILL(m1,0)` 함수를 사용합니다. 구성을 테스트하여 예상대로 작동하는지 확인하는 것이 중요합니다. 추가 옵션은 지표 수학을 사용하여 대상 추적 정책 생성을 참조하세요.

대상 추적 제거

엔드포인트에 대한 대상 추적을 제거하려면 AWS CLI 명령을 사용하여 조정 정책을 삭제한 다음 확장 가능 대상의 등록을 취소합니다.

대상 추적을 제거하려면

1. 조정 정책을 삭제합니다. 다음 예제에서는 지정한 조정 정책을 삭제합니다.

   문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

   ```
   aws application-autoscaling delete-scaling-policy \
   ```

```
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits \
    --policy-name TestPolicy \
```

개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling delete-scaling-policy \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --policy-name TestPolicy
```

2. 확장 가능 목표의 등록을 취소합니다. 다음 예제는 지정된 확장 가능 대상의 등록을 취소합니다.

   문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits
```

   개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits
```

예약된 크기 조정

예약된 조정을 사용하여 지정된 일정의 용량에 맞게 엔드포인트 프로비저닝을 조정할 수 있습니다. 예약된 조정은 특정 시간의 사용량 급증을 수용할 수 있도록 추론 단위 수를 자동으로 조정합니다. 문서 분류 엔드포인트 및 개체 인식기 엔드포인트에 예약된 조정을 사용할 수 있습니다. 예약된 조정에 대한 자세한 내용을 알아보려면 애플리케이션 Auto Scaling의 예약된 조정을 참조하세요.

> **Note**
>
> 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

예약된 조정 설정

엔드포인트에 대해 예약된 조정을 설정하려면 AWS CLI 명령을 사용하여 확장 가능한 대상을 등록한 다음 예약된 작업을 생성합니다. 확장 가능 대상은 추론 단위를 엔드포인트 프로비저닝을 조정하는 데 사용되는 리소스로 정의하고, 예정된 작업은 특정 시간의 프로비저닝된 용량의 Auto Scaling을 제어합니다.

예약된 조정을 설정하려면

1. 확장 가능 대상을 등록합니다. 다음 예에서는 확장 가능 대상을 등록하여 최소 용량은 추론 단위 1개, 최대 용량은 추론 단위 2개로 엔드포인트 프로비저닝을 조정합니다.

   문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

   ```
   aws application-autoscaling register-scalable-target \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:document-classifier-
   endpoint/name \
       --scalable-dimension comprehend:document-classifier-
   endpoint:DesiredInferenceUnits \
       --min-capacity 1 \
       --max-capacity 2
   ```

   개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

   ```
   aws application-autoscaling register-scalable-target \
       --service-namespace comprehend \
   ```

```
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --min-capacity 1 \
    --max-capacity 2
```

2. 예약된 작업을 생성합니다. 다음 예제에서는 최소 2개의 추론 단위와 최대 5개의 추론 단위를 사용하여 매일 12:00(UTC)에 프로비저닝된 용량을 자동으로 조정하는 예정된 작업을 생성합니다. 시간순 표현식 및 예약된 조정에 대한 자세한 내용은 예약 표현식을 참조하세요.

   문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling put-scheduled-action \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits \
    --scheduled-action-name TestScheduledAction \
    --schedule "cron(0 12 * * ? *)" \
    --scalable-target-action MinCapacity=2,MaxCapacity=5
```

   개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling put-scheduled-action \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --scheduled-action-name TestScheduledAction \
    --schedule "cron(0 12 * * ? *)" \
    --scalable-target-action MinCapacity=2,MaxCapacity=5
```

예약된 조정 제거

엔드포인트에 대해 예약된 조정을 제거하려면 AWS CLI 명령을 사용하여 예약된 작업을 삭제한 다음 확장 가능 대상의 등록을 취소합니다.

예약된 조정을 제거하려면

1. 예약된 작업을 삭제합니다. 다음 예제는 지정된 예약된 작업을 삭제합니다.

   문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling delete-scheduled-action \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits \
    --scheduled-action-name TestScheduledAction
```

   개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling delete-scheduled-action \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --scheduled-action-name TestScheduledAction
```

2. 확장 가능 목표의 등록을 취소합니다. 다음 예제는 지정된 확장 가능 대상의 등록을 취소합니다.

   문서 분류 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits
```

   개체 인식기 엔드포인트의 경우 다음 AWS CLI 명령을 사용합니다.

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
```

```
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits
```

리소스에 태그 지정

태그는 사용자가 Amazon Comprehend 리소스에 메타데이터로 추가할 수 있는 키-값 페어입니다. 분석 작업, 사용자 지정 분류 모델, 사용자 지정 개체 인식 모델 및 엔드포인트에서 태그를 사용할 수 있습니다. 태그에는 리소스 구성 및 태그 기반 액세스 제어 제공이라는 두 가지 주요 기능이 있습니다.

태그를 사용하여 리소스를 구성하려면 'Department'라는 태그 키와 'Sales' 또는 'Legal'이라는 태그 값을 추가할 수 있습니다. 그런 다음 회사의 법무 부서와 관련된 리소스를 검색하고 필터링할 수 있습니다.

태그 기반 액세스 제어를 제공하려면 태그 기반 권한이 포함된 IAM 정책을 생성하십시오. 정책은 요청에 제공된 태그(요청 태그) 또는 호출하는 리소스와 관련된 태그(리소스 태그)를 기반으로 작업을 허용하거나 허용하지 않을 수 있습니다. IAM 태그 사용에 대한 자세한 내용은 IAM 사용 설명서의 태그를 사용한 액세스 제어를 참조하세요.

Amazon Comprehend에서 태그를 사용할 때 고려할 사항:

- 리소스당 최대 50개의 태그를 추가할 수 있으며, 태그를 리소스 생성 시 추가하거나 소급하여 추가할 수 있습니다.
- 태그 키는 필수 필드이지만 태그 값은 선택 사항입니다.
- 태그는 리소스 간에 고유할 필요는 없지만 지정된 리소스에 중복된 태그 키를 가질 수는 없습니다.
- 태그 키와 값은 대소문자를 구분합니다.
- 태그 키는 최대 127자이고 태그 값은 최대 255자일 수 있습니다.
- 'aws:' 접두사는 AWS 사용을 위해 예약되어 있습니다. 키가 로 시작하는 태그를 추가, 편집 또는 삭제할 수 없습니다`aws:`. 이 태그는 리소스당 50자의 태그 수 제한에 포함되지 않습니다.

> **Note**
>
> 여러 AWS 서비스 및 리소스에서 태깅 스키마를 사용하려는 경우 다른 서비스에 허용되는 문자에 대한 요구 사항이 다를 수 있다는 점을 기억하세요.

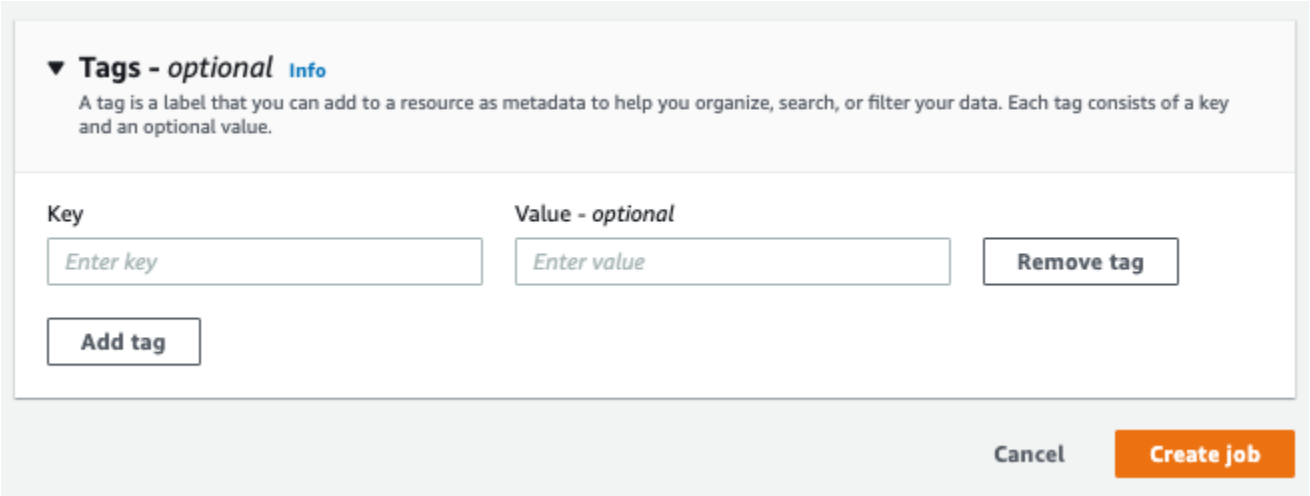
주제

- 새 리소스에 태그 지정
- 리소스와 연결된 태그의 보기, 편집, 삭제

새 리소스에 태그 지정

분석 작업, 사용자 지정 분류 모델 또는 사용자 지정 개체 인식 모델 또는 엔드포인트에 태그를 추가할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 만들려는 리소스(분석 작업, 사용자 지정 분류 또는 사용자 지정 개체 인식)를 선택합니다.
3. 작업 생성(또는 새 모델 생성)을 클릭합니다. 그러면 리소스의 기본 '생성' 페이지로 이동합니다. 이 페이지 하단에 '태그 - 선택 사항' 패널이 표시됩니다.



   태그 키와 태그 값(선택 사항)을 입력합니다. 리소스에 다른 태그를 추가하려면 태그 추가를 선택합니다. 모든 태그가 추가될 때까지 이 프로세스를 반복합니다. 태그 키는 리소스마다 고유해야 합니다.
4. 리소스 생성을 계속하려면 생성 또는 작업 생성 버튼을 선택합니다.

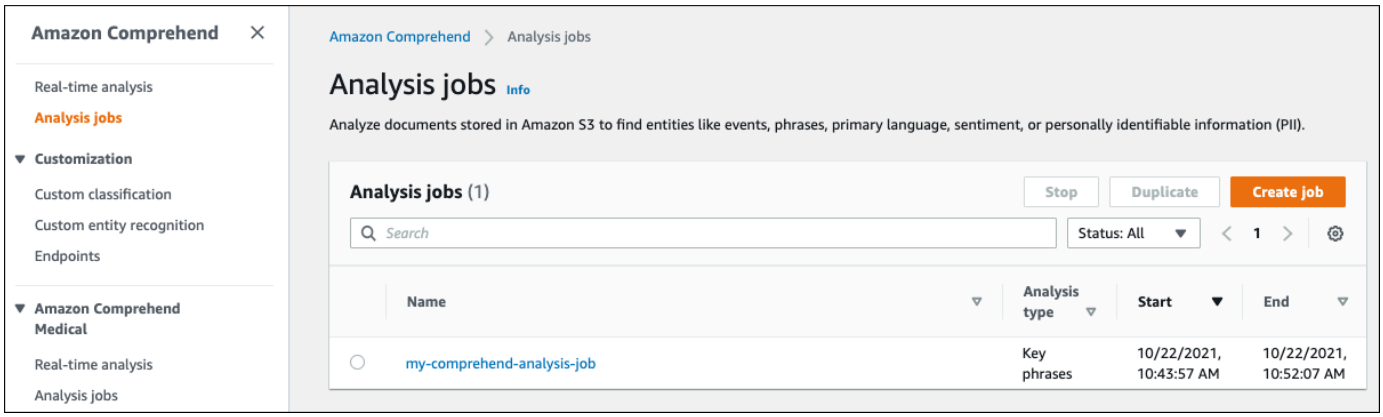
AWS CLI를 사용하여 태그를 추가할 수도 있습니다. 이 예제에서는 start-entities-detection-job 명령을 사용하여 태그를 추가하는 방법을 보여줍니다.

```
aws comprehend start-entities-detection-job \
--language-code "en" \
--input-data-config "{\"S3Uri\": \"s3://test-input/TEST.csv\"}" \
--output-data-config "{\"S3Uri\": \"s3://test-output\"}" \
--data-access-role-arn arn:aws:iam::123456789012:role/test \
--tags "[{\"Key\": \"color\",\"Value\": \"orange\"}]"
```

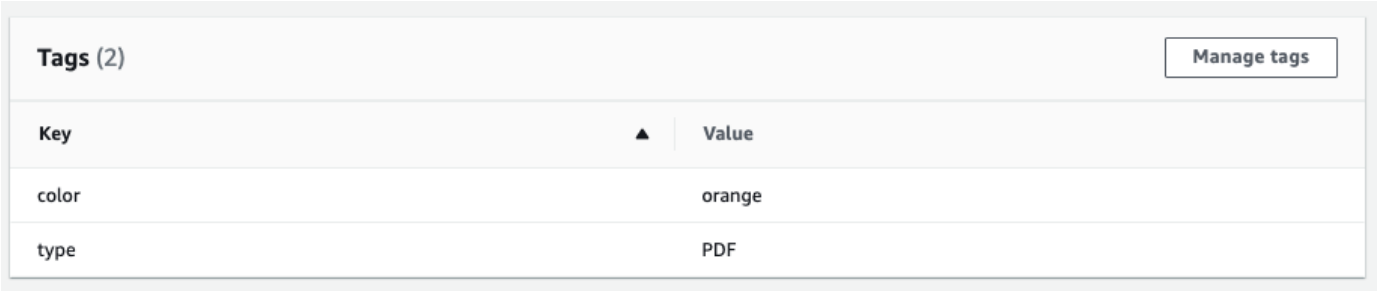
리소스와 연결된 태그의 보기, 편집, 삭제

분석 작업, 사용자 지정 분류 모델 또는 사용자 지정 개체 인식 모델과 관련된 태그를 볼 수 있습니다.

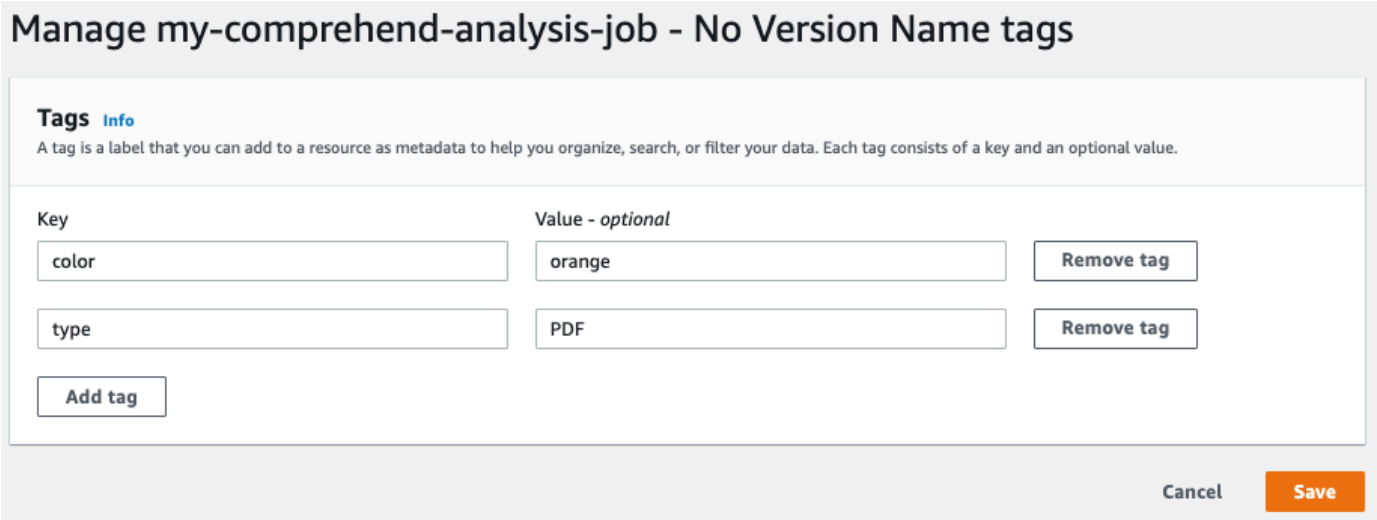
1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/comprehend/ Amazon Comprehend 콘솔을 엽니다.
2. 보거나, 수정하거나, 삭제하려는 태그가 있는 파일이 들어 있는 리소스(분석 작업, 사용자 지정 분류 또는 사용자 지정 개체 인식)를 선택합니다. 그러면 선택한 리소스의 기존 파일 목록이 표시됩니다.



3. 태그를 보거나, 수정하거나, 삭제하려는 파일(또는 모델)의 이름을 클릭합니다. 그러면 해당 파일(또는 모델)에 대한 세부 정보 페이지로 이동합니다. 태그 상자가 표시될 때까지 아래로 스크롤합니다. 여기에서 선택한 파일(또는 모델)과 관련된 모든 태그를 볼 수 있습니다.



   태그 관리를 선택하여 리소스에서 태그를 편집하거나 제거합니다.
4. 수정하려는 텍스트를 클릭한 다음 태그를 편집합니다. 태그 제거를 선택하여 태그를 제거할 수도 있습니다. 새 태그를 추가하려면 태그 추가를 선택한 다음 빈 필드에 원하는 텍스트를 입력합니다.



태그 수정을 완료하면 저장을 선택합니다.

AWS SDKs를 사용하는 Amazon Comprehend의 코드 예제

다음 코드 예제에서는 AWS 소프트웨어 개발 키트(SDK)와 함께 Amazon Comprehend를 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDKs 사용한 Amazon Comprehend의 기본 예제
  - AWS SDKs를 사용한 Amazon Comprehend 작업
    - AWS SDK 또는 CLI와 CreateDocumentClassifier 함께 사용
    - AWS SDK 또는 CLI와 DeleteDocumentClassifier 함께 사용
    - AWS SDK 또는 CLI와 DescribeDocumentClassificationJob 함께 사용
    - AWS SDK 또는 CLI와 DescribeDocumentClassifier 함께 사용
    - AWS SDK 또는 CLI와 DescribeTopicsDetectionJob 함께 사용
    - AWS SDK 또는 CLI와 DetectDominantLanguage 함께 사용
    - AWS SDK 또는 CLI와 DetectEntities 함께 사용
    - AWS SDK 또는 CLI와 DetectKeyPhrases 함께 사용
    - AWS SDK 또는 CLI와 DetectPiiEntities 함께 사용
    - AWS SDK 또는 CLI와 DetectSentiment 함께 사용
    - AWS SDK 또는 CLI와 DetectSyntax 함께 사용
    - AWS SDK 또는 CLI와 ListDocumentClassificationJobs 함께 사용
    - AWS SDK 또는 CLI와 ListDocumentClassifiers 함께 사용
    - AWS SDK 또는 CLI와 ListTopicsDetectionJobs 함께 사용
    - AWS SDK 또는 CLI와 StartDocumentClassificationJob 함께 사용
    - AWS SDK 또는 CLI와 StartTopicsDetectionJob 함께 사용

- AWS SDKs를 사용한 Amazon Comprehend 시나리오
  - Amazon Transcribe 스트리밍 앱 구축
  - 웹 사이트 방문자의 참여를 유도하는 Amazon Lex 챗봇 생성
  - Amazon SQS를 사용하여 메시지를 보내고 검색하는 웹 애플리케이션 생성
  - 고객 피드백을 분석하고 오디오를 합성하는 애플리케이션 생성
  - Amazon Comprehend 및 AWS SDK를 사용하여 문서 요소 감지
  - AWS SDK를 사용하여 이미지에서 추출된 텍스트의 개체 감지
  - AWS SDK를 사용하여 샘플 데이터에 대한 Amazon Comprehend 주제 모델링 작업 실행
  - AWS SDK를 사용하여 사용자 지정 Amazon Comprehend 분류자 훈련 및 문서 분류

AWS SDKs 사용한 Amazon Comprehend의 기본 예제

다음 코드 예제에서는 SDKs와 함께 AWS Amazon Comprehend의 기본 사항을 사용하는 방법을 보여 줍니다.

예시

- AWS SDKs를 사용한 Amazon Comprehend 작업
  - AWS SDK 또는 CLI와 CreateDocumentClassifier 함께 사용
  - AWS SDK 또는 CLI와 DeleteDocumentClassifier 함께 사용
  - AWS SDK 또는 CLI와 DescribeDocumentClassificationJob 함께 사용
  - AWS SDK 또는 CLI와 DescribeDocumentClassifier 함께 사용
  - AWS SDK 또는 CLI와 DescribeTopicsDetectionJob 함께 사용
  - AWS SDK 또는 CLI와 DetectDominantLanguage 함께 사용
  - AWS SDK 또는 CLI와 DetectEntities 함께 사용
  - AWS SDK 또는 CLI와 DetectKeyPhrases 함께 사용
  - AWS SDK 또는 CLI와 DetectPiiEntities 함께 사용
  - AWS SDK 또는 CLI와 DetectSentiment 함께 사용
  - AWS SDK 또는 CLI와 DetectSyntax 함께 사용
  - AWS SDK 또는 CLI와 ListDocumentClassificationJobs 함께 사용
  - AWS SDK 또는 CLI와 ListDocumentClassifiers 함께 사용
  - AWS SDK 또는 CLI와 ListTopicsDetectionJobs 함께 사용
  - AWS SDK 또는 CLI와 StartDocumentClassificationJob 함께 사용

- AWS SDK 또는 CLI와 StartTopicsDetectionJob 함께 사용

AWS SDKs를 사용한 Amazon Comprehend 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 Amazon Comprehend 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이들 발췌문은 Amazon Comprehend API를 직접 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDKs를 사용한 Amazon Comprehend 시나리오 에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon Comprehend API 참조를 참조하세요.

예시

- AWS SDK 또는 CLI와 CreateDocumentClassifier 함께 사용
- AWS SDK 또는 CLI와 DeleteDocumentClassifier 함께 사용
- AWS SDK 또는 CLI와 DescribeDocumentClassificationJob 함께 사용
- AWS SDK 또는 CLI와 DescribeDocumentClassifier 함께 사용
- AWS SDK 또는 CLI와 DescribeTopicsDetectionJob 함께 사용
- AWS SDK 또는 CLI와 DetectDominantLanguage 함께 사용
- AWS SDK 또는 CLI와 DetectEntities 함께 사용
- AWS SDK 또는 CLI와 DetectKeyPhrases 함께 사용
- AWS SDK 또는 CLI와 DetectPiiEntities 함께 사용
- AWS SDK 또는 CLI와 DetectSentiment 함께 사용
- AWS SDK 또는 CLI와 DetectSyntax 함께 사용
- AWS SDK 또는 CLI와 ListDocumentClassificationJobs 함께 사용
- AWS SDK 또는 CLI와 ListDocumentClassifiers 함께 사용
- AWS SDK 또는 CLI와 ListTopicsDetectionJobs 함께 사용
- AWS SDK 또는 CLI와 StartDocumentClassificationJob 함께 사용
- AWS SDK 또는 CLI와 StartTopicsDetectionJob 함께 사용

AWS SDK 또는 CLI와 **CreateDocumentClassifier** 함께 사용

다음 코드 예제는 CreateDocumentClassifier의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 지정 분류기 학습 및 문서 분류

CLI

AWS CLI

문서 분류자를 만들어 문서 분류

다음 create-document-classifier 예제에서는 문서 분류자 모델의 학습 프로세스를 시작합니다. 교육 데이터 파일 training.csv는 --input-data-config 태그에 있습니다. training.csv는 첫 번째 열에 레이블 또는 분류가 제공되고 두 번째 열에 문서가 제공되는 2열 문서입니다.

```
aws comprehend create-document-classifier \
    --document-classifier-name example-classifier \
    --data-access-arn arn:aws:comprehend:us-west-2:111122223333:pii-entities-
detection-job/123456abcdeb0e11022f22a11EXAMPLE \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --language-code en
```

출력:

```
{
    "DocumentClassifierArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/example-classifier"
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 사용자 지정 분류 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateDocumentClassifier를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
 software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-
using-amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
                  dataAccessRoleArn - The ARN value of the role used for this
 operation.
```

```
                  s3Uri - The Amazon S3 bucket that contains the CSV file.
                  documentClassifierName - The name of the document classifier.
                """";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dataAccessRoleArn = args[0];
        String s3Uri = args[1];
        String documentClassifierName = args[2];

        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
        comClient.close();
    }

    public static void createDocumentClassifier(ComprehendClient comClient,
String dataAccessRoleArn, String s3Uri,
            String documentClassifierName) {
        try {
            DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
                    .s3Uri(s3Uri)
                    .build();

            CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
                    .documentClassifierName(documentClassifierName)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .languageCode("en")
                    .inputDataConfig(config)
                    .build();

            CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
                    .createDocumentClassifier(createDocumentClassifierRequest);
```

```
            String documentClassifierArn =
 createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " +
 documentClassifierArn);

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x SDK API 참조의 CreateDocumentClassifier를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
```

```
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
immediately
        starts training on the data found in the specified Amazon S3 bucket.
Training
        can take 30 minutes or longer. The `describe_document_classifier`
function
        can be used to get training status and returns a status of TRAINED when
the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
data.
        :param training_key: The prefix used to find training data in the
training
                             bucket. If multiple objects have the same prefix,
all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
```

```
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateDocumentClassifier을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteDocumentClassifier** 함께 사용

다음 코드 예제는 DeleteDocumentClassifier의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 지정 분류기 학습 및 문서 분류

CLI

AWS CLI

사용자 지정 문서 분류자 삭제

다음 delete-document-classifier 예제에서는 사용자 지정 문서 분류자 모델을 삭제합니다.

```
aws comprehend delete-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/example-classifier-1
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Comprehend 개발자 안내서의 Amazon Comprehend 엔드포인트 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteDocumentClassifier를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
 self.classifier_arn)
            raise
```

- API에 대한 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteDocumentClassifier를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeDocumentClassificationJob** 함께 사용

다음 코드 예제는 DescribeDocumentClassificationJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 지정 분류기 학습 및 문서 분류

CLI

AWS CLI

문서 분류 작업 설명

다음 describe-document-classification-job 예제는 비동기 문서 분류 작업의 속성을 가져옵니다.

```
aws comprehend describe-document-classification-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

출력:

```
{
    "DocumentClassificationJobProperties": {
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "exampleclassificationjob",
        "JobStatus": "COMPLETED",
        "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
        "EndTime": "2023-06-14T17:15:58.582000+00:00",
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/mymodel/version/1",
```

```
        "InputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-destination-bucket/
testfolder/111122223333-CLN-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-servicerole"
    }
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 사용자 지정 분류를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 DescribeDocumentClassificationJob를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.
```

```
        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response =
self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeDocumentClassificationJob을 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeDocumentClassifier** 함께 사용

다음 코드 예제는 DescribeDocumentClassifier의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 지정 분류기 학습 및 문서 분류

CLI

AWS CLI

문서 분류기 설명

다음 describe-document-classifier 예제에서는 사용자 지정 문서 분류자 모델을 삭제합니다.

```
aws comprehend describe-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/example-classifier-1
```

출력:

```
{
    "DocumentClassifierProperties": {
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/example-classifier-1",
        "LanguageCode": "en",
        "Status": "TRAINED",
        "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
        "EndTime": "2023-06-13T19:42:31.752000+00:00",
        "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
        "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
        "InputDataConfig": {
            "DataFormat": "COMPREHEND_CSV",
            "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
        },
        "OutputDataConfig": {},
        "ClassifierMetadata": {
            "NumberOfLabels": 3,
            "NumberOfTrainedDocuments": 5016,
            "NumberOfTestDocuments": 557,
            "EvaluationMetrics": {
                "Accuracy": 0.9856,
                "Precision": 0.9919,
                "Recall": 0.9459,
                "F1Score": 0.9673,
                "MicroPrecision": 0.9856,
                "MicroRecall": 0.9856,
                "MicroF1Score": 0.9856,
                "HammingLoss": 0.0144
            }
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role",
        "Mode": "MULTI_CLASS"
    }
```

```
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 사용자 지정 모델 생성 및 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 DescribeDocumentClassifier를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
```

```
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeDocumentClassifier를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeTopicsDetectionJob** 함께 사용

다음 코드 예제는 DescribeTopicsDetectionJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 샘플 데이터에 대한 주제 모델링 작업 실행

CLI

AWS CLI

주제 탐지 작업 설명

다음 describe-topics-detection-job 예제는 비동기 주제 탐지 작업의 속성을 가져옵니다.

```
aws comprehend describe-topics-detection-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

출력:

```
{
```

```
    "TopicsDetectionJobProperties": {
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "example_topics_detection",
        "JobStatus": "IN_PROGRESS",
        "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
        "InputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-destination-bucket/
testfolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "NumberOfTopics": 10,
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-examplerole"
    }
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 Amazon Comprehend 통찰력 비동기 분석을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조에서 DescribeTopicsDetectionJob을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
```

```
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DescribeTopicsDetectionJob를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`DetectDominantLanguage`** 함께 사용

다음 코드 예제는 `DetectDominantLanguage`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 문서 요소 감지

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example calls the Amazon Comprehend service to determine the
    /// dominant language.
    /// </summary>
    public static class DetectDominantLanguage
    {
        /// <summary>
        /// Calls Amazon Comprehend to determine the dominant language used in
        /// the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle.";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            Console.WriteLine("Calling DetectDominantLanguage\n");
            var detectDominantLanguageRequest = new
DetectDominantLanguageRequest()
            {
                Text = text,
            };

            var detectDominantLanguageResponse = await
comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
            foreach (var dl in detectDominantLanguageResponse.Languages)
```

```
            {
                Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
 {dl.Score}");
            }

            Console.WriteLine("Done");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 DetectDominantLanguage를 참조하세요.

CLI

AWS CLI

입력 텍스트에서 주로 사용되는 언어 탐지

다음 `detect-dominant-language`은(는) 입력 텍스트를 분석하고 주로 사용되는 언어를 식별합니다. 사전 훈련된 모델의 신뢰도 점수도 출력됩니다.

```
aws comprehend detect-dominant-language \
    --text "It is a beautiful day in Seattle."
```

출력:

```
{
    "Languages": [
        {
            "LanguageCode": "en",
            "Score": 0.9877256155014038
        }
    ]
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 주로 사용되는 언어를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetectDominantLanguage를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
```

```
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient,
 String text) {
        try {
            DetectDominantLanguageRequest request =
 DetectDominantLanguageRequest.builder()
                    .text(text)
                    .build();

            DetectDominantLanguageResponse resp =
 comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectDominantLanguage를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DetectDominantLanguage를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DetectEntities** 함께 사용

다음 코드 예제는 DetectEntities의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 문서 요소 감지

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the AmazonComprehend service detect any
    /// entities in submitted text.
    /// </summary>
    public static class DetectEntities
    {
        /// <summary>
        /// The main method calls the DetectEntitiesAsync method to find any
        /// entities in the sample code.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new AmazonComprehendClient();

            Console.WriteLine("Calling DetectEntities\n");
            var detectEntitiesRequest = new DetectEntitiesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

            foreach (var e in detectEntitiesResponse.Entities)
            {
```

```
                Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score:
 {e.Score}, BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 DetectEntities를 참조하세요.

