aws

개발자 안내서

Amazon Textract

Amazon Textract: 개발자 안내서

Table of Contents

Amazon Textract Textract란 무엇인가?

Amazon Textract Textract를 사용하면 애플리케이션에 문서 텍스트 감지 및 분석 기능을 쉽게 추가할 수 있습니다. Amazon Textract 사용 고객은 다음을 수행할 수 있습니다.

- 재무 보고서, 의료 기록, 세금 양식 등 다양한 문서에서 입력 및 필기 텍스트를 감지합니다.
- Amazon Textract 문서 분석 API를 사용하여 구조화된 데이터가 있는 문서에서 텍스트, 양식 및 테이블을 추출합니다.
- 애널리즈경비 API를 사용하여 송장 및 영수증을 처리합니다.
- AnalyzeID API를 사용하여 미국 정부가 발급한 운전 면허증 및 여권과 같은 ID 문서를 처리합니다.

Amazon Textract Textract는 Amazon의 컴퓨터 비전 과학자들이 매일 수십억 개의 이미지와 비디오를 매일 분석할 목적으로 개발하여 성능이 검증되었을 뿐만 아니라 확장성까지 뛰어난 딥 러닝 기술을 기반으로 하고 있습니다. 를 사용하는 데는 머신 러닝에 관한 전문 지식이 필요하지 않습니다. Amazon Textract Textract에는 이미지 파일과 PDF 파일을 분석할 수 있는 간단하고 사용하기 쉬운 API가 포함되어 있습니다. Amazon Textract Textract는 항상 새로운 데이터에서 학습하고 있으며, Amazon은 서비스에 새로운 기능을 지속적으로 추가하고 있습니다.

Amazon Textract Textract를 사용하는 데는 다음과 같은 이점이 있습니다.

- 지능형 검색 인덱스 만들기— Amazon Textract Textract를 사용하면 이미지 및 PDF 파일에서 감지된 텍스트 라이브러리를 생성할 수 있습니다.
- 자연 언어 처리 (NLP) 를 위한 지능형 텍스트 추출 사용— Amazon Textract Textract는 텍스트를 NLP 애플리케이션의 입력으로 그룹화하는 방법을 제어할 수 있습니다. 텍스트를 단어와 줄로 추출할 수 있습니다. 또한 Amazon Textract 문서 테이블 분석이 활성화된 경우 테이블 셀별로 텍스트를 그룹화합니다.
- 다양한 소스의 데이터 캡처 및 정규화 가속화— Amazon Textract Textract를 사용하면 재무 문서, 연구 보고서 및 의료 메모와 같은 다양한 문서에서 텍스트 및 표 형식의 데이터를 추출할 수 있습니다. Amazon Textract Analyze 문서 API를 사용하면 문서에서 구조화되지 않은 데이터와 구조화된 데이터를 쉽고 빠르게 추출할 수 있습니다.
- 양식에서 데이터 캡처 자동화— Amazon Textract Textract를 사용하면 양식에서 구조화된 데이터를 추출할 수 있습니다. Amazon Textract Analysis API를 사용하면 양식을 통해 제출된 사용자 데이터를 사용 가능한 형식으로 추출할 수 있도록 기존 비즈니스 워크플로에 추출 기능을 구축할 수 있습니다.

Amazon Textract Textract를 사용하면 다음과 같은 이점이 있습니다.

- 앱에 문서 텍스트 감지 통합— Amazon Textract Textract는 간단한 API로 강력하고 정확한 분석을 사용할 수 있으므로 복잡하게 애플리케이션에 텍스트 감지 기능을 포함시킬 필요가 없습니다. Amazon Textract Textract를 사용하여 문서 텍스트를 감지하는 데는 컴퓨터 비전이나 딥 러닝에 관한 전문 지식이 필요하지 않습니다. Amazon Textract Text API를 사용하면 어떤 웹, 모바일 또는 연결된 장치의 애플리케이션에도 쉽게 텍스트 감지 기능을 통합할 수 있습니다.
- 확장 가능 문서 분석— Amazon Textract Textract를 사용하면 수백만 개의 문서에서 데이터를 신속하게 분석하고 추출하여 의사 결정을 가속화할 수 있습니다.
- 저렴한 비용— Amazon Textract Textract를 사용하면 분석하는 문서에 대해서만 비용을 지불합니다. 최소 요금이나 사전 약정은 없습니다. 무료로 사용을 시작한 다음 확장에 따라 의 계층별 요금 모델을 활용하여 더 많은 비용을 절감하십시오.

동기식 처리를 통해 Amazon Textract Textract는 지연 시간이 중요한 애플리케이션을 위해 단일 페이지 문서를 분석할 수 있습니다. 또한 Amazon Textract Textract는 다중 페이지 문서로 지원을 확장하는 비동기 작업을 제공합니다.

처음 Amazon Textract 사용자

Amazon Textract Textract를 처음 사용하는 경우, 먼저 다음 단원을 순서대로 읽어보십시오.

1. Amazon Textract 작동 방식- 이 단원에서는 Amazon Textract 구성 요소와 이러한 구성 요소가 엔드투 엔드 경험을 위해 함께 작동하는 방식을 소개합니다.
2. Amazon Textract 시작하기- 이 단원에서는 계정을 설정하고 Amazon Textract API를 테스트합니다.

와 함께 Amazon Textract 추출을 사용하여AWSSDK

다양한 프로그래밍 언어에 대해 AWS 소프트웨어 개발 키트(SDK)를 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제

SDK 설명서	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 .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제

가용성 예

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

Amazon Textract 작동 방식

Amazon Textract Textract를 사용하면 단일 또는 다중 페이지 입력 문서의 텍스트를 감지하고 분석할 수 있습니다 (입력 문서).

Amazon Textract Textract는 다음 작업에 대한 작업을 제공합니다.

- 텍스트만 감지합니다. 자세한 내용은 를 참조하십시오.텍스트 감지.
- 텍스트 간의 관계를 감지하고 분석합니다. 자세한 내용은 를 참조하십시오.문서 분석.
- 인보이스 및 영수증의 텍스트 감지 및 분석 자세한 내용은 를 참조하십시오.송장 및 수금 분석.
- 정부 신원 증명서의 텍스트 감지 및 분석 자세한 내용은 를 참조하십시오.ID 문서 분석.

Amazon Textract Textract는 작은 단일 페이지 문서를 처리하고 거의 실시간으로 응답하는 동기식 작업을 제공합니다. 자세한 정보는 동기식 작업을 통한 문서 처리을 참조하십시오. 또한 Amazon Textract Textract는 더 큰 다중 페이지 문서를 처리하는 데 사용할 수 있는 비동기 작업도 제공합니다. 비동기 응답은 실시간으로 이루어지지 않습니다. 자세한 정보는 비동기 작업을 사용한 문서 처리을 참조하십시오.

Amazon Textract 작업에서 문서를 처리할 때 결과는 다음과 같은 배열로 반환됩니다.the section called “Block”객체 또는 배열the section called “ExpenseDocument”객체입니다. 두 개체에는 문서상의 위치 및 문서의 다른 항목과의 관계 등 항목에 대해 감지된 정보가 들어 있습니다. 자세한 정보는 Amazon Textract 응답 객체을 참조하십시오. 사용 방법을 보여주는 예제의 경우`Block`객체, 참조자습서.

주제

- 텍스트 감지
- 문서 분석
- 송장 및 수금 분석
- ID 문서 분석
- 입력 문서
- Amazon Textract 응답 객체
- 문서 페이지의 항목 위치

텍스트 감지

Amazon Textract Textract는 문서에서 감지된 텍스트만 반환하는 동기 및 비동기 작업을 제공합니다. 두 작업 세트 모두에 대해 다음 정보가 여러 개로 반환됩니다.the section called “Block”객체입니다.

- 감지된 텍스트의 줄과 단어
- 검색된 텍스트의 줄과 단어 간의 관계
- 검색된 텍스트가 표시되는 페이지
- 문서 페이지의 텍스트 줄 및 단어 위치

자세한 정보는 the section called “줄 및 텍스트 단어”을 참조하십시오.

텍스트를 동기적으로 감지하려면DetectDocumentTextAPI 작업을 수행하고 문서 파일을 입력으로 전달합니다. 전체 결과 집합이 작업에 의해 반환됩니다. 자세한 내용과 예제는 동기식 작업을 통한 문서 처리 단원을 참조하십시오.

> **Note**
>
> Amazon Rekognition API 작업`DetectText`는 와 다릅니다.`DetectDocumentText`. 사용할 수 있습니다`DetectText`포스터 또는 도로 표지판과 같은 라이브 장면에서 텍스트를 감지합니다.

텍스트를 비동기적으로 감지하려면StartDocumentTextDetection입력 문서 파일 처리를 시작합니다. 결과를 가져오려면GetDocumentTextDetection. 결과는 다음 중 하나 이상의 응답으로 반환됩니다.`GetDocumentTextDetection`. 자세한 내용과 예제는 비동기 작업을 사용한 문서 처리 단원을 참조하십시오.

문서 분석

Amazon Textract Textract는 검색된 텍스트 간의 관계를 위해 문서와 양식을 분석합니다. Amazon Textract 분석 작업은 텍스트, 양식 및 표 등 세 가지 문서 추출 범주를 반환합니다. 송장 및 영수증 분석은 다른 프로세스를 통해 처리됩니다. 자세한 내용은송장 및 수금 분석.

텍스트 추출

문서에서 추출한 원시 텍스트입니다. 자세한 내용은 단원을 참조하십시오.줄 및 텍스트 단어.

양식 추출

양식 데이터는 문서에서 추출한 텍스트 항목에 링크됩니다. Amazon Textract Textract는 양식 데이터를 키-값 페어로 나타냅니다. 다음 예에서 Amazon Textract Textract에서 감지한 텍스트 줄 중 하나는 다음과 같습니다.이름: 제인 Doe. 또한 Amazon Textract Textract는 키를 식별합니다 (이름:) 및 값 (제인 Doe). 자세한 내용은 단원을 참조하십시오.양식 데이터 (키-값 쌍).

이름: 제인 Doe

Address: 123 애니 스트리트, 애니타운, 미국

생년월일: 12-26-1980

키-값 쌍은 양식에서 추출된 확인란 또는 옵션 버튼 (라디오 버튼) 을 나타내는 데에도 사용됩니다.

male:☑

자세한 내용은 단원을 참조하십시오.선택 요소.

테이블 추출

Amazon Textract Textract는 테이블, 테이블 셀 및 테이블 셀 내의 항목을 추출할 수 있으며 결과를 JSON, .csv 또는.txt 파일로 반환하도록 프로그래밍할 수 있습니다.

이름	Address
아나 캐롤라이나	123 Any Town

자세한 내용은 테이블을 참조하십시오. 테이블에서 선택 요소를 추출할 수도 있습니다. 자세한 내용은 단원을 참조하십시오.선택 요소.

분석된 아이템의 경우 Amazon Textract Textract는 다음을 여러 개로 반환합니다.the section called “Block”객체:

- 감지된 텍스트의 줄과 단어
- 탐지된 항목의 내용
- 탐지된 항목 간의 관계
- 항목이 감지된 페이지
- 문서 페이지에 있는 항목의 위치입니다.

동기식 또는 비동기 작업을 사용하여 문서의 텍스트를 분석할 수 있습니다. 텍스트를 동기적으로 분석하려면AnalyzeDocument작업을 수행하고 문서를 입력으로 전달합니다.AnalyzeDocument은 전체 결과 집합을 반환합니다. 자세한 정보는 Amazon Textract Textract를 사용하여 문서 텍스트 분석을 참조하십시오.

텍스트를 비동기적으로 감지하려면StartDocumentAnalysis처리를 시작합니다. 결과를 가져오려면GetDocumentAnalysis. 결과는 다음 중 하나 이상의 응답으로 반환됩니다.GetDocumentAnalysis. 자세한 내용과 예제는 다중 페이지 문서에서 텍스트 감지 또는 분석 단원을 참조하십시오.

수행할 분석 유형을 지정하려면FeatureTypes목록 입력 매개 변수입니다. 목록에 TABLES를 추가하여 입력 문서에서 검색된 테이블 (예: 표 셀, 셀 텍스트 및 셀의 선택 요소) 에 대한 정보를 반환합니다. FORMS를 추가하여 키-값 쌍 및 선택 요소와 같은 단어 관계를 반환합니다. 두 가지 유형의 분석을 모두 수행하려면 TABLES와 FORMS를 다음에 추가합니다.FeatureTypes.

문서에서 감지된 모든 줄과 단어가 응답에 포함됩니다 (값과 관련이 없는 텍스트 포함)FeatureTypes).

송장 및 수금 분석

Amazon Textract Textract는 템플릿 또는 구성 없이 거의 모든 인보이스 또는 영수증에서 연락처 정보, 구매한 아이템 및 공급업체 이름과 같은 관련 데이터를 추출합니다. 송장 및 영수증은 종종 다양한 레이아웃을 사용하므로 대규모로 데이터를 수동으로 추출하는 것이 어렵고 시간이 많이 걸립니다. Amazon Textract Textract는 ML을 사용하여 송장 및 영수증의 컨텍스트를 파악하고 송장 또는 영수증 날짜, 인보이스 또는 영수증 번호, 아이템 가격, 총액 및 결제 조건과 같은 데이터를 비즈니스 요구에 맞게 자동으로 추출합니다.

또한 Amazon Textract Textract는 워크플로에 중요하지만 명시적으로 레이블이 지정되지 않을 수 있는 공급업체 이름도 식별합니다. 예를 들어 Amazon Textract Textract는 명시적인 키-값 쌍 조합 없이 페이지 상단의 로고 안에만 표시되어 있더라도 영수증에서 공급업체 이름을 찾을 수 있습니다. 또한 Amazon Textract Textract를 사용하면 동일한 개념에 대해 서로 다른 단어를 사용하는 다양한 영수증 및 송장의 입력을 쉽게 통합할 수 있습니다. 예를 들어 Amazon Textract Textract는 고객 번호, 고객 번호, 계정 ID와 같은 서로 다른 문서의 필드 이름 간의 관계를 매핑하여 표준 분류법을 다음과 같이 출력합니다.INVOICE_RECEIPT_ID. 이 경우 Amazon Textract Textract는 여러 문서 유형에 걸쳐 일관되게 데이터를 나타냅니다. 표준 분류법에 정렬되지 않는 필드는 다음과 같이 분류됩니다.OTHER.

다음은 AnalyzeFreent가 현재 지원하는 표준 필드 목록입니다.

- 공급업체 이름:VENDOR_NAME

- 총:TOTAL
- 수신자 주소:RECEIVER_ADDRESS
- 송장/수금 일자:INVOICE_RECEIPT_DATE
- 송장/영수증 ID:INVOICE_RECEIPT_ID
- 결제 조건:PAYMENT_TERMS
- 소계:SUBTOTAL
- 기한:DUE_DATE
- 세금:TAX
- 송장 납세자 ID (SSN/ITIN 또는 EIN):TAX_PAYER_ID
- 아이템 이름:ITEM_NAME
- 아이템 가격:PRICE
- 품목 수량:QUANTITY

AnalyzeCreents API는 지정된 문서 페이지에 대해 다음 요소를 반환합니다.