CLI

AWS CLI

입력 텍스트에서 이름이 지정된 엔터티를 감지하려면

다음 detect-entities 예제에서는 입력 텍스트를 분석하고 이름이 지정된 엔터티를 반환합니다. 각 예측에 대한 사전 훈련된 모델의 신뢰도 점수도 출력됩니다.

```
aws comprehend detect-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
    account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by
 July 31st. Based on your autopay settings, \
    we will withdraw your payment on the due date from your bank account number
 XXXXXX1111 with the routing number XXXXX0000. \
    Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to
 Alice at AnySpa@example.com."
```

출력:

```
{
    "Entities": [
        {
            "Score": 0.9994556307792664,
            "Type": "PERSON",
            "Text": "Zhang Wei",
            "BeginOffset": 6,
            "EndOffset": 15
```

```
        },
        {
            "Score": 0.9981022477149963,
            "Type": "PERSON",
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9986887574195862,
            "Type": "ORGANIZATION",
            "Text": "AnyCompany Financial Services, LLC",
            "BeginOffset": 33,
            "EndOffset": 67
        },
        {
            "Score": 0.9959119558334351,
            "Type": "OTHER",
            "Text": "1111-XXXX-1111-XXXX",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9708039164543152,
            "Type": "QUANTITY",
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9987268447875977,
            "Type": "DATE",
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9858865737915039,
            "Type": "OTHER",
            "Text": "XXXXXX1111",
            "BeginOffset": 271,
            "EndOffset": 281
        },
        {
```

```
            "Score": 0.9700471758842468,
            "Type": "OTHER",
            "Text": "XXXXX0000",
            "BeginOffset": 306,
            "EndOffset": 315
        },
        {
            "Score": 0.9591118693351746,
            "Type": "ORGANIZATION",
            "Text": "Sunshine Spa",
            "BeginOffset": 340,
            "EndOffset": 352
        },
        {
            "Score": 0.9797496795654297,
            "Type": "LOCATION",
            "Text": "123 Main St",
            "BeginOffset": 354,
            "EndOffset": 365
        },
        {
            "Score": 0.994929313659668,
            "Type": "PERSON",
            "Text": "Alice",
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9949769377708435,
            "Type": "OTHER",
            "Text": "AnySpa@example.com",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```


자세한 내용은 Amazon Comprehend 개발자 안내서의 엔티티를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetectEntities를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
       Region region = Region.US_EAST_1;
       ComprehendClient comClient = ComprehendClient.builder()
               .region(region)
               .build();

       System.out.println("Calling DetectEntities");
       detectAllEntities(comClient, text);
       comClient.close();
    }
```

```
    public static void detectAllEntities(ComprehendClient comClient, String text)
 {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
 DetectEntitiesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectEntitiesResponse detectEntitiesResult =
 comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectEntities를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
```

```
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DetectEntities를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DetectKeyPhrases** 함께 사용

다음 코드 예제는 DetectKeyPhrases의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 문서 요소 감지

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the Amazon Comprehend service to
    /// search text for key phrases.
    /// </summary>
    public static class DetectKeyPhrase
    {
        /// <summary>
        /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync
        /// to detect any key phrases in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectKeyPhrases");
            var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
```

```
            var detectKeyPhrasesResponse = await
 comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
            foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
            {
                Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score},
 BeginOffset: {kp.BeginOffset}, EndOffset: {kp.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 DetectKeyPhrases를 참조하세요.

CLI

AWS CLI

입력 텍스트에서 핵심 문구 탐지

다음 `detect-key-phrases` 예제에서는 입력 텍스트를 분석하고 핵심 명사구를 식별합니다. 각 예측에 대한 사전 훈련된 모델의 신뢰도 점수도 출력됩니다.

```
aws comprehend detect-key-phrases \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

출력:

```
{
    "KeyPhrases": [
        {
            "Score": 0.8996376395225525,
```

```
            "Text": "Zhang Wei",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9992469549179077,
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.988385021686554,
            "Text": "Your AnyCompany Financial Services",
            "BeginOffset": 28,
            "EndOffset": 62
        },
        {
            "Score": 0.8740853071212769,
            "Text": "LLC credit card account 1111-XXXX-1111-XXXX",
            "BeginOffset": 64,
            "EndOffset": 107
        },
        {
            "Score": 0.9999437928199768,
            "Text": "a minimum payment",
            "BeginOffset": 112,
            "EndOffset": 129
        },
        {
            "Score": 0.9998900890350342,
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9979453086853027,
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9983011484146118,
            "Text": "your autopay settings",
            "BeginOffset": 172,
```

```
            "EndOffset": 193
        },
        {
            "Score": 0.9996572136878967,
            "Text": "your payment",
            "BeginOffset": 211,
            "EndOffset": 223
        },
        {
            "Score": 0.9995037317276001,
            "Text": "the due date",
            "BeginOffset": 227,
            "EndOffset": 239
        },
        {
            "Score": 0.9702621698379517,
            "Text": "your bank account number XXXXXX1111",
            "BeginOffset": 245,
            "EndOffset": 280
        },
        {
            "Score": 0.9179925918579102,
            "Text": "the routing number XXXXX0000.Customer feedback",
            "BeginOffset": 286,
            "EndOffset": 332
        },
        {
            "Score": 0.9978160858154297,
            "Text": "Sunshine Spa",
            "BeginOffset": 337,
            "EndOffset": 349
        },
        {
            "Score": 0.9706913232803345,
            "Text": "123 Main St",
            "BeginOffset": 351,
            "EndOffset": 362
        },
        {
            "Score": 0.9941995143890381,
            "Text": "comments",
            "BeginOffset": 379,
            "EndOffset": 387
        },
```

```
        {
            "Score": 0.9759287238121033,
            "Text": "Alice",
            "BeginOffset": 391,
            "EndOffset": 396
        },
        {
            "Score": 0.8376792669296265,
            "Text": "AnySpa@example.com",
            "BeginOffset": 400,
            "EndOffset": 415
        }
    ]
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 핵심 문구를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetectKeyPhrases를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String
 text) {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
 DetectKeyPhrasesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
 comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectKeyPhrases를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
 its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
```

```
            return phrases
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DetectKeyPhrases를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `DetectPiiEntities` 함께 사용

다음 코드 예제는 `DetectPiiEntities`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 문서 요소 감지

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the Amazon Comprehend service to find
/// personally identifiable information (PII) within text submitted to the
/// DetectPiiEntitiesAsync method.
/// </summary>
```

```
    public class DetectingPII
    {
        /// <summary>
        /// This method calls the DetectPiiEntitiesAsync method to locate any
        /// personally dientifiable information within the supplied text.
        /// </summary>
        public static async Task Main()
        {
            var comprehendClient = new AmazonComprehendClient();
            var text = @"Hello Paul Santos. The latest statement for your
                         credit card account 1111-0000-1111-0000 was
                         mailed to 123 Any Street, Seattle, WA 98109.";

            var request = new DetectPiiEntitiesRequest
            {
                Text = text,
                LanguageCode = "EN",
            };

            var response = await
comprehendClient.DetectPiiEntitiesAsync(request);

            if (response.Entities.Count > 0)
            {
                foreach (var entity in response.Entities)
                {
                    var entityValue = text.Substring(entity.BeginOffset,
entity.EndOffset - entity.BeginOffset);
                    Console.WriteLine($"{entity.Type}: {entityValue}");
                }
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DetectPiiEntities를 참조하세요.

CLI

AWS CLI

입력 텍스트에서 pii 엔티티 탐지

다음 detect-pii-entities 예제는 입력 텍스트를 분석하고 개인 식별 정보(PII)가 포함된 엔티티를 식별합니다. 각 예측에 대한 사전 훈련된 모델의 신뢰도 점수도 출력됩니다.

```
aws comprehend detect-pii-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

출력:

```
{
    "Entities": [
        {
            "Score": 0.9998322129249573,
            "Type": "NAME",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9998878240585327,
            "Type": "NAME",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9994089603424072,
            "Type": "CREDIT_DEBIT_NUMBER",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9999760985374451,
            "Type": "DATE_TIME",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
```

```
            "Score": 0.9999449253082275,
            "Type": "BANK_ACCOUNT_NUMBER",
            "BeginOffset": 271,
            "EndOffset": 281
        },
        {
            "Score": 0.9999847412109375,
            "Type": "BANK_ROUTING",
            "BeginOffset": 306,
            "EndOffset": 315
        },
        {
            "Score": 0.999925434589386,
            "Type": "ADDRESS",
            "BeginOffset": 354,
            "EndOffset": 365
        },
        {
            "Score": 0.9989161491394043,
            "Type": "NAME",
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9994171857833862,
            "Type": "EMAIL",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```


자세한 내용은 Amazon Comprehend 개발자 안내서의 개인 식별 정보(PII)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetectPiiEntities를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can
 be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DetectPiiEntities를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DetectSentiment** 함께 사용

다음 코드 예제는 DetectSentiment의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 문서 요소 감지

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service.
/// </summary>
public static class DetectSentiment
{
    /// <summary>
    /// This method calls the DetetectSentimentAsync method to analyze the
```

```
        /// supplied text and determine the overal sentiment.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectSentiment");
            var detectSentimentRequest = new DetectSentimentRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectSentimentResponse = await
comprehendClient.DetectSentimentAsync(detectSentimentRequest);
            Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
            Console.WriteLine("Done");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 DetectSentiment를 참조하세요.

CLI

AWS CLI

입력 텍스트의 감정 탐지

다음 `detect-sentiment` 예제는 입력 텍스트를 분석하고 일반적인 감정(`POSITIVE`, `NEUTRAL`, `MIXED` 또는 `NEGATIVE`)에 대한 추론을 반환합니다.

```
aws comprehend detect-sentiment \
    --language-code en \
    --text "It is a beautiful day in Seattle"
```

출력:

```
{
    "Sentiment": "POSITIVE",
    "SentimentScore": {
        "Positive": 0.9976957440376282,
        "Negative": 9.653854067437351e-05,
        "Neutral": 0.002169104292988777,
        "Mixed": 3.857641786453314e-05
    }
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 Sentiment 섹션을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetectSentiment를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectSentiment {
    public static void main(String[] args) {
```

```
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text)
 {
        try {
            DetectSentimentRequest detectSentimentRequest =
 DetectSentimentRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
 comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
 detectSentimentResult.sentimentScore().neutral());

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectSentiment를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DetectSentiment를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DetectSyntax** 함께 사용

다음 코드 예제는 DetectSyntax의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 문서 요소 감지

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use Amazon Comprehend to detect syntax
/// elements by calling the DetectSyntaxAsync method.
/// </summary>
public class DetectingSyntax
{
    /// <summary>
    /// This method calls DetectSynaxAsync to identify the syntax elements
    /// in the sample text.
    /// </summary>
```

```
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new AmazonComprehendClient();

            // Call DetectSyntax API
            Console.WriteLine("Calling DetectSyntaxAsync\n");
            var detectSyntaxRequest = new DetectSyntaxRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            DetectSyntaxResponse detectSyntaxResponse = await
 comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
            foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
            {
                Console.WriteLine($"Text: {s.Text}, PartOfSpeech:
 {s.PartOfSpeech.Tag}, BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for .NET API 참조의 DetectSyntax를 참조하세요.

CLI

AWS CLI

입력 텍스트에서 품사 탐지

다음 `detect-syntax` 예제에서는 입력 텍스트의 구문을 분석하고 품사의 여러 부분을 반환합니다. 각 예측에 대한 사전 훈련된 모델의 신뢰도 점수도 출력됩니다.

```
aws comprehend detect-syntax \
    --language-code en \
    --text "It is a beautiful day in Seattle."
```

출력:


```
{
    "SyntaxTokens": [
        {
            "TokenId": 1,
            "Text": "It",
            "BeginOffset": 0,
            "EndOffset": 2,
            "PartOfSpeech": {
                "Tag": "PRON",
                "Score": 0.9999740719795227
            }
        },
        {
            "TokenId": 2,
            "Text": "is",
            "BeginOffset": 3,
            "EndOffset": 5,
            "PartOfSpeech": {
                "Tag": "VERB",
                "Score": 0.999901294708252
            }
        },
        {
            "TokenId": 3,
            "Text": "a",
            "BeginOffset": 6,
            "EndOffset": 7,
            "PartOfSpeech": {
                "Tag": "DET",
                "Score": 0.9999938607215881
            }
        },
        {
            "TokenId": 4,
            "Text": "beautiful",
            "BeginOffset": 8,
            "EndOffset": 17,
            "PartOfSpeech": {
                "Tag": "ADJ",
                "Score": 0.9987351894378662
            }
        },
```

```
        {
            "TokenId": 5,
            "Text": "day",
            "BeginOffset": 18,
            "EndOffset": 21,
            "PartOfSpeech": {
                "Tag": "NOUN",
                "Score": 0.9999796748161316
            }
        },
        {
            "TokenId": 6,
            "Text": "in",
            "BeginOffset": 22,
            "EndOffset": 24,
            "PartOfSpeech": {
                "Tag": "ADP",
                "Score": 0.9998047947883606
            }
        },
        {
            "TokenId": 7,
            "Text": "Seattle",
            "BeginOffset": 25,
            "EndOffset": 32,
            "PartOfSpeech": {
                "Tag": "PROPN",
                "Score": 0.9940530061721802
            }
        }
    ]
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 구문 분석을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetectSyntax를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
       Region region = Region.US_EAST_1;
       ComprehendClient comClient = ComprehendClient.builder()
               .region(region)
               .build();

       System.out.println("Calling DetectSyntax");
       detectAllSyntax(comClient, text);
       comClient.close();
    }
```

```
    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest =
 DetectSyntaxRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSyntaxResponse detectSyntaxResult =
 comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
 token.partOfSpeech().tagAsString());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectSyntax를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
```

```
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DetectSyntax를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`ListDocumentClassificationJobs`** 함께 사용

다음 코드 예제는 `ListDocumentClassificationJobs`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 지정 분류기 학습 및 문서 분류

CLI

AWS CLI

모든 문서 분류 작업 나열

다음 `list-document-classification-jobs` 예제에는 모든 문서 분류 작업이 나열되어 있습니다.

```
aws comprehend list-document-classification-jobs
```

출력:

```
{
    "DocumentClassificationJobPropertiesList": [
        {
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName": "exampleclassificationjob",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
            "EndTime": "2023-06-14T17:15:58.582000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/1234567890101-CLN-e758dd56b824aa717ceab551f11749fb/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
```

```
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "exampleclassificationjob2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:22:39.829000+00:00",
            "EndTime": "2023-06-14T17:28:46.107000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/1234567890101-CLN-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 사용자 지정 분류를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListDocumentClassificationJobs를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListDocumentClassificationJobs를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListDocumentClassifiers** 함께 사용

다음 코드 예제는 ListDocumentClassifiers의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 지정 분류기 학습 및 문서 분류

CLI

AWS CLI

모든 문서 분류 작업 나열

다음 `list-document-classifiers` 예제에는 학습된 문서 분류자 모델과 학습 중인 문서 분류자 모델이 모두 나열되어 있습니다.

```
aws comprehend list-document-classifiers
```

출력:

```
{
    "DocumentClassifierPropertiesList": [
        {
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier1",
            "LanguageCode": "en",
            "Status": "TRAINED",
            "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
            "EndTime": "2023-06-13T19:42:31.752000+00:00",
            "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
            "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
                "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
            },
            "OutputDataConfig": {},
            "ClassifierMetadata": {
                "NumberOfLabels": 3,
                "NumberOfTrainedDocuments": 5016,
                "NumberOfTestDocuments": 557,
                "EvaluationMetrics": {
                    "Accuracy": 0.9856,
                    "Precision": 0.9919,
                    "Recall": 0.9459,
                    "F1Score": 0.9673,
                    "MicroPrecision": 0.9856,
                    "MicroRecall": 0.9856,
```

```
                    "MicroF1Score": 0.9856,
                    "HammingLoss": 0.0144
                }
            },
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        },
        {
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier2",
            "LanguageCode": "en",
            "Status": "TRAINING",
            "SubmitTime": "2023-06-13T21:20:28.690000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
                "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
            },
            "OutputDataConfig": {},
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        }
    ]
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 사용자 지정 모델 생성 및 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListDocumentClassifiers를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendClassifier:
```

```
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListDocumentClassifiers를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListTopicsDetectionJobs** 함께 사용

다음 코드 예제는 ListTopicsDetectionJobs의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 샘플 데이터에 대한 주제 모델링 작업 실행

CLI

AWS CLI

모든 주제 탐지 작업 나열

다음 `list-topics-detection-jobs` 예제는 진행 중인 모든 비동기 주제 탐지 작업과 완료된 비동기 주제 탐지 작업을 나열합니다.

```
aws comprehend list-topics-detection-jobs
```

출력:

```
{
    "TopicsDetectionJobPropertiesList": [
        {
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName" "topic-analysis-1"
            "JobStatus": "IN_PROGRESS",
            "SubmitTime": "2023-06-09T18:40:35.384000+00:00",
            "EndTime": "2023-06-09T18:46:41.936000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
```

```
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "topic-analysis-2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
            "EndTime": "2023-06-09T18:50:50.872000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE3",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE3",
            "JobName": "topic-analysis-2",
            "JobStatus": "IN_PROGRESS",
            "SubmitTime": "2023-06-09T18:50:56.737000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE3/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 Amazon Comprehend 통찰력 비동기 분석을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListTopicsDetectionJobs를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListTopicsDetectionJobs를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **StartDocumentClassificationJob** 함께 사용

다음 코드 예제는 StartDocumentClassificationJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 지정 분류기 학습 및 문서 분류

CLI

AWS CLI

문서 분류 작업 나열

다음 start-document-classification-job 예제에서는 --input-data-config 태그로 지정된 주소의 모든 파일에서 사용자 지정 모델을 사용하여 문서 분류 작업을 시작합니다. 이 예제에서 입력 S3 버킷에는 SampleSMStext1.txt, SampleSMStext2.txt 및 SampleSMStext3.txt가 포함되어 있습니다. 이 모델은 이전에 스팸과 비스팸 또는 “햄”, SMS 메시지의 문서 분류에 대해 학습되었습니다. 작업이 완료되면 --output-data-config 태그에 지정된 위치에 output.tar.gz가 배치됩니다. output.tar.gz에는 각 문서의 분류가 나열되는 predictions.jsonl이 들어 있습니다. JSON 출력은 파일당 한 줄로 출력되지만 여기서는 가독성을 위해 형식이 지정되어 있습니다.

```
aws comprehend start-document-classification-job \
    --job-name exampleclassificationjob \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket-INPUT/jobdata/" \
    --output-data-config "S3Uri=s3://amzn-s3-demo-destination-bucket/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
```

```
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-
classifier/mymodel/version/12
```

SampleSMStext1.txt의 콘텐츠:

```
"CONGRATULATIONS! TXT 2155550100 to win $5000"
```

SampleSMStext2.txt의 콘텐츠:

```
"Hi, when do you want me to pick you up from practice?"
```

SampleSMStext3.txt의 콘텐츠:

```
"Plz send bank account # to 2155550100 to claim prize!!"
```

출력:

```
{
    "JobId": "e758dd56b824aa717ceab551fEXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-classification-
job/e758dd56b824aa717ceab551fEXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

predictions.jsonl의 콘텐츠:

```
{"File": "SampleSMSText1.txt", "Line": "0", "Classes": [{"Name": "spam", "Score":
 0.9999}, {"Name": "ham", "Score": 0.0001}]}
{"File": "SampleSMStext2.txt", "Line": "0", "Classes": [{"Name": "ham", "Score":
 0.9994}, {"Name": "spam", "Score": 0.0006}]}
{"File": "SampleSMSText3.txt", "Line": "0", "Classes": [{"Name": "spam", "Score":
 0.9999}, {"Name": "ham", "Score": 0.0001}]}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 사용자 지정 분류를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartDocumentClassificationJob을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
 can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
```

```
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 StartDocumentClassificationJob을 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **StartTopicsDetectionJob** 함께 사용

다음 코드 예제는 StartTopicsDetectionJob의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 샘플 데이터에 대한 주제 모델링 작업 실행

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example scans the documents in an Amazon Simple Storage Service
/// (Amazon S3) bucket and analyzes it for topics. The results are stored
/// in another bucket and then the resulting job properties are displayed
/// on the screen. This example was created using the AWS SDK for .NEt
/// version 3.7 and .NET Core version 5.0.
/// </summary>
public static class TopicModeling
{
    /// <summary>
    /// This methos calls a topic detection job by calling the Amazon
    /// Comprehend StartTopicsDetectionJobRequest.
    /// </summary>
```

```
        public static async Task Main()
        {
            var comprehendClient = new AmazonComprehendClient();

            string inputS3Uri = "s3://input bucket/input path";
            InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
            string outputS3Uri = "s3://output bucket/output path";
            string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access
role";
            int numberOfTopics = 10;

            var startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
            {
                InputDataConfig = new InputDataConfig()
                {
                    S3Uri = inputS3Uri,
                    InputFormat = inputDocFormat,
                },
                OutputDataConfig = new OutputDataConfig()
                {
                    S3Uri = outputS3Uri,
                },
                DataAccessRoleArn = dataAccessRoleArn,
                NumberOfTopics = numberOfTopics,
            };

            var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

            var jobId = startTopicsDetectionJobResponse.JobId;
            Console.WriteLine("JobId: " + jobId);

            var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
            {
                JobId = jobId,
            };

            var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);
```

```
            var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new
ListTopicsDetectionJobsRequest());
            foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
            {
                PrintJobProperties(props);
            }
        }

        /// <summary>
        /// This method is a helper method that displays the job properties
        /// from the call to StartTopicsDetectionJobRequest.
        /// </summary>
        /// <param name="props">A list of properties from the call to
        /// StartTopicsDetectionJobRequest.</param>
        private static void PrintJobProperties(TopicsDetectionJobProperties
props)
        {
            Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
            Console.WriteLine($"NumberOfTopics:
{props.NumberOfTopics}\nInputS3Uri: {props.InputDataConfig.S3Uri}");
            Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 StartTopicsDetectionJob을 참조하세요.

CLI

AWS CLI

주제 탐지 분석 작업 시작

다음 `start-topics-detection-job` 예제에서는 `--input-data-config` 태그로 지정된 주소에 있는 모든 파일에 대해 비동기 주제 탐지 작업을 시작합니다. 작업이 완료되면 `--ouput-data-config` 태그로 지정된 위치에 `output` 폴더가 배치됩니다. `output`에는 topic-terms.csv 및 doc-topics.csv 파일이 들어 있습니다. 첫 번째 출력 파일 topic-terms.csv는 컬렉션

의 주제 목록입니다. 각 주제에 대해 목록에는 기본적으로 주제별 상위 용어가 가중치에 따라 포함됩니다. 두 번째 doc-topics.csv 파일에는 주제와 관련된 문서 및 해당 주제와 관련된 문서 비율이 나열되어 있습니다.

```
aws comprehend start-topics-detection-job \
    --job-name example_topics_detection_job \
    --language-code en \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --output-data-config "S3Uri=s3://amzn-s3-demo-destination-bucket/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --language-code en
```

출력:

```
{
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

자세한 내용은 Amazon Comprehend 개발자 안내서의 주제 모델링을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 StartTopicsDetectionJob을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
```

```
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix,
all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                          file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
```

```
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 StartTopicsDetectionJob을 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용한 Amazon Comprehend 시나리오

다음 코드 예제에서는 Amazon Comprehend AWS SDKs에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오는 Amazon Comprehend 내에서 또는 다른 함수와 결합된 여러 함수를 호출하여 특정 작업을 수행하는 방법을 보여줍니다 AWS 서비스. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- Amazon Transcribe 스트리밍 앱 구축
- 웹 사이트 방문자의 참여를 유도하는 Amazon Lex 챗봇 생성

- Amazon SQS를 사용하여 메시지를 보내고 검색하는 웹 애플리케이션 생성
- 고객 피드백을 분석하고 오디오를 합성하는 애플리케이션 생성
- Amazon Comprehend 및 AWS SDK를 사용하여 문서 요소 감지
- AWS SDK를 사용하여 이미지에서 추출된 텍스트의 개체 감지
- AWS SDK를 사용하여 샘플 데이터에 대한 Amazon Comprehend 주제 모델링 작업 실행
- AWS SDK를 사용하여 사용자 지정 Amazon Comprehend 분류자 훈련 및 문서 분류

Amazon Transcribe 스트리밍 앱 구축

다음 코드 예시는 라이브 오디오를 실시간으로 기록, 변환 및 번역하고 결과를 이메일로 보내는 앱을 구축하는 방법을 보여줍니다.

JavaScript

SDK for JavaScript (v3)

Amazon Transcribe를 사용하여 라이브 오디오를 실시간으로 기록, 변환 및 번역하고 Amazon Simple Email Service(Amazon SES)를 사용하여 결과를 이메일로 전송하는 앱을 구축하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예시를 참조하세요.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

웹 사이트 방문자의 참여를 유도하는 Amazon Lex 챗봇 생성

다음 코드 예제는 챗봇을 생성하여 웹사이트 방문자를 참여시키는 방법을 보여줍니다.

Java

SDK for Java 2.x

Amazon Lex API를 사용하여 웹 애플리케이션 내에 챗봇을 구축하여 웹 사이트 방문자의 참여를 유도하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK for JavaScript (v3)

Amazon Lex API를 사용하여 웹 애플리케이션 내에 챗봇을 구축하여 웹 사이트 방문자의 참여를 유도하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 AWS SDK for JavaScript 개발자 안내서의 Amazon Lex 챗봇 구축 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon SQS를 사용하여 메시지를 보내고 검색하는 웹 애플리케이션 생성

다음 코드 예제는 Amazon SQS를 사용하여 메시징 애플리케이션을 생성하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Amazon SQS API를 사용하여 메시지를 보내고 검색하는 Spring REST API를 개발하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon SQS

Kotlin

SDK for Kotlin

Amazon SQS API를 사용하여 메시지를 보내고 검색하는 Spring REST API를 개발하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon SQS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

고객 피드백을 분석하고 오디오를 합성하는 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

.NET

SDK for .NET

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK for Java 2.x

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK for JavaScript (v3)

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하십시오. 다음 발췌문은 AWS SDK for JavaScript 가 Lambda 함수 내에서 사용되는 방법을 보여줍니다.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
```

```
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
```

```
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);
```

```
  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK for Ruby

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Comprehend 및 AWS SDK를 사용하여 문서 요소 감지

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 문서에서 언어, 개체 및 핵심 문구를 감지합니다.
- 문서에서 개인 식별 정보(PII)를 감지합니다.
- 문서의 감성을 감지합니다.
- 문서의 구문 요소를 감지합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Comprehend 작업을 래핑하는 등급을 만듭니다.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
```

```
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
its
        modifiers.
```

```
        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can
be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
```

```
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

래퍼 클래스의 함수를 직접 호출하여 문서에 있는 개체, 문구 등을 감지합니다.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_detect = ComprehendDetect(boto3.client("comprehend"))
    with open("detect_sample.txt") as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print("-" * 88)
    print(sample_text)
    print("-" * 88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]["LanguageCode"]

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])

    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
    print("SentimentScore:")
```

```
    pprint(sentiment["SentimentScore"])

    print("Detecting syntax elements.")
    syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(syntax_tokens[:demo_size])

    print("Thanks for watching!")
    print("-" * 88)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - DetectDominantLanguage
  - DetectEntities
  - DetectKeyPhrases
  - DetectPiiEntities
  - DetectSentiment
  - DetectSyntax

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 이미지에서 추출된 텍스트의 개체 감지

다음 코드 예제는 Amazon Comprehend를 사용하여 Amazon S3에 저장된 이미지에서 Amazon Textract를 통해 추출한 텍스트의 개체를 감지하는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

Jupyter 노트북 AWS SDK for Python (Boto3) 에서를 사용하여 이미지에서 추출된 텍스트의 개체를 감지하는 방법을 보여줍니다. 이 예제에서는 Amazon Textract를 통해 Amazon Simple Storage Service (Amazon S3) 및 Amazon Comprehend에 저장된 이미지에서 텍스트를 추출하여 추출된 텍스트의 엔터티를 감지합니다.

이 예제는 Jupyter Notebook에 관한 것이며, 노트북을 호스팅할 수 있는 환경에서 실행되어야 합니다. Amazon SageMaker AI를 사용하여 예제를 실행하는 방법에 대한 지침은 TextractAndComprehendNotebook.ipynb의 지침을 참조하세요.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon S3
- Amazon Textract

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 샘플 데이터에 대한 Amazon Comprehend 주제 모델링 작업 실행

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 샘플 데이터에 대한 Amazon Comprehend 주제 모델링 작업 실행
- 작업에 대한 정보를 얻습니다.
- Amazon S3에서 작업 출력 데이터를 추출합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Comprehend 주제 모델링 작업을 직접 호출하는 래퍼 등급을 생성합니다.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix,
all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                          file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
```

```
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
```

```
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

래퍼 등급을 사용하여 주제 모델링 작업을 실행하고 작업 데이터를 가져옵니다.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    input_prefix = "input/"
    output_prefix = "output/"
    demo_resources = ComprehendDemoResources(
        boto3.resource("s3"), boto3.resource("iam")
    )
    topic_modeler = ComprehendTopicModeler(boto3.client("comprehend"))

    print("Setting up storage and security resources needed for the demo.")
    demo_resources.setup("comprehend-topic-modeler-demo")
    print("Copying sample data from public bucket into input bucket.")
    demo_resources.bucket.copy(
        {"Bucket": "public-sample-us-west-2", "Key": "TopicModeling/Sample.txt"},
        f"{input_prefix}sample.txt",
    )

    print("Starting topic modeling job on sample data.")
    job_info = topic_modeler.start_job(
        "demo-topic-modeling-job",
        demo_resources.bucket.name,
        input_prefix,
```

```
        JobInputFormat.per_line,
        demo_resources.bucket.name,
        output_prefix,
        demo_resources.data_access_role.arn,
    )

    print(
        f"Waiting for job {job_info['JobId']} to complete. This typically takes "
        f"20 - 30 minutes."
    )
    job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = topic_modeler.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from the output Amazon S3 bucket: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = demo_resources.extract_job_output(job)
    lines = 10
    print(f"First {lines} lines of document topics output:")
    pprint(job_output["doc-topics.csv"]["data"][:lines])
    print(f"First {lines} lines of terms output:")
    pprint(job_output["topic-terms.csv"]["data"][:lines])

    print("Cleaning up resources created for the demo.")
    demo_resources.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - DescribeTopicsDetectionJob
  - ListTopicsDetectionJobs
  - StartTopicsDetectionJob

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 사용자 지정 Amazon Comprehend 분류자 훈련 및 문서 분류

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Comprehend 멀티레이블 분류기를 생성합니다.
- 샘플 데이터를 기반으로 분류기를 훈련시킵니다.
- 두 번째 데이터 세트에 대한 분류 작업을 실행합니다.
- Amazon S3에서 작업 출력 데이터를 추출합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

래퍼 등급을 생성하여 Amazon Comprehend 문서 분류기 작업을 직접 호출합니다.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
```

```
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
immediately
        starts training on the data found in the specified Amazon S3 bucket.
Training
        can take 30 minutes or longer. The `describe_document_classifier`
function
        can be used to get training status and returns a status of TRAINED when
the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
data.
        :param training_key: The prefix used to find training data in the
training
                             bucket. If multiple objects have the same prefix,
all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
```

```
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
```

```
            )
            raise
        else:
            return classifiers


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
self.classifier_arn)
            raise


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
```

```
                            bucket. If multiple objects have the same prefix, all
                            of them are used.
        :param input_format: The format of the input data, either one document
per
                               file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
```

```
            response =
 self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

시나리오를 실행하는 클래스를 생성합니다.

```
class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """

    def __init__(self, demo_resources):
        """
```

```
        :param demo_resources: A ComprehendDemoResources class that manages
 resources
                               for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = "training/"
        self.input_prefix = "input/"
        self.input_format = JobInputFormat.per_line
        self.output_prefix = "output/"

    def setup(self):
        """Creates AWS resources used by the demo."""
        self.demo_resources.setup("comprehend-classifier-demo")

    def cleanup(self):
        """Deletes AWS resources used by the demo."""
        self.demo_resources.cleanup()

    @staticmethod
    def _sanitize_text(text):
        """Removes characters that cause errors for the document parser."""
        return text.replace("\r", " ").replace("\n", " ").replace(",", ";")

    @staticmethod
    def _get_issues(query, issue_count):
        """
        Gets issues from GitHub using the specified query parameters.

        :param query: The query string used to request issues from the GitHub
 API.
        :param issue_count: The number of issues to retrieve.
        :return: The list of issues retrieved from GitHub.
        """
        issues = []
        logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
        response = requests.get(f"{GITHUB_SEARCH_URL}?
{query}&per_page={issue_count}")
        if response.status_code == 200:
            issue_page = response.json()["items"]
            logger.info("Got %s issues.", len(issue_page))
            issues = [
                {
                    "title": ClassifierDemo._sanitize_text(issue["title"]),
                    "body": ClassifierDemo._sanitize_text(issue["body"]),
```

```
                    "labels": {label["name"] for label in issue["labels"]},
                }
                for issue in issue_page
            ]
        else:
            logger.error(
                "GitHub returned error code %s with message %s.",
                response.status_code,
                response.json(),
            )
        logger.info("Found %s issues.", len(issues))
        return issues

    def get_training_issues(self, training_labels):
        """
        Gets issues used for training the custom classifier. Training issues are
        closed issues from the Boto3 repo that have known labels. Comprehend
        requires a minimum of ten training issues per label.

        :param training_labels: The issue labels to use for training.
        :return: The set of issues used for training.
        """
        issues = []
        per_label_count = 15
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:closed+label:{label}",
                per_label_count,
            )
            for issue in issues:
                issue["labels"] = issue["labels"].intersection(training_labels)
        return issues

    def get_input_issues(self, training_labels):
        """
        Gets input issues from GitHub. For demonstration purposes, input issues
        are open issues from the Boto3 repo with known labels, though in practice
        any issue could be submitted to the classifier for labeling.

        :param training_labels: The set of labels to query for.
        :return: The set of issues used for input.
        """
        issues = []
        per_label_count = 5
```

```
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:open+label:{label}",
                per_label_count,
            )
        return issues

    def upload_issue_data(self, issues, training=False):
        """
        Uploads issue data to an Amazon S3 bucket, either for training or for
input.
        The data is first put into the format expected by Comprehend. For
training,
        the set of pipe-delimited labels is prepended to each document. For
        input, labels are not sent.

        :param issues: The set of issues to upload to Amazon S3.
        :param training: Indicates whether the issue data is used for training or
                         input.
        """
        try:
            obj_key = (
                self.training_prefix if training else self.input_prefix
            ) + "issues.txt"
            if training:
                issue_strings = [
                    f"{'|'.join(issue['labels'])},{issue['title']}
{issue['body']}"
                    for issue in issues
                ]
            else:
                issue_strings = [
                    f"{issue['title']} {issue['body']}" for issue in issues
                ]
            issue_bytes = BytesIO("\n".join(issue_strings).encode("utf-8"))
            self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
            logger.info(
                "Uploaded data as %s to bucket %s.",
                obj_key,
                self.demo_resources.bucket.name,
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't upload data to bucket %s.",
self.demo_resources.bucket.name
            )
            raise

    def extract_job_output(self, job):
        """Extracts job output from Amazon S3."""
        return self.demo_resources.extract_job_output(job)

    @staticmethod
    def reconcile_job_output(input_issues, output_dict):
        """
        Reconciles job output with the list of input issues. Because the input
issues
        have known labels, these can be compared with the labels added by the
        classifier to judge the accuracy of the output.

        :param input_issues: The list of issues used as input.
        :param output_dict: The dictionary of data that is output by the
classifier.
        :return: The list of reconciled input and output data.
        """
        reconciled = []
        for archive in output_dict.values():
            for line in archive["data"]:
                in_line = int(line["Line"])
                in_labels = input_issues[in_line]["labels"]
                out_labels = {
                    label["Name"]
                    for label in line["Labels"]
                    if float(label["Score"]) > 0.3
                }
                reconciled.append(
                    f"{line['File']}, line {in_line} has labels {in_labels}.\n"
                    f"\tClassifier assigned {out_labels}."
                )
        logger.info("Reconciled input and output labels.")
        return reconciled
```

레이블이 알려진 GitHub 이슈 세트에 대해 분류기를 학습시킨 다음, 분류기에 두 번째 GitHub 이슈 세트를 전송하여 레이블을 지정할 수 있도록 합니다.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_demo = ClassifierDemo(
        ComprehendDemoResources(boto3.resource("s3"), boto3.resource("iam"))
    )
    comp_classifier = ComprehendClassifier(boto3.client("comprehend"))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client
    )
    training_labels = {"bug", "feature-request", "dynamodb", "s3"}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = "doc-example-classifier"
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name,
        "en",
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label,
    )
    print(
        f"Waiting until {classifier_name} is trained. This typically takes "
        f"30-40 minutes."
    )
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
```

```
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
        "issue_classification_job",
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn,
    )
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = comp_classifier.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from Amazon S3: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = comp_demo.extract_job_output(job)
    print("Job output:")
    pprint(job_output)

    print("Reconciling job output with labels from GitHub:")
    reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
    print(*reconciled_output, sep="\n")

    answer = input(f"Do you want to delete the classifier {classifier_name} (y/
n)? ")
    if answer.lower() == "y":
        print(f"Deleting {classifier_name}.")
        comp_classifier.delete()

    print("Cleaning up resources created for the demo.")
    comp_demo.cleanup()
```

```
    print("Thanks for watching!")
    print("-" * 88)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreateDocumentClassifier
  - DeleteDocumentClassifier
  - DescribeDocumentClassificationJob
  - DescribeDocumentClassifier
  - ListDocumentClassificationJobs
  - ListDocumentClassifiers
  - StartDocumentClassificationJob

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Amazon Comprehend 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Comprehend의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델 에서는 이를 클라우드 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. Amazon Comprehend에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon Comprehend를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 Amazon Comprehend를 구성하는 방법을 보여줍니다. 또한 Amazon Comprehend 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

- Amazon Comprehend의 데이터 보호
- Amazon Comprehend 자격 증명 및 액세스 관리
- AWS CloudTrail를 사용하여 Amazon Comprehend API 호출 로깅
- Amazon Comprehend에 대한 규정 준수 검증
- Amazon Comprehend 복원성
- Amazon Comprehend의 인프라 보안

Amazon Comprehend의 데이터 보호

AWS 공동 책임 모델 Amazon Comprehend의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호

스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- AWS 암호화 솔루션과 내부의 모든 기본 보안 제어를 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon Comprehend 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

주제

- Amazon Comprehend에서의 KMS 암호화
- 교차 서비스 혼동된 대리인 방지
- Amazon Virtual Private Cloud를 사용한 작업 보호
- Amazon Comprehend와 인터페이스 VPC 엔드포인트(AWS PrivateLink)

Amazon Comprehend에서의 KMS 암호화

Amazon Comprehend는 AWS Key Management Service (AWS KMS)와 함께 작동하여 데이터에 대한 향상된 암호화를 제공합니다. Amazon S3는 텍스트 분석, 주제 모델링 또는 사용자 정의 Amazon Comprehend 작업을 생성할 때 사용자가 입력 문서를 암호화 할 수 있게 하였습니다. 와 통합 AWS KMS 하면 Start* 및 Create* 작업에 대한 스토리지 볼륨의 데이터를 암호화할 수 있으며 자체 KMS 키를 사용하여 Start* 작업의 출력 결과를 암호화할 수 있습니다.

의 경우 AWS Management Console Amazon Comprehend는 자체 KMS 키로 사용자 지정 모델을 암호화합니다. 의 경우 AWS CLI Amazon Comprehend는 자체 KMS 키 또는 제공된 고객 관리형 키(CMK)를 사용하여 사용자 지정 모델을 암호화할 수 있습니다.

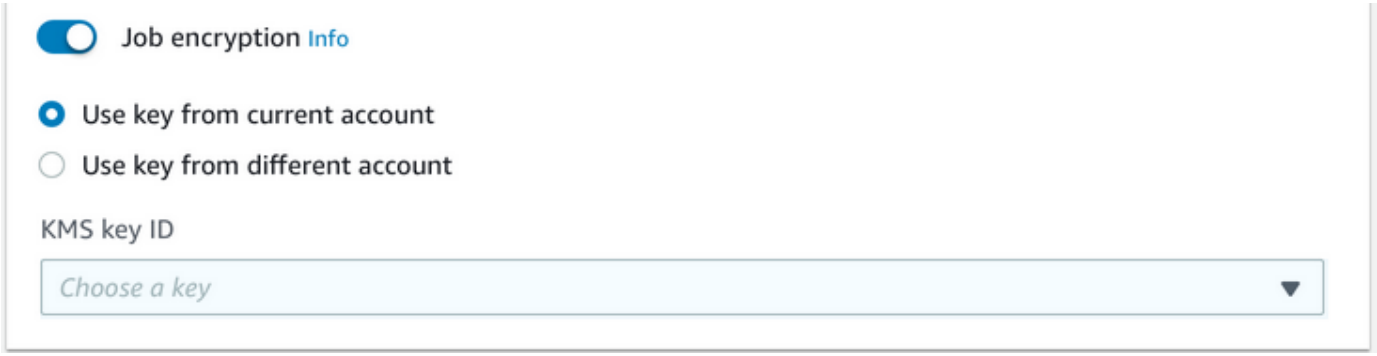
를 사용한 KMS 암호화 AWS Management Console

콘솔을 사용하면 두 가지 암호화 옵션을 사용할 수 있습니다.

- 볼륨 암호화
- 출력 결과 암호화

볼륨 암호화를 활성화하려면

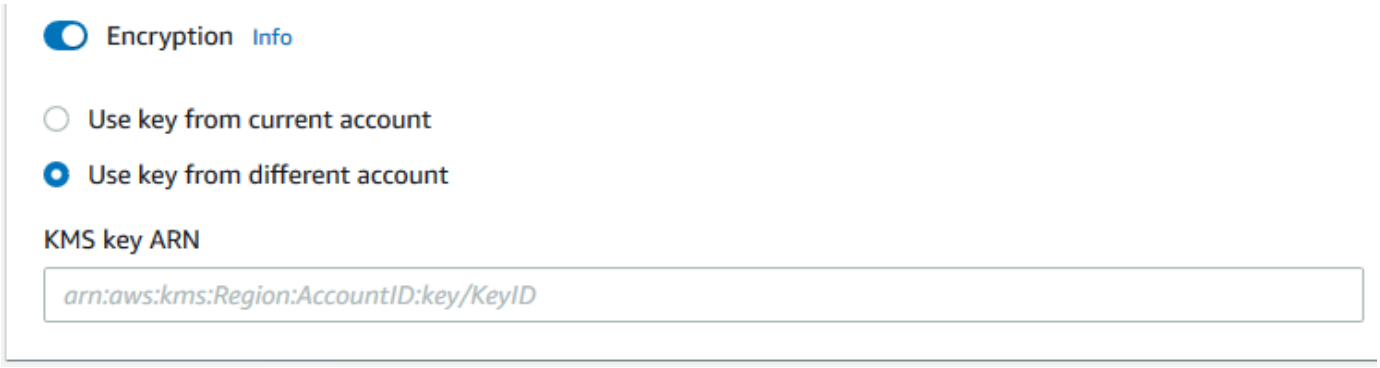
1. 작업 설정 에서 작업 암호화 옵션을 선택합니다.



2. KMS CMK(고객 관리 키)가 현재 사용 중인 계정의 키인지 아니면 다른 계정의 키인지 선택합니다. 현재 계정의 키를 사용하려면 KMS 키 ID에서 키 별칭을 선택합니다. 다른 계정의 키를 사용하려면 키의 ARN을 입력해야 합니다.

출력 결과 암호화를 활성화하려면

1. 출력 설정에서 암호화 옵션을 선택합니다.



2. CMK(고객 관리 키)가 현재 사용 중인 계정의 키인지 아니면 다른 계정의 키인지 선택합니다. 현재 계정의 키를 사용하려면 KMS 키 ID에서 키 ID를 선택합니다. 다른 계정의 키를 사용하려면 키의 ARN을 입력해야 합니다.

이전에 S3 입력 문서에 SSE-KMS를 사용하여 암호화를 설정하였다면 추가 보안을 제공할 수 있습니다. 하지만 이렇게 하면 사용한 IAM 역할은 입력 문서를 암호화하는 데 사용되는 KMS 키에 대한 `kms:Decrypt` 권한을 가지고 있어야 합니다. 자세한 내용은 KMS 암호화를 사용하는 데 필요한 권한을 참조하십시오.

API 작업을 사용한 KMS 암호화

모든 Amazon Comprehend `Start*`과 `Create*` API 작업은 KMS의 암호화된 입력 문서를 지원합니다. `Describe*`과 `List*` API 작업은 원래 작업이 `KmsKeyId`을 입력으로 사용했다면 `OutputDataConfig`에 `KmsKeyId`를 반환합니다. 입력으로 사용되지 않았다면 반환되지 않습니다.

이는 StartEntitiesDetectionJob 작업을 사용하여 다음 AWS CLI 예제에서 확인할 수 있습니다.

```
aws comprehend start-entities-detection-job \
     --region region \
     --data-access-role-arn "data access role arn" \
     --entity-recognizer-arn "entity recognizer arn" \
     --input-data-config "S3Uri=s3://Bucket Name/Bucket Path" \
     --job-name job name \
     --language-code en \
     --output-data-config "KmsKeyId=Output S3 KMS key ID" "S3Uri=s3://Bucket
 Name/Bucket Path/" \
     --volumekmskeyid "Volume KMS key ID"
```

> **Note**
>
> 이 예제는 Unix, Linux 및 macOS용 형식으로 표시됩니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

API 작업을 사용한 고객 관리 키 (CMK) 암호화

Amazon Comprehend 사용자 지정 모델 API 작업 `CreateEndpoint`, `CreateEntityRecognizer` `CreateDocumentClassifier`및는를 통해 고객 관리형 키를 사용한 암호화를 지원합니다 AWS CLI.

보안 주체가 고객 관리 키를 사용하거나 관리하도록 허용하려면 IAM 정책이 필요합니다. 이 키는 정책 설명의 `Resource` 요소에 지정되어 있습니다. 가장 좋은 방법은 정책 설명에 고객 관리형 키를 보안 주체만 사용해야 하는 키로 제한하는 것입니다.

다음 AWS CLI 예제에서는 CreateEntityRecognizer 작업을 사용하여 모델 암호화를 사용하여 사용자 지정 개체 인식기를 생성합니다.

```
aws comprehend create-entity-recognizer \
     --recognizer-name name \
     --data-access-role-arn data access role arn \
     --language-code en \
     --model-kms-key-id Model KMS Key ID \
     --input-data-config file:///path/input-data-config.json
```

> **Note**
>
> 이 예제는 Unix, Linux 및 macOS용 형식으로 표시됩니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 위장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에

서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

Amazon Comprehend가 리소스에 다른 서비스를 제공하는 권한을 제한하려면 리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하는 것이 좋습니다. 두 글로벌 조건 컨텍스트 키를 모두 사용하는 경우 aws:SourceAccount값과 aws:SourceArn값의 계정은 동일한 정책 문에서 사용할 경우 동일한 계정 ID를 사용해야 합니다.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모를 경우 또는 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 aws:SourceArn 전역 조건 컨텍스트 키를 사용합니다. 예: arn:aws:*servicename*::*123456789012*:*.

소스 계정 사용

다음 예제에서는 Amazon Comprehend에서 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하는 방법을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
          "Service": "comprehend.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringEquals": {
             "aws:SourceAccount":"111122223333"
        }
    }
  }
}
```

암호화된 모델의 엔드포인트에 대한 신뢰 정책

암호화된 모델의 엔드포인트를 생성하거나 업데이트하려면 신뢰 정책을 생성해야 합니다. aws:SourceAccount 값을 계정 ID로 설정합니다. ArnEquals 조건을 사용하는 경우, aws:SourceArn 값을 엔드포인트의 ARN으로 설정합니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
    {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
            "Service": "comprehend.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
               "aws:SourceAccount": "111122223333"
            },
            "ArnEquals": {
               "aws:SourceArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier-endpoint/endpoint-name"
            }
        }
    }
  ]
}
```

사용자 지정 모델 생성

사용자 지정 모델을 생성하려면 신뢰 정책을 생성해야 합니다. aws:SourceAccount 값을 계정 ID로 설정합니다. ArnEquals 조건을 사용하는 경우, aws:SourceArn 값을 사용자 지정 모델 버전의 ARN으로 설정합니다.

```
{
 "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "",
          "Effect": "Allow",
          "Principal": {
              "Service": "comprehend.amazonaws.com"
          },
          "Action": "sts:AssumeRole",
          "Condition": {
              "StringEquals": {
                  "aws:SourceAccount": "111122223333"
```

```
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:comprehend:us-west-2:111122223333:
                                document-classifier/smallest-classifier-test/
version/version-name"
                }
            }
        }
    ]
}
```

Amazon Virtual Private Cloud를 사용한 작업 보호

Amazon Comprehend는 Amazon Comprehend에서 데이터가 저장된 작업 컨테이너를 사용하는 동안 데이터의 안전을 보장하기 위해 다양한 보안 조치를 사용합니다. 그러나 작업 컨테이너는 인터넷을 통해 데이터 및 모델 아티팩트를 저장하는 Amazon S3 버킷과 같은 AWS 리소스에 액세스합니다.

사용자 데이터에 대한 액세스를 제어하려면 Virtual Private Cloud(VPC)를 생성하고 구성하여 데이터와 컨테이너가 인터넷을 통해 액세스 되지 않도록 구성하는 것이 좋습니다. VPC 생성 및 구성에 대한 내용은 Amazon VPC 사용 설명서의 Amazon VPC 시작하기를 참조하세요. VPC를 사용하면 사용자 데이터가 인터넷에 연결되지 않도록 VPC를 구성할 수 있기 때문에 데이터 보호에 도움이 됩니다. 또한 VPC를 사용하면 VPC 흐름 로그를 통하여 당사 작업 컨테이너에 들어오고 나가는 모든 네트워크 트래픽을 모니터링할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 흐름 로그를 참조하세요.

작업을 생성할 때 서브넷 및 보안 그룹을 지정하여 프라이빗 VPC 구성을 지정합니다. 서브넷 및 보안 그룹을 지정할 때 Amazon Comprehend가 서브넷 중 하나에 있는 보안 그룹과 연결된 탄력적 네트워크 인터페이스(ENI)를 생성합니다. ENI를 통해 작업 컨테이너는 사용자 VPC에 있는 리소스와 연결될 수 있습니다. ENI에 대한 자세한 내용은 Amazon VPC 사용 설명서의 탄력적 네트워크 인터페이스를 참조하세요.

> **Note**
>
> 작업의 경우, 사용자는 사용자 인스턴스가 실행되는 공유 하드웨어의 기본 테넌시 VPC 만을 사용하여 서브넷을 구성할 수 있습니다. VPCs의 테넌시 속성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 전용 인스턴스를 참조하세요.

Amazon VPC 액세스를 위한 작업 구성

사용자 VPC에서 서브넷 및 보안 그룹을 지정하려면 해당 API의 `VpcConfig` 요청 파라미터를 사용하거나 Amazon Comprehend 콘솔에서 작업을 생성할 때 이 정보를 제공합니다. Amazon Comprehend는 이 정보를 사용하여 ENI를 생성하고 이를 당사 작업 컨테이너에 연결합니다. ENI는 당사 작업 컨테이너에 인터넷과 연결되지 않은 사용자 VPC 내부에서 네트워크 연결을 제공합니다.

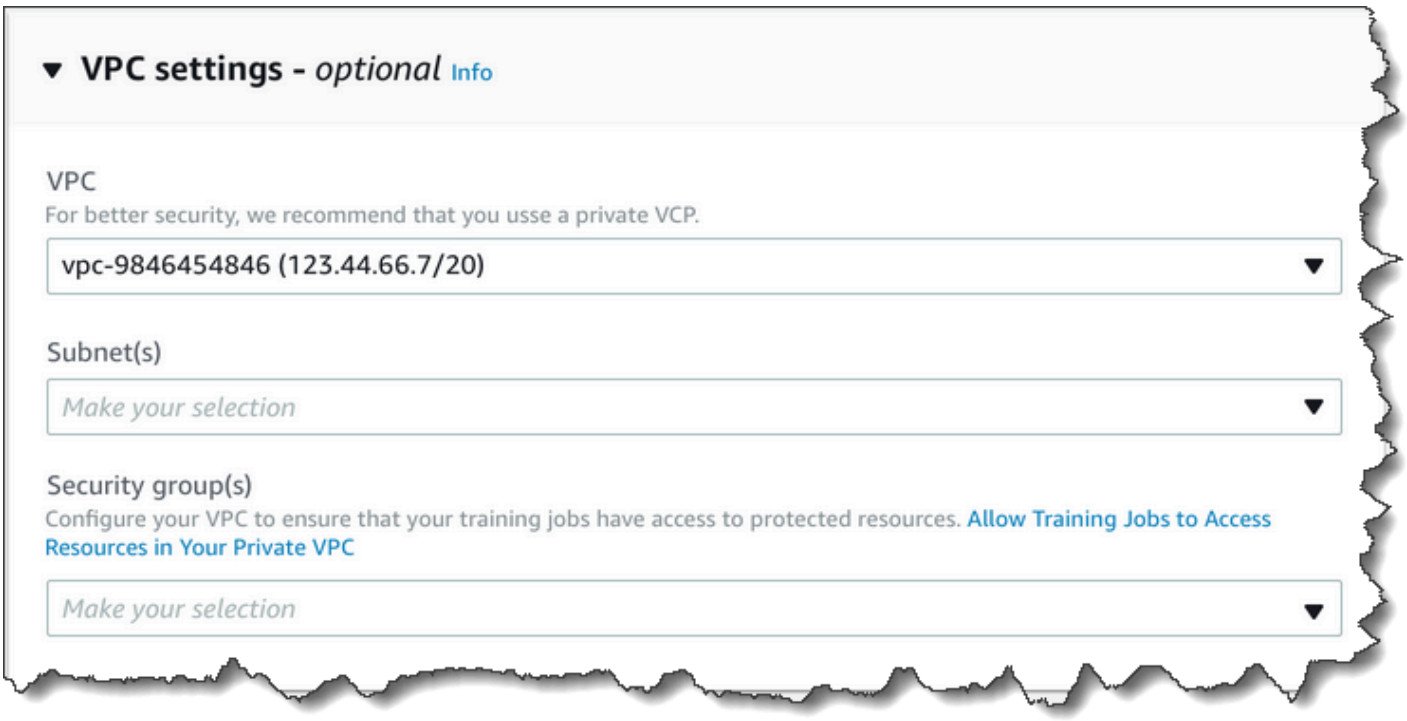
다음 API에는 `VpcConfig` 요청 파라미터가 포함되어 있습니다.

- `Create*`API: `CreateDocumentClassifier`, `CreateEntityRecognizer`
- `Start*`APIs: `StartDocumentClassificationJob`, `StartDominantLanguageDetectionJob`, `StartEntitiesDetectionJob`, `StartKeyPhrasesDetectionJob`, `StartSentimentDetectionJob`, `StartTargetedSentimentDetectionJob`, `StartTopicsDetectionJob`

다음은 사용자 API 직접 호출에 포함할 VpcConfig 파라미터의 예입니다.

```
"VpcConfig": {
      "SecurityGroupIds": [
          " sg-0123456789abcdef0"
          ],
      "Subnets": [
          "subnet-0123456789abcdef0",
          "subnet-0123456789abcdef1",
          "subnet-0123456789abcdef2"
          ]
      }
```

Amazon Comprehend 콘솔에서 VPC를 구성하려면 작업 생성 시 선택적 VPC 설정섹션에서 구성 세부 정보를 선택합니다.



Amazon Comprehend 작업을 위한 사용자 VPC 구성

사용자 Amazon Comprehend 용 VPC를 구성할 때 다음 지침을 따릅니다. VPC 설정에 관한 자세한 내용은 Amazon VPC 사용 설명서의 VPCs 및 서브넷 작업을 참조하세요.

서브넷에 충분한 IP 주소를 확보해야 합니다

사용자 VPC 서브넷에는 각 작업 인스턴스 별로 최소 2개 이상의 프라이빗 IP 주소가 있어야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷 IPv4 크기 조정을 참조하세요.

Amazon S3 VPC 엔드포인트 생성

작업 컨테이너가 인터넷에 액세스하지 않도록 사용자 VPC를 구성하면 사용자가 액세스를 허용하는 VPC 엔드포인트를 생성하지 않는 VPC는 사용자 데이터가 들어 있는 Amazon S3 버킷에 연결할 수 없습니다. VPC 엔드포인트를 생성하면 훈련 및 분석 작업 중에 작업 컨테이너가 데이터에 액세스할 수 있습니다.

VPC 엔드포인트를 생성할 때 다음 값을 구성합니다.

- 서비스 범주를 AWS 서비스로 선택
- 서비스를 로 지정 `com.amazonaws.region.s3`
- 게이트웨이를 VPC 엔드포인트 유형으로 선택

AWS CloudFormation 를 사용하여 VPC 엔드포인트를 생성하는 경우 AWS CloudFormation VPCEndpoint 설명서를 따릅니다. 다음 예제는 AWS CloudFormation 템플릿의 VPCEndpoint 구성을 보여줍니다.

```
VpcEndpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Action:
            - s3:GetObject
            - s3:PutObject
            - s3:ListBucket
            - s3:GetBucketLocation
            - s3:DeleteObject
            - s3:ListMultipartUploadParts
            - s3:AbortMultipartUpload
          Effect: Allow
          Resource:
            - "*"
          Principal: "*"
    RouteTableIds:
      - Ref: RouteTable
    ServiceName:
      Fn::Join:
        - ''
        - - com.amazonaws.
          - Ref: AWS::Region
          - ".s3"
    VpcId:
      Ref: VPC
```

사용자 VPC의 S3 버킷 액세스 요청 만을 허용하는 사용자 정의 정책의 생성을 권장합니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon S3용 엔드포인트 정책을 참조하세요.

다음 정책은 S3 버킷 액세스를 허용합니다. 작업에 필요한 리소스에만 액세스할 수 있도록 이 정책을 편집하십시오.

```
{
```

```
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:DeleteObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "*"
        }
    ]
}
```

사용자 엔드포인트 라우팅 테이블에 기본 DNS 설정을 사용하면 표준 Amazon S3 URL(예:`http://s3-aws-region.amazonaws.com/amzn-s3-demo-bucket`)이 결정됩니다. 기본 DNS 설정을 사용하지 않는다면 엔드포인트 라우팅 테이블을 구성하여 사용자 작업의 데이터 위치를 지정하는 데 사용한 URL이 결정되도록 합니다. VPC 엔드포인트 라추팅 테이블에 관한 내용은 Amazon VPC 사용 설명서의 게이트웨이 VPC 엔드포인트 라우팅을 참조하세요.

기본 엔드포인트 정책은 사용자가 당사 작업 컨테이너에 있는 Amazon Linux 및 Amazon Linux 2에서 패키지를 설치하도록 허용합니다. 사용자가 해당 리포지토리의 패키지를 설치하지 않도록 하려면 Amazon Linux 및 Amazon Linux 2 리포지토리에 대한 액세스를 명시적으로 거부하는 사용자 지정 엔드포인트 정책을 생성합니다. Comprehend 자체에는 이러한 패키지가 필요하지 않으므로 기능에 영향을 미치지 않습니다. 다음은 이러한 리포지토리에 대한 액세스를 거부하는 정책의 예입니다.