- 페이지 내의 영수증 또는 청구서 수는 다음과 같이 표시됩니다.ExpenseIndex
- 다음과 같이 표현되는 개별 필드의 표준화된 이름Type
- 다음과 같이 표현되는 필드의 실제 이름입니다.LabelDetection
- 다음과 같이 표시되는 해당 필드의 값입니다.ValueDetection
- 제출된 문서 내의 페이지 수는 다음과 같이 표시됩니다.Pages
- 필드, 값 또는 라인 항목이 감지된 페이지 번호로,PageNumber
- 페이지에 있는 개별 필드, 값 또는 라인 항목의 경계 상자와 좌표 위치를 포함하는 지오메트리로Geometry
- 문서에서 탐지된 각 데이터 조각과 관련된 신뢰 점수로, 다음과 같이 표시됩니다.Confidence
- 구매한 개별 품목의 전체 행으로,EXPENSE_ROW

다음은 AnalyzeFreents에서 처리한 수금에 대한 API 출력의 일부이며, 표준 필드로 추출된 문서의 합계: $55.64를 보여 줍니다.TOTAL, 문서의 실제 텍스트를 “합계”, 신뢰 점수 “97.1”, 페이지 번호 “1”, “$55.64"로 된 총 값 및 테두리 상자 및 다각형 좌표:

```
{
    "Type": {
        "Text": "TOTAL",
```

```
        "Confidence": 99.94717407226562
    },
    "LabelDetection": {
        "Text": "Total:",
        "Geometry": {
            "BoundingBox": {
                "Width": 0.09809663146734238,
                "Height": 0.0234375,
                "Left": 0.36822840571403503,
                "Top": 0.8017578125
            },
            "Polygon": [
                {
                    "X": 0.36822840571403503,
                    "Y": 0.8017578125
                },
                {
                    "X": 0.466325044631958,
                    "Y": 0.8017578125
                },
                {
                    "X": 0.466325044631958,
                    "Y": 0.8251953125
                },
                {
                    "X": 0.36822840571403503,
                    "Y": 0.8251953125
                }
        ]
    },
    "Confidence": 97.10792541503906
},
    "ValueDetection": {
        "Text": "$55.64",
        "Geometry": {
            "BoundingBox": {
                "Width": 0.10395314544439316,
                "Height": 0.0244140625,
                "Left": 0.66837477684021,
                "Top": 0.802734375
            },
            "Polygon": [
                {
                    "X": 0.66837477684021,
```

```
                    "Y": 0.802734375
                },
                {
                    "X": 0.7723279595375061,
                    "Y": 0.802734375
                },
                {
                    "X": 0.7723279595375061,
                    "Y": 0.8271484375
                },
                {
                    "X": 0.66837477684021,
                    "Y": 0.8271484375
                }
            ]
        },
    "Confidence": 99.85165405273438
},
"PageNumber": 1
}
```


동기 작업을 사용하여 송장 또는 영수증을 분석할 수 있습니다. 이러한 문서를 분석하려면 AnalyzeFreent 작업을 사용하여 영수증 또는 송장을 전달합니다.`AnalyzeExpense`은 전체 결과 집합을 반환합니다. 자세한 정보는 Amazon Textract Textract를 사용하여 송장 및 영수증 분석을 참조하십시오.

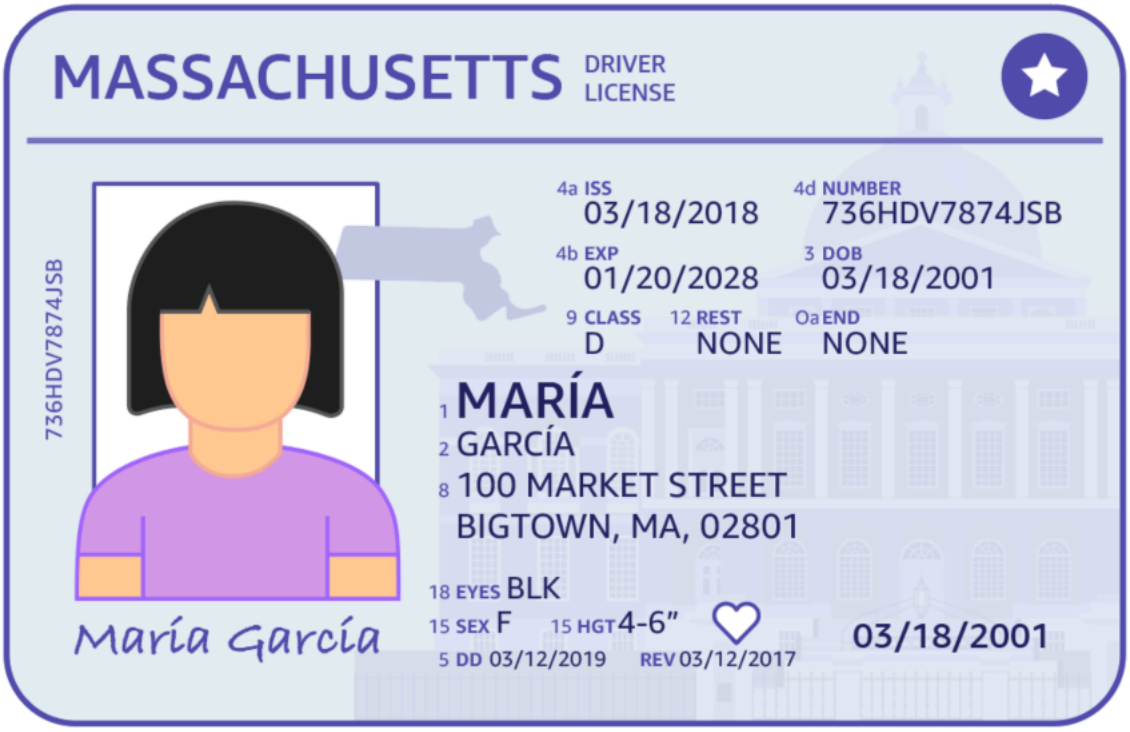
송장 및 영수증을 비동기적으로 분석하려면StartExpenseAnalysis입력 문서 파일 처리를 시작합니다. 결과를 가져오려면GetExpenseAnalysis. 지정된 호출에 대한 결과StartExpenseAnalysis에서 반환됩니다.`GetExpenseAnalysis`. 자세한 내용과 예제는 비동기 작업을 사용한 문서 처리 단원을 참조하십시오.

ID 문서 분석

Amazon Textract Textract는 AnalyzeID API를 사용하여 미국 정부가 발행한 여권, 운전 면허증 및 기타 자격 증명 문서에서 관련 정보를 추출할 수 있습니다. Analyze ID를 사용하면 기업들은 미국 운전 면허증, 주 ID 및 템플릿이나 형식이 다른 여권과 같은 ID에서 정보를 빠르고 정확하게 추출할 수 있습니다. AnalyzeID API는 다음 두 가지 범주의 데이터 유형을 반환합니다.

- 생년월일, 발행일, ID 번호, 클래스 및 제한과 같은 ID에서 사용할 수 있는 키-값 쌍입니다.
- 이름, 주소, 발행자 등 명시적 키가 연결되어 있지 않을 수 있는 문서의 암시적 필드입니다.

주요 이름은 응답 내에서 표준화됩니다. 예를 들어 운전 면허증에 LIC# (라이센스 번호) 이 표시되고 여권에 여권 아니오가 표시되면 분석 ID 응답은 표준 키를 원시 키 (예: LIC#) 와 함께 “문서 ID”로 반환합니다. 이러한 표준화를 통해 고객은 동일한 개념에 대해 서로 다른 용어를 사용하는 여러 ID에서 정보를 쉽게 결합할 수 있습니다.



분석 ID는 다음과 같은 구조에서 정보를 반환합니다.`IdentityDocumentFields`. 이것들은JSON정규화된 유형 및 유형과 연관된 값의 두 가지 정보를 포함하는 구조체입니다. 이 두 가지 모두 신뢰 점수도 있습니다. 자세한 정보는 ID 문서 응답 객체을 참조하십시오.

동기 작업을 사용하여 운전 면허증이나 여권을 분석할 수 있습니다. 이러한 문서를 분석하려면 AnalyzeID 작업을 사용하여 ID 문서를 전달합니다.`AnalyzeID`은 전체 결과 집합을 반환합니다. 자세한 정보는 Amazon Textract Textract를 사용하여 자격 증명 문서 분석을 참조하십시오.

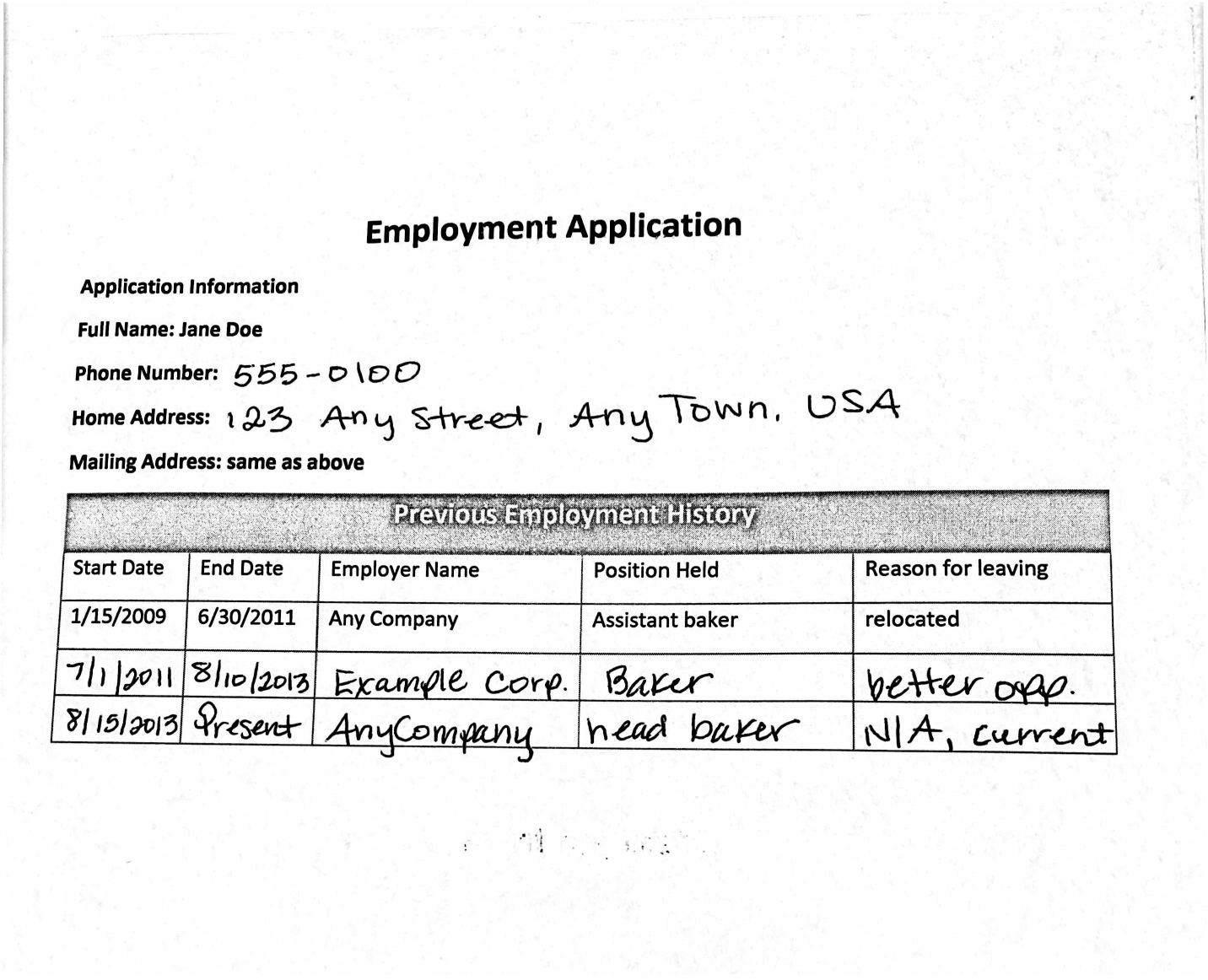
> **Note**
>
> 운전 면허증과 같은 일부 신분 증명서에는 양면이 있습니다. 동일한 Analyze ID API 요청 내에서 운전면허증의 전면 및 후면 이미지를 별도의 이미지로 전달할 수 있습니다.

입력 문서

Amazon Textract 작업에 적합한 입력은 단일 또는 다중 페이지 문서입니다. 일부 예로는 법률 문서, 양식, 신분증 또는 서신이 있습니다. 양식은 사용자가 답변을 제공하라는 질문이나 프롬프트가 있는 문서입니다. 환자 등록 양식, 세금 양식 또는 보험 청구 양식을 예로 들 수 있습니다.

문서는 JPEG, PNG, PDF 또는 TIFF 형식일 수 있습니다. PDF 및 TIFF 형식 파일을 사용하면 여러 페이지로 된 문서를 처리할 수 있습니다. Amazon Textract Textract가 문서를 다음과 같이 표현하는 방법에 대한 자세한 내용은Block객체, 참조텍스트 감지 및 문서 분석 응답 객체.

다음은 허용 가능한 입력 문서 예제입니다.



Employment Application

Application Information

Full Name: Jane Doe

Phone Number: 555-0100

Home Address: 123 Any Street, Any Town, USA

Mailing Address: same as above

Previous Employment History

Start Date	End Date	Employer Name	Position Held	Reason for leaving
1/15/2009	6/30/2011	Any Company	Assistant baker	relocated
7/1/2011	8/10/2013	Example Corp.	Baker	better opp.
8/15/2013	Present	AnyCompany	head baker	N/A, current

문서 제한에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

Amazon Textract 동기 작업의 경우 Amazon S3 버킷에 저장된 입력 문서를 사용하거나 base64로 인코딩된 이미지 바이트를 전달할 수 있습니다. 자세한 정보는 Amazon Textract 동기 운영 호출을 참조하십시오. 비동기 작업의 경우 Amazon S3 버킷에 입력 문서를 제공해야 합니다. 자세한 정보는 Amazon Textract 비동기 작업 호출을 참조하십시오.

Amazon Textract 응답 객체

Amazon Textract 작업은 실행 작업에 따라 다른 유형의 객체를 반환합니다. 텍스트를 감지하고 일반 문서를 분석하기 위해 작업은 Block 객체를 반환합니다. 송장 또는 영수증을 분석하기 위해 작업은 ExpenseDocuments 객체를 반환합니다. ID 문서를 분석하기 위해 이 작업은 IdentityDocumentFields 객체를 반환합니다. 이러한 응답 객체에 대한 자세한 내용은 다음 단원을 참조하십시오.

주제

- 텍스트 감지 및 문서 분석 응답 객체
- 송장 및 수금 응답 객체
- ID 문서 응답 객체

텍스트 감지 및 문서 분석 응답 객체

Amazon Textract Textract가 문서를 처리할 때 다음과 같은 목록을 생성합니다.Block탐지되거나 분석된 텍스트의 개체입니다. 각 블록에는 탐지된 아이템, 위치 및 Amazon Textract Textract가 처리의 정확성에 대한 확신에 대한 정보가 포함되어 있습니다.

문서는 다음과 같은 유형으로 구성됩니다.`Block`객체입니다.

- 페이지
- 줄 및 텍스트 단어
- 양식 데이터 (키-값 쌍)
- 표 및 셀
- 선택 요소

블록의 내용은 호출하는 작업에 따라 다릅니다. 텍스트 감지 작업 중 하나를 호출하면 검색된 텍스트의 페이지, 줄 및 단어가 반환됩니다. 자세한 정보는 텍스트 감지을 참조하십시오. 문서 분석 작업 중 하나를 호출하면 검색된 페이지, 키-값 쌍, 테이블, 선택 요소 및 텍스트에 대한 정보가 반환됩니다. 자세한 정보는 문서 분석을 참조하십시오.

약간Block객체 필드는 두 유형의 처리에 공통적입니다. 예를 들어, 각 블록에는 고유한 식별자가 있습니다.

사용 방법을 보여주는 예제의 경우Block객체, 참조자습서.

문서 레이아웃

Amazon Textract Textract는 문서의 표현을 여러 유형의 목록으로 반환합니다.Block상위-하위 관계 또는 키-값 쌍으로 연결된 객체입니다. 문서의 페이지 수를 제공하는 메타데이터도 반환됩니다. 다음은 전형적인 JSON 입니다.Block형식의 객체PAGE.

```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
                ]
            },
            "Relationships": [
                {
                    "Type": "CHILD",
```

```
                    "Ids": [
                        "2602b0a6-20e3-4e6e-9e46-3be57fd0844b",
                        "82aedd57-187f-43dd-9eb1-4f312ca30042",
                        "52be1777-53f7-42f6-a7cf-6d09bdc15a30",
                        "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"
        }.....

    ],
    "DocumentMetadata": {
        "Pages": 1
    }
}
```

문서가 하나 이상에서 작성됩니다.PAGE블록을 지원합니다. 각 페이지에는 페이지에서 감지된 기본 항목 (예: 텍스트 및 표 줄) 에 대한 하위 블록 목록이 포함되어 있습니다. 자세한 정보는 페이지을 참조하십시오.

의 유형을 결정할 수 있습니다.Block객체를 검사하여BlockType필드.

ABlock객체가 관련 목록을 포함하고 있습니다.Block의 객체Relationships필드, 배열입니다.Relationship객체입니다. ARelationships배열은 CHILD 유형 또는 VALUE 유형입니다. CHILD 유형의 배열은 현재 블록의 하위인 항목을 나열하는 데 사용됩니다. 예를 들어, 현재 블록이 LINE 유형일 경우Relationships에는 텍스트 줄을 구성하는 WORD 블록의 ID 목록이 포함되어 있습니다. VALUE 유형의 배열은 키-값 페어를 포함하는 데 사용됩니다. 다음을 검사하여 관계 유형을 확인할 수 있습니다.Type의 필드Relationship객체입니다.

하위 블록에는 상위 블록 오브젝트에 대한 정보가 없습니다.

다음과 같은 예시Block정보는 단원을 참조하십시오.동기식 작업을 통한 문서 처리.