```
{
    "Statement": [
      {
        "Sid": "AmazonLinuxAMIRepositoryAccess",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::packages.*.amazonaws.com/*",
```

```
            "arn:aws:s3:::repo.*.amazonaws.com/*"
        ]
      }
    ]
}

{
    "Statement": [
        { "Sid": "AmazonLinux2AMIRepositoryAccess",
          "Principal": "*",
          "Action": [
              "s3:GetObject"
              ],
          "Effect": "Deny",
          "Resource": [
              "arn:aws:s3:::amazonlinux.*.amazonaws.com/*"
              ]
         }
    ]
}
```

DataAccessRole에 대한 권한

분석 작업에 VPC를 사용하는 경우 `Create*`및 `Start*`작업에 사용되는 `DataAccessRole`은 입력 문서 및 출력 버킷에 액세스하는 VPC에 대한 권한도 있어야 합니다.

다음 정책은 `Create*` 및 `Start*` 작업에 사용되는 `DataAccessRole`에게 필요한 액세스를 제공합니다.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
              "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeSubnets",
```

```
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

VPC 보안 그룹 구성

분산형 작업에서 동일한 작업의 다른 컨테이너 사이의 통신을 허용해야 합니다. 이렇게 하려면 동일한 보안 그룹의 멤버 간의 인바운드 연결을 허용하는 보안 그룹 규칙을 구성합니다. 자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹 규칙을 참조하세요.

VPC 외부 리소스에 연결

인터넷 액세스가 되지 않도록 사용자 VPC를 구성하면 해당 VPC를 사용하는 작업은 사용자 VPC 외부의 리소스에 액세스할 수 없습니다. 사용자 작업에 VPC 외부 리소스 액세스가 필요하면 다음 옵션 중 하나를 사용하여 액세스를 제공합니다.

- 작업에서 인터페이스 VPC 엔드포인트를 지원하는 AWS 서비스에 액세스해야 하는 경우 해당 서비스에 연결할 엔드포인트를 생성합니다. 인터페이스 엔드포인트를 지원하는 서비스 목록은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하세요. 인터페이스 VPC 엔드포인트 생성에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.
- 작업에 인터페이스 VPC 엔드포인트를 지원하지 않는 AWS 서비스 또는 외부 리소스에 대한 액세스가 필요한 경우 NAT 게이트웨이를 AWS생성하고 아웃바운드 연결을 허용하도록 보안 그룹을 구성합니다. VPC용 NAT 게이트웨이에 대한 자세한 내용은 Amazon VPC 사용 설명서의 시나리오 2: 퍼블릭 서브넷과 프라이빗 서브넷 VPC(NAT)를 참조하세요.

Amazon Comprehend와 인터페이스 VPC 엔드포인트(AWS PrivateLink)

인터페이스 VPC 엔드포인트를 생성하여 VPC와 Amazon Comprehend 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 인터넷 게이트웨이AWS PrivateLink, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 Amazon Comprehend APIs에 비공개로 액세스할 수 있는 기술인로 구동됩니다. VPC의 인스턴스는 Amazon Comprehend API와 통신하는 데 퍼블릭 IP 주소를 필요로 하지 않습니다. VPC와 Amazon Comprehend 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 나타냅니다.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.

Amazon Comprehend VPC 엔드포인트 고려 사항

Amazon Comprehend를 위한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다.

Amazon Comprehend 엔드포인트는 리전의 모든 가용 영역에서 사용할 수 있는 것은 아닙니다. 엔드포인트를 생성할 때 다음 명령을 사용하여 가용 영역 목록을 만드십시오.

```
aws ec2 describe-vpc-endpoint-services \
  --service-names com.amazonaws.us-west-2.comprehend
```

Amazon Comprehend는 사용자 VPC에서 모든 API 작업 호출을 지원합니다.

Amazon Comprehend용 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 ()를 사용하여 Amazon Comprehend 서비스에 대한 VPC 엔드포인트를 생성할 수 있습니다AWS CLI. AWS Command Line Interface 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 사용하여 Amazon Comprehend를 위한 VPC 엔드포인트를 생성합니다.

- com.amazonaws.*region*.comprehend

엔드포인트에 프라이빗 DNS를 사용할 수 있게 하면, 리전의 기본 DNS 이름을 사용하여 Amazon Comprehend에 API 요청을 할 수 있습니다, 예: *comprehend.us-east-1.amazonaws.com*.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통한 서비스 액세스를 참조하세요.

Amazon Comprehend를 위한 VPC 엔드포인트 정책 생성

Amazon Comprehend에 대한 액세스를 제어하는 사용자 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 위탁자.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통한 서비스에 대한 액세스 제어를 참조하세요.

예제: Amazon Comprehend 작업을 위한 VPC 엔드포인트 정책

다음은 Amazon Comprehend용 엔드포인트 정책의 예입니다. 이 정책을 엔드포인트에 연결하면 모든 리소스의 전체 보안 주체가 Amazon Comprehend `DetectEntities` 작업에 대한 액세스를 허용합니다.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "comprehend:DetectEntities"
         ],
         "Resource":"*"
      }
   ]
}
```

Amazon Comprehend 자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는 입니다. IAM 관리자는 어떤 사용자가 Amazon Comprehend 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon Comprehend에서 IAM을 사용하는 방법
- Amazon Comprehend의 자격 증명 기반 정책 예제
- AWS Amazon Comprehend에 대한 관리형 정책
- Amazon Comprehend 자격 증명 및 액세스 문제 해결

대상

사용 방법 AWS Identity and Access Management (IAM)은 Amazon Comprehend에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 - Amazon Comprehend 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증 정보와 권한을 관리자가 제공합니다. 더 많은 Amazon Comprehend 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. Amazon Comprehend의 기능에 액세스할 수 없는 경우 Amazon Comprehend 자격 증명 및 액세스 문제 해결을 참조하세요.

서비스 관리자 - 회사에서 Amazon Comprehend 리소스를 책임지고 있는 경우 Amazon Comprehend에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon Comprehend 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 Amazon Comprehend에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon Comprehend에서 IAM을 사용하는 방법를 참조하세요.

IAM 관리자 - IAM 관리자라면 Amazon Comprehend에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 Amazon Comprehend 자격 증명 기반 정책 예제를 보려면 Amazon Comprehend의 자격 증명 기반 정책 예제를 참조하세요.

ID를 통한 인증

인증은 자격 증명 AWS 으로에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를

사용하지 않는 경우 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 다중 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 테 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스 에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스 에 액세스하는 모든 사용자의 사용자입니다. 페더레이션 자격 증명 액세스 시 역할을 AWS 계정수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 및 애플리케이션에서 사용할 수 있도록 자체 자격 증명 소스의 사용자 AWS 계정 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 AWS 서비스 수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나

Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.

- 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
- 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결된 AWS 경우 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) - SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유 AWS 계정 한 여러를 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.
- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이를 포함한 자격 증명에 대한 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCPs)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 가 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon Comprehend에서 IAM을 사용하는 방법

IAM을 사용하여 Amazon Comprehend에 대한 액세스를 관리하기 전에 Amazon Comprehend에서 사용할 수 있는 IAM 기능을 알아봅니다.

Amazon Comprehend에서 사용할 수 있는 IAM 기능

IAM 기능	Amazon Comprehend 지원
ID 기반 정책	예
리소스 기반 정책	예
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	아니요
ABAC(정책 내 태그)	부분
임시 자격 증명	예
전달 액세스 세션(FAS)	예
서비스 역할	예
서비스 연결 역할	아니요

Amazon Comprehend 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Amazon Comprehend의 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지

를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. ID 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로 위탁자를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon Comprehend의 자격 증명 기반 정책 예제

Amazon Comprehend 자격 증명 기반 정책 예제를 보려면 Amazon Comprehend의 자격 증명 기반 정책 예제를 참조하세요.

Amazon Comprehend 내의 리소스 기반 정책

리소스 기반 정책 지원: 예

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 위탁자로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 다른 경우 신뢰할 수 AWS 계정있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 위탁자에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

Amazon Comprehend 서비스는 리소스 기반 정책 중 한 가지 유형(사용자 지정 모델 정책)만 지원하며, 이 유형은 사용자 지정 모델에 연결됩니다. 이 정책은 사용자 지정 모델을 사용할 수 있는 다른 계정을 정의합니다.

컨테이너에 리소스 기반 정책을 연결하는 방법은 사용자 지정 모델을 위한 리소스 기반 정책을(를) 참조하세요.

Amazon Comprehend 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

Amazon Comprehend 작업 목록을 보려면 서비스 권한 부여 참조에서 Amazon Comprehend에서 정의한 작업을 참조하세요.

Amazon Comprehend의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
comprehend
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "comprehend:DetectSentiment",
      "comprehend:ClassifyDocument"
         ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `Describe`라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "comprehend:Describe*"
```

서비스에 대한 모든 작업을 지정하는 데 와일드카드를 사용하지 마십시오. 최소 권한을 부여하는 모범 사례를 사용하고 정책에 사용되는 권한의 범위를 좁히십시오.

Amazon Comprehend 자격 증명 기반 정책 예제를 보려면 Amazon Comprehend의 자격 증명 기반 정책 예제를 참조하세요.

Amazon Comprehend 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon Comprehend 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조에서 Amazon Comprehend에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Comprehend에서 정의한 작업을 참조하세요.

Amazon Comprehend에 사용되는 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 `OR` 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon Comprehend 조건 키 목록을 보려면 서비스 권한 부여 참조에서 Amazon Comprehend의 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Comprehend에서 정의한 작업을 참조하세요.

Amazon Comprehend 자격 증명 기반 정책 예제를 보려면 Amazon Comprehend의 자격 증명 기반 정책 예제를 참조하세요.

Amazon Comprehend의 ACL

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon Comprehend와 ABAC

ABAC 지원(정책의 태그): 부분적

속성 기반 액세스 제어(ABAC)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS 리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 위탁자의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Amazon Comprehend 리소스 태그 지정에 대한 자세한 내용은 리소스에 태그 지정을 참조하세요.

Amazon Comprehend에서 임시 보안 인증 사용

임시 자격 증명 지원: 예

임시 자격 증명을 사용하여 로그인할 때 일부 AWS 서비스 가 작동하지 않습니다. 임시 자격 증명으로 AWS 서비스 작업하는을 비롯한 자세한 내용은 AWS 서비스 IAM 사용 설명서의 IAM으로 작업하는를 참조하세요.

사용자 이름과 암호를 제외한 방법을 AWS Management Console 사용하여에 로그인하는 경우 임시 자격 증명을 사용합니다. 예를 들어 회사의 SSO(Single Sign-On) 링크를 AWS 사용하여에 액세스하면 해당 프로세스가 임시 자격 증명을 자동으로 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 자격 증명을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 사용자에서 IAM 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다. 그런 다음 이러한 임시 자격 증명을 사용하여 장기 액세스 키를 사용하는 대신 동적으로 임시 자격 증명을 생성하는 access AWS. AWS recommends에 액세스할 수 있습니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하세요.

Amazon Comprehend용 전달 액세스 세션

전달 액세스 세션(FAS) 지원: 예

IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Amazon Comprehend 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon Comprehend 기능이 중단될 수 있습니다. Amazon Comprehend가 관련 지침을 제공하는 경우에만 서비스 역할을 편집하십시오.

Amazon Comprehend 비동기 작업을 사용하려면 문서 컬렉션이 포함된 Amazon S3 버킷에 대한 Amazon Comprehend의 액세스 권한을 부여해야 합니다. Amazon Comprehend 서비스 보안 주체를 신뢰하는 신뢰 정책을 사용하여 계정에 데이터 액세스 역할을 생성하면 됩니다.

정책 예제는 비동기 작업에 필요한 역할 기반 권한을 참조하세요.

Amazon Comprehend 서비스 연결 역할

서비스 링크 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 `Yes`이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon Comprehend의 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할은 Amazon Comprehend 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 비롯하여 Amazon Comprehend에서 정의되는 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon Comprehend에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- Amazon Comprehend 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 문서 분석 작업을 수행하는 데 필요한 권한
- KMS 암호화를 사용하는 데 필요한 권한
- AWS Amazon Comprehend에 대한 관리형(미리 정의된) 정책
- 비동기 작업에 필요한 역할 기반 권한
- 모든 Amazon Comprehend 작업을 허용할 수 있는 권한
- 주제 모델링 작업을 허용할 수 있는 권한
- 사용자 지정 비동기 분석 작업에 필요한 권한

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 Amazon Comprehend 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 조건을 사용하여 AWS 서비스와 같은 특정를 통해 사용되는 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을

확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.

- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

Amazon Comprehend 콘솔 사용

Amazon Comprehend 콘솔에 액세스하려면 최소한의 권한 세트가 있어야 합니다. 이러한 권한을 통해의 Amazon Comprehend 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API에만 호출하는 사용자에 대해 최소 콘솔 권한을 허용할 필요는 없습니다. 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

최소 Amazon Comprehend 콘솔 권한의 경우 *ComprehendReadOnly* AWS 관리형 정책을 엔터티에 연결할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

Amazon Comprehend 콘솔을 사용하려면 다음 정책에 표시된 작업에 대한 권한도 필요합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
      "Action": [
          "iam:ListRoles",
          "iam:GetRole",
          "s3:ListAllMyBuckets",
          "s3:ListBucket",
          "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": "*"
  }
  ]
  }
```

Amazon Comprehend 콘솔에 이러한 추가 권한이 필요한 이유는 다음과 같습니다.

- 계정에 사용 가능한 IAM 역할을 나열할 `iam` 권한.
- Amazon S3 버킷과 주제 모델링에 대한 데이터가 포함된 객체에 액세스할 `s3` 권한.

콘솔을 사용하여 비동기 배치 작업 또는 주제 모델링 작업을 생성할 때 콘솔이 해당 작업에 대한 IAM 역할을 생성하도록 할 수 있습니다. IAM 역할을 생성하려면 IAM 역할 및 정책을 생성하고 역할에 정책을 연결할 수 있는 다음과 같은 추가 권한을 사용자에게 부여해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Action":
          [
          "iam:CreateRole",
          "iam:CreatePolicy",
          "iam:AttachRolePolicy"
          ],
        "Effect": "Allow",
        "Resource": "*"
      },
      {
        "Action":
          [
          "iam:PassRole"
          ],
        "Effect": "Allow",
        "Resource": "arn:aws:iam::*:role/*Comprehend*"
      }
    ]
  }
```

Amazon Comprehend 콘솔에 이러한 추가 권한이 필요한 이유는 다음과 같습니다.

- 역할과 정책을 생성하고 역할과 정책을 연결할 수 있는 `iam` 권한. 이 `iam:PassRole` 작업을 통해 콘솔은 Amazon Comprehend에 역할을 전달할 수 있습니다.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함되어 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

문서 분석 작업을 수행하는 데 필요한 권한

다음 예제 정책은 Amazon Comprehend 문서 분석 작업을 사용할 수 있는 권한을 부여합니다.

```
{
 "Version": "2012-10-17",
 "Statement": [{
    "Sid": "AllowDetectActions",
    "Effect": "Allow",
    "Action": [
              "comprehend:DetectEntities",
              "comprehend:DetectKeyPhrases",
              "comprehend:DetectDominantLanguage",
              "comprehend:DetectSentiment",
              "comprehend:DetectTargetedSentiment",
              "comprehend:DetectSyntax",
              "textract:DetectDocumentText",
              "textract:AnalyzeDocument"
           ],
    "Resource": "*"
    }
 ]
}
```

정책에는 `DetectEntities`, `DetectKeyPhrases`, `DetectDominantLanguage`, `DetectTargetedSentiment`, `DetectSentiment`, 및 `DetectSyntax` 작업을 사용할 수 있는 권한을 부여하는 명령문이 하나 있습니다. 또한 이 정책 설명서는 두 가지 Amazon Textract API 메서드를 사용할 수 있는 권한을 부여합니다. Amazon Comprehend는 이러한 메서드를 호출하여 이미지 파일 및 스캔한 PDF 문서에서 텍스트를 추출합니다. 이러한 유형의 입력 파일에 대해 사용자 지정 추론을 실행하지 않는 사용자에 대해서는 이러한 권한을 제거할 수 있습니다.

이 정책이 적용되는 사용자는 계정에서 배치 작업이나 비동기 작업을 수행할 수 없습니다.

자격 증명 기반 정책에서 권한을 가질 보안 주체를 지정하지 않으므로 이 정책은 `Principal` 요소를 지정하지 않습니다. 정책을 사용자에게 연결할 경우 사용자는 암시적인 보안 주체입니다. IAM 역할에 권한 정책을 연결하면 역할의 신뢰 정책에서 식별된 보안 주체가 권한을 얻습니다.

모든 Amazon Comprehend API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 서비스 승인 참조의 Amazon Comprehend의 작업, 리소스 및 조건 키를 참조하세요.

KMS 암호화를 사용하는 데 필요한 권한

비동기 작업의 데이터 및 작업 암호화에 Amazon KMS(Amazon Key Management Service)를 온전히 사용하려면 다음 정책에 표시된 작업에 권한을 부여해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Action": [
            "kms:CreateGrant"
        ],
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDatakey"
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": [
                    "s3.region.amazonaws.com"
                ]
            }
        }
    }
  ]
}
```

Amazon Comprehend로 비동기 작업을 생성할 때는 Amazon S3에 저장된 입력 데이터를 사용합니다. S3를 사용하면 저장된 데이터를 암호화할 수 있는 옵션이 있습니다. 이 암호화는 Amazon Comprehend가 아닌 S3에 의해 이루어집니다. Amazon Comprehend 작업에서 사용하는 데이터 액세스 역할에 원본 입력 데이터를 암호화하는 데 사용한 키에 대한 `kms:Decrypt` 권한을 제공하면 암호화된 입력 데이터를 해독하고 읽을 수 있습니다.

KMS 고객 관리형 키(CMK)를 사용하여 S3의 출력 결과와 작업 처리 중에 사용되는 스토리지 볼륨을 암호화하는 옵션도 있습니다. 이렇게 하면 두 가지 유형의 암호화에 동일한 KMS 키를 사용할 수 있지만 반드시 그럴 필요는 없습니다. 작업을 생성할 때 별도의 필드를 사용하여 출력 암호화 및 볼륨 암호화를 위한 키를 지정할 수 있으며, 다른 계정의 KMS 키를 사용할 수도 있습니다.

KMS 암호화를 사용하는 경우 볼륨 암호화에는 `kms:CreateGrant` 권한이 필요하고 출력 데이터 암호화에는 `kms:GenerateDataKey` 권한이 필요합니다. 암호화된 입력을 읽으려면(Amazon S3에서

입력 데이터를 이미 암호화한 경우), `kms:Decrypt` 권한이 필요합니다. IAM 역할은 필요에 따라 이러한 권한을 부여해야 합니다. 하지만 현재 사용 중인 계정과 다른 계정에서 키를 가져온 경우, 해당 KMS 키에 대한 KMS 키 정책은 작업의 데이터 액세스 역할에도 이러한 권한을 부여해야 합니다.

AWS Amazon Comprehend에 대한 관리형(미리 정의된) 정책

AWS 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 이러한 AWS 관리형 정책은 일반적인 사용 사례에 필요한 권한을 부여하므로 필요한 권한을 조사할 필요가 없습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은 Amazon Comprehend에만 해당됩니다.

- ComprehendFullAccess – 주제 모델링 작업 실행을 포함하여 Amazon Comprehend 리소스에 대한 완전한 액세스 권한을 부여합니다. IAM 역할을 나열하고 가져올 수 있는 권한을 포함합니다.
- ComprehendReadOnly – `StartDominantLanguageDetectionJob`, `StartEntitiesDetectionJob`, `StartKeyPhrasesDetectionJob`, `StartSentimentDetectionJob`, `StartTargetedSentimentDetectionJob`, `StartTopicsDetectionJob`을 제외한 모든 Amazon Comprehend 작업을 실행할 권한을 부여합니다.

Amazon Comprehend를 사용할 모든 사용자에게 다음과 같은 추가 정책을 적용해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement":
      [
        {
          "Action":
            [
              "iam:PassRole"
            ],
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/*Comprehend*"
        }
      ]
  }
```

IAM 콘솔에 로그인하고 이 콘솔에서 특정 정책을 검색하여 관리형 권한 정책을 검토할 수 있습니다.

이러한 정책은 AWS SDKs 또는 AWS CLI를 사용할 때 작동합니다.

Amazon Comprehend 작업 및 리소스에 대한 권한을 허용하는 고유의 사용자 정의 IAM 정책을 생성할 수도 있습니다. 해당 권한이 필요한 사용자, 역할 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

비동기 작업에 필요한 역할 기반 권한

Amazon Comprehend 비동기 작업을 사용하려면 문서 컬렉션이 포함된 Amazon S3 버킷에 대한 Amazon Comprehend의 액세스 권한을 부여해야 합니다. Amazon Comprehend 서비스 보안 주체를 신뢰하는 신뢰 정책을 사용하여 계정에 데이터 액세스 역할을 생성하면 됩니다. 자세한 내용은 AWS 자격 증명 및 액세스 관리 사용 설명서에서 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

다음 예제는 역할에 연결할 수 있는 신뢰 정책을 보여줍니다. 혼동된 대리인 방지를 위해 하나 이상의 글로벌 조건 컨텍스트 키를 사용하여 권한 범위를 제한합니다. `aws:SourceAccount` 값을 계정 ID로 설정합니다. `ArnEquals` 조건을 사용하는 경우, `aws:SourceArn` 값을 작업의 ARN으로 설정합니다. Amazon Comprehend가 작업 생성의 일부로 이 번호를 생성하므로 ARN의 작업 번호에는 와일드카드를 사용하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "comprehend.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringEquals": {
            "aws:SourceAccount": "111122223333"
          },
          "ArnEquals": {
            "aws:SourceArn": "arn:aws:comprehend:us-west-2:111122223333:pii-entities-
detection-job/*"
          }
        }
      }
    ]
 }
```

역할을 생성한 후 해당 역할에 대한 액세스 정책을 생성하십시오. 이렇게 하면 입력 데이터가 들어 있는 Amazon S3 버킷에 Amazon S3 `GetObject` 및 `ListBucket` 권한을 부여하고, Amazon S3 출력 데이터 버킷에 Amazon S3 `PutObject` 권한을 부여합니다.

모든 Amazon Comprehend 작업을 허용할 수 있는 권한

에 가입 AWS한 후 사용자 생성 및 권한 관리를 포함하여 계정을 관리할 관리자 사용자를 생성합니다.

Amazon Comprehend를 사용하기 위해 모든 Amazon Comprehend 작업에 대한 권한을 가진 사용자(서비스별 관리자)를 생성할 수도 있습니다. 이 사용자에게 다음 권한 정책을 연결할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Sid": "AllowAllComprehendActions",
        "Effect": "Allow",
        "Action":
        [
            "comprehend:*",
            "iam:ListRoles",
            "iam:GetRole",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:GetBucketLocation",
            "iam:CreateRole",
            "iam:CreatePolicy",
            "iam:AttachRolePolicy",
            "kms:CreateGrant",
            "kms:Decrypt",
            "kms:GenerateDatakey"
      ],
      "Resource": "*"
    },
    {
      "Action":
        [
          "iam:PassRole"
        ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/*Comprehend*"
    }
```

```
  ]
}
```

암호화와 관련하여 이러한 권한은 다음과 같은 방법으로 수정할 수 있습니다.

- Amazon Comprehend가 암호화된 S3 버킷에 저장된 문서를 분석할 수 있도록 하려면 IAM 역할에 `kms:Decrypt` 권한이 있어야 합니다.
- Amazon Comprehend가 분석 작업을 처리하는 컴퓨팅 인스턴스에 연결된 스토리지 볼륨에 저장된 문서를 암호화할 수 있도록 하려면 IAM 역할에 `kms:CreateGrant` 권한이 있어야 합니다.
- Amazon Comprehend가 S3 버킷의 출력 결과를 암호화할 수 있도록 하려면 IAM 역할에 `kms:GenerateDataKey` 권한이 있어야 합니다.

주제 모델링 작업을 허용할 수 있는 권한

다음 권한 정책은 사용자에게 Amazon Comprehend 주제 모델링 작업을 수행할 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
  "Sid": "AllowTopicModelingActions",
  "Effect": "Allow",
  "Action": [
            "comprehend:DescribeTopicsDetectionJob",
            "comprehend:ListTopicsDetectionJobs",
            "comprehend:StartTopicsDetectionJob",
         ],
        "Resource": "*"
        ]
    }
  ]
  }
```

사용자 지정 비동기 분석 작업에 필요한 권한

> **Important**
>
> 모델 액세스를 제한하는 IAM 정책이 있는 경우 사용자 지정 모델로는 추론 작업을 완료할 수 없습니다. 사용자 지정 비동기 분석 작업을 위한 와일드카드 리소스를 포함하도록 IAM 정책을 업데이트해야 합니다.

StartDocumentClassificationJob 및 StartEntitiesDetectionJob API를 사용하는 경우, 현재 와일드카드를 리소스로 사용하고 있지 않는 한 IAM 정책을 업데이트해야 합니다. 사전 학습된 모델을 사용하여 StartEntitiesDetectionJob을 사용하는 경우 이는 아무런 영향을 주지 않으며 아무것도 변경할 필요가 없습니다.

다음 예제 정책에는 오래된 참조가 포함되어 있습니다.

```
{
    "Action": [
        "comprehend:StartDocumentClassificationJob",
        "comprehend:StartEntitiesDetectionJob",
    ],
    "Resource": [
        "arn:aws:comprehend:us-east-1:123456789012:document-classifier/myClassifier",
        "arn:aws:comprehend:us-east-1:123456789012:entity-recognizer/myRecognizer"
    ],
    "Effect": "Allow"
}
```

이 정책은 StartDocumentClassificationJob 및 StartEntitiesDetectionJob을 성공적으로 실행하는 데 사용해야 하는 업데이트된 정책입니다.

```
{
    "Action": [
        "comprehend:StartDocumentClassificationJob",
        "comprehend:StartEntitiesDetectionJob",
    ],
    "Resource": [
        "arn:aws:comprehend:us-east-1:123456789012:document-classifier/myClassifier",
        "arn:aws:comprehend:us-east-1:123456789012:document-classification-job/*",
        "arn:aws:comprehend:us-east-1:123456789012:entity-recognizer/myRecognizer",
        "arn:aws:comprehend:us-east-1:123456789012:entities-detection-job/*"
    ],
    "Effect": "Allow"
}
```

AWS Amazon Comprehend에 대한 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 직접 정책을 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과

전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이 정책은 일반적인 사용 사례를 다루며 사용자의 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스에서 때때로 추가 권한을 AWS 관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새로운 기능이 시작되거나 새 작업을 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트가 기존 권한을 손상시키지 않습니다.

또한는 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnlyAccess AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면는 새 작업 및 리소스에 대한 읽기 전용 권한을 AWS 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: ComprehendFullAccess

이 정책은 주제 모델링 작업 실행을 포함하여 Amazon Comprehend 리소스에 대한 모든 액세스 권한을 부여합니다. 또한 이 정책은 Amazon S3 버킷 및 IAM 역할에 대한 목록 작성 및 가져오기 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "comprehend:*",
                "iam:GetRole",
                "iam:ListRoles",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

AWS 관리형 정책: ComprehendReadOnly

이 정책은 다음을 제외한 모든 Amazon Comprehend 작업을 실행할 수 있는 읽기 전용 권한을 부여합니다.

- `StartDominantLanguageDetectionJob`
- `StartEntitiesDetectionJob`
- `StartKeyPhrasesDetectionJob`
- `StartSentimentDetectionJob`
- `StartTargetedSentimentDetectionJob`
- `StartTopicsDetectionJob`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "comprehend:BatchDetectDominantLanguage",
                "comprehend:BatchDetectEntities",
                "comprehend:BatchDetectKeyPhrases",
                "comprehend:BatchDetectSentiment",
                "comprehend:BatchDetectSyntax",
                "comprehend:ClassifyDocument",
                "comprehend:ContainsPiiEntities",
                "comprehend:DescribeDocumentClassificationJob",
                "comprehend:DescribeDocumentClassifier",
                "comprehend:DescribeDominantLanguageDetectionJob",
                "comprehend:DescribeEndpoint",
                "comprehend:DescribeEntitiesDetectionJob",
                "comprehend:DescribeEntityRecognizer",
                "comprehend:DescribeKeyPhrasesDetectionJob",
                "comprehend:DescribePiiEntitiesDetectionJob",
                "comprehend:DescribeResourcePolicy",
                "comprehend:DescribeSentimentDetectionJob",
                "comprehend:DescribeTargetedSentimentDetectionJob",
                "comprehend:DescribeTopicsDetectionJob",
                "comprehend:DetectDominantLanguage",
                "comprehend:DetectEntities",
```

```
                "comprehend:DetectKeyPhrases",
                "comprehend:DetectPiiEntities",
                "comprehend:DetectSentiment",
                "comprehend:DetectSyntax",
                "comprehend:ListDocumentClassificationJobs",
                "comprehend:ListDocumentClassifiers",
                "comprehend:ListDocumentClassifierSummaries",
                "comprehend:ListDominantLanguageDetectionJobs",
                "comprehend:ListEndpoints",
                "comprehend:ListEntitiesDetectionJobs",
                "comprehend:ListEntityRecognizers",
                "comprehend:ListEntityRecognizerSummaries",
                "comprehend:ListKeyPhrasesDetectionJobs",
                "comprehend:ListPiiEntitiesDetectionJobs",
                "comprehend:ListSentimentDetectionJobs",
                "comprehend:ListTargetedSentimentDetectionJobs",
                "comprehend:ListTagsForResource",
                "comprehend:ListTopicsDetectionJobs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AWS 관리형 정책에 대한 Amazon Comprehend 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 Amazon Comprehend의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon Comprehend 문서 기록 페이지에서 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
ComprehendReadOnly – 기존 정책에 업데이트	Amazon Comprehend는 이제 ComprehendReadOnly 정책의 `comprehend:DescribeTargetedSentimentDetectionJob` 및 `comprehend:ListTargetedSentimentDete`	2022년 3월 30일

변경 사항	설명	날짜
	ctionJobs 작업을 허용합니다.	
ComprehendReadOnly – 기존 정책에 업데이트	Amazon Comprehend는 이제 ComprehendReadOnly 정책의 comprehend:DescribeResourcePolicy 작업을 허용합니다.	2022년 2월 2일
ComprehendReadOnly – 기존 정책에 업데이트	Amazon Comprehend는 이제 ComprehendReadOnly 정책의 ListDocumentClassifierSummaries 및 ListEntityRecognizerSummaries 작업을 허용합니다.	2021년 9월 21일
ComprehendReadOnly – 기존 정책에 업데이트	Amazon Comprehend는 이제 ComprehendReadOnly 정책의 ContainsPIIEntities 작업을 허용합니다.	2021년 3월 26일
Amazon Comprehend에서 변경 내용 추적 시작	Amazon Comprehend는 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 3월 1일

Amazon Comprehend 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amazon Comprehend와 IAM에서 작업할 때 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon Comprehend에서 작업을 수행할 권한이 없음
- 저에게 iam:PassRole을 수행할 권한이 없습니다.
- 내 외부의 사람이 내 Amazon Comprehend 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

Amazon Comprehend에서 작업을 수행할 권한이 없음

작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 comprehend:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 comprehend:GetWidget on resource: my-example-widget
```

이 경우 Mateo의 정책은 comprehend:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스하도록 허용하도록 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

저에게 iam:PassRole을 수행할 권한이 없습니다.

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon Comprehend에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 Amazon Comprehend에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 Amazon Comprehend 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제

어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- Amazon Comprehend에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon Comprehend에서 IAM을 사용하는 방법을 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유 AWS 계정 한 다른의 IAM 사용자에게 액세스 권한 제공을 참조하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

AWS CloudTrail를 사용하여 Amazon Comprehend API 호출 로깅

Amazon Comprehend는 Amazon Comprehend에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Amazon Comprehend에 대한 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Amazon Comprehend 콘솔로부터의 호출과 Amazon Comprehend API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 Amazon Comprehend 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon Comprehend에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

구성 및 사용 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Amazon Comprehend 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. Amazon Comprehend에서 지원되는 이벤트 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. 에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다 AWS 계정. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하십시오.

Amazon Comprehend에 대한 이벤트를 AWS 계정포함하여에서 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 기본적으로 콘솔에서 추적을 생성하면 추적이 모든 AWS 리전에 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

Amazon Comprehend는 CloudTrail 로그 파일의 이벤트로 다음 작업의 로깅을 지원합니다.

- BatchDetectDominantLanguage
- BatchDetectEntities
- BatchDetectKeyPhrases
- BatchDetectSentiment
- BatchDetectSyntax
- ClassifyDocument
- CreateDocumentClassifier
- CreateEndpoint
- CreateEntityRecognizer
- DeleteDocumentClassifier
- DeleteEndpoint
- DeleteEntityRecognizer
- DescribeDocumentClassificationJob
- DescribeDocumentClassifier
- DescribeDominantLanguageDetectionJob
- DescribeEndpoint
- DescribeEntitiesDetectionJob
- DescribeEntityRecognizer
- DescribeKeyPhrasesDetectionJob

- DescribePiiEntitiesDetectionJob
- DescribeSentimentDetectionJob
- DescribeTargetedSentimentDetectionJob
- DescribeTopicsDetectionJob
- DetectDominantLanguage
- DetectEntities
- DetectKeyPhrases
- DetectPiiEntities
- DetectSentiment
- DetectSyntax
- ListDocumentClassificationJobs
- ListDocumentClassifiers
- ListDominantLanguageDetectionJobs
- ListEndpoints
- ListEntitiesDetectionJobs
- ListEntityRecognizers
- ListKeyPhrasesDetectionJobs
- ListPiiEntitiesDetectionJobs
- ListSentimentDetectionJobs
- ListTargetedSentimentDetectionJobs
- ListTagsForResource
- ListTopicsDetectionJobs
- StartDocumentClassificationJob
- StartDominantLanguageDetectionJob
- StartEntitiesDetectionJob
- StartKeyPhrasesDetectionJob
- StartPiiEntitiesDetectionJob
- StartSentimentDetectionJob
- StartTargetedSentimentDetectionJob
- StartTopicsDetectionJob

- StopDominantLanguageDetectionJob
- StopEntitiesDetectionJob
- StopKeyPhrasesDetectionJob
- StopPiiEntitiesDetectionJob
- StopSentimentDetectionJob
- StopTargetedSentimentDetectionJob
- StopTrainingDocumentClassifier
- StopTrainingEntityRecognizer
- TagResource
- UntagResource
- UpdateEndpoint

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 사용자 자격 증명으로 했는지 여부.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청을 했는지 여부입니다.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

예제: Amazon Comprehend 로그 파일 항목

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `ClassifyDocument` 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예시입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAICFHPEXAMPLE",
```

```
        "arn": "arn:aws:iam::12345678910:user/myadmin2",
        "accountId": "12345678910",
        "accessKeyId": "ASIA3VZEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-10-19T14:22:09Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-10-19T17:31:20Z",
    "eventSource": "comprehend.amazonaws.com",
    "eventName": "ClassifyDocument",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "3.21.185.237",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0)
 Gecko/20100101 Firefox/115.0",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "fd916e66-caac-46c9-a1fc-81a0ef33e61b",
    "eventID": "535ca22b-b3a3-4c13-b2c5-bf51ab082794",
    "readOnly": false,
    "resources": [
        {
            "accountId": "12345678910",
            "type": "AWS::Comprehend::DocumentClassifierEndpoint",
            "ARN": "arn:aws:comprehend:us-east-2:12345678910:document-classifier-
endpoint/endpointExample"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "12345678910"
}
```


Amazon Comprehend에 대한 규정 준수 검증

타사 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 Amazon Comprehend의 보안 및 규정 준수를 평가합니다. 여기에는 PCI, FedRAMP, HIPAA 등이 포함됩니다. 를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 AWS 아티팩트에서 보고서 다운로드를 참조하세요.

Amazon Comprehend 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS 에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 -이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공합니다 AWS.
- HIPAA 보안 및 규정 준수 백서 설계 -이 백서에서는 기업이를 사용하여 HIPAA 준수 애플리케이션을 AWS 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 업계 및 위치에 적용될 수 있습니다.
- AWS Config -이 AWS 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

특정 규정 준수 프로그램 범위의 AWS 서비스 목록은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 참조하세요. 일반 정보는 AWS 규정 준수 프로그램을 참조하세요.

Amazon Comprehend 복원성

AWS 글로벌 인프라는 및 가용 영역을 기반으로 구축됩니다.는 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공하며, 이는 지연 시간이 짧고 처리량이 AWS 리전 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

Amazon Comprehend의 인프라 보안

관리형 서비스인 Amazon Comprehend는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Amazon Comprehend에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

지침 및 할당량

달리 지정하지 않는 한, Amazon Comprehend 할당량은 리전별로 적용됩니다. 애플리케이션에 필요한 경우 조정 가능한 할당량 증가를 요청할 수 있습니다. 할당량과 할당량 증가를 요청하는 방법에 대한 자세한 내용은 AWS Service Quotas을 참조하세요.

주제

지원되는 리전

Amazon Comprehend는 다음 AWS 리전에서 사용할 수 있습니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(오리건)
- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- AWS GovCloud(미국 서부)

기본적으로 Amazon Comprehend는 지원되는 각 리전에서 모든 API 작업을 제공합니다. 예외 사항은 문서 처리를 참조하세요.

API 엔드포인트 사용에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon Comprehend 리전 및 엔드포인트를 참조하세요.

리전의 현재 할당량을 검토하거나 조정 가능한 할당량에 대한 할당량 증가를 요청하려면 Service Quotas 콘솔을 여십시오.

내장 모델의 할당량

Amazon Comprehend는 UTF-8 텍스트 문서를 분석할 수 있는 내장 모델을 제공합니다. Amazon Comprehend는 내장 모델을 사용하는 동기 및 비동기 작업을 제공합니다.

주제

- 실시간(동기) 분석
- 비동기 분석

실시간(동기) 분석

이 섹션에서는 내장 모델을 사용한 실시간 분석과 관련된 할당량을 설명합니다.

주제

- 단일 문서 작업
- 다중 문서 작업
- 실시간(동기식) 요청에 대한 요청 제한

단일 문서 작업

Amazon Comprehend API는 단일 문서를 입력으로 취하는 작업을 제공합니다. 다음 할당량이 이러한 작업에 적용됩니다.

단일 문서 작업에 대한 일반 할당량

다음 할당량은 개체, 핵심 문구 또는 지배적 언어를 감지하기 위한 실시간 분석에 적용됩니다. 개체 감지의 경우 이러한 할당량은 기본 제공 모델을 사용한 감지에 적용됩니다. 사용자 지정 개체 감지에 대해서는 사용자 지정 개체 인식 의 할당량을 참조하세요.

설명	할당량/지침
최대 문서 크기	100KB

단일 문서 작업에 대한 작업별 할당량

다음 할당량은 감성, 대상 감성 및 구문을 감지하기 위한 실시간 분석에 적용됩니다.

설명	할당량/지침
최대 문서 크기	5KB

다중 문서 작업

Amazon Comprehend API는 단일 API 요청으로 여러 문서를 처리하는 일괄 작업을 제공합니다. 다음 할당량이 일괄 작업에 적용됩니다.

설명	할당량/지침
최대 문서 크기	5KB
요청당 최대 문서 수	25

일괄 작업의 사용에 대한 자세한 내용은 다중 문서 동기 처리을 참조하세요.

실시간(동기식) 요청에 대한 요청 제한

Amazon Comprehend는 동기 요청에 동적 제한을 적용합니다. 시스템 처리 대역폭을 사용할 수 있는 경우 Amazon Comprehend는 처리하는 요청 수를 점진적으로 늘립니다. 애플리케이션의 동기 API 작업 사용을 제어하려면 결제 알림을 켜거나 애플리케이션에서 속도 제한을 구현하는 것이 좋습니다.

비동기 분석

이 섹션에서는 내장 모델을 사용한 비동기 분석과 관련된 할당량을 설명합니다.

비동기 API 작업은 각각 최대 10개의 활성 작업을 지원합니다. 각 API 작업에 대한 할당량을 보려면 Amazon Comprehend 엔드포인트의 Service Quotas 표와 Amazon Web Services 일반 참조의 할당량을 참조하세요.

조정 가능한 할당량의 경우 Service Quotas 콘솔을 사용하여 할당량 증가를 요청할 수 있습니다.

주제

- 비동기 작업에 대한 일반 할당량
- 비동기 작업에 대한 작업별 할당량
- 비동기 요청에 대한 요청 제한

비동기 작업에 대한 일반 할당량

콘솔이나 API `Start*` 작업을 사용하여 비동기 분석 작업을 실행할 수 있습니다. 비동기 작업을 사용할 시점에 대한 자세한 내용은 비동기 일괄 처리을 참조하세요. 다음 할당량은 내장 모델의 대부분의 API `Start*` 작업에 적용됩니다. 예외 사항은 비동기 작업에 대한 작업별 할당량을 참조하세요.

설명	할당량/지침
항목, 핵심 문구, PII 및 언어를 감지하는 작업에서 각 문서의 최대 크기	1MB
요청 내 모든 파일 크기 합계의 최대치	5GB
요청 내 모든 파일 크기 합계의 최소치	500바이트
최대 파일 수, 파일당 문서 하나	1,000,000
라인 수 합계의 최대치, 라인 하나당 문서 하나	1,000,000

비동기 작업에 대한 작업별 할당량

이 섹션에서는 특정 비동기 작업의 할당량을 설명합니다. 다음 표에 할당량이 지정되지 않은 경우 일반 할당량 값이 적용됩니다.

주제

- 감성

- 대상 감성
- 이벤트
- 주제 모델링

감성

StartSentimentDetectionJob 작업으로 생성하는 비동기 감성 작업의 할당량은 다음과 같습니다.

설명	할당량/지침
각 입력 문서의 최대 크기	5KB

대상 감성

StartTargetedSentimentDetectionJob 작업으로 생성하는 비동기 대상 감성 작업의 할당량은 다음과 같습니다.

설명	할당량/지침
지원되는 문서 형식	UTF-8
한 작업 내 각 문서의 최대 크기	10KB
한 작업 내 모든 문서의 최대 크기	300MB
최대 파일 수, 파일당 문서 하나	30,000개
라인 수 합계의 최대치, 줄 하나에 문서 하나 (요청 내 모든 파일)	30,000개

이벤트

StartEventsDetectionJob 작업으로 생성하는 비동기 이벤트 감지 작업의 할당량은 다음과 같습니다.

설명	할당량
문자 인코딩	UTF-8

설명	할당량
한 작업 내 모든 파일 크기의 합계	50MB
한 작업 내 각 문서의 최대 크기	10KB
최대 파일 수, 파일당 문서 하나	5,000
라인 수 합계의 최대치, 라인 하나에 문서 하나 (요청된 모든 파일에 대하여)	5,000

주제 모델링

StartTopicsDetectionJob 작업으로 생성하는 비동기 주제 모델링 작업의 할당량은 다음과 같습니다.

설명	할당량/지침
문자 인코딩	UTF-8
리턴할 주제의 최대 수	100
파일 1개에 대한 최대 파일 크기, 파일당 문서 하나	100MB

자세한 정보는 주제 모델링을 참조하십시오.

비동기 요청에 대한 요청 제한

각 비동기 API 작업은 초당 최대 요청 수(리전별, 계정당)와 최대 10개의 활성 작업을 지원합니다. 각 API 작업에 대한 할당량을 보려면 Amazon Comprehend 엔드포인트의 Service Quotas 표와 Amazon Web Services 일반 참조의 할당량을 참조하세요.

조정 가능한 할당량의 경우 Service Quotas 콘솔을 사용하여 할당량 증가를 요청할 수 있습니다.

사용자 지정 모델의 할당량

Amazon Comprehend를 사용하여 사용자 지정 분류 및 사용자 지정 개체 인식을 위한 고유의 사용자 지정 모델을 구축할 수 있습니다. 이 섹션에서는 사용자 지정 모델 학습 및 사용과 관련된 지침 및 할당

량이 나와 있습니다. 사용자 지정 모델에 대한 자세한 내용은 Amazon Comprehend 사용자 정의을 참조하세요.

주제

일반 할당량

Amazon Comprehend는 사용자 지정 모델로 분석할 수 있는 각 입력 문서 유형에 대해 일반적인 크기 할당량을 설정합니다. 실시간 분석 할당량은 실시간 분석을 위한 최대 문서 크기을 참조하세요. 비동기 분석 할당량은 비동기 사용자 지정 분석을 위한 입력을 참조하세요.

각 비동기 API 작업은 초당 최대 요청 수(리전별, 계정당)와 최대 10개의 활성 작업을 지원합니다. 각 API 작업에 대한 할당량을 보려면 Amazon Comprehend 엔드포인트의 Service Quotas 표와 Amazon Web Services 일반 참조의 할당량을 참조하세요.

조정 가능한 할당량의 경우 Service Quotas 콘솔을 사용하여 할당량 증가를 요청할 수 있습니다.

엔드포인트의 할당량

사용자 지정 모델을 사용하여 실시간 분석을 실행할 엔드포인트를 생성합니다. 엔드포인트에 대한 정보는 Amazon Comprehend 엔드포인트 관리를 참조하세요.

엔드포인트에는 다음과 같은 할당량이 적용됩니다. 할당량 증가 요청에 대한 자세한 내용은 AWS Service Quotas를 참조하세요.

설명	할당량/지침
각 계정의 리전별 최대 활성 엔드포인트 수	20
각 계정의 리전별 최대 추론 단위 수	200
리전별 엔드포인트당 최대 추론 단위 수	50
추론 단위당 최대 처리량(문자)	100개/초

설명	할당량/지침
추론 단위당 최대 처리량(문서)	2개/초

문서 분류

이 섹션에서는 다음과 같은 문서 분류 작업에 대한 지침 및 할당량을 설명합니다.

- CreateDocumentClassifier 작업으로 시작하는 분류기 학습 작업.
- StartDocumentClassificationJob 작업으로 시작하는 비동기 문서 분류 작업.
- ClassifyDocument 작업을 사용하는 동기식 문서 분류 요청.

문서 분류를 위한 일반 할당량

다음 표에는 사용자 지정 분류기 학습과 관련된 일반 할당량이 설명되어 있습니다.

설명	할당량/지침
클래스 명칭의 최대 길이	5,000자
클래스 수(멀티클래스 모드)	2~1,000개
클래스 수(멀티레이블 모드)	2-100개
주석 형식	
클래스당 최소 주석 수(멀티클래스 모드)	10
클래스당 최소 주석 수(멀티레이블 모드)	10
최소 주석 수(멀티레이블 모드)	50
CSV 파일 형식	
클래스당 최소 학습 문서 수(멀티클래스 모드)	50
클래스당 최소 학습 문서 수(멀티레이블 모드)	10
최소 학습 문서 수(멀티레이블 모드)	50

일반 텍스트 문서의 분류

일반 텍스트 입력 문서를 사용하여 일반 텍스트 모델을 만들고 학습시킵니다. Amazon Comprehend는 일반 텍스트 모델을 사용하여 일반 텍스트 문서를 분류하는 실시간 및 비동기 작업을 제공합니다.

학습

다음 표에는 일반 텍스트 문서를 사용한 사용자 지정 분류기 학습과 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
학습 작업에 있는 모든 파일의 크기의 합계	5GB
사용자 지정 분류기를 학습하기 위한 증강 매니페스트 파일의 최대 수	5
각 증강 매니페스트 파일의 최대 속성 이름 수	5
속성 이름의 최대 길이	63자

실시간(동기) 분석

다음 표에는 일반 텍스트 문서를 사용한 실시간 분류와 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
동기 요청당 최대 문서 수	1
최대 텍스트 문서 크기(UTF-8 인코딩)	10KB

비동기 분석

다음 표에는 일반 텍스트 문서를 사용한 비동기 분류와 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
비동기 작업에 있는 모든 파일 크기의 합계	5GB

설명	할당량/지침
파일 1개에 대한 최대 파일 크기, 파일당 문서 하나	10MB
최대 파일 수, 파일당 문서 하나	1,000,000
라인 수 합계의 최대치, 라인 하나에 문서 하나 (요청된 모든 파일에 대하여)	1,000,000

반정형 문서의 분류

이 섹션에서는 반정형 문서의 문서 분류에 대한 지침 및 할당량에 대해 설명합니다. 반정형 문서를 분류하려면 네이티브 입력 문서로 학습시킨 네이티브 문서 모델을 사용하십시오.

반정형 문서로 네이티브 문서 모델 학습

다음 표에는 PDF 문서, Word 문서, 이미지 파일과 같은 반정형 문서를 사용한 사용자 지정 분류기 학습과 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
모든 문서의 최대 페이지 수	10,000개
최대 주석 파일 크기(모든 CSV 파일 크기 합산)	5MB
문서 코퍼스 크기(학습 및 테스트 문서)	10GB
학습 및 테스트 파일의 파일 크기	
이미지 파일 크기 (JPG, PNG, TIFF)	1바이트–10MB TIFF 파일: 최대 한 페이지
PDF 문서의 페이지 크기	1바이트–10MB
Word 문서의 페이지 크기	1바이트–10MB
Amazon Textract API 출력 JSON 크기	1바이트–1MB

실시간(동기) 분석

이 섹션에서는 반정형 문서의 실시간 분류와 관련된 할당량에 대해 설명합니다.

다음 표는 입력 문서의 최대 파일 크기를 보여줍니다. 모든 입력 문서 유형의 경우 입력 파일의 최대 크기는 1페이지이며 10,000자를 넘지 않아야 합니다.

파일 유형	최대 크기(API)	최대 크기(콘솔)
UTF-8 텍스트 문서	10KB	10KB
PDF 문서	10MB	5MB
Word 문서	10MB	5MB
이미지 파일	10MB	5MB
Amazon Textract API 출력 크기	1MB	해당 사항 없음

비동기 분석

다음 표에는 반정형 문서의 비동기 분류와 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
한 작업에 대한 모든 입력 문서의 최대 페이지 수	25,000
문서 코퍼스 크기	25GB
이미지 파일 크기 (JPG, PNG 또는 TIFF)	1바이트–10MB TIFF 파일: 최대 한 페이지
PDF 문서의 페이지 크기	1바이트–10MB
Word 문서의 페이지 크기	1바이트–10MB
Textract API 출력 JSON 크기	1바이트–1MB

사용자 지정 개체 인식

이 섹션에서는 다음과 같은 문서 분류 작업에 대한 지침 및 할당량을 설명합니다.

- 개체 인식기 학습 작업은 CreateEntityRecognizer 작업으로 시작되었습니다.
- 비동기 개체 인식 작업은 StartEntitiesDetectionJob 작업으로 시작되었습니다.
- 동기식 개체 인식에서 DetectEntities 작업의 사용을 요청합니다.

일반 텍스트 문서에 대한 사용자 지정 개체 인식

Amazon Comprehend는 사용자 지정 개체 인식기를 사용하여 일반 텍스트 문서를 분석하기 위한 비동기 및 동기화 작업을 제공합니다.

학습

이 섹션에서는 일반 텍스트 문서를 분석하도록 사용자 지정 개체 인식기를 훈련시키는 것과 관련된 할당량을 설명합니다. 모델을 학습시키기 위해 개체 목록 또는 주석이 달린 텍스트 문서 세트를 제공할 수 있습니다.

다음 표에는 개체 목록을 사용한 모델 학습과 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
모델별 개체 수	1~25개
문서 크기(UTF-8)	1~5,000바이트
개체 목록의 항목 수	1~1백만 개
항목 리스트 상의 개별 항목(포스트 스트립) 길이	1~5,000자
항목 리스트 코퍼스 크기(일반 텍스트로 된 모든 문서 합산)	5KB~200MB

다음 표에는 주석이 달린 텍스트 문서를 사용한 모델 학습과 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
모델당 개체 수/사용자 지정 개체 인식기	1~25개

설명	할당량/지침
문서 크기(UTF-8)	1~5,000바이트
문서 수(일반 텍스트 주석 참조)	3~200,000개
문서 코퍼스 크기(일반 텍스트의 모든 문서 합산)	5KB~200MB
개체당 최소 주석 수	25

실시간(동기) 분석

다음 표에는 일반 텍스트 문서의 실시간 분석과 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
동기 요청당 최대 문서 수	1
최대 텍스트 문서 크기(UTF-8 인코딩)	5KB

비동기 분석

다음 표에는 일반 텍스트 문서의 비동기 개체 인식과 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
문서 크기(UTF-8)	1바이트–1MB
최대 파일 수, 파일당 문서 하나	1,000,000
라인 수 합계의 최대치, 라인 하나에 문서 하나 (요청된 모든 파일에 대하여)	1,000,000
문서 코퍼스 크기(일반 텍스트의 모든 문서 합산)	1바이트–5GB

반정형 문서에 대한 사용자 지정 개체 인식

Amazon Comprehend는 사용자 지정 개체 인식기를 사용하여 일반 텍스트 문서를 분석하기 위한 비동기 및 동기 작업을 제공합니다. 주석이 달린 PDF 문서를 사용하여 모델을 학습시켜야 합니다.

학습

다음 표에는 반정형 문서를 분석하기 위한 사용자 지정 개체 인식기(CreateEntityRecognizer)를 학습시키는 것과 관련된 할당량이 설명되어 있습니다.

설명	할당량/지침
모델당 개체 수/사용자 지정 개체 인식기	1~25개
최대 주석 파일 크기(UTF-8 JSON)	5MB
문서 개수	250~10,000개
문서 코퍼스 크기(일반 텍스트의 모든 문서 합산)	5KB~1GB
개체당 최소 주석 수	100
사용자 지정 분류기 학습을 위한 증강 매니페스트 파일의 최대 수	5
각 증강 매니페스트 파일의 최대 속성 이름 수	5
속성 이름의 최대 길이	63자

실시간(동기) 분석

이 섹션에서는 반정형 문서의 실시간 분석와 관련된 할당량에 대해 설명합니다.

다음 표는 입력 문서의 최대 파일 크기를 보여줍니다. 모든 입력 문서 유형의 경우 입력 파일의 최대 크기는 1페이지이며 10,000자를 넘지 않아야 합니다.

파일 유형	최대 크기(API)	최대 크기(콘솔)
UTF-8 텍스트 문서	10KB	10KB
PDF 문서	10MB	5MB

파일 유형	최대 크기(API)	최대 크기(콘솔)
Word 문서	10MB	5MB
이미지 파일	10MB	5MB
Textract 출력 파일	1MB	해당 사항 없음

비동기 분석

이 섹션에서는 반정형 문서의 비동기 분석을 위한 할당량에 대해 설명합니다.

설명	할당량/지침
이미지 크기(JPG 또는 PNG)	1바이트–10MB
이미지 크기(TIFF)	1바이트–10MB 최대 한 페이지.
문서 크기(PDF)	1바이트–50MB
문서 크기(Docx)	1바이트–5MB
문서 크기(UTF-8)	1바이트–1MB
최대 파일 수, 파일당 문서 하나 (이미지 파일 또는 PDF/Word 문서에는 라인 하나에 문서 하나가 허용되지 않음)	500
PDF 또는 Docx 파일의 최대 페이지 수	100
텍스트 추출 후의 문서 코퍼스 크기(일반 텍스트, 모든 파일 합산)	1바이트–5GB

이미지 제한에 대한 자세한 내용은 Amazon Textract의 엄격한 제한을 참조하세요.

플라이휠 할당량

플라이휠을 사용하여 사용자 지정 분류 및 사용자 지정 개체 인식을 위한 사용자 지정 모델 버전의 학습 및 추적을 관리할 수 있습니다. 플리이휠에 대한 자세한 내용은 플라이 휠을 참조하세요.

플라이휠 일반 할당량

플라이휠 및 플라이휠 반복에는 다음 할당량이 적용됩니다.

설명	할당량/지침
최대 채널 수	50
CREATING 상태의 플라이휠 최대 수	10
플라이휠당 학습 데이터세트 최대 수	50
플라이휠당 테스트 데이터세트 최대 수	50
INGESTING 상태의 데이터세트 최대 수	10
계정당 진행 중인 플라이휠 반복 최대 횟수	10

사용자 지정 분류 모델의 데이터세트 할당량

사용자 지정 분류 모델과 관련된 플라이휠용 데이터세트를 주입하는 경우 다음 할당량이 적용됩니다.

설명	할당량/지침
클래스당 최소 학습 문서 수(멀티레이블 모드)	50
학습 문서 최대 개수	1,000,000
최소 데이터세트 크기	500바이트
데이터세트 최대 크기	5GB
파일 1개에 대한 최대 파일 크기, 파일당 문서 하나	10MB

사용자 지정 개체 인식 모델의 데이터세트 할당량

사용자 지정 개체 인식 모델과 관련된 플라이휠용 데이터세트를 주입하는 경우 다음 할당량이 적용됩니다.

설명	할당량/지침
최대 문서 크기	5KB
학습 문서 최소 개수	3
학습 문서 최대 개수	200,000
개체당 최소 주석 수	25
데이터세트 최대 크기	200MB

자습서 및 기타 리소스

Amazon Comprehend 자습서 및 기타 리소스

주제

- 자습서: Amazon Comprehend 고객 리뷰를 통한 인사이트 분석
- 개인 식별 정보(PII)를 위한 Amazon S3 객체 Lambda 액세스 포인트 사용
- 솔루션: Amazon Comprehend와 OpenSearch를 사용한 텍스트 분석

자습서: Amazon Comprehend 고객 리뷰를 통한 인사이트 분석

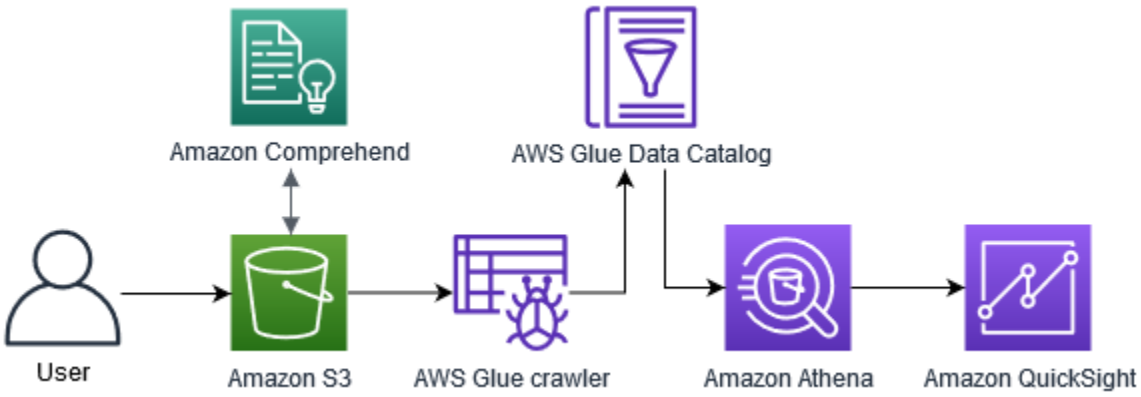
이 자습서에서는 Amazon Comprehend를 Amazon Simple Storage Service, AWS Glue, Amazon Athena, Amazon QuickSight와 함께 사용하여 문서에 대한 귀중한 인사이트를 얻는 방법을 설명합니다. Amazon Comprehend는 구조화되지 않은 텍스트에서 감성(문서의 분위기)과 개체(사람, 조직, 이벤트, 날짜, 제품, 장소, 수량, 제목)를 추출할 수 있습니다.

예를 들어 고객 리뷰를 통해 실행 가능한 인사이트를 얻을 수 있습니다. 이 자습서에서는 소설에 대한 고객 리뷰의 샘플 데이터 세트를 분석합니다. Amazon Comprehend 감성 분석을 사용하여 고객이 소설에 대해 긍정적으로 느끼는지 부정적으로 느끼는지 판단할 수 있습니다. 또한 Amazon Comprehend 개체 분석을 사용하여 관련 소설이나 저자와 같은 중요한 개체에 대한 멘션을 발견할 수 있습니다. 이 자습서를 따라하면 50% 이상의 리뷰가 긍정적임을 발견할 수 있습니다. 또한 고객이 저자를 비교하고 다른 고전 소설에 관심을 보인다는 것을 발견할 수도 있습니다.

이 자습서에서는 다음을 수행합니다.

- 리뷰의 샘플 데이터 세트를 Amazon Simple Storage Service(Amazon S3)에 저장합니다. Amazon Simple Storage Service(S3)는 스토리지 서비스입니다.
- Amazon Comprehend를 사용하여 리뷰 문서의 감성과 개체를 분석합니다.
- AWS Glue 크롤러를 사용하여 분석 결과를 데이터베이스에 저장합니다. AWS Glue 는 분석을 위해 데이터를 카탈로그화하고 정리할 수 있는 추출, 변환 및 로드(ETL) 서비스입니다.
- Amazon Athena 쿼리를 실행하여 데이터를 정리합니다. Amazon Athena 는 서버리스 대화형 쿼리 서비스입니다.
- Amazon QuickSight에서 데이터를 사용하여 시각화를 생성합니다. QuickSight는 데이터에서 인사이트를 추출하기 위한 서버리스 비즈니스 인텔리전스 도구입니다.