신뢰도

Amazon Textract 작업은 감지된 아이템의 정확성에 대해 Amazon Textract Textract가 가진 비율의 신뢰도를 반환합니다. 자신감을 얻으려면Confidence의 필드Block객체입니다. 값이 높을수록 신뢰도가 높아집니다. 시나리오에 따라 신뢰도가 낮은 탐지는 사람의 시각적 확인이 필요할 수 있습니다.

Geometry

Amazon Textract 작업은 자격 증명 분석을 제외하고 문서 페이지에서 탐지된 항목의 위치에 대한 위치 정보를 반환합니다. 위치를 가져오려면`Geometry`의 필드`Block`객체입니다. 자세한 내용은 단원을 참조하십시오.문서 페이지의 항목 위치

페이지

문서는 하나 이상의 페이지로 구성됩니다. Athe section called “Block”형식의 객체`PAGE`문서의 각 페이지에 대해 존재합니다. A`PAGE`블록 객체에는 문서 페이지에서 검색된 텍스트 줄, 키-값 쌍 및 테이블의 하위 ID 목록이 포함되어 있습니다.

a를 위한 JSON`PAGE`block은 다음과 비슷합니다.

```
{

    "Geometry": ....
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "2602b0a6-20e3-4e6e-9e46-3be57fd0844b", // Line - Hello, world.
                "82aedd57-187f-43dd-9eb1-4f312ca30042", // Line - How are you?
                "52be1777-53f7-42f6-a7cf-6d09bdc15a30",
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"  // Page identifier
},
```

PDF 형식의 여러 페이지로 된 문서에서 비동기 작업을 사용하는 경우 다음을 검사하여 블록이 있는 페이지를 확인할 수 있습니다.`Page`의 필드`Block`객체입니다. 스캔한 이미지 (JPEG, PNG, PDF 또는 TIFF 형식의 이미지) 는 이미지에 문서 페이지가 두 개 이상인 경우에도 단일 페이지 문서로 간주됩니다. 비동기 작업은 항상 a를 반환합니다.`Page`스캔한 이미지의 값 1입니다.

총 페이지 수가 반환됩니다.`Pages`의 필드`DocumentMetadata`.`DocumentMetadata`각 목록과 함께 반환됩니다.`Block`Amazon Textract 작업에서 반환된 객체입니다.

줄 및 텍스트 단어

Amazon Textract 작업에서 반환된 감지된 텍스트는 다음 목록에 반환됩니다.the section called "Block"객체입니다. 이러한 개체는 문서 페이지에서 감지되는 텍스트 또는 텍스트 단어를 나타냅니다. 다음 텍스트는 여러 단어로 만들어진 두 줄의 텍스트를 보여 줍니다.

이것은 텍스트입니다.

두 줄로 구분됩니다.

감지된 텍스트는 다음 위치에 반환됩니다.`Text`필드`Block`객체입니다. 이`BlockType`필드는 텍스트가 텍스트 줄 (LINE) 인지 또는 단어 (WORD) 인지 결정합니다. A단어는 공백으로 구분되지 않은 하나 이상의 ISO 기본 라틴 스크립트 문자입니다. A선는 탭으로 구분된 연속 단어 문자열입니다.

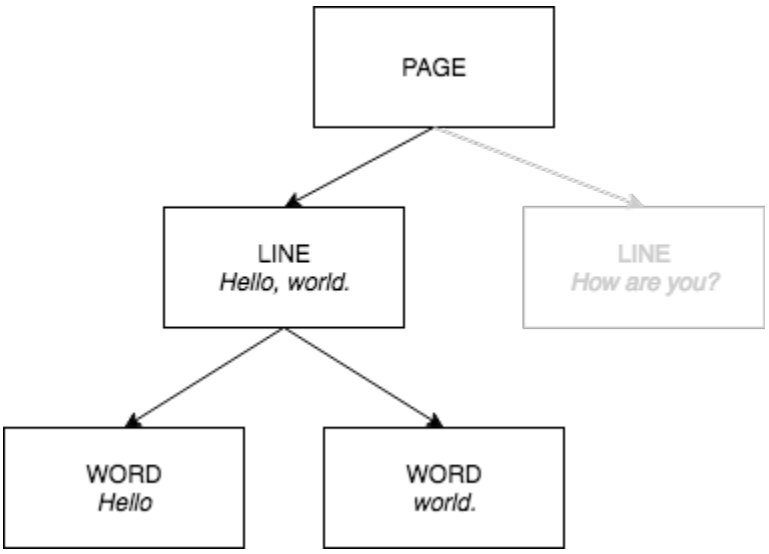
또한 Amazon Textract Textract는 텍스트를 사용하여 손글씨 또는 인쇄되었는지 여부를 결정합니다.`TextTypes`필드. 이들은 각각 필기 및 인쇄로 반환됩니다.

기타`Block`특성은 ID, 신뢰 및 지오메트리 정보와 같은 모든 블록 유형에 공통적입니다. 자세한 정보는 the section called "텍스트 감지 및 문서 분석 응답 객체"을 참조하십시오.

줄과 단어만 감지하려면 다음을 사용할 수 있습니다.DetectDocumentText또는StartDocumentTextDetection. 자세한 정보는 텍스트 감지을 참조하십시오. 감지된 텍스트 (선 및 단어) 와 문서의 다른 부분 (예: 표) 과 관련되는 방법에 대한 정보를 얻으려면 다음을 사용할 수 있습니다.AnalyzeDocument또는StartDocumentAnalysis. 자세한 정보는 문서 분석을 참조하십시오.

`PAGE`,`LINE`, 및`WORD`블록은 상위-하위 관계에서 서로 관련되어 있습니다. A`PAGE`블록은 모두의 상위입니다.`LINE`문서 페이지에서 객체를 차단합니다. LINE에는 하나 이상의 단어가 있을 수 있습니다.`Relationships`LINE 블록에 대한 배열은 텍스트 행을 구성하는 하위 WORD 블록의 ID를 저장합니다.

다음 다이어그램은 라인을 보여 줍니다.Hello world.텍스트에서Hello world. 어떻게 지내세요?는 다음과 같이 표시됩니다.`Block`객체입니다.



다음은 의 JSON 출력입니다.`DetectDocumentText`때 문장Hello world. 어떻게 지내세요?가 감지됩니다. 첫 번째 예제는 문서 페이지의 JSON입니다. 하위 ID를 사용하여 문서를 탐색할 수 있는 방법에 유의하십시오.

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "d7fbd604-d609-4d69-857d-247a3f591238", // Line - Hello, world.
                "b6c19a93-6493-4d8e-958f-853c8f7ca055" //  Line - How are you?
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "56ec1d77-171f-4881-9852-2b5b7e761608"
},
```

다음은 “Hello, World”라는 줄을 구성하는 LINE 블록에 대한 JSON입니다.

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "7f97e2ca-063e-47a8-981c-8beee31afc01", // Word - Hello,
```

```
                "4b990aa0-af96-4369-b90f-dbe02538ed21"  // Word - world.
            ]
        }
    ],
    "Confidence": 99.63229370117188,
    "Geometry": {...},
    "Text": "Hello, world.",
    "BlockType": "LINE",
    "Id": "d7fbd604-d609-4d69-857d-247a3f591238"
},
```

다음은 단어의 WORD 블록에 대한 JSON입니다.안녕하세요.:

```
{
    "Geometry": {...},
    "Text": "Hello,",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.74746704101562,
    "Id": "7f97e2ca-063e-47a8-981c-8beee31afc01"
},
```

마지막 JSON은 단어의 WORD 블록입니다.세계.:

```
{
    "Geometry": {...},
    "Text": "world.",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.5171127319336,
    "Id": "4b990aa0-af96-4369-b90f-dbe02538ed21"
},
```

양식 데이터 (키-값 쌍)

Amazon Textract Textract는 문서에서 양식 데이터를 키-값 페어로 추출할 수 있습니다. 예를 들어 다음 텍스트에서 Amazon Textract Textract가 키를 식별할 수 있습니다.이름:) 및 값 (아나 캐롤라이나).

이름: 아나 캐롤라이나

감지된 키-값 페어는Block응답의 객체AnalyzeDocument과GetDocumentAnalysis.
이FeatureTypes키-값 쌍, 테이블 또는 둘 다에 대한 정보를 검색하는 입력 매개 변수입니다. 키-값

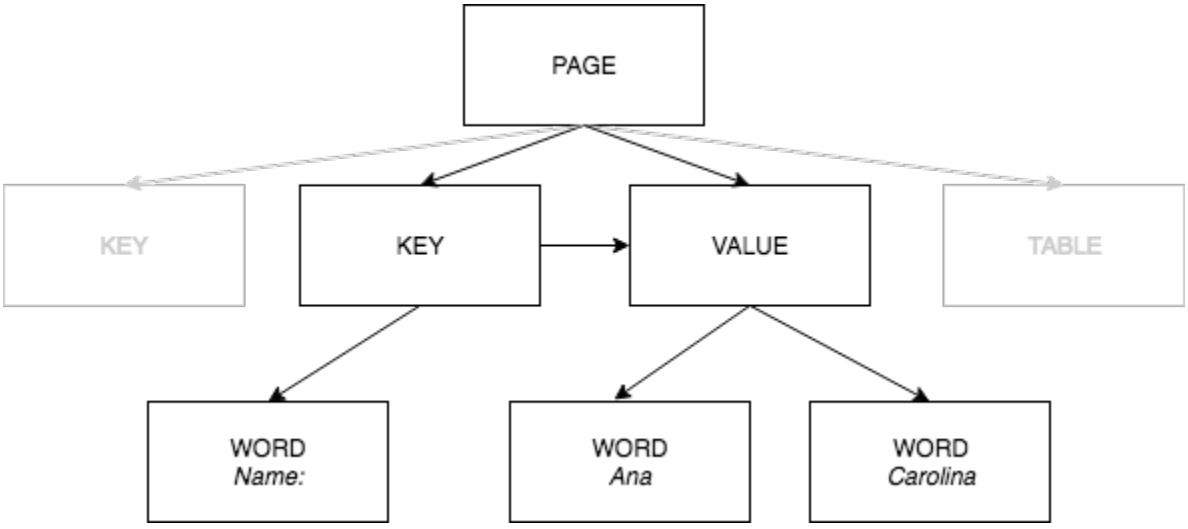
쌍에만 해당 값을 사용하십시오.FORMS. 문제 해결 예는 양식 문서에서 키-값 쌍 추출을(를) 참조하십시오. 문서를 표현하는 방법에 대한 일반 정보Block객체, 참조텍스트 감지 및 문서 분석 응답 객체.

KEY_VALUE_SET 유형을 가진 블록 객체는 문서에서 탐지된 링크된 텍스트 항목에 대한 정보를 저장하는 KEY 또는 VALUE 블록 객체의 컨테이너입니다. 이EntityType블록이 KEY인지 VALUE인지 여부를 결정하는 속성입니다.

- A키object는 링크된 텍스트의 키에 대한 정보를 포함합니다. 예,이름:. KEY 블록에는 두 개의 관계 목록이 있습니다. VALUE 유형의 관계는 키와 연관된 VALUE 블록의 ID를 포함하는 목록입니다. CHILD 유형의 관계는 키의 텍스트를 구성하는 WORD 블록의 ID 목록입니다.
- A값객체에는 키와 연결된 텍스트에 대한 정보가 들어 있습니다. 이전 예제에서아나 캐롤라이나키의 값입니다.이름:. VALUE 블록은 WORD 블록을 식별하는 자식 블록 목록과 관계가 있습니다. 각 WORD 블록에는 값의 텍스트를 구성하는 단어 중 하나가 포함되어 있습니다. AVALUE객체에는 선택한 요소에 대한 정보도 포함될 수 있습니다. 자세한 정보는 선택 요소을 참조하십시오.

KEY_VALUE_SET의 각 인스턴스Block객체는 PAGE의 하위입니다.Block현재 페이지에 해당하는 객체입니다.

다음 다이어그램은 키-값 페어의 방식을 보여 줍니다.이름: 아나 캐롤라이나는 다음과 같이 표시됩니다.Block객체입니다.



다음 예제에서는 키-값 페어의 방식을 보여 줍니다.이름: 아나 캐롤라이나는 JSON으로 표시됩니다.

PAGE 블록에는 유형의 하위 블록이 있습니다.KEY_VALUE_SET문서에서 감지된 각 KEY 및 VALUE 블록에 대해

```
{
```

```
    "Geometry": ....
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "2602b0a6-20e3-4e6e-9e46-3be57fd0844b",
                "82aedd57-187f-43dd-9eb1-4f312ca30042",
                "52be1777-53f7-42f6-a7cf-6d09bdc15a30", // Key - Name:
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"  // Value - Ana Caroline
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"  // Page identifier
},
```

다음 JSON은 키 블록 (52베1777-53f7-42f6-A7cF-6d09BDC15A30) 이 밸류 블록 (7ca6-00ef-4CDA-바1AA-5571dfed1a7c) 과의 관계가 있음을 보여줍니다. 또한 키에 대한 텍스트가 포함 된 워드 블록 (c734fca6-c4c4-415c-b6c1-30f7510b72ee) 에 대한 자식 블록이 있습니다.이름:).

```
{
    "Relationships": [
        {
            "Type": "VALUE",
            "Ids": [
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"  // Value identifier
            ]
        },
        {
            "Type": "CHILD",
            "Ids": [
                "c734fca6-c4c4-415c-b6c1-30f7510b72ee"  // Name:
            ]
        }
    ],
    "Confidence": 51.55965805053711,
    "Geometry": ....,
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "KEY"
    ],
    "Id": "52be1777-53f7-42f6-a7cf-6d09bdc15a30"  //Key identifier
```

```
},
```

다음 JSON은 값 블록 7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c가 값의 텍스트를 구성하는 WORD 블록에 대한 ID의 자식 목록을 가지고 있음을 보여줍니다.애나과캐롤라이나).

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "db553509-64ef-4ecf-ad3c-bea62cc1cd8a", // Ana
                "e5d7646c-eaa2-413a-95ad-f4ae19f53ef3"  // Carolina
            ]
        }
    ],
    "Confidence": 51.55965805053711,
    "Geometry": ....,
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "VALUE"
    ],
    "Id": "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c" // Value identifier
}
```

다음 JSON은Block단어를 위한 객체이름:,애나, 및캐롤라이나.