다음 다이어그램은 워크플로를 보여줍니다.



이 자습서를 완료하는 데 걸리는 예상 시간: 1시간

예상 비용: 이 자습서의 일부 작업은 AWS 계정에 요금이 부과됩니다. 각 서비스의 요금에 대한 자세한 내용은 다음 요금 페이지를 참조하세요.

- Amazon S3 요금
- Amazon Comprehend 요금
- AWS Glue 요금
- Amazon Athena 요금
- QuickSight 요금

주제

사전 조건

이 자습서를 완료하려면 다음이 필요합니다.

- AWS 계정. 설정에 대한 자세한 내용은 단원을 AWS 계정참조하십시오설정.

- IAM 개체(사용자, 그룹 또는 역할). 계정의 사용자 및 그룹을 설정하는 방법을 알아보려면 IAM 사용 설명서의 시작하기 자습서를 참조하세요.
- 사용자, 그룹 또는 역할에 연결된 다음 권한 정책. 정책은 이 자습서를 완료하는 데 필요한 일부 권한을 부여합니다. 다음 사전 조건에는 필요한 추가 권한이 설명되어 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement":
  [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action":
      [
        "comprehend:*",
        "ds:AuthorizeApplication",
        "ds:CheckAlias",
        "ds:CreateAlias",
        "ds:CreateIdentityPoolDirectory",
        "ds:DeleteDirectory",
        "ds:DescribeDirectories",
        "ds:DescribeTrusts",
        "ds:UnauthorizeApplication",
        "iam:AttachRolePolicy",
        "iam:CreatePolicy",
        "iam:CreatePolicyVersion",
        "iam:CreateRole",
        "iam:DeletePolicyVersion",
        "iam:DeleteRole",
        "iam:DetachRolePolicy",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetRole",
        "iam:ListAccountAliases",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy",
        "iam:ListPolicies",
        "iam:ListPolicyVersions",
        "iam:ListRoles",
        "quicksight:*",
        "s3:*",
        "tag:GetResources"
      ],
```

```
      "Resource": "*"
    },
    {
      "Action":
      [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource":
      [
        "arn:aws:iam::*:role/*Comprehend*"
      ]
    }
  ]
}
```

이전 정책을 사용하여 IAM 정책을 생성해 그룹 또는 사용자에게 연결합니다. IAM 정책 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하세요. IAM 자격 증명에 정책을 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

- IAM 그룹 또는 사용자에 연결된 관리형 정책. 이전 정책 외에도 다음 AWS 관리형 정책을 그룹 또는 사용자에게 연결해야 합니다.
  - `AWSGlueConsoleFullAccess`
  - `AWSQuicksightAthenaAccess`

  이러한 관리형 정책은 AWS Glue Amazon Athena, 및 QuickSight를 사용할 수 있는 권한을 부여합니다. IAM 자격 증명에 정책을 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

1단계: Amazon S3에 문서 추가

Amazon Comprehend 분석 작업을 시작하기 전에 Amazon Simple Storage Service(Amazon S3)에 샘플 고객 리뷰 데이터 세트를 저장해야 합니다. Amazon S3는 버킷이라는 컨테이너에서 데이터를 호스팅합니다. Amazon Comprehend는 버킷에 저장된 문서를 분석하고 분석 결과를 버킷으로 전송할 수 있습니다. 이 단계에서는 S3 버킷을 생성하고 이 버킷에 입력 및 출력 폴더를 생성하고 이 버킷에 샘플 데이터 세트를 업로드합니다.

주제

- 사전 조건

- 샘플 데이터 다운로드
- Amazon S3 버킷 생성
- (콘솔 전용) 폴더 생성
- 입력 데이터 업로드

사전 조건

시작하기 전에 자습서: Amazon Comprehend 고객 리뷰를 통한 인사이트 분석를 검토하고 필수 조건을 충족해야 합니다.

샘플 데이터 다운로드

다음 샘플 데이터세트에는 “텍스트 분류를 위한 문자 수준 컨볼루션 네트워크”(Xiang Zhang et al., 2015)라는 기사와 함께 게시된 대규모 데이터 세트 “Amazon 리뷰 - 전체”에서 가져온 Amazon 리뷰가 포함되어 있습니다. 데이터세트를 컴퓨터에 다운로드합니다.

샘플 데이터 가져오기

1. tutorial-reviews-data.zip zip 파일을 컴퓨터에 다운로드합니다.
2. 컴퓨터에서 zip 파일을 추출합니다. 두 개의 파일이 있습니다. `THIRD_PARTY_LICENSES.txt` 파일은 Xiang Zhang 등이 게시한 데이터세트의 오픈 소스 라이선스입니다. `amazon-reviews.csv` 파일은 자습서에서 분석하는 데이터세트입니다.

Amazon S3 버킷 생성

샘플 데이터세트를 다운로드한 후 Amazon S3 버킷을 생성하여 입력 및 출력 데이터를 저장합니다. Amazon S3 콘솔 또는 AWS Command Line Interface (AWS CLI)를 사용하여 S3 버킷을 생성할 수 있습니다.

Amazon S3 버킷 생성(콘솔)

Amazon S3 콘솔에서 모든 AWS에 고유한 이름을 가진 버킷을 생성합니다.

S3 버킷 생성(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 버킷에서 버킷 생성을 선택합니다.

3. 버킷 이름에서, 버킷의 용도를 설명하는 전역적으로 고유한 이름을 입력합니다.
4. 리전에서 버킷을 생성할 AWS 리전을 선택합니다. 선택한 리전은 Amazon Comprehend를 지원해야 합니다. 지연 시간을 줄이려면 Amazon Comprehend에서 지원하는 지리적 위치와 가장 가까운 AWS 리전을 선택합니다. Amazon Comprehend를 지원하는 리전 목록은 글로벌 인프라 안내서의 리전 표를 참조하세요.
5. 객체 소유권, 퍼블릭 액세스 차단에 대한 버킷 설정, 버킷 버전 관리 및 태그에 대한 기본 설정은 그대로 두십시오.
6. 기본 암호화의 경우, 비활성화를 선택합니다.

> **Tip**
>
> 이 자습서에서는 암호화를 사용하지 않지만 중요한 데이터를 분석할 때는 암호화를 사용하는 것이 좋습니다. 엔드-투-엔드 암호화의 경우, 버킷에 저장된 데이터를 암호화할 수 있으며 분석 작업을 실행할 때도 암호화할 수 있습니다. 를 사용한 암호화에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 란 무엇입니까 AWS Key Management Service?를 AWS참조하세요.

7. 버킷 구성을 검토한 다음 버킷 생성을 선택합니다.

Amazon S3 버킷(AWS CLI)을 생성합니다.

를 연 후 `create-bucket` 명령을 AWS CLI실행하여 입력 및 출력 데이터를 저장할 버킷을 생성합니다.

Amazon S3 버킷 생성(AWS CLI)

1. 버킷을 생성하려면, AWS CLI에서 다음 명령을 실행합니다. amzn-s3-demo-bucket을 모든에서 고유한 버킷의 이름으로 바꿉니다 AWS.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket
```

기본적으로 `create-bucket` 명령은 `us-east-1` AWS 리전에 버킷을 생성합니다. `us-east-1`과 다른 AWS 리전 에 버킷을 생성하려면 `LocationConstraint` 파라미터를 추가하여 리전을 지정합니다. 예를 들어, 다음 명령은 `us-west-2` 리전에서 파일 시스템을 생성합니다.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket
--region us-west-2 --create-bucket-configuration LocationConstraint=us-west-2
```

단, 특정 리전에서만 Amazon Comprehend를 지원합니다. Amazon Comprehend를 지원하는 리전 목록은 글로벌 인프라 안내서의 리전 표를 참조하세요.

2. 버킷이 성공적으로 생성되었는지 확인하려면 다음 명령을 실행합니다. 이 명령은 계정에 연결된 모든 S3 버킷을 나열합니다.

```
aws s3 ls
```

(콘솔 전용) 폴더 생성

다음으로, S3 버킷에 폴더 두 개를 생성합니다. 첫 번째 폴더는 입력 데이터 폴더입니다. 두 번째 폴더는 Amazon Comprehend가 분석 결과를 보내는 곳입니다. Amazon S3 콘솔을 사용하는 경우 폴더를 수동으로 생성해야 합니다. 를 사용하는 경우 샘플 데이터 세트를 업로드하거나 분석 작업을 실행할 때 폴더를 생성할 AWS CLI수 있습니다. 이러한 이유로 콘솔 사용자만을 위한 폴더를 생성하는 절차를 제공합니다. AWS CLI를 사용하는 경우 입력 데이터 업로드 및 3단계: Amazon S3의 문서에 대한 분석 작업 실행에 폴더를 생성합니다.

S3 버킷에서 폴더 생성(콘솔)

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷에서, 버킷 목록 중 해당 버킷을 선택합니다.
3. 개요 탭에서 폴더 생성을 선택합니다.
4. 폴더의 새 이름으로 `input`을 입력합니다.
5. 암호화 설정에서 없음(버킷 설정 사용)을 선택합니다.
6. 저장을 선택합니다.
7. 3~6단계를 반복하여 분석 작업 출력을 위한 또 다른 폴더를 생성하되, 4단계에서 새 폴더 이름 `output`을 입력합니다.

입력 데이터 업로드

이제 버킷이 생겼으니 샘플 데이터세트 `amazon-reviews.csv`를 업로드하십시오. Amazon S3 콘솔 또는 AWS CLI를 사용하여 S3 버킷에 데이터를 업로드할 수 있습니다.

버킷에 샘플 문서 업로드(콘솔)

Amazon S3 콘솔에서 샘플 데이터 세트 파일을 입력 폴더에 업로드합니다.

샘플 문서 업로드(콘솔)

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷에서, 버킷 목록 중 해당 버킷을 선택합니다.
3. `input` 폴더를 선택한 다음 업로드를 선택합니다.
4. 파일 추가를 선택한 다음 컴퓨터에 있는 `amazon-reviews.csv` 파일을 선택합니다.
5. 기타 설정은 기본값을 유지합니다.
6. 업로드를 선택합니다.

샘플 문서를 버킷에 업로드(AWS CLI)

S3 버킷에 입력 폴더를 생성하고 `cp` 명령을 사용하여 데이터세트 파일을 새 폴더에 업로드합니다.

샘플 문서 업로드(AWS CLI)

1. 버킷의 새 폴더에 `amazon-reviews.csv` 파일을 업로드하려면 다음 AWS CLI 명령을 실행합니다. amzn-s3-demo-bucket을 버킷 이름으로 바꿉니다. 끝에 `/input/` 경로를 추가하면 Amazon S3는 버킷에 `input`이라는 새 폴더를 자동으로 생성하고 데이터세트 파일을 해당 폴더에 업로드합니다.

   ```
   aws s3 cp amazon-reviews.csv s3://amzn-s3-demo-bucket/input/
   ```

2. 파일이 성공적으로 업로드되었는지 확인하려면 다음 명령을 실행합니다. 명령어는 버킷 `input` 폴더의 콘텐츠를 나열합니다.

   ```
   aws s3 ls s3://amzn-s3-demo-bucket/input/
   ```

이제 `input`라는 폴더에 `amazon-reviews.csv` 파일이 있는 S3 버킷이 생겼습니다. 콘솔을 사용한 경우 버킷에도 `output` 폴더가 있습니다. 를 사용한 경우 Amazon Comprehend 분석 작업을 실행할 때 출력 폴더를 AWS CLI생성합니다.

2단계: (CLI만 해당) Amazon Comprehend에 대한 IAM 역할 생성

이 단계는 AWS Command Line Interface (AWS CLI)를 사용하여이 자습서를 완료하는 경우에만 필요합니다. Amazon Comprehend 콘솔을 사용하여 분석 작업을 실행하는 경우 3단계: Amazon S3의 문서에 대한 분석 작업 실행으로 건너뛰십시오.

분석 작업을 실행하려면 Amazon Comprehend이 샘플 데이터 세트가 들어 있고 작업 출력을 포함할 Amazon S3 버킷에 액세스할 수 있어야 합니다. IAM 역할을 사용하면 AWS 서비스 또는 사용자의 권한을 제어할 수 있습니다. 이 단계에서는 Amazon Comprehend를 위한 IAM 역할을 생성합니다. 그런 다음 Amazon Comprehend이 S3 버킷에 액세스할 수 있는 권한을 부여하는 리소스 기반 정책을 생성하여 이 역할에 연결합니다. 이 단계가 끝나면 Amazon Comprehend는 입력 데이터에 액세스하고, 출력을 저장하고, 감성 및 개체 분석 작업을 실행하는 데 필요한 권한을 갖게 됩니다.

Amazon Comprehend에서 IAM을 사용하는 방법에 대한 자세한 내용은 Amazon Comprehend에서 IAM을 사용하는 방법을 참조하세요.

주제

- 사전 조건
- IAM 역할 생성
- IAM 정책을 IAM 역할에 연결합니다.

사전 조건

시작하기 전에 다음을 수행하십시오.

- 1단계: Amazon S3에 문서 추가을 완료합니다.
- JSON 정책을 저장하고 Amazon 리소스 이름(ARN)을 추적할 수 있는 코드 또는 텍스트 편집기가 있어야 합니다.

IAM 역할 생성

Amazon Simple Storage Service(Amazon S3) 버킷에 액세스하려면 Amazon Comprehend가 AWS Identity and Access Management (IAM) 역할을 맡아야 합니다. IAM 역할은 Amazon Comprehend가 신뢰할 수 있는 개체임을 선언합니다. Amazon Comprehend가 역할을 수임해 신뢰할 수 있는 개체가 된 후에 Amazon Comprehend에 버킷 액세스 권한을 부여할 수 있습니다. 이 단계에서 Amazon Comprehend를 신뢰할 수 있는 개체로 레이블을 지정하는 역할을 생성합니다. AWS CLI 또는 Amazon Comprehend 콘솔을 사용하여 역할을 생성할 수 있습니다. 콘솔을 사용하려면 3단계: Amazon S3의 문서에 대한 분석 작업 실행로 건너뛰십시오.

Amazon Comprehend 콘솔에서 역할 이름에 'Comprehend'가 포함되고 신뢰 정책에 comprehend.amazonaws.com이 포함된 역할을 선택할 수 있습니다. 콘솔에 해당 역할을 표시하려면 CLI에서 생성한 역할을 이러한 기준을 충족하도록 구성합니다.

Amazon Comprehend에 대한 IAM 역할을 생성하려면(AWS CLI)

1. 다음 신뢰 정책을 컴퓨터의 코드 또는 텍스트 편집기에서 `comprehend-trust-policy.json`이라는 JSON 문서로 저장합니다. 이 신뢰 정책은 Amazon Comprehend를 신뢰할 수 있는 개체로 선언하고 Amazon Comprehend가 IAM 역할을 수임하는 것을 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "comprehend.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. IAM 역할을 생성하려면 다음 AWS CLI 명령을 실행합니다. 이 명령은 `AmazonComprehendServiceRole-access-role`이라는 IAM 역할을 생성하고 신뢰 정책을 이 역할에 연결합니다. *`path/`*를 로컬 컴퓨터의 JSON 문서 경로로 변경합니다.

```
aws iam create-role --role-name AmazonComprehendServiceRole-access-role
--assume-role-policy-document file://path/comprehend-trust-policy.json
```

> **Tip**
>
> 파라미터 구문 분석 오류 메시지가 표시되면 JSON 신뢰 정책 파일의 경로가 잘못된 것일 수 있습니다. 홈 디렉터리를 기반으로 파일의 상대 경로를 제공하십시오.

3. Amazon 리소스 이름(ARN)을 복사하여 텍스트 편집기에서 저장합니다. ARN에는 *`arn:aws:iam::123456789012:role/AmazonComprehendServiceRole-access-role`*과 비슷한 형식이 있습니다. Amazon Comprehend 분석 작업을 실행하려면 이 ARN이 필요합니다.

IAM 정책을 IAM 역할에 연결합니다.

Amazon S3 버킷에 액세스하려면 Amazon Comprehend가 나열, 읽기 및 쓰기를 할 수 있는 권한이 필요합니다. Amazon Comprehend에 필요한 권한을 부여하려면 IAM 정책을 생성하여 IAM 역할에 연결

합니다. IAM 정책은 Amazon Comprehend가 버킷에서 입력 데이터를 검색하고 분석 결과를 버킷에 기록하도록 허용합니다. 정책을 생성한 후 이 정책을 IAM 역할에 연결합니다.

IAM 정책을 생성하려면(AWS CLI)

1. 다음 정책을 `comprehend-access-policy.json`이라는 JSON 문서로 로컬에 저장합니다. 그러면 Amazon Comprehend에 지정된 S3 버킷에 액세스할 수 있는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

2. S3 버킷 액세스 정책을 생성하려면 다음 AWS CLI 명령을 실행합니다. *path/*를 로컬 컴퓨터의 JSON 문서 경로로 변경합니다.

```
aws iam create-policy --policy-name comprehend-access-policy
--policy-document file://path/comprehend-access-policy.json
```

3. 액세스 정책 ARN을 복사하여 텍스트 편집기에 저장합니다. ARN에는 *arn:aws:iam::123456789012:policy/comprehend-access-policy*과 비슷한 형식이 있습니다. 액세스 정책을 IAM 역할에 연결하려면 이 ARN이 필요합니다.

IAM 역할에 IAM 정책을 연결하려면(AWS CLI)

- 다음 명령을 실행합니다. *policy-arn*을 이전 단계에서 복사한 액세스 정책 ARN으로 변경합니다.

```
aws iam attach-role-policy --policy-arn policy-arn
--role-name AmazonComprehendServiceRole-access-role
```

이제 Amazon Comprehend에 대한 신뢰 정책과 Amazon Comprehend가 S3 버킷에 액세스할 수 있는 권한을 부여하는 액세스 정책을 포함하는 AmazonComprehendServiceRole-access-role이라는 IAM 역할을 갖게 되었습니다. IAM 역할에 대한 ARN을 텍스트 편집기에 복사했습니다.

3단계: Amazon S3의 문서에 대한 분석 작업 실행

데이터를 Amazon S3에 저장한 후 Amazon Comprehend 분석 작업을 실행할 수 있습니다. 감성 분석 작업은 문서의 전반적인 분위기(긍정적, 부정적, 중립적 또는 혼합)를 결정합니다. 개체 분석 작업은 문서에서 실제 객체의 이름을 추출합니다. 이러한 객체에는 사람, 장소, 제목, 이벤트, 날짜, 수량, 제품 및 조직이 포함됩니다. 이 단계에서는 2개의 Amazon Comprehend 분석 작업을 실행하여 샘플 데이터 세트에서 감성 및 개체를 추출합니다.

주제

- 사전 조건
- 감성과 개체를 분석합니다.

사전 조건

시작하기 전에 다음을 수행하십시오.

- 1단계: Amazon S3에 문서 추가을 완료합니다.

- (선택 사항)를 사용하는 경우 2단계: (CLI만 해당) Amazon Comprehend에 대한 IAM 역할 생성를 AWS CLI완료하고 IAM 역할 ARN을 준비합니다.

감성과 개체를 분석합니다.

첫 번째로 실행하는 작업은 샘플 데이터 세트에 있는 각 고객 리뷰의 감성을 분석합니다. 두 번째 작업은 각 고객 리뷰의 개체를 추출합니다. Amazon Comprehend 분석 작업은 Amazon Comprehend 콘솔 또는 AWS CLI를 사용하여 수행할 수 있습니다.

> **Tip**
>
> Amazon Comprehend를 지원하는 AWS 리전에 있는지 확인합니다. 자세한 내용은 글로벌 인프라 가이드의 리전 테이블을 참조하세요.

감성 및 개체 분석(콘솔)

Amazon Comprehend 콘솔 사용 시, 한 번에 하나의 작업을 생성합니다. 감성과 개체 분석 작업을 모두 실행하려면 다음 단계를 반복해야 합니다. 첫 번째 작업을 위해서는 IAM 역할을 생성하지만 두 번째 작업에서는 첫 번째 작업의 IAM 역할을 재사용할 수 있습니다. 동일한 S3 버킷과 폴더를 사용하는 한 IAM 역할을 재사용할 수 있습니다.

감성 및 개체 분석 작업을 실행하려면(콘솔)

1. Amazon Simple Storage Service(S3) 버킷을 생성한 리전과 동일한 리전에 있는지 확인합니다. 다른 리전에 있는 경우 탐색 모음에서 AWS 리전 선택기에서 S3 버킷을 생성한 리전을 선택합니다.
2. https://console.aws.amazon.com/comprehend/에서 Amazon Comprehend 콘솔을 엽니다.
3. Amazon Comprehend 시작을 선택합니다.
4. 탐색 창에서 분석 작업을 선택합니다.
5. 작업 생성을 선택합니다.
6. 작업 설정 섹션에서 다음을 수행합니다.
   a. 이름에 `reviews-sentiment-analysis`를 입력합니다.
   b. 분석 유형에서 감성을 선택합니다.
   c. 언어에서 영어를 선택합니다.
   d. 작업 암호화 설정은 비활성 상태를 유지합니다.

7. 입력 데이터 섹션에서 다음을 수행합니다.

   a. 데이터 소스로 내 문서를 선택합니다.

   b. S3 위치로 S3 찾아보기를 선택한 다음 버킷 목록에서 버킷을 선택합니다.

   c. S3 버킷에서 객체로 `input` 폴더를 선택합니다.

   d. `input` 폴더에서 샘플 데이터세트 `amazon-reviews.csv`를 선택한 다음 선택하기를 선택합니다.

   e. 입력 형식으로 라인 하나에 문서 하나 선택합니다.

8. 출력 데이터 섹션에서 다음을 수행합니다.

   a. S3 위치로 S3 찾아보기를 선택한 다음 버킷 목록에서 버킷을 선택합니다.

   b. S3 버킷에서 객체로 `output` 폴더를 선택한 다음 선택하기를 선택합니다.

   c. 암호화는 꺼진 상태로 둡니다.

9. 권한 연결 섹션에서 다음을 수행합니다.

   a. IAM 역할에서 IAM 역할 생성을 선택합니다.

   b. 액세스 권한은 입력 및 출력 S3 버킷을 선택합니다.

   c. 이름 접미사에 `comprehend-access-role`을 입력합니다. 이 역할은 Amazon S3 버킷에 대한 액세스를 제공합니다.

10. 작업 생성을 선택합니다.

11. 1~10단계를 반복하여 개체 분석 작업을 생성합니다. 다음과 같이 변경합니다.

    a. 작업 설정에서 이름에 `reviews-entities-analysis`를 입력합니다.

    b. 작업 설정에서 분석 유형으로 개체를 선택합니다.

    c. 액세스 권한에서 기존 IAM 역할 사용을 선택합니다. 역할 이름으로 `AmazonComprehendServiceRole-comprehend-access-role`을 선택합니다(감성 작업에 대해 생성한 역할과 동일).

감성 및 개체 분석(AWS CLI)

`start-sentiment-detection-job` 및 `start-entities-detection-job` 명령을 사용하여 감성 및 개체 분석 작업을 실행합니다. 각 명령을 실행한 후 에는 출력 S3 위치를 포함하여 작업에 대한 세부 정보에 액세스할 수 있는 `JobId` 값이 있는 JSON 객체가 AWS CLI 표시됩니다.

감정 및 개체 분석 작업을 실행하려면(AWS CLI)

1. AWS CLI에서 다음 명령을 실행하여 감성 분석 작업을 시작합니다. *arn:aws:iam::123456789012:role/comprehend-access-role*을 이전에 텍스트 편집기에 복사한 IAM 역할 ARN으로 바꿉니다. 기본 AWS CLI 리전이 Amazon S3 버킷을 생성한 리전과 다른 경우 `--region` 파라미터를 포함하고를 버킷이 있는 리전*us-east-1*으로 바꿉니다.

```
aws comprehend start-sentiment-detection-job
--input-data-config S3Uri=s3://amzn-s3-demo-bucket/input/
--output-data-config S3Uri=s3://amzn-s3-demo-bucket/output/
--data-access-role-arn arn:aws:iam::123456789012:role/comprehend-access-role
--job-name reviews-sentiment-analysis
--language-code en
[--region us-east-1]
```

2. 작업을 제출한 후 `JobId`를 복사하여 텍스트 편집기에 저장합니다. 분석 작업에서 출력 파일을 찾으려면 `JobId`가 필요합니다.
3. 다음 명령을 실행하여 개체 분석 작업을 시작합니다.

```
aws comprehend start-entities-detection-job
--input-data-config S3Uri=s3://amzn-s3-demo-bucket/input/
--output-data-config S3Uri=s3://amzn-s3-demo-bucket/output/
--data-access-role-arn arn:aws:iam::123456789012:role/comprehend-access-role
--job-name reviews-entities-analysis
--language-code en
[--region us-east-1]
```

4. 작업을 제출한 후 `JobId`를 복사하여 텍스트 편집기에 저장합니다.
5. 작업의 상태를 확인합니다. `JobId`를 추적하여 작업 진행률을 볼 수 있습니다.

   감성 분석 작업 진행률을 추적하려면 다음 명령을 실행합니다. *sentiment-job-id*를 감성 분석을 실행한 후 복사한 `JobId`로 바꿉니다.

```
aws comprehend describe-sentiment-detection-job
--job-id sentiment-job-id
```

   개체 분석 작업을 추적하려면 다음 명령을 실행합니다. *entities-job-id*를 개체 분석을 실행한 후 복사한 `JobId`로 바꿉니다.

```
aws comprehend describe-entities-detection-job
```

```
--job-id entities-job-id
```

`JobStatus`가 `COMPLETED`으로 표시되는 데 몇 분 정도 걸립니다.

감성 및 개체 분석 작업을 완료했습니다. 다음 단계로 이동하기 전에 두 작업을 모두 완료해야 합니다. 작업을 완료하는 데 몇 분 정도 걸릴 수 있습니다.

4단계: 데이터 시각화를 위한 Amazon Comprehend 출력 준비

데이터 시각화를 생성하기 위한 감성 및 개체 분석 작업의 결과를 준비하려면 AWS Glue 및 Amazon Athena를 사용합니다. 이 단계에서는 Amazon Comprehend 결과 파일을 추출합니다. 그런 다음 데이터를 탐색하고 AWS Glue Data Catalog의 테이블에 자동으로 카탈로그화하는 AWS Glue 크롤러를 생성합니다. 그런 다음 서버리스 및 대화형 쿼리 서비스 Amazon Athena인를 사용하여 이러한 테이블에 액세스하고 변환합니다. 이 단계를 완료하면 Amazon Comprehend 결과가 정리되어 시각화할 준비가 된 것입니다.

PII 개체 감지 작업의 경우 출력 파일은 압축된 아카이브가 아닌 일반 텍스트입니다. 출력 파일 이름은 입력 파일과 동일하며 끝에 `.out`이 추가됩니다. 출력 파일의 압축을 푸는 단계가 필요없습니다. [로 데이터 로드로 건너뜁니다 AWS Glue Data Catalog](#).

주제

- [사전 조건](#)
- [출력 다운로드](#)
- [출력 파일 추출](#)
- [추출한 파일 업로드](#)
- [AWS Glue Data Catalog에 데이터를 로드합니다.](#)
- [분석을 위한 데이터 준비](#)

사전 조건

시작하기 전에 [3단계: Amazon S3의 문서에 대한 분석 작업 실행](#)를 완료합니다.

출력 다운로드

Amazon Comprehend는 Gzip 압축을 사용하여 출력 파일을 압축하고 이를 tar 아카이브로 저장합니다. 출력 파일의 압축을 푸는 가장 간단한 방법은 `output.tar.gz` 아카이브를 로컬로 다운로드하는 것입니다.

이 단계에서는 감성 및 개체 출력 아카이브를 다운로드합니다.

출력 파일 다운로드(콘솔)

각 작업의 출력 파일을 찾으려면 Amazon Comprehend 콘솔의 분석 작업으로 돌아갑니다. 분석 작업은 출력 파일을 다운로드할 수 있는 출력을 위한 S3 위치를 제공합니다.

출력 파일을 다운로드하려면(콘솔)

1. Amazon Comprehend 콘솔의 탐색 창에서 분석 작업으로 돌아갑니다.
2. 감성 분석 작업 `reviews-sentiment-analysis`를 선택합니다.
3. 출력에서 출력 데이터 위치 옆에 표시된 링크를 선택합니다. 그러면 S3 버킷의 `output.tar.gz` 아카이브로 리디렉션됩니다.
4. 개요 탭에서 다운로드를 선택합니다.
5. 컴퓨터에서 아카이브 이름을 `sentiment-output.tar.gz`로 바꿉니다. 모든 출력 파일의 이름이 동일하므로 감성 및 개체 파일을 추적하는 데 도움이 됩니다.
6. 1~4단계를 반복하여 `reviews-entities-analysis` 작업의 결과를 찾아 다운로드합니다. 컴퓨터에서 아카이브 이름을 `entities-output.tar.gz`로 바꿉니다.

출력 파일 다운로드(AWS CLI)

각 작업의 출력 파일을 찾으려면 분석 작업의 `JobId`를 사용하여 출력의 S3 위치를 찾습니다. 그런 다음 `cp` 명령을 사용하여 출력 파일을 컴퓨터에 다운로드합니다.

출력 파일을 다운로드하려면(AWS CLI)

1. 감성 분석 작업에 대한 세부 정보를 나열하려면 다음 명령을 실행합니다. *`sentiment-job-id`*를 저장한 감성 `JobId`로 바꿉니다.

   ```
   aws comprehend describe-sentiment-detection-job --job-id sentiment-job-id
   ```

   `JobId`를 추적하지 못한 경우 다음 명령을 실행하여 감성 작업을 모두 나열하고 작업을 이름별로 필터링할 수 있습니다.

   ```
   aws comprehend list-sentiment-detection-jobs
   --filter JobName="reviews-sentiment-analysis"
   ```

2. OutputDataConfig 객체에서 S3Uri 값을 찾습니다. S3Uri 값은 다음 형식: *s3://amzn-s3-demo-bucket/.../output/output.tar.gz*과 유사해야 합니다. 이 값을 텍스트 편집기에 복사합니다.
3. 감성 출력 아카이브를 로컬 디렉터리에 다운로드하려면 다음 명령을 실행합니다. S3 버킷 경로를 이전 단계에서 복사한 S3Uri로 교체합니다. *path/*를 로컬 디렉터리의 폴더 경로로 바꿉니다. 원래 아카이브 이름을 sentiment-output.tar.gz라는 이름으로 대체해 감성 및 개체 파일을 추적하는 데 도움이 됩니다.

```
aws s3 cp s3://amzn-s3-demo-bucket/.../output/output.tar.gz
path/sentiment-output.tar.gz
```

4. 개체 분석 작업에 대한 세부 정보를 나열하려면 다음 명령을 실행합니다.

```
aws comprehend describe-entities-detection-job
--job-id entities-job-id
```

JobId를 모르면 다음 명령을 실행하여 개체 작업을 모두 나열하고 작업을 이름별로 필터링합니다.

```
aws comprehend list-entities-detection-jobs
--filter JobName="reviews-entities-analysis"
```

5. 개체 작업 설명에 있는 OutputDataConfig 객체에서 S3Uri 값을 복사합니다.
6. 개체 출력 아카이브를 로컬 디렉터리에 다운로드하려면 다음 명령을 실행합니다. S3 버킷 경로를 이전 단계에서 복사한 S3Uri로 교체합니다. *path/*를 로컬 디렉터리의 폴더 경로로 바꿉니다. 원래 아카이브 이름을 entities-output.tar.gz라는 이름으로 대체합니다.

```
aws s3 cp s3://amzn-s3-demo-bucket/.../output/output.tar.gz
path/entities-output.tar.gz
```

출력 파일 추출

Amazon Comprehend 결과에 액세스하려면 먼저 감성 및 개체 아카이브의 압축을 풀어야 합니다. 로컬 파일 시스템이나 터미널을 사용하여 아카이브의 압축을 풀 수 있습니다.

출력 파일 추출(GUI 파일 시스템)

macOS를 사용하는 경우 GUI 파일 시스템에서 아카이브를 두 번 클릭하여 아카이브에서 출력 파일 추출합니다.

Windows를 사용하는 경우 7-Zip과 같은 타사 도구를 사용하여 GUI 파일 시스템에서 출력 파일을 추출합니다. Windows에서는 아카이브의 출력 파일에 액세스하기 위해 두 단계를 수행해야 합니다. 먼저 아카이브를 압축 해제한 다음 아카이브를 추출합니다.

결과 파일을 구분할 수 있도록 감성 파일의 이름을 `sentiment-output`으로 바꾸고 개체 파일의 이름을 `entities-output`으로 바꿉니다.

출력 파일 추출(터미널)

Linux 또는 macOS를 사용하는 경우 표준 터미널을 사용할 수 있습니다. Windows를 사용하는 경우 tar 명령을 실행하려면 Cygwin과 같은 유닉스 스타일 환경에 액세스할 수 있어야 합니다.

감성 아카이브에서 감성 출력 파일을 추출하려면 로컬 터미널에서 다음 명령을 실행합니다.

```
tar -xvf sentiment-output.tar.gz --transform 's,^,sentiment-,'
```

참고로 `--transform` 파라미터는 아카이브 내 출력 파일에 접두사 `sentiment-`을 추가하고 파일 이름을 `sentiment-output`으로 변경합니다. 이렇게 하면 감성과 개체 출력 파일을 구분하고 덮어쓰기를 방지할 수 있습니다.

개체 아카이브에서 개체 출력 파일을 추출하려면 로컬 터미널에서 다음 명령을 실행합니다.

```
tar -xvf entities-output.tar.gz --transform 's,^,entities-,'
```

`--transform` 파라미터는 출력 파일 이름에 접두사 `entities-`를 추가합니다.

> **Tip**
>
> Amazon S3의 스토리지 비용을 절약하기 위해 파일을 업로드하기 전에 Gzip으로 다시 압축할 수 있습니다. 는 tar 아카이브에서 데이터를 자동으로 읽을 AWS Glue 수 없으므로 원본 아카이브의 압축을 풀고 압축을 해제하는 것이 중요합니다. 그러나는 Gzip 형식의 파일에서 읽을 AWS Glue 수 있습니다.

추출한 파일 업로드

파일을 추출한 후 버킷에 업로드합니다. 가 데이터를 제대로 AWS Glue 읽으려면 감성 및 개체 출력 파일을 별도의 폴더에 저장해야 합니다. 버킷에 추출된 감성 결과를 위한 폴더와 추출된 개체 결과를 위한 두 번째 폴더를 생성합니다. Amazon S3 콘솔 또는 AWS CLI에서 폴더를 생성할 수 있습니다.

추출한 파일을 Amazon S3에 업로드(콘솔)

S3 버킷에 추출된 감성 결과 파일을 위한 폴더 하나와 추출된 개체 결과 파일를 위한 두 번째 폴더 하나를 생성합니다. 그런 다음 추출된 결과 파일을 해당 폴더에 업로드합니다.

추출한 파일을 Amazon S3에 업로드하려면(콘솔)

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷에서 버킷을 선택한 다음 폴더 생성을 선택합니다.
3. 새 폴더 이름에 `sentiment-results`를 입력하고 저장을 선택합니다. 이 폴더에는 추출된 감성 출력 파일이 포함됩니다.
4. 버킷의 개요 탭에 있는 버킷 콘텐츠 목록에서 새 폴더 `sentiment-results`를 선택합니다. 업로드를 선택합니다.
5. 파일 추가를 선택하고 로컬 컴퓨터에서 `sentiment-output` 파일을 선택한 후 다음을 선택합니다.
6. 사용자 관리, 다른에 대한 액세스 AWS 계정 및 퍼블릭 권한 관리 옵션을 기본값으로 둡니다. Next(다음)를 선택합니다.
7. 스토리지 등급에서 표준을 선택합니다. 암호화, 메타데이터 및 태그 옵션을 기본값으로 유지합니다. Next(다음)를 선택합니다.
8. 업로드 옵션을 검토한 다음 업로드를 선택합니다.
9. 1~8단계를 반복해 `entities-results`라고 하는 폴더를 만들고 `entities-output` 파일을 이 폴더에 업로드합니다.

추출한 파일 Amazon S3에 업로드하기(AWS CLI)

cp 명령으로 파일을 업로드하는 동안 S3 버킷에 폴더를 생성할 수 있습니다.

추출한 파일을 Amazon S3에 업로드하려면(AWS CLI)

1. 다음 명령을 실행하여 감성 폴더를 생성하고 감성 파일을 이 폴더에 업로드합니다. *path/*를 추출된 감성 출력 파일의 로컬 경로로 바꿉니다.

```
aws s3 cp path/sentiment-output s3://amzn-s3-demo-bucket/sentiment-results/
```

2. 다음 명령을 실행하여 개체 폴더를 생성하고 개체 파일을 이 폴더에 업로드합니다. *path/*를 추출된 개체 출력 파일의 로컬 경로로 바꿉니다.

```
aws s3 cp path/entities-output s3://amzn-s3-demo-bucket/entities-results/
```

AWS Glue Data Catalog에 데이터를 로드합니다.

결과를 데이터베이스로 가져오려면 AWS Glue 어크롤러를 사용할 수 있습니다. 어 AWS Glue 크롤러는 파일을 스캔하고 데이터의 스키마를 검색합니다. 그런 다음 AWS Glue Data Catalog (서버리스 데이터베이스)의 테이블에 있는 데이터를 정렬합니다. AWS Glue 콘솔 또는를 사용하여 크롤러를 생성할 수 있습니다 AWS CLI.

데이터를 AWS Glue Data Catalog 에 로드하기(콘솔)

`sentiment-results` 및 `entities-results` 폴더를 별도로 스캔하는 AWS Glue 크롤러를 생성합니다. AWS Glue 의 IAM 역할이 S3 버킷에 액세스할 수 있는 크롤러 권한을 부여합니다. 이 IAM 역할은 크롤러를 설정할 때 생성합니다.

데이터를에 로드하려면 AWS Glue Data Catalog (콘솔)

1. 가 지원하는 리전에 있는지 확인합니다 AWS Glue. 다른 리전에 있는 경우, 탐색 모음의 리전 선택기에서 지원되는 리전을 선택합니다. 가 지원하는 리전 목록은 글로벌 인프라 안내서의 리전 테이블을 AWS Glue참조하세요.
2. https://console.aws.amazon.com/glue/ AWS Glue 콘솔을 엽니다.
3. 탐색 창에서 크롤러를 선택한 후 크롤러 추가를 선택합니다.
4. 크롤러 이름에 `comprehend-analysis-crawler`를 입력한 후 다음을 선택합니다.
5. 크롤러 소스 유형에 데이터 스토어를 선택한 후 다음을 선택합니다.
6. 데이터 스토어 페이지 추가에 대해 다음을 수행합니다.

    a. 데이터 스토어 선택에서 S3을 선택합니다.

    b. 연결은 비워 둡니다.

    c. 데이터 크롤링에 내 계정에 지정된 경로를 선택합니다.

    d. 포함 경로에 감성 출력 폴더의 전체 S3 경로를 입력합니다: `s3://amzn-s3-demo-bucket/sentiment-results`.

e. Next(다음)를 선택합니다.

7. 다른 데이터 스토어 추가에서 예를 선택한 다음 다음을 선택합니다. 6단계를 반복하되 개체 출력 폴더의 전체 S3 경로를 입력합니다: `s3://amzn-s3-demo-bucket/entities-results`.
8. 다른 데이터 스토어 추가에서 아니요를 선택한 다음 다음을 선택합니다.
9. IAM 역할 선택에서 다음을 수행합니다.

   a. IAM 역할 생성을 선택합니다.

   b. IAM 역할에 `glue-access-role`을 입력한 후 다음을 선택합니다.

10. 이 크롤러의 일정 생성에 온디맨드로 실행을 선택한 다음 다음을 선택합니다.
11. 크롤러 출력 구성에서 다음을 수행합니다.

    a. 데이터베이스에 데이터베이스 추가를 선택합니다.

    b. 데이터베이스 이름에 `comprehend-results`를 입력합니다. 이 데이터베이스는 Amazon Comprehend 출력 테이블을 저장합니다.

    c. 다른 옵션은 기본 설정으로 두고 다음을 선택합니다.

12. 크롤러 정보를 검토한 후 마침을 선택합니다.
13. Glue 콘솔의 크롤러에서 `comprehend-analysis-crawler`를 선택하고 크롤러 실행을 선택합니다. 크롤러를 완료하는 데 몇 분 정도 걸릴 수 있습니다.

데이터를 AWS Glue Data Catalog 에 로드하기(AWS CLI)

S3 버킷에 액세스할 수 AWS Glue 있는 권한을 제공하는에 대한 IAM 역할을 생성합니다. 그런 다음 AWS Glue Data Catalog에서 데이터베이스를 생성합니다. 마지막으로 데이터베이스의 테이블에 데이터를 로드하는 크롤러를 만들고 실행합니다.

데이터를에 로드하려면 AWS Glue Data Catalog (AWS CLI)

1. 에 대한 IAM 역할을 생성하려면 다음을 AWS Glue수행합니다.

   a. 다음 신뢰 정책을 `glue-trust-policy.json`이라는 JSON 문서로 컴퓨터에 저장합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "glue.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. IAM 역할을 생성하려면 다음 명령을 실행합니다. *path/*를 로컬 컴퓨터의 JSON 문서 경로로 변경합니다.

```
aws iam create-role --role-name glue-access-role
--assume-role-policy-document file://path/glue-trust-policy.json
```

c. 에 새 역할에 대한 Amazon 리소스 번호(ARN)가 AWS CLI 나열되면 복사하여 텍스트 편집기에 저장합니다.

d. 다음 IAM 정책을 `glue-access-policy.json`이라는 JSON 문서로 컴퓨터에 저장합니다. 이 정책은 결과 폴더를 크롤링할 수 있는 AWS Glue 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/sentiment-results*",
                "arn:aws:s3:::amzn-s3-demo-bucket/entities-results*"
            ]
        }
    ]
}
```

e. IAM 정책을 생성하려면 다음 명령을 실행합니다. *path/*를 로컬 컴퓨터의 JSON 문서 경로로 변경합니다.

```
aws iam create-policy --policy-name glue-access-policy
--policy-document file://path/glue-access-policy.json
```

f. 에 액세스 정책의 ARN이 AWS CLI 나열되면 복사하여 텍스트 편집기에 저장합니다.

g. 다음 명령을 실행하여 새 정책을 IAM 역할에 연결합니다. *policy-arn*을 이전 단계에서 복사한 IAM 정책 ARN으로 변경합니다.

```
aws iam attach-role-policy --policy-arn policy-arn
--role-name glue-access-role
```

h. 다음 명령을 실행AWSGlueServiceRole하여 AWS 관리형 정책을 IAM 역할에 연결합니다.

```
aws iam attach-role-policy --policy-arn
arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
--role-name glue-access-role
```

2. 다음 명령을 실행하여 AWS Glue 데이터베이스를 생성합니다.

```
aws glue create-database
--database-input Name="comprehend-results"
```

3. 다음 명령을 실행하여 새 AWS Glue 크롤러를 생성합니다. 를 AWS Glue IAM 역할의 ARN*glue-iam-role-arn*으로 바꿉니다.

```
aws glue create-crawler
--name comprehend-analysis-crawler
--role glue-iam-role-arn
--targets S3Targets=[
{Path="s3://amzn-s3-demo-bucket/sentiment-results"},
{Path="s3://amzn-s3-demo-bucket/entities-results"}]
--database-name comprehend-results
```

4. 다음 명령을 실행하여 크롤러를 시작합니다.

```
aws glue start-crawler --name comprehend-analysis-crawler
```

크롤러를 완료하는 데 몇 분 정도 걸릴 수 있습니다.

분석을 위한 데이터 준비

이제 데이터베이스가 Amazon Comprehend 결과로 채워졌습니다. 하지만 결과는 중첩됩니다. 중첩을 해제하려면에서 몇 가지 SQL 문을 실행합니다 Amazon Athena. Amazon Athena 는 표준 SQL을 사용하여 Amazon S3의 데이터를 쉽게 분석할 수 있는 대화형 쿼리 서비스입니다. Athena는 서버리스 서비스이므로 관리할 인프라가 없으며 쿼리당 지불 요금 모델이 있습니다. 이 단계에서는 분석과 시각화

에 사용할 수 있는 정리된 데이터로 구성된 새 테이블을 생성합니다. Athena 콘솔을 사용하여 데이터를 준비합니다.

데이터를 준비하려면

1. https://console.aws.amazon.com/athena/에서 Athena 콘솔을 엽니다.
2. 쿼리 편집기에서 설정을 선택한 다음 관리를 선택합니다.
3. 쿼리 결과 위치에 `s3://amzn-s3-demo-bucket/query-results/`를 입력합니다. 이렇게 하면 실행 중인 Amazon Athena 쿼리의 출력을 저장하는 라는 새 폴더가 `query-results` 버킷에 생성됩니다. 저장(Save)을 선택합니다.
4. 쿼리 편집기에서 편집기를 선택합니다.
5. 데이터베이스에서 `comprehend-results` 생성한 AWS Glue 데이터베이스를 선택합니다.
6. 테이블 섹션에 `sentiment_results` 및 `entities_results`라는 두 개의 테이블이 있어야 합니다. 테이블 미리 보기로 크롤러가 데이터를 로드했는지 확인합니다. 각 테이블의 옵션(테이블 이름 옆에 있는 세 개의 점)에서 테이블 미리 보기를 선택합니다. 짧은 쿼리가 자동으로 실행됩니다. 결과 창에서 테이블에 데이터가 포함되어 있는지 확인합니다.

   > **Tip**
   >
   > 테이블에 데이터가 없는 경우 S3 버킷의 폴더를 확인합니다. 개체 결과 폴더와 감성 결과 폴더가 하나씩 있는지 확인합니다. 그런 다음 새 AWS Glue 크롤러를 실행해 봅니다.

7. `sentiment_results` 테이블의 중첩을 해제하려면 쿼리 편집기에서 다음 쿼리를 입력한 다음 실행을 선택합니다.

```
CREATE TABLE sentiment_results_final AS
SELECT file, line, sentiment,
sentimentscore.mixed AS mixed,
sentimentscore.negative AS negative,
sentimentscore.neutral AS neutral,
sentimentscore.positive AS positive
FROM sentiment_results
```

8. 개체 테이블의 중첩을 해제하려면 쿼리 편집기에서 다음 쿼리를 입력한 다음 실행을 선택합니다.

```
CREATE TABLE entities_results_1 AS
SELECT file, line, nested FROM entities_results
CROSS JOIN UNNEST(entities) as t(nested)
```

9. 개체 테이블의 중첩 해제를 완료하려면 쿼리 편집기에서 다음 쿼리를 입력한 다음 쿼리 실행을 선택합니다.

```
CREATE TABLE entities_results_final AS
SELECT file, line,
nested.beginoffset AS beginoffset,
nested.endoffset AS endoffset,
nested.score AS score,
nested.text AS entity,
nested.type AS category
FROM entities_results_1
```

`sentiment_results_final` 테이블은 파일, 라인, 감성, 혼합, 부정, 중립, 긍정이라는 열을 포함하는 다음과 같은 모양이어야 합니다. 테이블에는 셀당 하나의 값이 있어야 합니다. 감성 열에는 특정 리뷰의 가장 가능성이 높은 전반적인 감성이 설명되어 있습니다. 혼합, 부정, 중립, 긍정 열은 각 감성 유형에 대한 점수를 제공합니다.

Results

	file	line	sentiment	mixed	negative	neutral	positive
1	amazon-reviews.csv	6	MIXED	0.9862896203994751	0.0015502438182011247	1.6660270921420306E-4	0.0119935879483
2	amazon-reviews.csv	8	POSITIVE	0.0012987082591280341	0.01186690479516983	0.174478679895401	0.8123556375503
3	amazon-reviews.csv	11	POSITIVE	6.5368581090297084E-6	0.0013866390800103545	0.007405391428619623	0.9912014007568
4	amazon-reviews.csv	13	POSITIVE	4.7155481297522783E-4	0.24615342915058136	0.017713148146867752	0.7356618046760
5	amazon-reviews.csv	14	POSITIVE	1.5821871784282848E-5	0.06828905642032623	0.014075091108679771	0.9176200628280
6	amazon-reviews.csv	16	MIXED	0.9864791035652161	8.548551704734564E-4	1.0789262159960344E-4	0.0125581491738
7	amazon-reviews.csv	20	NEGATIVE	1.1621621524682269E-4	0.9815887212753296	0.004688907880336046	0.0136061981320
8	amazon-reviews.csv	21	POSITIVE	4.663573781726882E-5	0.009533549658954144	0.0015825830632820725	0.9888372421264
9	amazon-reviews.csv	23	POSITIVE	1.7699007003102452E-4	0.40269607305526733	0.0018250439316034317	0.5953019261360
10	amazon-reviews.csv	25	POSITIVE	1.8434448065818287E-6	1.15832663141191E-4	0.0010993879986926913	0.9987829327583

`entities_results_final` 테이블은 파일, 라인, 시작오프셋, 종료오프셋, 점수, 개체, 카테고리라는 열을 포함하는 다음과 같은 모양이어야 합니다. 테이블에는 셀당 하나의 값이 있어야 합니다. 점수 열은 감지한 개체에 대한 Amazon Comprehend의 신뢰도를 나타냅니다. 카테고리는 Comprehend가 감지한 개체의 유형을 나타냅니다.

Results

	file	line	beginoffset	endoffset	score	entity	category
1	amazon-reviews.csv	0	15	22	0.9885989378545348	English	OTHER
2	amazon-reviews.csv	2	24	28	0.9699371997593782	2 me	QUANTITY
3	amazon-reviews.csv	2	94	95	0.6523066984191679	2	QUANTITY
4	amazon-reviews.csv	2	125	126	0.713791396412543	2	QUANTITY
5	amazon-reviews.csv	4	30	36	0.9957169942979278	kindle	COMMERCIAL_ITEM
6	amazon-reviews.csv	5	1	10	0.9979111763962706	Hawthorne	PERSON
7	amazon-reviews.csv	5	135	142	0.5065408081314243	Puritan	OTHER
8	amazon-reviews.csv	5	143	148	0.7702269458801602	Salem	LOCATION
9	amazon-reviews.csv	5	211	229	0.999675563687763	The Scarlet Letter	TITLE
10	amazon-reviews.csv	5	233	236	0.8944631322676461	one	QUANTITY

이제 Amazon Comprehend 결과가 테이블에 로드되었으므로 데이터를 시각화하고 데이터에서 의미 있는 인사이트를 추출할 수 있습니다.

5단계: QuickSight에서 Amazon Comprehend 출력 시각화

Amazon Comprehend 결과를 테이블에 저장한 후 QuickSight를 사용하여에 연결하고 데이터를 시각화할 수 있습니다. QuickSight는 데이터를 시각화하기 위한 AWS 관리형 비즈니스 인텔리전스(BI) 도구입니다. QuickSight를 사용하면 데이터 소스에 쉽게 연결하고 강력한 시각적 객체를 생성할 수 있습니다. 이 단계에서는 QuickSight를 데이터에 연결하고, 데이터에서 인사이트를 추출하는 시각화를 생성하고, 시각화 대시보드를 게시합니다.

주제

- 사전 조건
- QuickSight 액세스 권한 부여
- 데이터 세트 가져오기
- 감성 시각화 생성
- 개체 시각화 생성
- 대시보드 게시
- 정리

사전 조건

시작하기 전에 4단계: 데이터 시각화를 위한 Amazon Comprehend 출력 준비를 완료합니다.

QuickSight 액세스 권한 부여

데이터를 가져오려면 QuickSight에서 Amazon Simple Storage Service(Amazon S3) 버킷 및 Amazon Athena 테이블에 대한 액세스 권한이 필요합니다. QuickSight에 데이터에 대한 액세스 권한을 부여하려면 QuickSight 관리자로 로그인하고 리소스 권한을 편집할 수 있는 액세스 권한이 있어야 합니다. 다음 단계를 완료할 수 없는 경우 개요 페이지 자습서: Amazon Comprehend 고객 리뷰를 통한 인사이트 분석에서 IAM 필수 조건을 검토하십시오.

QuickSight에 데이터에 대한 액세스 권한을 부여하려면

1. QuickSight 콘솔을 엽니다.
2. QuickSight를 처음 사용하는 경우 콘솔에 이메일 주소를 제공하여 새 관리자 사용자를 생성하라는 메시지가 표시됩니다. 이메일 주소에 AWS 계정과 동일한 이메일 주소를 입력합니다. 계속을 선택합니다.
3. 로그인한 후 탐색 모음에서 프로파일 이름을 선택하고 QuickSight 관리를 선택합니다. QuickSight 관리 옵션을 보려면 관리자로 로그인해야 합니다.
4. 보안 및 권한을 선택합니다.
5. AWS 서비스에 대한 QuickSight 액세스에서 추가 또는 제거를 선택합니다.
6. Amazon S3를 선택합니다.
7. Amazon S3 버킷 선택에서 S3 버킷과 Athena 작업 그룹에 대한 쓰기 권한 모두에 사용할 S3 버킷을 선택합니다.
8. 마침을 클릭합니다.
9. 업데이트를 선택합니다.

데이터 세트 가져오기

시각화를 생성하기 전에 QuickSight에 감성 및 개체 데이터 세트를 추가해야 합니다. QuickSight 콘솔을 사용하여이 작업을 수행할 수 있습니다. 중첩되지 않은 감성 및 중첩되지 않은 엔터티 테이블을 가져옵니다 Amazon Athena.

데이터 세트를 가져오려면

1. QuickSight 콘솔을 엽니다.

2. 탐색 모음의 데이터 세트에서 새 데이터 세트를 선택합니다.
3. 데이터 세트 생성에서 Athena를 선택합니다.
4. 데이터 소스 이름에 `reviews-sentiment-analysis`를 입력하고 데이터 소스 생)을 선택합니다.
5. 데이터베이스로 `comprehend-results` 데이터베이스를 선택합니다.
6. 테이블에 감성 테이블 `sentiment_results_final`를 선택한 다음 선택을 선택합니다.
7. 더 빠른 분석을 위해 SPICE로 불러오기를 선택하고 시각화를 선택합니다. SPICE는 QuickSight의 인 메모리 계산 엔진으로, 시각화를 생성할 때 직접 쿼리보다 더 빠른 분석을 제공합니다.
8. QuickSight 콘솔로 돌아가서 데이터 세트를 선택합니다. 1~7단계를 반복하여 개체 데이터 세트를 생성하되 다음과 같이 변경합니다.

   a. 데이터 소스 이름에 `reviews-entities-analysis`를 입력합니다.

   b. 테이블에 개체 테이블 `entities_results_final`를 선택합니다.

감성 시각화 생성

이제 QuickSight에서 데이터에 액세스할 수 있으므로 시각화 생성을 시작할 수 있습니다. Amazon Comprehend 감성 데이터를 사용하여 파이형 차트를 생성합니다. 파이형 차트는 긍정적인 리뷰, 중립적 리뷰, 혼합 리뷰, 부정적인 리뷰의 비율을 보여줍니다.

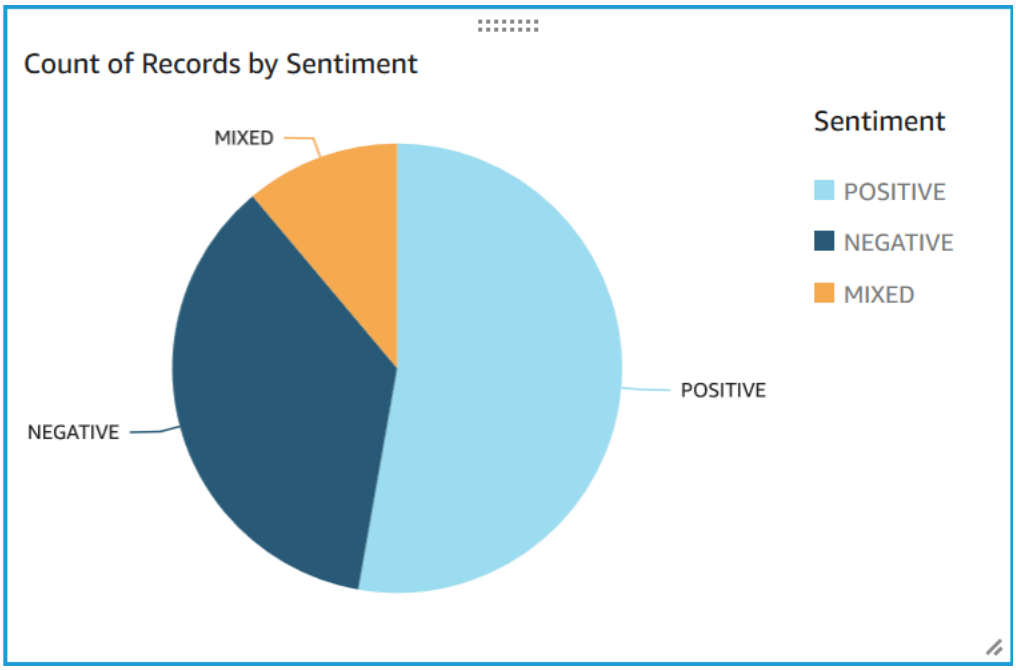
감성 데이터를 시각화하려면

1. QuickSight 콘솔에서 분석을 선택한 다음 새 분석을 선택합니다.
2. 데이터 세트에서 감성 데이터 세트 `sentiment_results_final`를 선택한 다음 분석 생성을 선택합니다.
3. 시각적 편집기의 필드 목록에서 감성을 선택합니다.

> **Note**
>
> 필드 목록의 값은 Amazon Athena에서 테이블을 만들 때 사용한 열 이름에 따라 달라집니다. SQL 쿼리에서 제공된 열 이름을 변경한 경우 필드 목록 이름은 이 시각화 예제에 사용된 이름과 다를 것입니다.

4. 시각적 유형에서 파이형 차트를 선택합니다.

긍정, 중립, 혼합 및 부정 섹션이 있는 다음과 비슷한 파이형 차트가 표시됩니다. 섹션 개수와 백분율을 보려면 그 위에 마우스를 가져갑니다.



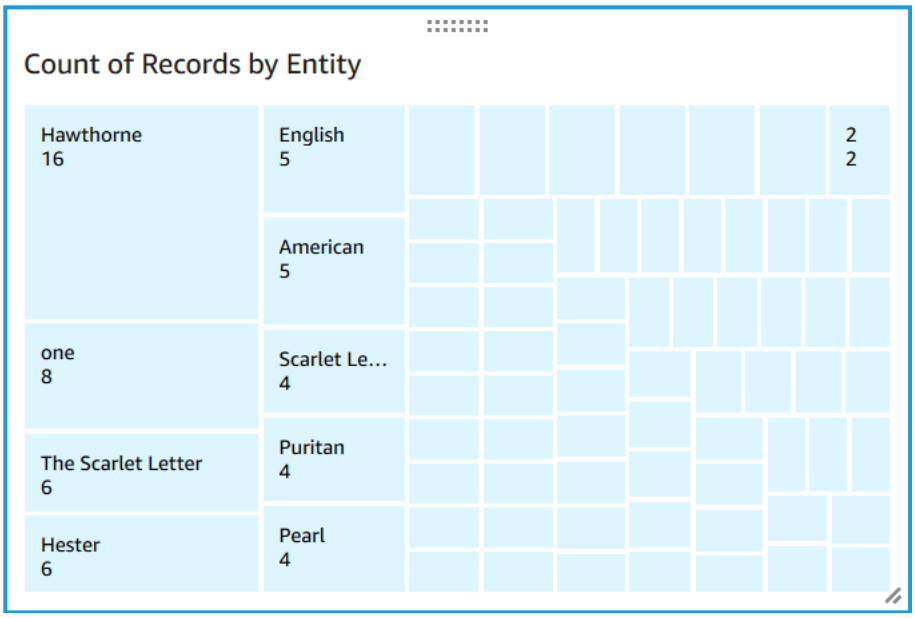
개체 시각화 생성

이제 개체 데이터 세트로 두 번째 시각화를 생성합니다. 데이터에 있는 개별 개체의 트리 맵을 만듭니다. 트리 맵의 각 블록은 개체를 나타내며, 블록 크기는 개체가 데이터 세트에 나타나는 횟수와 상관 관계가 있습니다.

개체 데이터를 시각화하려면

1. 시각화 제어판의 데이터 세트 옆에서 데이터 세트 추가, 편집, 교체, 및 제거 아이콘을 선택합니다.
2. (데이터 세트 추가를 선택합니다.
3. 추가할 데이터 세트 선택으로 데이터 세트 목록에서 개체 데이터 세트 `entities_results_final`를 선택하고 선택를 선택합니다.
4. 시각화 제어판에서 데이터 세트 드롭다운 메뉴를 선택하고 개체 데이터 세트 `entities_results_final`를 선택합니다.
5. 필드 목록에서 개체를 선택합니다.
6. 시각적 유형으로 트리 맵을 선택합니다.