```
{
    "Geometry": {...},
    "Text": "Name:",
    "TextType": "PRINTED".
    "BlockType": "WORD",
    "Confidence": 99.56285858154297,
    "Id": "c734fca6-c4c4-415c-b6c1-30f7510b72ee"
},
 {
    "Geometry": {...},
    "Text": "Ana",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.52057647705078,
    "Id": "db553509-64ef-4ecf-ad3c-bea62cc1cd8a"
},
{
```

```
    "Geometry": {...},
    "Text": "Carolina",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.84207916259766,
    "Id": "e5d7646c-eaa2-413a-95ad-f4ae19f53ef3"
},
```

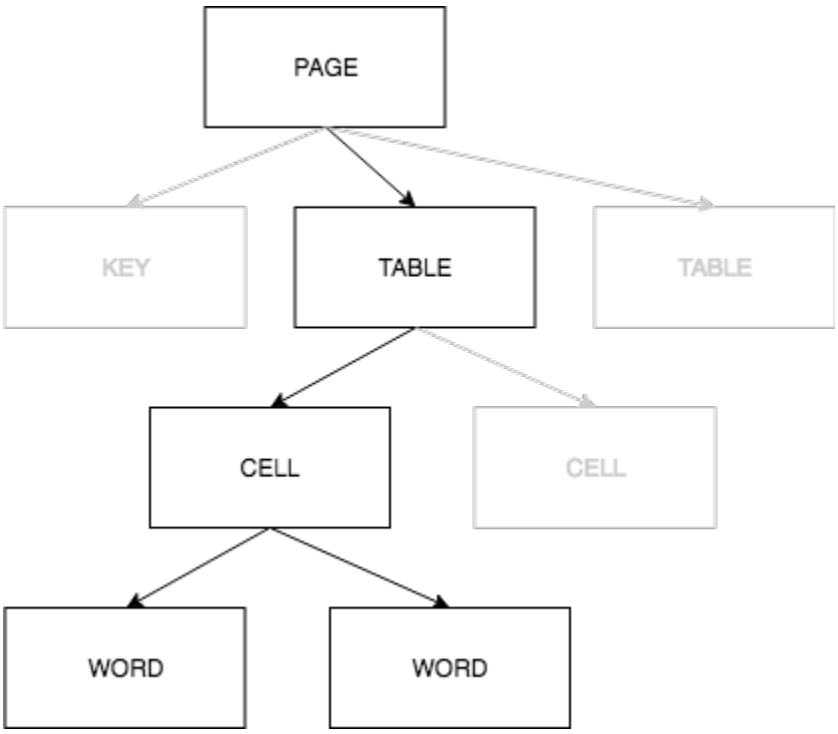
테이블

Amazon Textract Textract는 테이블의 테이블과 셀을 추출할 수 있습니다. 예를 들어 양식에서 다음 표가 감지되면 Amazon Textract Textract는 네 개의 셀이 있는 테이블을 감지합니다.

이름	Address
아나 캐롤라이나	123 Any Stown

감지된 테이블은 다음과 같이 반환됩니다Block응답의 객체AnalyzeDocument과GetDocumentAnalysis. 이FeatureTypes키-값 쌍, 테이블 또는 둘 다에 대한 정보를 검색하는 입력 매개 변수입니다. 테이블에만 해당 값을 사용하십시오.TABLES. 문제 해결 예는 테이블을 CSV 파일로 내보내기을(를) 참조하십시오. 문서를 표현하는 방법에 대한 일반 정보Block객체, 참조텍스트 감지 및 문서 분석 응답 객체.

다음 다이어그램은 테이블의 단일 셀이 어떻게 표현되는지 보여줍니다.Block객체입니다.



셀에는 다음이 포함됩니다.WORD탐지된 단어에 대한 블록SELECTION_ELEMENT확인란과 같은 선택 요소에 대한 블록입니다.

다음은 네 개의 셀이 있는 위 테이블의 부분 JSON입니다.

PAGE Block 객체에는 TABLE 블록에 대한 자식 블록 ID 목록과 감지된 각 텍스트 라인이 있습니다.

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "f2a4ad7b-f21d-4966-b548-c859b84f66a4",   // Line - Name
                "4dce3516-ffeb-45e0-92a2-60770e9cb744",   // Line  - Address
                "ee506578-768f-4696-8f4b-e4917e429f50",   // Line - Ana Carolina
                "33fc7223-411b-4399-8a90-ccd3c5a2c196",   // Line  - 123 Any Town
                "3f9665be-379d-4ae7-be44-d02f32b049c2"    // Table
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "78c3ce84-ae70-418e-add7-27058418adf6"
},
```

TABLE 블록에는 테이블 내의 셀에 대한 하위 ID 목록이 포함되어 있습니다. TABLE 블록에는 문서의 테이블 위치에 대한 형상 정보도 포함됩니다. 다음 JSON은 테이블에 네 개의 셀이 있음을 보여 줍니다.`Ids`어레이.

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "505e9581-0d1c-42fb-a214-6ff736822e8c",
                "6fca44d4-d3d3-46ab-b22f-7fca1fbaaf02",
                "9778bd78-f3fe-4ae1-9b78-e6d29b89e5e9",
                "55404b05-ae12-4159-9003-92b7c129532e"
            ]
        }
    ],
    "BlockType": "TABLE",
    "Confidence": 92.5705337524414,
    "Id": "3f9665be-379d-4ae7-be44-d02f32b049c2"
},
```

테이블 셀의 블록 유형은 CELL입니다. 이`Block`각 셀의 객체에는 테이블의 다른 셀과 비교하여 셀 위치에 대한 정보가 포함됩니다. 또한 문서에서 셀 위치에 대한 형상 정보도 포함됩니다. 이전 예제에서`505e9581-0d1c-42fb-a214-6ff736822e8c`는 단어가 포함된 셀의 하위 ID입니다.이름. 다음 예제는 셀에 대한 정보입니다.

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "e9108c8e-0167-4482-989e-8b6cd3c3653e"
            ]
        }
    ],
    "Confidence": 100.0,
    "RowSpan": 1,
    "RowIndex": 1,
    "ColumnIndex": 1,
    "ColumnSpan": 1,
```

```
    "BlockType": "CELL",
    "Id": "505e9581-0d1c-42fb-a214-6ff736822e8c"
},
```

각 셀은 테이블의 위치를 가지며 첫 번째 셀은 1,1입니다. 이전 예제에서 값이 있는 셀입니다.이름은 1행, 열 1에 있습니다. 값이 있는 셀123 Any Stown은 2행, 2열에 있습니다. 셀 블록 객체에는 이 정보가RowIndex과ColumnIndex필드. 하위 목록에는 셀 내에 있는 텍스트가 포함된 WORD Block 개체의 ID가 포함되어 있습니다. 목록의 단어는 셀의 왼쪽 상단에서 셀의 오른쪽 하단까지 검색된 순서대로 표시됩니다. 위의 예에서 셀에는 값 e9108c8e-0167-4482-989e-8b6CD3c3653e인 하위 ID가 있습니다. 다음 출력은 ID 값이 e9108c8e-0167-4482-989e-8b6CD3c3653e인 워드 블록에 대한 출력입니다.

```
"Geometry": {...},
"Text": "Name",
"TextType": "Printed",
"BlockType": "WORD",
"Confidence": 99.81139373779297,
"Id": "e9108c8e-0167-4482-989e-8b6cd3c3653e"
},
```

선택 요소

Amazon Textract Textract는 문서 페이지의 옵션 버튼 (라디오 버튼) 및 확인란과 같은 선택 요소를 감지할 수 있습니다. 선택 요소는 다음에서 감지할 수 있습니다.양식 데이터그리고테이블. 예를 들어 양식에서 다음 표가 검색되면 Amazon Textract Textract는 표 셀의 확인란을 감지합니다.

	동의	보통	동의하지 않음
서비스	☑	☐	☐
사용이 간편합니다.	☐	☑	☐
공정 요금	☑	☐	☐

감지된 선택 요소는 다음과 같이 반환됩니다.Block응답의 객체AnalyzeDocument과GetDocumentAnalysis.

> **Note**
>
> 이FeatureTypes키-값 쌍, 테이블 또는 둘 다에 대한 정보를 검색하는 입력 매개 변수입니다. 예를 들어, 테이블을 필터링하는 경우 응답에는 테이블에서 탐지된 선택 요소가 포함됩니다. 키-값 쌍에서 감지된 선택 요소는 응답에 포함되지 않습니다.

선택 요소에 대한 정보는Block형식의 객체SELECTION_ELEMENT. 선택 가능한 요소의 상태를 확인하려면SelectionStatus의 필드SELECTION_ELEMENT블록을 지원합니다. 상태는 다음과 같을 수 있습니다.이 선택되어 있습니다또는NOT_선택됨. 예를 들어, 값은SelectionStatus이전 이미지의 경우이 선택되어 있습니다.

ASELECTION_ELEMENT Block객체는 키-값 쌍 또는 테이블 셀과 연결됩니다. ASELECTION_ELEMENT Block객체에 선택 요소에 대한 경계 상자 정보가 포함되어 있습니다.Geometry필드. ASELECTION_ELEMENT Block개체가PAGE Block객체입니다.

양식 데이터 (키-값 쌍)

키-값 쌍은 양식에서 감지된 선택 요소를 나타내는 데 사용됩니다. 이KEYblock에는 선택 요소의 텍스트가 들어 있습니다. 이VALUE블록에는 셀렉션_요소 블록이 포함되어 있습니다. 다음 다이어그램은 선택 요소를 다음과 같이 표현하는 방식을 보여 줍니다.the section called “Block”객체입니다.

키-값 페어에 대한 자세한 내용은 단원을 참조하십시오.양식 데이터 (키-값 쌍).

다음 JSON 스니펫은 선택 요소를 포함하는 키-값 쌍의 키를 보여 줍니다 (남성 ☑). 하위 ID (아이디 bd14cfd5-9005-498b-a7f3-45ceb171f0ff) 는 선택 요소의 텍스트를 포함하는 WORD 블록의 ID입니다.남성). 값 ID (아이디 24AAAC7F - FC-49C7-A4F0-3688B05586D4) 는VALUE다음을 포함하는 블록SELECTION_ELEMENT블록 객체입니다.

```
{
    "Relationships": [
        {
            "Type": "VALUE",
            "Ids": [
                "24aaac7f-fcce-49c7-a4f0-3688b05586d4"  // Value containing Selection
 Element
            ]
        },
        {
            "Type": "CHILD",
```

```
            "Ids": [
                "bd14cfd5-9005-498b-a7f3-45ceb171f0ff"  // WORD - male
            ]
        }
    ],
    "Confidence": 94.15619659423828,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.022914813831448555,
            "Top": 0.08072036504745483,
            "Left": 0.18966935575008392,
            "Height": 0.014860388822853565
        },
        "Polygon": [
            {
                "Y": 0.08072036504745483,
                "X": 0.18966935575008392
            },
            {
                "Y": 0.08072036504745483,
                "X": 0.21258416771888733
            },
            {
                "Y": 0.09558075666427612,
                "X": 0.21258416771888733
            },
            {
                "Y": 0.09558075666427612,
                "X": 0.18966935575008392
            }
        ]
    },
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "KEY"
    ],
    "Id": "a118dc43-d5f7-49a2-a20a-5f876d9ffd79"
}
```

다음 JSON 스니펫은 해당 단어에 대한 WORD 블록입니다.남성. WORD 블록에는 상위 LINE 블록도 있습니다.

```
{
```

```
    "Geometry": {
        "BoundingBox": {
            "Width": 0.022464623674750328,
            "Top": 0.07842985540628433,
            "Left": 0.18863198161125183,
            "Height": 0.01617223583161831
        },
        "Polygon": [
            {
                "Y": 0.07842985540628433,
                "X": 0.18863198161125183
            },
            {
                "Y": 0.07842985540628433,
                "X": 0.2110965996980667
            },
            {
                "Y": 0.09460209310054779,
                "X": 0.2110965996980667
            },
            {
                "Y": 0.09460209310054779,
                "X": 0.18863198161125183
            }
        ]
    },
    "Text": "Male",
    "BlockType": "WORD",
    "Confidence": 54.06439208984375,
    "Id": "bd14cfd5-9005-498b-a7f3-45ceb171f0ff"
},
```

값 블록에는 셀렉션_요소 블록인 자식 (아이디 f2f5e8CD-e73A-4E99-A095-053ACD3b6bfb) 이 있습니다.

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "f2f5e8cd-e73a-4e99-a095-053acd3b6bfb"  // Selection element
            ]
        }
```

```
    ],
    "Confidence": 94.15619659423828,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.017281491309404373,
            "Top": 0.07643391191959381,
            "Left": 0.2271782010793686,
            "Height": 0.026274094358086586
        },
        "Polygon": [
            {
                "Y": 0.07643391191959381,
                "X": 0.2271782010793686
            },
            {
                "Y": 0.07643391191959381,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.2271782010793686
            }
        ]
    },
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "VALUE"
    ],
    "Id": "24aaac7f-fcce-49c7-a4f0-3688b05586d4"
},
}
```

다음 JSON은 셀렉션_엘리먼트 블록입니다. 의 가치SelectionStatus확인란이 선택되었음을 나타냅니다.

```
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.020316146314144135,
```

```
            "Top": 0.07575977593660355,
            "Left": 0.22590067982673645,
            "Height": 0.027631107717752457
        },
        "Polygon": [
            {
                "Y": 0.07575977593660355,
                "X": 0.22590067982673645
            },
            {
                "Y": 0.07575977593660355,
                "X": 0.2462168186903
            },
            {
                "Y": 0.1033908873796463,
                "X": 0.2462168186903
            },
            {
                "Y": 0.1033908873796463,
                "X": 0.22590067982673645
            }
        ]
    },
    "BlockType": "SELECTION_ELEMENT",
    "SelectionStatus": "SELECTED",
    "Confidence": 74.14942932128906,
    "Id": "f2f5e8cd-e73a-4e99-a095-053acd3b6bfb"
}
```


테이블 셀

Amazon Textract Textract는 테이블 셀 내에서 선택 요소를 감지할 수 있습니다. 예를 들어 다음 표의 셀에는 확인란이 있습니다.

	동의	보통	동의하지 않음
서비스 좋음	☑	☐	☐
사용이 간편합니다.	☐	☑	☐
공정 요금	☑	☐	☐

ACELL하위를 포함할 수 있는 블록SELECTION_ELEMENT하위 요소뿐만 아니라 선택 요소에 대한 객체WORD감지된 텍스트의 블록입니다.

테이블에 대한 자세한 내용은 단원을 참조하십시오.테이블.

테이블Block이전 테이블의 객체는 이와 비슷합니다.

```
{
    "Geometry": {.....},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "652c09eb-8945-473d-b1be-fa03ac055928",
                "37efc5cc-946d-42cd-aa04-e68e5ed4741d",
                "4a44940a-435a-4c5c-8a6a-7fea341fa295",
                "2de20014-9a3b-4e26-b453-0de755144b1a",
                "8ed78aeb-5c9a-4980-b669-9e08b28671d2",
                "1f8e1c68-2c97-47b2-847c-a19619c02ca9",
                "9927e1d1-6018-4960-ac17-aadb0a94f4d9",
                "68f0ed8b-a887-42a5-b618-f68b494a6034",
                "fcba16e0-6bd7-4ea5-b86e-36e8330b68ea",
                "2250357c-ae34-4ed9-86da-45dac5a5e903",
                "c63ad40d-5a14-4646-a8df-2d4304213dbc",   // Cell
                "2b8417dc-e65f-4fcd-aa0f-61a23f1e8cb0",
                "26c62932-72f0-4dc2-9893-1ae27829c060",
                "27f291cc-abf4-4c23-aa24-676abe99cb1e",
                "7e5ce028-1bcd-4d9f-ad42-15ac181c5b47",
                "bf32e3d2-efa2-4fc1-b09b-ab9cc52ff734"
            ]
        }
    ],
    "BlockType": "TABLE",
    "Confidence": 99.99993896484375,
    "Id": "f66eac36-2e74-406e-8032-14d1c14e0b86"
}
```

CellBLOCK확인란이 포함된 셀에 대한 개체 (ID c63ad40d-5A14-4646-A8df-2d4304213dbc)서비스 좋음는 다음과 같습니다. 어린이가 포함되어 있습니다.Block(아이디 = 26d122fd-C5f4-4B53-92c4-0애92730e1e)SELECTION_ELEMENT Block확인란의 객체입니다.

```
{
```

```
    "Geometry": {.....},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "26d122fd-c5f4-4b53-92c4-0ae92730ee1e"  // Selection Element
            ]
        }
    ],
    "Confidence": 79.741689682006836,
    "RowSpan": 1,
    "RowIndex": 3,
    "ColumnIndex": 3,
    "ColumnSpan": 1,
    "BlockType": "CELL",
    "Id": "c63ad40d-5a14-4646-a8df-2d4304213dbc"
}
```

더 셀렉션_엘리먼트Block확인란의 객체는 다음과 같습니다. 의 가치SelectionStatus확인란이 선택되었음을 나타냅니다.

```
{
    "Geometry": {.......},
    "BlockType": "SELECTION_ELEMENT",
    "SelectionStatus": "SELECTED",
    "Confidence": 88.79517364501953,
    "Id": "26d122fd-c5f4-4b53-92c4-0ae92730ee1e"
}
```

송장 및 수금 응답 객체

인보이스 또는 영수증을 AnalyzeFreents API에 제출하면 일련의 ExpenseDocuments 객체를 반환합니다. 각 비용 문서는 다음과 같이 추가로 분리됩니다.LineItemGroups과SummaryFields. 대부분의 송장 및 영수증에는 공급업체 이름, 영수증 번호, 영수증 날짜 또는 총액과 같은 정보가 포함되어 있습니다. 애널리즈비용에서 이 정보를 반환합니다.SummaryFields. 영수증과 송장에는 구매한 품목에 대한 세부 정보도 포함되어 있습니다. 애널리즈경비 API는 이 정보를 아래에 반환합니다.LineItemGroups. 이ExpenseIndex필드는 비용을 고유하게 식별하고 적절한 비용을 연관시킵니다.SummaryFields과LineItemGroups해당 비용으로 감지되었습니다.

Analyze비용 응답에서 가장 세분화된 데이터 수준은 다음과 같이 구성됩니다.Type,ValueDetection, 및LabelDetection(선택). 개별 엔티티는 다음과 같습니다.

- 유형: 상위 수준에서 감지되는 정보의 종류를 나타냅니다.
- 라벨 감지: 문서 텍스트 내에서 연관된 값의 레이블을 참조합니다.LabelDetection은 선택 사항이며 레이블이 쓰여진 경우에만 반환됩니다.
- 가치 감지: 반환되는 레이블 또는 유형의 값을 나타냅니다.

애널리즈비용 API도 감지합니다.ITEM,QUANTITY, 및PRICE라인 항목 내에서 정규화된 필드로 SKU 또는 상세 설명과 같은 영수증 이미지의 라인 항목에 다른 텍스트가 있는 경우 JSON에 다음과 같이 포함됩니다.EXPENSE_ROW아래 예제와 같이:

```
{
                    "Type": {
                        "Text": "EXPENSE_ROW",
                        "Confidence": 99.95216369628906
                    },
                    "ValueDetection": {
                        "Text": "Banana 5 $2.5",
                        "Geometry": {
                          …
                        },
                        "Confidence": 98.11214447021484
                    }
```

위의 예에서는 AnalyzeFreent API가 영수증에서 2.5달러에 판매된 5개 바나나에 대한 라인 항목 정보가 포함된 전체 행을 반환하는 방법을 보여줍니다.

유형

다음은 키-값 페어의 표준 또는 정규화된 유형의 예입니다.