다음과 비슷한 트리 맵이 파이형 차트 옆에 표시됩니다. 특정 개체의 개수를 보려면 마우스를 블록 위로 가져갑니다.



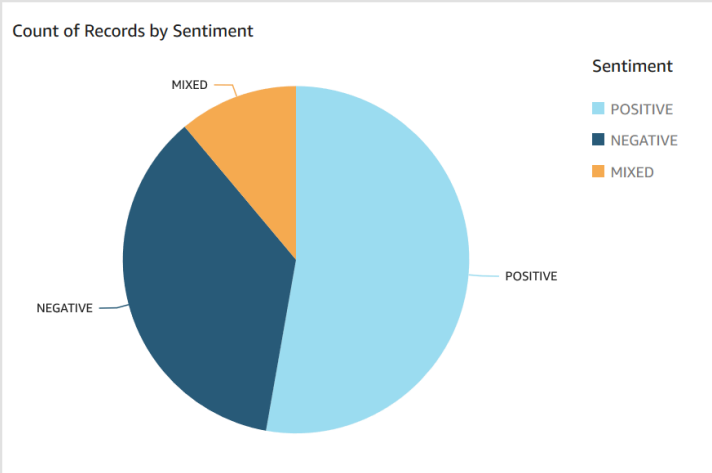
대시보드 게시

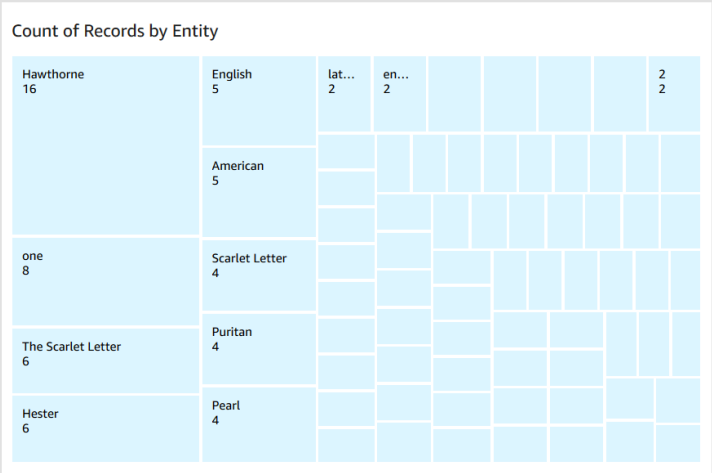
시각화를 생성한 후 대시보드로 게시할 수 있습니다. 대시보드를 사용하여 대시보드를의 사용자와 공유 AWS 계정하거나, PDF로 저장하거나, 보고서로 이메일을 보내는 등 다양한 작업을 수행할 수 있습니다(QuickSight 엔터프라이즈 에디션으로 제한됨). 이 단계에서는 시각화를 계정의 대시보드로 게시합니다.

대시보드를 게시하려면

1. 탐색 모음에서 공유를 선택합니다.
2. 대시보드 게시를 선택합니다.
3. 새 대시보드를 다른 이름으로 게시를 선택하고 대시보드 이름 `comprehend-analysis-reviews`를 입력합니다.
4. 대시보드 게시를 선택합니다.
5. 오른쪽 상단의 닫기 버튼을 선택하여 사용자와 대시보드 공유 창을 닫습니다.
6. QuickSight 콘솔의 탐색 창에서 대시보드를 선택합니다. 새 대시보드 `comprehend-analysis-reviews`의 썸네일이 대시보드 아래에 나타나야 합니다. 대시보드를 선택하면 확인할 수 있습니다.

이제 다음 예제와 비슷한 감성 및 개체 시각화가 포함된 대시보드가 생겼습니다.





> **Tip**
>
> 대시보드에서 시각화를 편집하려면 분석으로 돌아가서 업데이트하려는 시각화를 편집합니다. 그런 다음 대시보드를 새 대시보드로 다시 게시하거나 기존 대시보드를 대체하여 게시합니다.

정리

이 자습서를 완료한 후 더 이상 사용하지 않을 AWS 리소스를 정리할 수 있습니다. 활성 AWS 리소스는 계정에 계속 요금이 부과될 수 있습니다.

요금이 계속 발생하는 것을 방지하는 데 도움이 되는 것:

- QuickSight 구독을 취소합니다. QuickSight는 월간 구독 서비스입니다. 구독을 취소하려면 QuickSight 사용 설명서의 구독 취소를 참조하세요.
- Amazon S3 버킷 삭제. Amazon S3는 스토리지 비용을 부과합니다, Amazon S3 리소스를 정리하려면 버킷을 삭제하십시오. 버킷 삭제에 대한 자세한 내용은 Amazon Simple Storage Service 콘솔 사용 설명서의 S3 버킷을 삭제하려면 어떻게 해야 합니까?를 참조하세요. 버킷을 삭제하기 전에 중요한 파일을 모두 저장해야 합니다.
- AWS Glue Data Catalog를 지웁니다. 는 매월 스토리지에 대해 AWS Glue Data Catalog 요금을 부과합니다. 데이터베이스를 삭제하여 요금이 계속 발생하지 않도록 할 수 있습니다. AWS Glue Data Catalog 데이터베이스 관리에 대한 자세한 내용은 AWS Glue 개발자 안내서의 AWS Glue 콘솔에서 데이터베이스 작업을 참조하세요. 데이터베이스나 테이블을 정리하기 전에 데이터를 내보내야 합니다.

개인 식별 정보(PII)를 위한 Amazon S3 객체 Lambda 액세스 포인트 사용

개인 식별 정보 (PII)를 위한 Amazon S3 객체 Lambda 액세스 포인트를 사용하여 Amazon S3 버킷에서 문서를 검색하는 방법을 구성할 수 있습니다. PII가 포함된 문서에 대한 액세스를 제어하고 문서에서 PII를 교정할 수 있습니다. Amazon Comprehend가 문서 내 PII를 감지하는 방법에 대한 자세한 내용은 PII 개체 감지을 참조하세요. Amazon S3 객체 Lambda 액세스 포인트는 AWS Lambda 함수를 사용하여 표준 Amazon S3 GET 요청의 출력을 자동으로 변환합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 S3 객체 Lambda를 사용한 객체 변환을 참조하세요.

PII를 위한 Amazon S3 객체 Lambda 액세스 포인트를 생성하면 Amazon Comprehend Lambda 함수를 사용하여 문서를 처리해 PII가 포함된 문서에 대한 액세스를 제어하고 문서에서 PII를 교정합니다.

PII를 위한 Amazon S3 객체 Lambda 액세스 포인트를 생성하면 다음 Amazon Comprehend Lambda 함수를 사용하여 문서가 처리됩니다.

- `ComprehendPiiAccessControlS3ObjectLambda`- S3 버킷에 저장된 PII가 포함된 문서에 대한 액세스를 제어합니다. 이 Lambda 함수에 대한 자세한 내용은에 로그인하여에서 ComprehendPiiAccessControlS3ObjectLambda 함수를 AWS Management Console 확인합니다 AWS Serverless Application Repository.
- `ComprehendPiiRedactionS3ObjectLambda`- Amazon S3 버킷에 있는 문서에서 PII를 교정합니다. 이 Lambda 함수에 대한 자세한 내용은에 로그인하여에서 ComprehendPiiRedactionS3ObjectLambda 함수를 AWS Management Console 확인합니다 AWS Serverless Application Repository.

에서 서버리스 애플리케이션을 배포하는 방법에 대한 자세한 내용은 AWS Serverless Application Repository 개발자 안내서의 애플리케이션 배포를 AWS Serverless Application Repository참조하세요.

주제

- 개인 식별 정보(PII)가 포함된 문서에 대한 액세스 제어
- 문서에서 개인 식별 정보(PII) 교정하기

개인 식별 정보(PII)가 포함된 문서에 대한 액세스 제어

Amazon S3 객체 Lambda 액세스 포인트를 사용하여 개인 식별 정보(PII)가 포함된 문서에 대한 액세스를 제어할 수 있습니다.

권한이 있는 사용자만 Amazon S3 버킷에 저장된 PII가 포함된 문서에 액세스할 수 있도록 하려면 ComprehendPiiAccessControlS3ObjectLambda 함수를 사용합니다. 이 Lambda 함수는 문서 객체에 대한 표준 Amazon S3 GET 요청을 처리할 때 ContainsPiiEntities 작업을 사용합니다.

예를 들어, 신용카드 번호나 은행 계좌 번호와 같은 PII가 포함된 문서가 S3 버킷에 있는 경우 이러한 PII 개체 유형을 감지하고 권한이 없는 사용자에 대한 액세스를 제한하는 ComprehendPiiAccessControlS3ObjectLambda 함수를 구성할 수 있습니다. 지원되는 PII 개체 유형에 대한 자세한 내용은 PII 범용 개체 유형을 참조하세요.

이 Lambda 함수에 대한 자세한 내용은에 로그인하여에서 ComprehendPiiAccessControlS3ObjectLambda 함수를 AWS Management Console 확인합니다 AWS Serverless Application Repository.

문서에 대한 액세스를 제어하기 위한 Amazon S3 객체 Lambda 액세스 포인트 생성

다음 예제에서는 주민등록번호가 포함된 문서에 대한 액세스를 제어하는 Amazon S3 객체 Lambda 액세스 포인트를 생성합니다.

를 사용하여 Amazon S3 객체 Lambda 액세스 포인트 생성 AWS Command Line Interface

Amazon S3 객체 Lambda 액세스 포인트 구성을 생성하고 config.json이라는 파일에 구성을 저장합니다.

```
{
    "SupportingAccessPoint": "s3-default-access-point-name-arn",
    "TransformationConfigurations": [
        {
            "Actions": [
                "s3:GetObject"
            ],
            "ContentTransformation": {
                "AwsLambda": {
                    "FunctionArn": "comprehend-pii-access-control-s3-object-lambda-
arn",
                    "FunctionPayload": "{\"pii_entities_types\": \"SSN\"}"
                }
            }
        }
    ]
}
```

다음 예제에서는 config.json 파일에 정의된 구성을 기반으로 Amazon S3 객체 Lambda 액세스 포인트를 생성합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws s3control create-banner-access-point \
    --region region \
    --account-id account-id \
    --name s3-object-lambda-access-point \
    --configuration file://config.json
```

문서에 대한 액세스를 제어하기 위한 Amazon S3 객체 Lambda 액세스 포인트 간접 호출

다음 예제에서는 Amazon S3 객체 Lambda 액세스 포인트를 간접 호출하여 문서에 대한 액세스를 제어합니다.

를 사용하여 Amazon S3 객체 Lambda 액세스 포인트 호출 AWS Command Line Interface

다음 예제에서는 AWS CLI를 사용하여 Amazon S3 객체 Lambda 액세스 포인트를 간접 호출합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws s3api get-object \
    --region region \
    --bucket s3-object-lambda-access-point-name-arn \
    --key object-prefix-key output-file-name
```

문서에서 개인 식별 정보(PII) 교정하기

Amazon S3 객체 Lambda 액세스 포인트를 사용하여 문서에서 개인 식별 정보(PII)를 교정할 수 있습니다.

S3 버킷에 저장된 문서에서 PII 개체 유형을 교정하려면 ComprehendPiiRedactionS3ObjectLambda 함수를 사용합니다. 이 Lambda 함수는 문서 객체에 대한 표준 Amazon S3 GET 요청을 처리할 때 ContainsPiiEntities 및 DetectPiiEntities 작업을 사용합니다.

예를 들어, S3 버킷에 신용카드 번호 또는 은행 계좌 정보와 같은 PII 정보가 포함된 문서가 있는 경우 PII를 감지한 다음 PII 개체 유형이 교정된 문서의 복사본을 반환하는 `ComprehendPiiRedactionS3ObjectLambda` 함수를 구성할 수 있습니다. 지원되는 PII 개체 유형에 대한 자세한 내용은 PII 범용 개체 유형을 참조하세요.

이 Lambda 함수에 대한 자세한 내용은에 로그인하여에서 ComprehendPiiRedactionS3ObjectLambda 함수를 AWS Management Console 확인합니다 AWS Serverless Application Repository.

문서에서 PII를 교정하기 위한 Amazon S3 객체 Lambda 액세스 포인트 생성

다음 예제에서는 문서에서 신용카드 번호를 교정하는 Amazon S3 객체 Lambda 액세스 포인트를 생성합니다.

를 사용하여 Amazon S3 객체 Lambda 액세스 포인트 생성 AWS Command Line Interface

Amazon S3 객체 Lambda 액세스 포인트 구성을 생성하고 `config.json`이라는 파일에 구성을 저장합니다.

```
{
    "SupportingAccessPoint": "s3-default-access-point-name-arn",
    "TransformationConfigurations": [
        {
            "Actions": [
                "s3:GetObject"
            ],
            "ContentTransformation": {
                "AwsLambda": {
                    "FunctionArn": "comprehend-pii-redaction-s3-object-lambda-arn",
                    "FunctionPayload": "{\"pii_entities_types\": \"CREDIT_DEBIT_NUMBER
\"}"
                }
            }
        }
    ]
}
```

다음 예제에서는 `config.json`에 정의된 구성을 기반으로 Amazon S3 객체 Lambda 액세스 포인트 생성을 보여줍니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws s3control create-access-point-for-object-lambda \
    --region region \
    --account-id account-id \
    --name s3-object-lambda-access-point \
    --configuration file://config.json
```

문서에서 PII를 교정하기 위한 Amazon S3 객체 Lambda 액세스 포인트 간접 호출

다음 예제에서는 문서에서 PII를 교정하기 위한 Amazon S3 객체 Lambda 액세스 포인트를 간접 호출합니다.

를 사용하여 Amazon S3 객체 Lambda 액세스 포인트 호출 AWS Command Line Interface

다음 예제에서는 AWS CLI를 사용하여 Amazon S3 객체 Lambda 액세스 포인트를 간접 호출합니다.

다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws s3api get-object \
    --region region \
    --bucket s3-object-lambda-access-point-name-arn \
    --key object-prefix-key output-file-name
```

솔루션: Amazon Comprehend와 OpenSearch를 사용한 텍스트 분석

AWS 는 Amazon Comprehend 및 OpenSearch 서비스를 사용하여 텍스트 분석의 참조 구현을 제공합니다. Amazon Comprehend는 텍스트 분석을 제공하고 OpenSearch는 문서 인덱싱, 검색 및 시각화를 제공합니다.

자세한 내용은 OpenSearch 및 Amazon Comprehend를 사용한 텍스트 분석을 참조하세요.

API 참조

API 참조는 이제 별도의 문서입니다. 자세한 내용은 Amazon Comprehend API 참조를 참조하세요.

Amazon Comprehend의 문서 기록

다음 표에서는 이 Amazon Comprehend 릴리스 관련 문서를 소개합니다.

변경 사항	설명	날짜
네이티브 문서를 사용한 사용자 지정 분류기 학습	Amazon Comprehend는 이제 네이티브 문서를 사용한 사용자 지정 분류기 학습을 지원합니다. 자세한 내용은 Amazon Comprehend의 학습 분류 모델을 참조하세요.	2023년 4월 19일
사용자 지정 모델 관리를 위한 플라이휠	Amazon Comprehend는 이제 사용자 지정 모델의 모델 버전 학습 및 추적을 관리하는 데 도움이 되는 플라이휠을 지원합니다. 자세한 내용은 Amazon Comprehend의 플라이휠을 참조하세요.	2023년 2월 28일
IAM 보안 주제 업데이트	페더레이션 자격 증명을 포함하도록 IAM 보안 주제를 업데이트했습니다. 자세한 내용은 Amazon Comprehend의 자격 증명 및 액세스 관리를 참조하세요.	2022년 12월 22일
사용자 지정 모델을 사용한 추론을 위한 원스텝 프로세싱	Amazon Comprehend는 이제 사용자 지정 분류 또는 사용자 지정 개체 인식을 실행하기 전에 이미지, PDF 또는 Word 입력 문서에 대한 텍스트 추출을 자동으로 수행합니다. 자세한 내용은 Amazon Comprehend에서의 문서 처리를 참조하세요.	2022년 12월 1일

대상 감성을 위한 동기식 API	Amazon Comprehend는 이제 대상 감성을 위한 동기식 API 및 콘솔 실시간 분석을 지원합니다. 대상 감성은 문서 내 특정 개체와 관련된 감정을 결정합니다. 자세한 내용은 Amazon Comprehend의 대상 감성을 참조하세요.	2022년 9월 21일
학습 인식기의 최소 주석 수 줄임	Amazon Comprehend는 일반 텍스트 CSV 주석 파일을 사용하여 인식기를 학습시키는 데 필요한 최소 요구 사항을 줄였습니다. 이제 최소 3개의 주석이 달린 문서와 개체 유형당 최소 25개의 주석을 포함하는 사용자 지정 개체 인식 모델을 구축할 수 있습니다. 자세한 내용은 학습 데이터 준비를 참조하세요.	2022년 8월 3일
실시간 API의 입력 문서 크기 증가	Amazon Comprehend는 이제 대부분의 실시간 API에 대해 최대 100KB의 입력 문서를 지원합니다. 자세한 내용은 지침 및 할당량을 참조하세요.	2022년 7월 18일
추가 PII 개체 유형	이제 Amazon Comprehend에서 추가 PII 개체 유형을 감지합니다. 자세한 내용은 Amazon Comprehend에서 PII 개체 감지를 참조하세요.	2022년 5월 20일

목차 재구성	더 쉽게 탐색할 수 있도록 Amazon Comprehend 개발자 안내서 목차를 재구성했습니다. 자세한 내용은 Amazon Comprehend란?을 참조하세요.	2022년 4월 7일
대상 감성	Amazon Comprehend는 이제 문서의 특정 개체와 관련된 감정을 결정하는 대상 감성 분석을 지원합니다. 자세한 내용은 Amazon Comprehend의 대상 감성을 참조하세요.	2022년 3월 9일
새 기능	Amazon Comprehend에서는 이제 사용자 지정 개체 인식을 위해 이미지를 분석할 수 있습니다. 자세한 내용은 Amazon Comprehend에서 사용자 지정 개체 감지를 참조하세요.	2022년 2월 28일
새 기능	이제 학습된 사용자 지정 모델을 AWS 계정간에 복사할 수 있습니다. 자세한 내용은 Amazon Comprehend에서 계정 간 사용자 지정 모델 복사를 참조하세요.	2022년 2월 2일
새 기능	이제 AWS Trusted Advisor 를 사용하여 Amazon Comprehend 엔드포인트의 비용 및 보안을 최적화하는 데 도움이 되는 권장 사항을 볼 수 있습니다. 자세한 정보는 Amazon Comprehend로 Trusted Advisor 사용을 참조하세요.	2021년 9월 29일

새 기능	Amazon Comprehend는 새 모델 버전을 생성하고, 특정 테스트 세트에서 지속적으로 테스트하고, 새 모델 엔드포인트로 실시간 마이그레이션을 수행할 수 있는 기능을 제공함으로써 모델을 지속적으로 개선할 수 있는 Comprehend 사용자 지정 기능 모음을 출시했습니다.	2021년 9월 21일
새 기능	Amazon Comprehend에서는 이제 사용자 지정 개체 인식을 위해 PDF 및 Word 문서를 분석할 수 있습니다. PDF 및 Word 형식을 사용하면 헤더, 목록 및 표가 포함된 문서에서 정보를 추출할 수 있습니다.	2021년 9월 14일
새 기능	Amazon Comprehend는 엔드포인트를 전체적으로 볼 수 있는 새로운 엔드포인트 개요 기능을 출시했습니다. 엔드포인트 개요 페이지에서는 모든 엔드포인트를 한 곳에서 확인하여 엔드포인트 사용량과 실제 리소스 사용량을 비교할 수 있습니다.	2021년 8월 24일
새 기능	이제 Amazon Comprehend Medical은 인터페이스 VPC 엔드포인트를 생성하여 Virtual Private Cloud(VPC)와 프라이빗 연결을 설정할 수 있습니다. 자세한 내용은 VPC 엔드포인트(PrivateLink)를 참조하세요.	2021년 6월 13일

언어 확장	Amazon Comprehend는 주요 언어 기능에 하우사어(ha), 라오스어(lo), 몰타어(mt), 오로모어(om) 등 4개 언어를 추가했습니다. 자세한 내용은 Amazon Comprehend에서 지원되는 언어를 참조하세요.	2021년 5월 10일
새 기능	Amazon Comprehend에서는 이제 고객 관리형 키(CMK)를 사용하여 사용자 지정 모델을 암호화할 수 있습니다. 자세한 내용은 Amazon Comprehend의 KMS 암호화를 참조하세요.	2021년 3월 31일
새 기능	이제 Amazon S3 객체 Lambda 액세스 포인트를 사용하여 개인 식별 정보(PII)가 포함된 문서를 Amazon S3 버킷에서 검색하는 방법을 구성할 수 있습니다. PII가 포함된 문서에 대한 액세스를 제어하고 문서에서 PII를 교정할 수 있습니다. 자세한 내용은 개인 식별 정보(PII)를 위한 Amazon S3 객체 Lambda 액세스 포인트 사용을 참조하세요.	2021년 3월 18일
새 기능	이제 개인 식별 정보(PII)로 문서에 레이블을 지정할 수 있습니다. Amazon Comprehend는 문서에 PII가 있는지 분석하여 식별된 PII 개체 유형(예: 이름, 주소, 은행 계좌 번호 또는 전화번호)의 레이블을 반환할 수 있습니다. 자세한 내용은 PII로 문서 레이블 지정을 참조하세요.	2021년 3월 11일

새 기능	Amazon Comprehend를 사용하면 이제 문서 집합에서 이벤트를 감지할 수 있습니다. 비동기 이벤트 감지 작업을 생성하면 Amazon Comprehend는 지원되는 유형의 금융 이벤트를 감지할 수 있습니다. 자세한 내용은 이벤트 감지를 참조하세요.	2020년 11월 24일
새 기능	Amazon Comprehend에서는 이제 사용자 지정 개체 인식기 엔드포인트에 Auto Scaling을 사용할 수 있습니다. Auto Scaling을 사용하면 용량 요구 사항에 맞게 엔드포인트 프로비저닝을 자동으로 설정할 수 있습니다. 자세한 내용은 엔드포인트를 사용한 Auto Scaling을 참조하세요.	2020년 9월 28일
새 기능	사용자 지정 분류기 또는 개체 인식기를 훈련하기 위해 이제 Amazon SageMaker AI Ground Truth에서 생성한 레이블이 지정된 데이터 세트인 증강 매니페스트 파일을 제공할 수 있습니다. 이러한 파일에 대한 자세한 내용 및 예제를 보려면 다중 클래스 모드, 다중 레이블 모드 및 주석을 참조하세요.	2020년 9월 22일

새 자습서	Amazon Comprehend는 이제 고객 리뷰를 분석하고 분석 결과를 시각화하는 다중 서비스 워크플로를 안내하는 자습서를 제공합니다. 자세한 내용은 자습서: 리뷰에서 얻은 인사이트 분석을 참조하세요.	2020년 9월 17일
새 기능	Amazon Comprehend를 사용하면 텍스트에서 주소, 은행 계좌 번호 또는 전화번호와 같은 개인 식별 정보(PII)가 포함된 개체를 감지할 수 있습니다. Amazon Comprehend는 텍스트 내 각 PII 항목의 위치를 제공하거나 PII가 수정된 텍스트 사본을 제공할 수 있습니다. 자세한 정보는 개인 식별 정보(PII) 감지를 참조하세요.	2020년 9월 17일
새 기능	이전에는 최대 12개의 사용자 지정 개체에 대해서만 모델을 학습시킬 수 있었습니다. 이제 Amazon Comprehend를 사용하면 한 번에 최대 25개의 사용자 지정 개체에 대해 모델을 학습시킬 수 있습니다. 자세한 내용은 사용자 지정 개체 인식을 참조하세요.	2020년 8월 12일

언어 확장	Amazon Comprehend는 사용자 지정 개체 인식 기능을 위해 독일어(de), 스페인어(es), 프랑스어(fr), 이탈리아어(it), 포르투갈어(pt) 등 5개 언어를 추가했습니다. 자세한 내용은 Amazon Comprehend에서 지원되는 언어를 참조하세요.	2020년 8월 12일
새 기능	이제 Amazon Comprehend는 인터페이스 VPC 엔드포인트를 생성하여 Virtual Private Cloud(VPC)와 프라이빗 연결을 설정할 수 있습니다. 자세한 내용은 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.	2020년 8월 11일
새 기능	Amazon Comprehend를 사용하면 이제 실시간 분석을 실행하여 개별 텍스트 문서에서 사용자 지정 개체를 빠르게 감지할 수 있습니다. 자세한 내용은 Amazon Comprehend를 사용한 사용자 지정 개체 실시간 감지를 참조하세요.	2020년 7월 9일

새 기능 추가	Amazon Comprehend는 이제 문서에 대한 비동기 사용자 지정 분류의 두 번째 모드를 지원합니다. 이 모드는 문서에 사용자 지정 클래스를 적용할 때 더 큰 유연성을 제공합니다. 다중 클래스 모드는 단일 클래스만 각 문서와 연결하지만 새로운 다중 레이블 모드는 둘 이상의 클래스를 연결할 수 있습니다. 예를 들어, 영화를 공상 과학 영화와 액션 영화로 동시에 분류할 수 있습니다. 자세한 내용은 사용자 지정 분류의 다중 클래스 및 다중 레이블 모드를 참조하세요.	2019년 12월 19일
새 기능 추가	Amazon Comprehend는 이제 구조화되지 않은 텍스트가 포함된 문서에 대한 실시간 사용자 지정 분류 지원을 제공합니다. 고객은 실시간 사용자 지정 분류를 사용하여 자체 비즈니스 규칙을 기반으로 정보를 동시에 이해하고 레이블을 지정하고 라우팅할 수 있습니다. 자세한 내용은 사용자 지정 분류로 실시간 분석을 참조하세요.	2019년 11월 25일

새로운 언어 추가	Amazon Comprehend는 몇 가지 기능에 아랍어(ar), 힌디어(hi), 일본어(ja), 한국어(ko), 중국어 간체(zh), 중국어 번체(zh-TW) 등 6개 언어를 추가했습니다. 이러한 새 언어는 감정 파악, 핵심 문구 감지 및 비-사용자 지정 감지 개체 작업에만 지원됩니다. 자세한 내용은 지원되는 언어를 참조하세요.	2019년 11월 6일
새 기능	이전에는 단일 사용자 지정 개체에서만 모델을 학습시킬 수 있었습니다. 따라서 개체 인식 작업을 통해 해당 개체 하나만 검색할 수 있었습니다. Amazon Comprehend는 이를 변경하여 이제 한 번에 최대 12개의 사용자 지정 개체에 대해 모델을 학습시킬 수 있습니다. 자세한 내용은 사용자 지정 개체 인식을 참조하세요.	2019년 7월 9일
새 기능	Amazon Comprehend는 이제 사용자 지정 분류기를 학습시킬 때 지표를 분석하는 추가 기능을 위한 다중 클래스 혼동 지표를 제공합니다. 이는 현재 API만 사용하여 지원됩니다. 자세한 내용은 Amazon Comprehend에서 리소스에 태그 지정을 참조하세요.	2019년 4월 5일

새 기능	Amazon Comprehend는 사용자 지정 분류자 및 사용자 지정 개체 인식기용 태그를 제공합니다. 이 태그를 메타데이터로 사용하면 그 어느 때보다 세밀한 제어 수준으로 리소스를 구성, 필터링 및 제어할 수 있습니다. 자세한 내용은 Amazon Comprehend에서 리소스에 태그 지정을 참조하세요.	2019년 4월 3일
새 기능	Amazon S3에서는 이미 입력 문서를 암호화할 수 있도록 지원하고 있으며, Amazon Comprehend는 이를 훨씬 더 확장합니다. 자체 KMS 키를 사용하면 작업의 출력 결과뿐만 아니라 분석 작업을 처리하는 컴퓨팅 인스턴스에 연결된 스토리지 볼륨의 데이터도 암호화할 수 있습니다. 그 결과 엔드-투-엔드 보안이 이루어집니다. 자세한 내용은 Amazon Comprehend의 KMS 암호화를 참조하세요.	2019년 3월 28일

새 기능	사용자 지정 개체 인식은 사전 설정된 일반 개체 유형 중 하나로 지원되지 않는 새 개체 유형을 식별할 수 있도록 하여 Amazon Comprehend의 기능을 확장합니다. 즉, 문서를 분석하고 특정 요구 사항에 맞는 제품 코드나 비즈니스별 개체와 같은 개체를 추출할 수 있습니다. 자세한 내용은 사용자 지정 개체 인식을 참조하세요.	2018년 11월 16일
새 기능	Amazon Comprehend를 사용하여 문서를 클래스 또는 범주에 할당하여 사용자 지정 분류를 위한 자체 모델을 구축할 수 있습니다. 자세한 내용은 데이터 분류를 참조하세요.	2018년 11월 15일
리전 확장	이제 유럽(프랑크푸르트)(eu-central-1)에서 Amazon Comprehend를 사용할 수 있습니다.	2018년 10월 10일
언어 확장	Amazon Comprehend는 이제 영어와 스페인어 외에도 프랑스어, 독일어, 이탈리아어 및 포르투갈어로 된 문서를 검사할 수 있습니다. 자세한 내용은 Amazon Comprehend에서 지원되는 언어를 참조하세요.	2018년 10월 10일
리전 확장	이제 아시아 태평양(시드니)(ap-southeast-2)에서 Amazon Comprehend를 사용할 수 있습니다.	2018년 8월 15일

새 기능	Amazon Comprehend는 이제 문서를 분석하여 문서 구문과 각 단어의 품사를 찾아냅니다. 자세한 내용은 구문을 참조하세요.	2018년 7월 17일
새 기능	Amazon Comprehend는 이제 언어, 핵심 문구, 개체 및 감성 감지를 위한 비동기식 배치 처리를 지원합니다. 자세한 내용은 비동기식 배치 처리를 참조하세요.	2018년 27월 6일
새 안내서	이 설명서는 Amazon Comprehend 개발자 안내서의 최초 릴리스입니다.	2017년 11월 29일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.

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