```
{
     "PageNumber": 1,
     "Type": {
         "Text": "VENDOR_NAME",
         "Confidence": 70.0
     },
     "ValueDetection": {
         "Geometry": { ... },
```

```
                "Text": "AMAZON",
                "Confidence": 87.89806365966797
            }
        }
```

영수증에 “공급업체 이름”이 명시적으로 나열되지 않았습니다. 그러나 비용 분석 API는 문서를 영수증으로 인식하고 “AMAZON” 값을 유형으로 분류했습니다.VENDOR_NAME.

라벨 감지

다음은 고객 문서 페이지에 표시된 텍스트의 예입니다.

```
        {
            "PageNumber": 1,
            "Type": {
                "Text": "OTHER",
                "Confidence": 70.0
            },
            "LabelDetection": {
                "Geometry": { ... },
                "Text": "CASHIER",
                "Confidence": 88.19171142578125
            },
            "ValueDetection": {
                "Geometry": { ... },
                "Text": "Mina",
                "Confidence": 87.89806365966797
            }
        }
```

예제 문서에는 “캐셔 미나”가 포함되어 있습니다. 비용 분석 API는 있는 그대로 값을 추출하고 다음 값을 반환합니다.LabelDetection. 영수증에 “키”가 명시적으로 표시되지 않는 “공급업체 이름”과 같은 묵시적 값의 경우LabelDetection은 (는) 분석 비용 요소에 포함되지 않습니다. 이 경우 애널리즈경비 API가 반환되지 않습니다.LabelDetection.

가치 감지

다음은 키-값 페어의 “value”를 보여 줍니다.

```
            {
                 "PageNumber": 1,
                 "Type": {
                     "Text": "OTHER",
                     "Confidence": 70.0
                 },
                 "LabelDetection": {
                     "Geometry": { ... },
                     "Text": "CASHIER",
                     "Confidence": 88.19171142578125
                 },
                 "ValueDetection": {
                     "Geometry": { ... },
                     "Text": "Mina",
                     "Confidence": 87.89806365966797
                 }
             }
```

이 예에서 문서에는 “캐셔 미나”가 포함되어 있습니다. 애널리즈경비 API는 계산원 값을 Mina로 감지하여 반환했습니다.`ValueDetection`.

ID 문서 응답 객체

AnalyzeId API에 ID 문서를 제출하면 일련의 파일이 반환됩니다.`IdentityDocumentField`객체입니다. 이러한 각 객체에는 다음이 포함됩니다.`Type`, 및`Value`.`Type`Amazon Textract Textract가 감지하는 정규화된 필드를 기록합니다.`Value`정규화된 필드와 연관된 텍스트를 기록합니다.

다음은 의 예입니다.`IdentityDocumentField`간결성을 위해 단축되었습니다.

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "IdentityDocumentFields": [
        {
            "Type": {
                "Text": "first name"
            },
            "ValueDetection": {
```

```
                "Text": "jennifer",
                "Confidence": 99.99908447265625
            }
        },
        {
            "Type": {
                "Text": "last name"
            },
            "ValueDetection": {
                "Text": "sample",
                "Confidence": 99.99758911132812
            }
        },
```

다음은 더 긴 응답에서 잘라낸 Identity문서 필드의 두 가지 예입니다. 감지된 형식과 해당 유형의 값 사이에는 분리되어 있습니다. 여기서는 각각 이름과 성입니다. 이 구조는 포함된 모든 정보와 함께 반복됩니다. 유형이 정규화된 필드로 인식되지 않으면 “기타”로 나열됩니다.

다음은 운전 면허증에 대한 정규화된 필드 목록입니다.

- 이름
- 성
- Middle
- 접미사
- 주소 도시
- 주소에 있는 우편 번호
- 주소 상태
- 카운티
- 문서 번호
- 만료 날짜
- 생년월일
- 상태 이름
- 발행일
- class
- 제한 사항
- 보증

- ID 유형
- 재향 군인
- address

다음은 미국 여권에 대한 정규화된 필드 목록입니다.

- 이름
- 성
- Middle
- 문서 번호
- 만료 날짜
- 생년월일
- 출생지
- 발행일
- ID 유형

문서 페이지의 항목 위치

Amazon Textract 작업은 문서 페이지에 있는 항목의 위치와 형상을 반환합니다.DetectDocumentText과GetDocumentTextDetection선과 단어의 위치 및 지오메트리를 반환하는 동안AnalyzeDocument과GetDocumentAnalysis키-값 페어, 테이블, 셀 및 선택 요소의 위치 및 형상을 반환합니다.

문서 페이지에서 항목이 있는 위치를 확인하려면 테두리 상자 (Geometry) Amazon Textract 작업에서 반환한 정보Block객체입니다. 이Geometryobject는 탐지된 항목에 대한 두 가지 유형의 위치 및 형상 정보를 포함합니다.

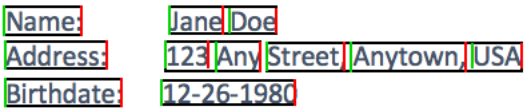
- 축 정렬BoundingBox왼쪽 위 좌표와 항목의 너비와 높이가 포함된 객체입니다.
- 항목의 아웃라인을 설명하는 폴리곤 객체로, 배열로 지정됩니다.Point다음을 포함하는 객체X(가로 축) 및Y(세로 축) 각 점의 페이지 좌표를 문서화합니다.

a를 위한 JSONBlock객체는 다음과 비슷합니다. 참고BoundingBox과Polygon필드.

```
{
    "Geometry": {
```

```
        "BoundingBox": {
            "Width": 0.053907789289951324,
            "Top": 0.08913730084896088,
            "Left": 0.11085548996925354,
            "Height": 0.013171200640499592
        },
        "Polygon": [
            {
                "Y": 0.08985357731580734,
                "X": 0.11085548996925354
            },
            {
                "Y": 0.08913730084896088,
                "X": 0.16447919607162476
            },
            {
                "Y": 0.10159222036600113,
                "X": 0.16476328670978546
            },
            {
                "Y": 0.10230850428342819,
                "X": 0.11113958805799484
            }
        ]
    },
    "Text": "Name:",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.56285858154297,
    "Id": "c734fca6-c4c4-415c-b6c1-30f7510b72ee"
},
```


형상 정보를 사용하여 탐지된 항목 주위에 경계 상자를 그릴 수 있습니다. 를 사용하는 예제의 경우BoundingBox과Polygon각 단어의 시작과 끝에 선 및 수직선 주위에 상자를 그리는 정보는Amazon Textract Textract를 사용하여 문서 텍스트 감지. 예제 출력은 다음과 비슷합니다.



Bounding Box

경계 상자 (BoundingBox) 에는 다음과 같은 속성이 있습니다.

- 높이 — 전체 문서 페이지 높이에 대한 비율로서 경계 상자의 높이입니다.
- Left — 전체 문서 페이지 너비에 대한 비율로서 경계 상자의 상단 좌측 포인트에 대한 X 좌표입니다.
- Top — 전체 문서 페이지 높이에 대한 비율로서 경계 상자의 상단 지점에 대한 Y 좌표입니다.
- 너비 — 전체 문서 페이지 너비에 대한 비율로서 경계 상자의 너비입니다.

각 BoundingBox 속성은 0~1 사이의 값을 가집니다. 값은 전체 이미지 너비에 대한 비율입니다.`Left`과`Width`) 또는 높이 (적용`Height`과`Top`). 예를 들어 입력 이미지가 700x200픽셀이고 경계 상자의 상단 좌측 좌표가 (350,50) 픽셀일 경우 API는`Left`값 0.5 (350/700) 및`Top`값은 0.25 (50/200) 입니다.

다음 다이어그램은 각 BoundingBox 속성이 다루는 문서 페이지의 범위를 보여 줍니다.

경계 상자를 올바른 위치 및 크기로 표시하려면, (원하는 값에 따라) BoundingBox 값을 문서 페이지 너비 또는 높이에 곱해 픽셀 값을 구해야 합니다. 이 픽셀 값을 사용하여 경계 상자를 표시합니다. 한 예로 608픽셀 너비 x 588픽셀 높이의 문서 페이지와 분석된 텍스트에 대해 다음 경계 상자 값을 사용하는 것입니다.

```
BoundingBox.Left: 0.3922065
BoundingBox.Top: 0.15567766
BoundingBox.Width: 0.284666
BoundingBox.Height: 0.2930403
```

텍스트 경계 상자의 위치는 다음과 같이 계산됩니다.

```
Left coordinate = BoundingBox.Left (0.3922065) * document page width (608)
= 238
```

```
Top coordinate = BoundingBox.Top (0.15567766) * document page height (588)
= 91
```

```
Bounding box width = BoundingBox.Width (0.284666) * document page width
(608) = 173
```

```
Bounding box height = BoundingBox.Height (0.2930403) * document page height
(588) = 172
```

이러한 값을 사용하여 분석된 텍스트 주위에 경계 상자를 표시합니다. 다음 Java 및 Python 예제에서는 경계 상자를 표시하는 방법을 보여줍니다.

Java

```
   public void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox box,
Graphics2D g2d) {

       float left = imageWidth * box.getLeft();
       float top = imageHeight * box.getTop();

       // Display bounding box.
       g2d.setColor(new Color(0, 212, 0));
       g2d.drawRect(Math.round(left / scale), Math.round(top / scale),
               Math.round((imageWidth * box.getWidth()) / scale),
Math.round((imageHeight * box.getHeight())) / scale);

   }
```

Python

이 파이썬 예제는`response`에서 반환한DetectDocumentTextAPI 연산.

```
def process_text_detection(response):

    # Get the text blocks
    blocks = response['Blocks']
    width, height = image.size
    draw = ImageDraw.Draw(image)
    print('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:

        draw = ImageDraw.Draw(image)

        if block['BlockType'] == "LINE":
            box=block['Geometry']['BoundingBox']
            left = width * box['Left']
            top = height * box['Top']
            draw.rectangle([left,top, left + (width * box['Width']), top +(height *
box['Height'])],outline='black')

    # Display the image
    image.show()
```

```
    return len(blocks)
```

다각형

에 의해 반환된 폴리곤AnalyzeDocument는 다음과 같은 배열입니다.Point객체입니다. EachPoint에는 문서 페이지의 특정 위치에 대한 X 및 Y 좌표가 있습니다. BoundingBox 좌표와 마찬가지로 다각형 좌표는 문서 너비와 높이로 정규화되고 0에서 1 사이입니다.

폴리곤 배열의 점을 사용하여Block객체입니다. 문서 페이지에서 각 다각형 점의 위치를 계산하는 방법은 다음과 같습니다.BoundingBoxes. X 좌표에 문서 페이지 너비를 곱하고 Y 좌표에 문서 페이지 높이를 곱합니다.

다음 예제에서는 다각형의 수직선을 표시하는 방법을 보여 줍니다.

```
   public void ShowPolygonVerticals(int imageHeight, int imageWidth, List <Point>
points, Graphics2D g2d) {

       g2d.setColor(new Color(0, 212, 0));
       Object[] parry = points.toArray();
       g2d.setStroke(new BasicStroke(2));

       g2d.drawLine(Math.round(((Point) parry[0]).getX() * imageWidth),
               Math.round(((Point) parry[0]).getY() * imageHeight),
Math.round(((Point) parry[3]).getX() * imageWidth),
               Math.round(((Point) parry[3]).getY() * imageHeight));

       g2d.setColor(new Color(255, 0, 0));
       g2d.drawLine(Math.round(((Point) parry[1]).getX() * imageWidth),
               Math.round(((Point) parry[1]).getY() * imageHeight),
Math.round(((Point) parry[2]).getX() * imageWidth),
               Math.round(((Point) parry[2]).getY() * imageHeight));

   }
```

Amazon Textract 시작하기

이 단원에서는 Amazon Textract 사용을 시작하는 방법을 알아봅니다. Amazon Textract 을 처음 사용하는 경우, 먼저 의 개념과 용어를 검토하는 것이 좋습니다.Amazon Textract 작동 방식.

Amazon Textract 콘솔의 데모를 사용하여 API를 사용해 볼 수 있습니다. 자세한 내용은 단원을 참조하십시오.https://console.aws.amazon.com/textract/.

주제

- 1단계: AWS 계정 설정 및 IAM 사용자 만들기
- 2단계: 설정AWS CLI과AWSSDK
- 3단계: 사용 시작하기AWS CLI과AWSSDK API

1단계: AWS 계정 설정 및 IAM 사용자 만들기

Amazon Textract 을 처음 사용하기 전에 다음 작업을 완료해야 합니다.

1. AWS에 가입.
2. IAM 사용자 생성.

AWS에 가입

Amazon Web Services (AWS) 에 가입하면 AWS에서 릴리스된 모든 서비스에 AWS 계정이 자동 등록됩니다. 사용한 서비스에 대해서만 청구됩니다.

Amazon Textract 을 사용하면 사용한 리소스에 대해서만 비용을 지불합니다. Amazon Textract 사용 요금에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract 요금. AWS를 처음 사용하는 고객인 경우 Amazon Textract 을 무료로 시작할 수 있습니다. 자세한 내용은 AWS 프리 티어를 참조하십시오.

이미 AWS 계정이 있다면 다음 작업으로 건너뛰십시오. AWS 계정이 없는 경우에는 다음 절차의 단계를 수행하여 계정을 만듭니다.

AWS 계정을 만들려면 다음을 수행합니다.

1. https://portal.aws.amazon.com/billing/signup을 엽니다.

2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화를 받고 전화 키패드를 사용하여 확인 코드를 입력하는 과정이 있습니다.

다음 작업에 필요하므로 AWS 계정 ID를 기록합니다.

IAM 사용자 생성

Amazon Textract 같은 AWS 서비스를 사용하려면 액세스할 때 자격 증명을 제공해야 합니다. 이를 통해 서비스가 소유한 리소스에 액세스할 수 있는 권한이 있는지를 확인합니다. 콘솔은 암호를 요구합니다. AWS 계정에 대한 액세스 키를 생성하면 AWS CLI 또는 API에 액세스할 수 있습니다. 그러나 AWS 계정용 자격 증명을 사용하여 AWS에 액세스하는 것은 권장되지 않습니다. 그 대신 다음과 같이 하는 것이 좋습니다.

- 사용AWS Identity and Access ManagementIAM 사용자를 생성하려면 (IAM) 입니다.
- 관리 권한을 가진 IAM 그룹에 사용자를 추가합니다.

그러면 특정 URL이나 그 IAM 사용자의 자격 증명을 사용하여 AWS에 액세스할 수 있습니다.

AWS에 가입했지만 IAM 사용자를 만들지 않았다면 IAM 콘솔을 사용하여 IAM 사용자를 만들 수 있습니다. 절차에 따라 계정에 IAM 사용자를 만듭니다.

IAM 사용자를 만들고 콘솔에 로그인하려면

1. AWS 계정에서 관리자 권한을 가진 IAM 사용자를 만듭니다. 지침은 단원을 참조하십시오.IAM 사용자와 관리자 그룹 처음 만들기의IAM 사용 설명서.
2. IAM 사용자로 특정 URL을 사용하여 AWS Management Console에 로그인합니다. 자세한 내용은 단원을 참조하십시오.사용자의 계정 로그인 방법의IAM 사용 설명서.

> **Note**
>
> 관리자 권한을 가진 IAM 사용자는 다음 작업에 제한 없이 액세스할 수 있습니다.AWS계정에 있는 서비스. 이 가이드의 코드 예제에서는 다음과 같은 사용자가 있다고 가정합니다.`AmazonTextractFullAccess`권한.`AmazonS3ReadOnlyAccess`Amazon S3 버킷에 저장된 문서에 액세스하는 예제에서는 이 필요합니다. 보안 요구 사항에 따라 이러한 권한으로 제한한 IAM 그룹을 사용할 수 있습니다. 자세한 내용은 단원을 참조하십시오.IAM 그룹 생성.

IAM에 대한 자세한 내용은 다음을 참조하세요.

- AWS Identity and Access Management(IAM)
- 시작하기
- IAM 사용 설명서

다음 단계

2단계: 설정AWS CLI과AWSSDK

2단계: 설정AWS CLI과AWSSDK

다음 단계는 이 설명서의 예제에서 사용하는 AWS Command Line Interface(AWS CLI)와 AWS SDK를 설치하는 방법을 보여줍니다.

AWS SDK 호출을 인증하는 방법에는 여러 가지가 있습니다. 이 가이드의 예제에서는 AWS CLI 명령과 AWS SDK API 작업을 호출하기 위해 기본 자격 증명 프로파일을 사용합니다. 기본 자격 증명은 여러 서비스에서 작동하므로 자격 증명을 이미 구성한 경우 다시 설정할 필요가 없습니다. 그러나 이 서비스에 대한 다른 자격 증명 집합을 만들려면 이름 프로필을 만들 수 있습니다. 프로파일 생성에 대한 자세한 내용은 다음을 참조하십시오.명된 프로필을 참조하십시오.

사용 가능한 목록은 다음과 같습니다.AWS지역, 참조리전 및 엔드포인트의Amazon Web Services 일반 참조.

AWS CLI와 AWS SDK를 설정하려면

1. 사용하려는 AWS CLI와 AWS SDK를 다운로드하여 설치합니다. 이 가이드에서는 다음과 같은 예제를 제공합니다.AWS CLI, 자바, 파이썬입니다. 그 밖의 AWS SDK에 대한 자세한 정보는 Amazon Web Services용 도구를 참조하십시오.
   - AWS CLI
   - AWS SDK for Java
   - AWS SDK for Python (Boto3)
2. 에서 만든 사용자의 액세스 키를 만듭니다.IAM 사용자 생성.
   a. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

b. 탐색 창에서 사용자를 선택합니다.

c. 에서 만든 사용자의 이름을 선택합니다.IAM 사용자 생성.

d. Security credentials(보안 자격 증명) 탭을 선택합니다.

e. 액세스 키 생성을 선택합니다. 그런 다음 .csv 파일 다운로드를 선택하여 액세스 키 ID 및 보안 액세스 키를 컴퓨터에 CSV 파일로 저장합니다. 안전한 위치에 파일을 저장합니다. 이 대화 상자를 닫은 후에는 보안 액세스 키에 다시 액세스할 수 없습니다. CSV 파일을 다운로드한 후 다음을 선택합니다.Close.

3. 로컬 시스템의 다음 위치에 있는 AWS 자격 증명 프로필 파일에서 자격 증명을 설정합니다.

- `~/.aws/credentials`Linux, macOS 또는 Unix.
- `C:\Users\USERNAME\.aws\credentials`Windows의 경우.

이.awsAWS 인스턴스를 처음 처음 구성하기 전에는 폴더가 존재하지 않습니다. CLI를 사용하여 자격 증명을 처음 구성하면 이 폴더가 생성됩니다. AWS 자격 증명에 대한 자세한 내용은 단원을 참조하십시오.구성 및 자격 증명 파일 설정.

이 파일에는 다음 형식의 행이 포함되어야 합니다.

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

다음 주소로 액세스 키 ID와 보안 액세스 키를 대체합니다.너_액세스_키_id과너_비밀_접근_키.

4. AWS에서 기본 AWS 리전 설정config로컬 시스템의 파일은 다음 위치에 있습니다.

- `~/.aws/config`Linux, macOS 또는 Unix.
- `C:\Users\USERNAME\.aws\config`Windows의 경우.

이.awsAWS 인스턴스를 처음 처음 구성하기 전에는 폴더가 존재하지 않습니다. CLI를 사용하여 자격 증명을 처음 구성하면 이 폴더가 생성됩니다. AWS 자격 증명에 대한 자세한 내용은 단원을 참조하십시오.구성 및 자격 증명 파일 설정.

이 파일에는 다음 행이 포함되어야 합니다.

```
[default]
region = your_aws_region
```

원하는 AWS 리전 (예를 들면 “us-west-2") 으로 대체합니다.YOUR_aws_리전.

> **Note**
>
> 리전을 선택하지 않으면 기본적으로 us-east-1이 사용됩니다.

다음 단계

3단계: 사용 시작하기AWS CLI과AWSSDK API

3단계: 사용 시작하기AWS CLI과AWSSDK API

를 설정한 후AWS CLI과AWSAmazon Textract 을 사용하는 애플리케이션을 빌드할 수 있습니다. 다음 주제에서는 Amazon Textract 을 시작하는 방법을 보여 줍니다.

- Amazon Textract Textract를 사용하여 문서 텍스트 분석

AWS CLI 예제 형식

이 안내서의 AWS CLI 예제는 Linux 운영 체제용입니다. Microsoft Windows에서 샘플을 사용하려면 `--document` 파라미터의 JSON 형식을 변경하고 줄 바꿈을 백슬래시(\)에서 캐럿 기호(^)로 변경해야 합니다. JSON 형식에 대한 자세한 내용은 AWS 명령줄 인터페이스 파라미터 값 지정을 참조하십시오.

동기식 작업을 통한 문서 처리

Amazon Textract Textract는 JPEG, PNG, PDF 및 TIFF 형식의 이미지로 제공되는 단일 페이지 문서의 텍스트를 감지하고 분석할 수 있습니다. 작업은 동기식이며 거의 실시간으로 결과를 반환합니다. 문서에 대한 자세한 정보는 텍스트 감지 및 문서 분석 응답 객체 섹션을 참조하세요.

이 단원에서는 Amazon Textract Textract를 사용하여 단일 페이지 문서의 텍스트를 동기적으로 감지하고 분석하는 방법에 대해 설명합니다. 여러 페이지로 된 문서에서 텍스트를 감지 및 분석하거나 JPEG 및 PNG 문서를 비동기적으로 감지하려면비동기 작업을 사용한 문서 처리.

다음과 같은 목적으로 Amazon Textract 동기 작업을 사용할 수 있습니다.

- 텍스트 감지 — 단일 페이지 문서 이미지에서 줄 및 단어를 검색할 수 있습니다.DetectDocumentText작업을 수행합니다. 자세한 정보는 텍스트 감지을 참조하십시오.
- 텍스트 분석 — 단일 페이지 문서에서 검색된 텍스트 간의 관계를 식별할 수 있습니다.AnalyzeDocument작업을 수행합니다. 자세한 정보는 문서 분석을 참조하십시오.
- 송장 및 수금 분석 — AnalyzeFement 작업을 사용하여 단일 페이지 송장에서 감지된 텍스트 또는 수금 간의 재무 관계를 식별할 수 있습니다. 자세한 내용은 단원을 참조하십시오.송장 및 수금 분석
- 신원 문서 분석 — 미국 정부가 발행한 신분 증명 문서를 분석하고 신분 증명서에 있는 일반적인 유형의 정보와 함께 정보를 추출할 수 있습니다. 자세한 내용은 를 참조하십시오.ID 문서 분석.

주제

- Amazon Textract 동기 운영 호출
- Amazon Textract Textract를 사용하여 문서 텍스트 감지
- Amazon Textract Textract를 사용하여 문서 텍스트 분석
- Amazon Textract Textract를 사용하여 송장 및 영수증 분석
- Amazon Textract Textract를 사용하여 자격 증명 문서 분석

Amazon Textract 동기 운영 호출

Amazon Textract 작업은 로컬 파일 시스템에 저장된 문서 이미지 또는 Amazon S3 버킷에 저장된 문서 이미지를 처리합니다. 다음을 사용하여 입력 문서의 위치를 지정합니다.Document매개 변수를 입력합니다. 문서 이미지는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다. 동기 작업에 대한 결과는 즉시 반환되며 검색을 위해 저장되지 않습니다.

전체 예제는 단원을 참조하십시오.Amazon Textract Textract를 사용하여 문서 텍스트 감지.

요청

다음은 Amazon Textract Textract에서 요청이 작동하는 방식에 대해 설명합니다.

이미지 바이트로 전달된 문서

이미지를 base64로 인코딩된 바이트 배열로 전달하여 문서 이미지를 Amazon Textract 작업에 전달할 수 있습니다. 로컬 파일 시스템에서 로드된 문서 이미지를 예로 들 수 있습니다. 를 사용하는 경우 코드가 문서 파일 바이트를 인코딩할 필요가 없을 수도 있습니다.AWSAmazon Textract API 작업을 호출하는 SDK입니다.

이미지 바이트는Bytes의 필드Document매개 변수를 입력합니다. 다음 예제에서는 에서 이미지 바이트를 전달하는 Amazon Textract 작업에 대한 입력 JSON을 보여 줍니다.Bytes매개 변수를 입력합니다.

```
{
    "Document": {
        "Bytes": "/9j/4AAQSk....."
    }
}
```

> **Note**
>
> 사용 중인AWS CLI, Amazon Textract 작업에 이미지 바이트를 전달할 수 없습니다. 대신 Amazon S3 버킷에 저장된 이미지를 참조해야 합니다.

다음 Java 코드는 로컬 파일 시스템에서 이미지를 로드하고 Amazon Textract 작업을 호출하는 방법을 보여줍니다.

```
String document="input.png";

ByteBuffer imageBytes;
try (InputStream inputStream = new FileInputStream(new File(document))) {
    imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
}
AmazonTextract client = AmazonTextractClientBuilder.defaultClient();
```

```
DetectDocumentTextRequest request = new DetectDocumentTextRequest()
        .withDocument(new Document()
                .withBytes(imageBytes));


DetectDocumentTextResult result = client.detectDocumentText(request);
```

Amazon S3 버킷에 저장된 문서

Amazon Textract TExtract는 Amazon S3 버킷에 저장된 문서 이미지를 분석할 수 있습니다. 을 사용하여 버킷 및 파일 이름을 지정합니다.S3Object의 필드Document매개 변수를 입력합니다. 다음 예제에서는 Amazon S3 버킷에 저장된 문서를 처리하는 Amazon Textract 작업에 대한 입력 JSON을 보여 줍니다.

```
{
    "Document": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.png"
        }
    }
}
```

다음 예제에서는 Amazon S3 버킷에 저장된 이미지를 사용하여 Amazon Textract 작업을 호출하는 방법을 보여줍니다.

```
String document="input.png";
String bucket="bucket";

AmazonTextract client = AmazonTextractClientBuilder.defaultClient();

DetectDocumentTextRequest request = new DetectDocumentTextRequest()
        .withDocument(new Document()
                .withS3Object(new S3Object()
                        .withName(document)
                        .withBucket(bucket)));

DetectDocumentTextResult result = client.detectDocumentText(request);
```

응답

다음 예제는 호출로부터의 JSON 응답입니다.DetectDocumentText. 자세한 정보는 텍스트 감지을 참조하십시오.


```
{
{
  "DocumentMetadata": {
    "Pages": 1
  },
  "Blocks": [
    {
      "BlockType": "PAGE",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.9995205998420715,
          "Height": 1.0,
          "Left": 0.0,
          "Top": 0.0
        },
        "Polygon": [
          {
            "X": 0.0,
            "Y": 0.0
          },
          {
            "X": 0.9995205998420715,
            "Y": 2.297314024515845E-16
          },
          {
            "X": 0.9995205998420715,
            "Y": 1.0
          },
          {
            "X": 0.0,
            "Y": 1.0
          }
        ]
      },
      "Id": "ca4b9171-7109-4adb-a811-e09bbe4834dd",
      "Relationships": [
        {
          "Type": "CHILD",
```

```
        "Ids": [
          "26085884-d005-4144-b4c2-4d83dc50739b",
          "ee9d01bc-d91c-401d-8c0a-eec76f5f7862",
          "404bb3d3-d7ab-4008-a195-5dec87a08664",
          "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
          "47aab5ab-be2c-4c73-97c7-d0a45454e843",
          "dd06bb49-6a56-4ea7-beec-a2aa09835c3c",
          "8837153d-81b8-4031-a49f-83a3d81803c2",
          "5dae3b74-9e95-4b62-99b7-93b88fe70648",
          "4508da80-64d8-42a8-8846-cfafe6eab10c",
          "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
          "f04bb223-d075-41c3-b328-7354611c826b",
          "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
          "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
          "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
          "359f3870-7183-43f5-b638-970f5cefe4d5",
          "b9deea0a-244c-4d54-b774-cf03fbaaa8b1",
          "e2a43881-f620-44f2-b067-500ce7dc8d4d",
          "41756974-64ef-432d-b4b2-34702505975a",
          "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
          "bc907357-63d6-43c0-ab87-80d7e76d377e",
          "2d727ca7-3acb-4bb9-a564-5885c90e9325",
          "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
          "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
          "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
          "ac4b9ee0-c9b2-4239-a741-5753e5282033",
          "ebc18885-48d7-45b8-90e3-d172b4357802",
          "babf6360-789e-49c1-9c78-0784acc14a0c"
        ]
      }
    ]
  },
  {
    "BlockType": "LINE",
    "Confidence": 99.93761444091797,
    "Text": "Employment Application",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.3391372561454773,
        "Height": 0.06906412541866302,
        "Left": 0.29548385739326477,
        "Top": 0.027493247762322426
      },
      "Polygon": [
```

```
          {
            "X": 0.29548385739326477,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.0965573713183403
          },
          {
            "X": 0.29548385739326477,
            "Y": 0.0965573713183403
          }
        ]
      },
      "Id": "26085884-d005-4144-b4c2-4d83dc50739b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "ed48dacc-d089-498f-8e93-1cee1e5f39f3",
            "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.91246795654297,
      "Text": "Application Information",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.19878505170345306,
          "Height": 0.03754019737243652,
          "Left": 0.03988289833068848,
          "Top": 0.14050349593162537
        },
        "Polygon": [
          {
            "X": 0.03988289833068848,
            "Y": 0.14050349593162537
```

```
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.1780436933040619
          },
          {
            "X": 0.03988289833068848,
            "Y": 0.1780436933040619
          }
        ]
      },
      "Id": "ee9d01bc-d91c-401d-8c0a-eec76f5f7862",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "efe3fc6d-becb-4520-80ee-49a329386aee",
            "c2260852-6cfd-4a71-9fc6-62b2f9b02355"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.88693237304688,
      "Text": "Full Name: Jane Doe",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16733919084072113,
          "Height": 0.031106337904930115,
          "Left": 0.03899926319718361,
          "Top": 0.21361036598682404
        },
        "Polygon": [
          {
            "X": 0.03899926319718361,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.20633845031261444,
```

```
            "Y": 0.21361036598682404
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.24471670389175415
          },
          {
            "X": 0.03899926319718361,
            "Y": 0.24471670389175415
          }
        ]
      },
      "Id": "404bb3d3-d7ab-4008-a195-5dec87a08664",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "e94eb587-9545-4215-b0fc-8e8cb1172958",
            "090aeba5-8428-4b7a-a54b-7a95a774120e",
            "64ff0abb-736b-4a6b-aa8d-ad2c0086ae1d",
            "565ffc30-89d6-4295-b8c6-d22b4ed76584"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9206314086914,
      "Text": "Phone Number: 555-0100",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3115004599094391,
          "Height": 0.047169625759124756,
          "Left": 0.03604753687977791,
          "Top": 0.2812676727771759
        },
        "Polygon": [
          {
            "X": 0.03604753687977791,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.2812676727771759
```

```
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.32843729853630066
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.32843729853630066
          }
        ]
      },
      "Id": "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "d782f847-225b-4a1b-b52d-f252f8221b1f",
            "fa69c5cd-c80d-4fac-81df-569edae8d259",
            "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.48902893066406,
      "Text": "Home Address: 123 Any Street, Any Town. USA",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.7431139945983887,
          "Height": 0.09577702730894089,
          "Left": 0.03359385207295418,
          "Top": 0.3258342146873474
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.3258342146873474
          },
          {
```

```
            "X": 0.7767078280448914,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "47aab5ab-be2c-4c73-97c7-d0a45454e843",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "acfbed90-4a00-42c6-8a90-d0a0756eea36",
            "046c8a40-bb0e-4718-9c71-954d3630e1dd",
            "82b838bc-4591-4287-8dea-60c94a4925e4",
            "5cdcde7a-f5a6-4231-a941-b6396e42e7ba",
            "beafd497-185f-487e-b070-db4df5803e94",
            "ef1b77fb-8ba6-41fe-ba53-dce039af22ed",
            "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e",
            "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.89382934570312,
      "Text": "Mailing Address: same as above",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.26575741171836853,
          "Height": 0.039571404457092285,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
            "X": 0.03068041242659092,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.2964377999305725,
```

```
            "Y": 0.43351811170578003
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4730895161628723
          },
          {
            "X": 0.03068041242659092,
            "Y": 0.4730895161628723
          }
        ]
      },
      "Id": "dd06bb49-6a56-4ea7-beec-a2aa09835c3c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "d7261cdc-6ac5-4711-903c-4598fe94952d",
            "287f80c3-6db2-4dd7-90ec-5f017c80aa31",
            "ce31c3ad-b51e-4068-be64-5fc9794bc1bc",
            "e96eb92c-6774-4d6f-8f4a-68a7618d4c66",
            "88b85c05-427a-4d4f-8cc4-3667234e8364"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 94.67343139648438,
      "Text": "Previous Employment History",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3309842050075531,
          "Height": 0.051920413970947266,
          "Left": 0.3194798231124878,
          "Top": 0.5172380208969116
        },
        "Polygon": [
          {
            "X": 0.3194798231124878,
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
```

```
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
            "Y": 0.5691584348678589
          },
          {
            "X": 0.3194798231124878,
            "Y": 0.5691584348678589
          }
        ]
      },
      "Id": "8837153d-81b8-4031-a49f-83a3d81803c2",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "8b324501-bf38-4ce9-9777-6514b7ade760",
            "b0cea99a-5045-464d-ac8a-a63ab0470995",
            "b92a6ee5-ca59-44dc-9c47-534c133b11e7"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.66949462890625,
      "Text": "Start Date",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08310240507125854,
          "Height": 0.030944595113396645,
          "Left": 0.034429505467414856,
          "Top": 0.6123942136764526
        },
        "Polygon": [
          {
            "X": 0.034429505467414856,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319030880928,
            "Y": 0.6123942136764526
          },
```

```
          {
            "X": 0.1175319030880928,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
          }
        ]
      },
      "Id": "5dae3b74-9e95-4b62-99b7-93b88fe70648",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45",
            "91e582cd-9871-4e9c-93cc-848baa426338"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86717224121094,
      "Text": "End Date",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07581500709056854,
          "Height": 0.03223184868693352,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
```

```
          },
          {
            "X": 0.14846202731132507,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "4508da80-64d8-42a8-8846-cfafe6eab10c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "7c97b56b-699f-49b0-93f4-98e6d90b107c",
            "7af04e27-0c15-447e-a569-b30edb99a133"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9539794921875,
      "Text": "Employer Name",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1347292959690094,
          "Height": 0.0392492413520813,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.3994368314743042,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.3994368314743042,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
```

```
            "Y": 0.6533204317092896
          }
        ]
      },
      "Id": "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9bfeb55-75cd-47cd-b953-728e602a3564",
            "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.35584259033203,
      "Text": "Position Held",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11393272876739502,
          "Height": 0.03415105864405632,
          "Left": 0.49973347783088684,
          "Top": 0.614840030670166
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.6489911079406738
          },
          {
            "X": 0.49973347783088684,
            "Y": 0.6489911079406738
          }
        ]
```

```
      },
      "Id": "f04bb223-d075-41c3-b328-7354611c826b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "6d5edf02-845c-40e0-9514-e56d0d652ae0",
            "3297ab59-b237-45fb-ae60-a108f0c95ac2"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9817886352539,
      "Text": "Reason for leaving",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16511960327625275,
          "Height": 0.04062700271606445,
          "Left": 0.7430596351623535,
          "Top": 0.6116235852241516
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6522505879402161
          }
        ]
      },
      "Id": "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
      "Relationships": [
```

```
        {
          "Type": "CHILD",
          "Ids": [
            "f4b8cf26-d2da-4a76-8345-69562de3cc11",
            "386d4a63-1194-4c0e-a18d-4d074a0b1f93",
            "a8622541-1896-4d54-8d10-7da2c800ec5c"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.77413177490234,
      "Text": "1/15/2009",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08799663186073303,
          "Height": 0.03832906484603882,
          "Left": 0.03175082430243492,
          "Top": 0.691371738910675
        },
        "Polygon": [
          {
            "X": 0.03175082430243492,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.7297008037567139
          },
          {
            "X": 0.03175082430243492,
            "Y": 0.7297008037567139
          }
        ]
      },
      "Id": "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
      "Relationships": [
        {
          "Type": "CHILD",
```

```
          "Ids": [
            "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.72286224365234,
      "Text": "6/30/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843101561069489,
          "Height": 0.03991425037384033,
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
          }
        ]
      },
      "Id": "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
          ]
        }
```

```
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86936950683594,
      "Text": "Any Company",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11800950765609741,
          "Height": 0.03943679481744766,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
          {
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.736709475517273
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.736709475517273
          }
        ]
      },
      "Id": "359f3870-7183-43f5-b638-970f5cefe4d5",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "77749c2b-aa7f-450e-8dd2-62bcaf253ba2",
            "713bad19-158d-4e3e-b01f-f5707ddb04e5"
          ]
        }
      ]
    },
    {
```

```
      "BlockType": "LINE",
      "Confidence": 99.582275390625,
      "Text": "Assistant baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.13280922174453735,
          "Height": 0.032666124403476715,
          "Left": 0.49814170598983765,
          "Top": 0.699238657951355
        },
        "Polygon": [
          {
            "X": 0.49814170598983765,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.7319048047065735
          },
          {
            "X": 0.49814170598983765,
            "Y": 0.7319048047065735
          }
        ]
      },
      "Id": "b9deea0a-244c-4d54-b774-cf03fbaaa8b1",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "989944f9-f684-4714-87d8-9ad9a321d65c",
            "ae82e2aa-1601-4e0c-8340-1db7ad0c9a31"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
```

```
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994903564453,
          "Height": 0.033302485942840576,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "e2a43881-f620-44f2-b067-500ce7dc8d4d",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98190307617188,
      "Text": "7/1/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067441940307617,
```

```
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.8451982140541077
          },
          {
            "X": 0.028500309213995934,
            "Y": 0.8451982140541077
          }
        ]
      },
      "Id": "41756974-64ef-432d-b4b2-34702505975a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0f711065-1872-442a-ba6d-8fababaa452a"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98418426513672,
      "Text": "8/10/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10664612054824829,
          "Height": 0.06439518928527832,
          "Left": 0.14159755408763885,
          "Top": 0.7791688442230225
        },
        "Polygon": [
```

```
          {
            "X": 0.14159755408763885,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.8435640335083008
          },
          {
            "X": 0.14159755408763885,
            "Y": 0.8435640335083008
          }
        ]
      },
      "Id": "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a92d8eef-db28-45ba-801a-5da0f589d277"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98075866699219,
      "Text": "Example Corp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.2114926278591156,
          "Height": 0.058415766805410385,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
        "Polygon": [
          {
            "X": 0.26764172315597534,
            "Y": 0.794414758682251
          },
```

```
          {
            "X": 0.47913435101509094,
            "Y": 0.794414758682251
          },
          {
            "X": 0.47913435101509094,
            "Y": 0.8528305292129517
          },
          {
            "X": 0.26764172315597534,
            "Y": 0.8528305292129517
          }
        ]
      },
      "Id": "bc907357-63d6-43c0-ab87-80d7e76d377e",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "d6962efb-34ab-4ffb-9f2f-5f263e813558",
            "1876c8ea-d3e8-4c39-870e-47512b3b5080"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.91166687011719,
      "Text": "Baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09931200742721558,
          "Height": 0.06008726358413696,
          "Left": 0.5098910331726074,
          "Top": 0.787897527217865
        },
        "Polygon": [
          {
            "X": 0.5098910331726074,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
```

```
          },
          {
            "X": 0.609203040599823,
            "Y": 0.847984790802002
          },
          {
            "X": 0.5098910331726074,
            "Y": 0.847984790802002
          }
        ]
      },
      "Id": "2d727ca7-3acb-4bb9-a564-5885c90e9325",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "00adeaef-ed57-44eb-b8a9-503575236d62"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.93852233886719,
      "Text": "better opp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18919607996940613,
          "Height": 0.06994765996932983,
          "Left": 0.7428008317947388,
          "Top": 0.7928366661071777
        },
        "Polygon": [
          {
            "X": 0.7428008317947388,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
            "Y": 0.8627843260765076
```

```
          },
          {
            "X": 0.7428008317947388,
            "Y": 0.8627843260765076
          }
        ]
      },
      "Id": "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "c0fc9a58-7a4b-4f69-bafd-2cff32be2665",
            "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.92573547363281,
      "Text": "8/15/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10257463902235031,
          "Height": 0.05412459373474121,
          "Left": 0.027909137308597565,
          "Top": 0.8608770370483398
        },
        "Polygon": [
          {
            "X": 0.027909137308597565,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.915001630783081
          },
          {
            "X": 0.027909137308597565,
```

```
            "Y": 0.915001630783081
          }
        ]
      },
      "Id": "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "5384f860-f857-4a94-9438-9dfa20eed1c6"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.99625396728516,
      "Text": "Present",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888341903686523,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
          {
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
```

```
      "Id": "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611276149749756,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
      },
      "Id": "ac4b9ee0-c9b2-4239-a741-5753e5282033",
      "Relationships": [
        {
          "Type": "CHILD",
```

```
          "Ids": [
            "25343360-d906-440a-88b7-92eb89e95949"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.99549102783203,
      "Text": "head baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1937451809644699,
          "Height": 0.056156039237976074,
          "Left": 0.49359121918678284,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "ebc18885-48d7-45b8-90e3-d172b4357802",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0ef3c194-8322-4575-94f1-82819ee57e3a",
            "d296acd9-3e9a-4985-95f8-f863614f2c46"
          ]
```

```
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98360443115234,
      "Text": "N/A, current",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.22544169425964355,
          "Height": 0.06588292121887207,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
        },
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.9381561279296875
          },
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "babf6360-789e-49c1-9c78-0784acc14a0c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "195cfb5b-ae06-4203-8520-4e4b0a73b5ce",
            "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
          ]
        }
      ]
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.94815826416016,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.17462396621704102,
          "Height": 0.06266549974679947,
          "Left": 0.29548385739326477,
          "Top": 0.03389188274741173
        },
        "Polygon": [
          {
            "X": 0.29548385739326477,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.0965573862195015
          },
          {
            "X": 0.29548385739326477,
            "Y": 0.0965573862195015
          }
        ]
      },
      "Id": "ed48dacc-d089-498f-8e93-1cee1e5f39f3"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92706298828125,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.15933875739574432,
          "Height": 0.062391020357608795,
          "Left": 0.47528234124183655,
          "Top": 0.027493247762322426
```

```
        },
        "Polygon": [
          {
            "X": 0.47528234124183655,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.08988427370786667
          },
          {
            "X": 0.47528234124183655,
            "Y": 0.08988427370786667
          }
        ]
      },
      "Id": "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9821548461914,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09610454738140106,
          "Height": 0.03656719997525215,
          "Left": 0.03988289833068848,
          "Top": 0.14147649705410004
        },
        "Polygon": [
          {
            "X": 0.03988289833068848,
            "Y": 0.14147649705410004
          },
          {
            "X": 0.13598744571208954,
            "Y": 0.14147649705410004
          },
          {
```

```
            "X": 0.13598744571208954,
            "Y": 0.1780436933040619
          },
          {
            "X": 0.03988289833068848,
            "Y": 0.1780436933040619
          }
        ]
      },
      "Id": "efe3fc6d-becb-4520-80ee-49a329386aee"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.84278106689453,
      "Text": "Information",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10029315203428268,
          "Height": 0.03209415823221207,
          "Left": 0.13837480545043945,
          "Top": 0.14050349593162537
        },
        "Polygon": [
          {
            "X": 0.13837480545043945,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.17259766161441803
          },
          {
            "X": 0.13837480545043945,
            "Y": 0.17259766161441803
          }
        ]
      },
      "Id": "c2260852-6cfd-4a71-9fc6-62b2f9b02355"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.83993530273438,
      "Text": "Full",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03039788082242012,
          "Height": 0.031106330454349518,
          "Left": 0.03899926319718361,
          "Top": 0.21361036598682404
        },
        "Polygon": [
          {
            "X": 0.03899926319718361,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.24471670389175415
          },
          {
            "X": 0.03899926319718361,
            "Y": 0.24471670389175415
          }
        ]
      },
      "Id": "e94eb587-9545-4215-b0fc-8e8cb1172958"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93611907958984,
      "Text": "Name:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05555811896920204,
          "Height": 0.030184319242835045,
          "Left": 0.07123806327581406,
          "Top": 0.2137702852487564
```

```
        },
        "Polygon": [
          {
            "X": 0.07123806327581406,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2439546138048172
          },
          {
            "X": 0.07123806327581406,
            "Y": 0.2439546138048172
          }
        ]
      },
      "Id": "090aeba5-8428-4b7a-a54b-7a95a774120e"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.91043853759766,
      "Text": "Jane",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03905024006962776,
          "Height": 0.02941947989165783,
          "Left": 0.12933772802352905,
          "Top": 0.214289128780365
        },
        "Polygon": [
          {
            "X": 0.12933772802352905,
            "Y": 0.214289128780365
          },
          {
            "X": 0.16838796436786652,
            "Y": 0.214289128780365
          },
          {
```

```
            "X": 0.16838796436786652,
            "Y": 0.24370861053466797
          },
          {
            "X": 0.12933772802352905,
            "Y": 0.24370861053466797
          }
        ]
      },
      "Id": "64ff0abb-736b-4a6b-aa8d-ad2c0086ae1d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86123657226562,
      "Text": "Doe",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.035229459404945374,
          "Height": 0.030427640303969383,
          "Left": 0.17110899090766907,
          "Top": 0.21377210319042206
        },
        "Polygon": [
          {
            "X": 0.17110899090766907,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.244199737906456
          },
          {
            "X": 0.17110899090766907,
            "Y": 0.244199737906456
          }
        ]
      },
      "Id": "565ffc30-89d6-4295-b8c6-d22b4ed76584"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.92633056640625,
      "Text": "Phone",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.052783288061618805,
          "Height": 0.03104414977133274,
          "Left": 0.03604753687977791,
          "Top": 0.28701552748680115
        },
        "Polygon": [
          {
            "X": 0.03604753687977791,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.31805968284606934
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.31805968284606934
          }
        ]
      },
      "Id": "d782f847-225b-4a1b-b52d-f252f8221b1f"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86275482177734,
      "Text": "Number:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07424934208393097,
          "Height": 0.030300479382276535,
          "Left": 0.0915418416261673,
          "Top": 0.28639692068099976
```

```
        },
        "Polygon": [
          {
            "X": 0.0915418416261673,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.3166973888874054
          },
          {
            "X": 0.0915418416261673,
            "Y": 0.3166973888874054
          }
        ]
      },
      "Id": "fa69c5cd-c80d-4fac-81df-569edae8d259"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97282409667969,
      "Text": "555-0100",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.17021971940994263,
          "Height": 0.047169629484415054,
          "Left": 0.17732827365398407,
          "Top": 0.2812676727771759
        },
        "Polygon": [
          {
            "X": 0.17732827365398407,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.2812676727771759
          },
          {
```

```
            "X": 0.3475480079650879,
            "Y": 0.32843729853630066
          },
          {
            "X": 0.17732827365398407,
            "Y": 0.32843729853630066
          }
        ]
      },
      "Id": "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.66238403320312,
      "Text": "Home",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.049357783049345016,
          "Height": 0.03134990110993385,
          "Left": 0.03359385207295418,
          "Top": 0.36172014474868774
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.3930700421333313
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.3930700421333313
          }
        ]
      },
      "Id": "acfbed90-4a00-42c6-8a90-d0a0756eea36"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.6871109008789,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07411003112792969,
          "Height": 0.0314042791724205,
          "Left": 0.08516156673431396,
          "Top": 0.3600046932697296
        },
        "Polygon": [
          {
            "X": 0.08516156673431396,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3914089798927307
          },
          {
            "X": 0.08516156673431396,
            "Y": 0.3914089798927307
          }
        ]
      },
      "Id": "046c8a40-bb0e-4718-9c71-954d3630e1dd"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93781280517578,
      "Text": "123",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05761868134140968,
          "Height": 0.05008566007018089,
          "Left": 0.1750781387090683,
          "Top": 0.35484206676483154
```

```
        },
        "Polygon": [
          {
            "X": 0.1750781387090683,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.40492773056030273
          },
          {
            "X": 0.1750781387090683,
            "Y": 0.40492773056030273
          }
        ]
      },
      "Id": "82b838bc-4591-4287-8dea-60c94a4925e4"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.96530151367188,
      "Text": "Any",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06814215332269669,
          "Height": 0.06354366987943649,
          "Left": 0.2550157308578491,
          "Top": 0.35471394658088684
        },
        "Polygon": [
          {
            "X": 0.2550157308578491,
            "Y": 0.35471394658088684
          },
          {
            "X": 0.3231579065322876,
            "Y": 0.35471394658088684
          },
          {
```

```
            "X": 0.3231579065322876,
            "Y": 0.41825762391090393
          },
          {
            "X": 0.2550157308578491,
            "Y": 0.41825762391090393
          }
        ]
      },
      "Id": "5cdcde7a-f5a6-4231-a941-b6396e42e7ba"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.87527465820312,
      "Text": "Street,",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12156613171100616,
          "Height": 0.05449587106704712,
          "Left": 0.3357025980949402,
          "Top": 0.3550415635108948
        },
        "Polygon": [
          {
            "X": 0.3357025980949402,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.4095374345779419
          },
          {
            "X": 0.3357025980949402,
            "Y": 0.4095374345779419
          }
        ]
      },
      "Id": "beafd497-185f-487e-b070-db4df5803e94"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.99514770507812,
      "Text": "Any",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07748188823461533,
          "Height": 0.07339789718389511,
          "Left": 0.47723668813705444,
          "Top": 0.3482133150100708
        },
        "Polygon": [
          {
            "X": 0.47723668813705444,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.47723668813705444,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "ef1b77fb-8ba6-41fe-ba53-dce039af22ed"
    },
    {
      "BlockType": "WORD",
      "Confidence": 96.80656433105469,
      "Text": "Town.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11213835328817368,
          "Height": 0.057233039289712906,
          "Left": 0.5563329458236694,
          "Top": 0.3331930637359619
```

```
        },
        "Polygon": [
          {
            "X": 0.5563329458236694,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3904260993003845
          },
          {
            "X": 0.5563329458236694,
            "Y": 0.3904260993003845
          }
        ]
      },
      "Id": "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98260498046875,
      "Text": "USA",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08771833777427673,
          "Height": 0.05706935003399849,
          "Left": 0.6889894604682922,
          "Top": 0.3258342146873474
        },
        "Polygon": [
          {
            "X": 0.6889894604682922,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.3258342146873474
          },
          {
```

```
            "X": 0.7767078280448914,
            "Y": 0.3829035460948944
          },
          {
            "X": 0.6889894604682922,
            "Y": 0.3829035460948944
          }
        ]
      },
      "Id": "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9583969116211,
      "Text": "Mailing",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06291338801383972,
          "Height": 0.03957144916057587,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
            "X": 0.03068041242659092,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.4730895459651947
          },
          {
            "X": 0.03068041242659092,
            "Y": 0.4730895459651947
          }
        ]
      },
      "Id": "d7261cdc-6ac5-4711-903c-4598fe94952d"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.87476348876953,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07364854216575623,
          "Height": 0.03147412836551666,
          "Left": 0.0954652726650238,
          "Top": 0.43450701236724854
        },
        "Polygon": [
          {
            "X": 0.0954652726650238,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.465981125831604
          },
          {
            "X": 0.0954652726650238,
            "Y": 0.465981125831604
          }
        ]
      },
      "Id": "287f80c3-6db2-4dd7-90ec-5f017c80aa31"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94071960449219,
      "Text": "same",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.04640670120716095,
          "Height": 0.026415130123496056,
          "Left": 0.17156922817230225,
          "Top": 0.44010937213897705
```

```
        },
        "Polygon": [
          {
            "X": 0.17156922817230225,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.46652451157569885
          },
          {
            "X": 0.17156922817230225,
            "Y": 0.46652451157569885
          }
        ]
      },
      "Id": "ce31c3ad-b51e-4068-be64-5fc9794bc1bc"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.76510620117188,
      "Text": "as",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.02041218988597393,
          "Height": 0.025104399770498276,
          "Left": 0.2207803726196289,
          "Top": 0.44124215841293335
        },
        "Polygon": [
          {
            "X": 0.2207803726196289,
            "Y": 0.44124215841293335
          },
          {
            "X": 0.24119256436824799,
            "Y": 0.44124215841293335
          },
          {
```

```
            "X": 0.24119256436824799,
            "Y": 0.4663465619087219
          },
          {
            "X": 0.2207803726196289,
            "Y": 0.4663465619087219
          }
        ]
      },
      "Id": "e96eb92c-6774-4d6f-8f4a-68a7618d4c66"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9301528930664,
      "Text": "above",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05268359184265137,
          "Height": 0.03216424956917763,
          "Left": 0.24375422298908234,
          "Top": 0.4354657828807831
        },
        "Polygon": [
          {
            "X": 0.24375422298908234,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4676300287246704
          },
          {
            "X": 0.24375422298908234,
            "Y": 0.4676300287246704
          }
        ]
      },
      "Id": "88b85c05-427a-4d4f-8cc4-3667234e8364"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 85.3905029296875,
      "Text": "Previous",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09860499948263168,
          "Height": 0.04000622034072876,
          "Left": 0.3194798231124878,
          "Top": 0.5194430351257324
        },
        "Polygon": [
          {
            "X": 0.3194798231124878,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5594492554664612
          },
          {
            "X": 0.3194798231124878,
            "Y": 0.5594492554664612
          }
        ]
      },
      "Id": "8b324501-bf38-4ce9-9777-6514b7ade760"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.14524841308594,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.14039960503578186,
          "Height": 0.04645847901701927,
          "Left": 0.4214291274547577,
          "Top": 0.5219109654426575
```

```
        },
        "Polygon": [
          {
            "X": 0.4214291274547577,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.568369448184967
          },
          {
            "X": 0.4214291274547577,
            "Y": 0.568369448184967
          }
        ]
      },
      "Id": "b0cea99a-5045-464d-ac8a-a63ab0470995"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.48454284667969,
      "Text": "History",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08361124992370605,
          "Height": 0.05192042887210846,
          "Left": 0.5668527483940125,
          "Top": 0.5172380208969116
        },
        "Polygon": [
          {
            "X": 0.5668527483940125,
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
            "Y": 0.5172380208969116
          },
          {
```

```
            "X": 0.6504639983177185,
            "Y": 0.5691584348678589
          },
          {
            "X": 0.5668527483940125,
            "Y": 0.5691584348678589
          }
        ]
      },
      "Id": "b92a6ee5-ca59-44dc-9c47-534c133b11e7"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.78699493408203,
      "Text": "Start",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.041341401636600494,
          "Height": 0.030926469713449478,
          "Left": 0.034429505467414856,
          "Top": 0.6124123334884644
        },
        "Polygon": [
          {
            "X": 0.034429505467414856,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
          }
        ]
      },
      "Id": "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.55198669433594,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03923053666949272,
          "Height": 0.03072454035282135,
          "Left": 0.07830137014389038,
          "Top": 0.6123942136764526
        },
        "Polygon": [
          {
            "X": 0.07830137014389038,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6431187391281128
          },
          {
            "X": 0.07830137014389038,
            "Y": 0.6431187391281128
          }
        ]
      },
      "Id": "91e582cd-9871-4e9c-93cc-848baa426338"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8897705078125,
      "Text": "End",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03212086856365204,
          "Height": 0.03193363919854164,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
```

```
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6439804434776306
          },
          {
            "X": 0.14846202731132507,
            "Y": 0.6439804434776306
          }
        ]
      },
      "Id": "7c97b56b-699f-49b0-93f4-98e6d90b107c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8445816040039,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03987143933773041,
          "Height": 0.03142518177628517,
          "Left": 0.1844055950641632,
          "Top": 0.612853467464447
        },
        "Polygon": [
          {
            "X": 0.1844055950641632,
            "Y": 0.612853467464447
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.612853467464447
          },
          {
```

```
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
          },
          {
            "X": 0.1844055950641632,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "7af04e27-0c15-447e-a569-b30edb99a133"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9652328491211,
      "Text": "Employer",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08150768280029297,
          "Height": 0.0392492301762104,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
            "Y": 0.6533204317092896
          }
        ]
      },
      "Id": "a9bfeb55-75cd-47cd-b953-728e602a3564"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.94273376464844,
      "Text": "Name",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05018233880400658,
          "Height": 0.03248906135559082,
          "Left": 0.34925445914268494,
          "Top": 0.6162016987800598
        },
        "Polygon": [
          {
            "X": 0.34925445914268494,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6486907601356506
          },
          {
            "X": 0.34925445914268494,
            "Y": 0.6486907601356506
          }
        ]
      },
      "Id": "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
    },
    {
      "BlockType": "WORD",
      "Confidence": 98.85071563720703,
      "Text": "Position",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07007700204849243,
          "Height": 0.03255689889192581,
          "Left": 0.49973347783088684,
          "Top": 0.6164342164993286
```

```
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6489911079406738
          },
          {
            "X": 0.49973347783088684,
            "Y": 0.6489911079406738
          }
        ]
      },
      "Id": "6d5edf02-845c-40e0-9514-e56d0d652ae0"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86096954345703,
      "Text": "Held",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.04017873853445053,
          "Height": 0.03292537108063698,
          "Left": 0.5734874606132507,
          "Top": 0.614840030670166
        },
        "Polygon": [
          {
            "X": 0.5734874606132507,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136662364006042,
            "Y": 0.614840030670166
          },
          {
```

```
            "X": 0.6136662364006042,
            "Y": 0.6477653980255127
          },
          {
            "X": 0.5734874606132507,
            "Y": 0.6477653980255127
          }
        ]
      },
      "Id": "3297ab59-b237-45fb-ae60-a108f0c95ac2"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97740936279297,
      "Text": "Reason",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06497219949960709,
          "Height": 0.03248770162463188,
          "Left": 0.7430596351623535,
          "Top": 0.6136704087257385
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6461580991744995
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6461580991744995
          }
        ]
      },
      "Id": "f4b8cf26-d2da-4a76-8345-69562de3cc11"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.98371887207031,
      "Text": "for",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.029645200818777084,
          "Height": 0.03462234139442444,
          "Left": 0.8108851909637451,
          "Top": 0.6117717623710632
        },
        "Polygon": [
          {
            "X": 0.8108851909637451,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6463940739631653
          },
          {
            "X": 0.8108851909637451,
            "Y": 0.6463940739631653
          }
        ]
      },
      "Id": "386d4a63-1194-4c0e-a18d-4d074a0b1f93"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98424530029297,
      "Text": "leaving",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06517849862575531,
          "Height": 0.040626998990774155,
          "Left": 0.8430007100105286,
          "Top": 0.6116235852241516
```

```
        },
        "Polygon": [
          {
            "X": 0.8430007100105286,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
          {
            "X": 0.8430007100105286,
            "Y": 0.6522505879402161
          }
        ]
      },
      "Id": "a8622541-1896-4d54-8d10-7da2c800ec5c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.77413177490234,
      "Text": "1/15/2009",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08799663186073303,
          "Height": 0.03832906112074852,
          "Left": 0.03175082430243492,
          "Top": 0.691371738910675
        },
        "Polygon": [
          {
            "X": 0.03175082430243492,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.691371738910675
          },
          {
```

```
            "X": 0.11974745243787766,
            "Y": 0.7297008037567139
          },
          {
            "X": 0.03175082430243492,
            "Y": 0.7297008037567139
          }
        ]
      },
      "Id": "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.72286224365234,
      "Text": "6/30/2011",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843102306127548,
          "Height": 0.03991425037384033,
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
          }
        ]
      },
      "Id": "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.92295837402344,
      "Text": "Any",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.034067559987306595,
          "Height": 0.037968240678310394,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
          {
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.7352409362792969
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.7352409362792969
          }
        ]
      },
      "Id": "77749c2b-aa7f-450e-8dd2-62bcaf253ba2"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.81578063964844,
      "Text": "Company",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08160992711782455,
          "Height": 0.03890080004930496,
          "Left": 0.29906952381134033,
          "Top": 0.6978086829185486
```

```
        },
        "Polygon": [
          {
            "X": 0.29906952381134033,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.736709475517273
          },
          {
            "X": 0.29906952381134033,
            "Y": 0.736709475517273
          }
        ]
      },
      "Id": "713bad19-158d-4e3e-b01f-f5707ddb04e5"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.37964630126953,
      "Text": "Assistant",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.0789310410618782,
          "Height": 0.03139699995517731,
          "Left": 0.49814170598983765,
          "Top": 0.7005078196525574
        },
        "Polygon": [
          {
            "X": 0.49814170598983765,
            "Y": 0.7005078196525574
          },
          {
            "X": 0.5770727396011353,
            "Y": 0.7005078196525574
          },
          {
```

```
            "X": 0.5770727396011353,
            "Y": 0.7319048047065735
          },
          {
            "X": 0.49814170598983765,
            "Y": 0.7319048047065735
          }
        ]
      },
      "Id": "989944f9-f684-4714-87d8-9ad9a321d65c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.784912109375,
      "Text": "baker",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.050264399498701096,
          "Height": 0.03237773850560188,
          "Left": 0.5806865096092224,
          "Top": 0.699238657951355
        },
        "Polygon": [
          {
            "X": 0.5806865096092224,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.7316163778305054
          },
          {
            "X": 0.5806865096092224,
            "Y": 0.7316163778305054
          }
        ]
      },
      "Id": "ae82e2aa-1601-4e0c-8340-1db7ad0c9a31"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994158506393,
          "Height": 0.03330250084400177,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98190307617188,
      "Text": "7/1/2011",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067439705133438,
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
```

```
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.8451982140541077
          },
          {
            "X": 0.028500309213995934,
            "Y": 0.8451982140541077
          }
        ]
      },
      "Id": "0f711065-1872-442a-ba6d-8fababaa452a"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98418426513672,
      "Text": "8/10/2013",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10664612054824829,
          "Height": 0.06439515948295593,
          "Left": 0.14159755408763885,
          "Top": 0.7791688442230225
        },
        "Polygon": [
          {
            "X": 0.14159755408763885,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.7791688442230225
          },
          {
```

```
            "X": 0.24824367463588715,
            "Y": 0.843563973903656
          },
          {
            "X": 0.14159755408763885,
            "Y": 0.843563973903656
          }
        ]
      },
      "Id": "a92d8eef-db28-45ba-801a-5da0f589d277"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97722625732422,
      "Text": "Example",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12127546221017838,
          "Height": 0.05682983994483948,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
        "Polygon": [
          {
            "X": 0.26764172315597534,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.8512446284294128
          },
          {
            "X": 0.26764172315597534,
            "Y": 0.8512446284294128
          }
        ]
      },
      "Id": "d6962efb-34ab-4ffb-9f2f-5f263e813558"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.98429870605469,
      "Text": "Corp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07650306820869446,
          "Height": 0.05481306090950966,
          "Left": 0.4026312530040741,
          "Top": 0.7980174422264099
        },
        "Polygon": [
          {
            "X": 0.4026312530040741,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.8528305292129517
          },
          {
            "X": 0.4026312530040741,
            "Y": 0.8528305292129517
          }
        ]
      },
      "Id": "1876c8ea-d3e8-4c39-870e-47512b3b5080"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.91166687011719,
      "Text": "Baker",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09931197017431259,
          "Height": 0.06008723005652428,
          "Left": 0.5098910331726074,
          "Top": 0.787897527217865
```

```
        },
        "Polygon": [
          {
            "X": 0.5098910331726074,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.8479847311973572
          },
          {
            "X": 0.5098910331726074,
            "Y": 0.8479847311973572
          }
        ]
      },
      "Id": "00adeaef-ed57-44eb-b8a9-503575236d62"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98870849609375,
      "Text": "better",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10782185196876526,
          "Height": 0.06207133084535599,
          "Left": 0.7428008317947388,
          "Top": 0.7928366661071777
        },
        "Polygon": [
          {
            "X": 0.7428008317947388,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.8506226539611816,
            "Y": 0.7928366661071777
          },
          {
```

```
            "X": 0.8506226539611816,
            "Y": 0.8549079895019531
          },
          {
            "X": 0.7428008317947388,
            "Y": 0.8549079895019531
          }
        ]
      },
      "Id": "c0fc9a58-7a4b-4f69-bafd-2cff32be2665"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8883285522461,
      "Text": "opp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07421936094760895,
          "Height": 0.058906231075525284,
          "Left": 0.8577775359153748,
          "Top": 0.8038780689239502
        },
        "Polygon": [
          {
            "X": 0.8577775359153748,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8627843260765076
          },
          {
            "X": 0.8577775359153748,
            "Y": 0.8627843260765076
          }
        ]
      },
      "Id": "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.92573547363281,
      "Text": "8/15/2013",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10257463902235031,
          "Height": 0.05412459000945091,
          "Left": 0.027909137308597565,
          "Top": 0.8608770370483398
        },
        "Polygon": [
          {
            "X": 0.027909137308597565,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.915001630783081
          },
          {
            "X": 0.027909137308597565,
            "Y": 0.915001630783081
          }
        ]
      },
      "Id": "5384f860-f857-4a94-9438-9dfa20eed1c6"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99625396728516,
      "Text": "Present",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888339668512344,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
```

```
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
          {
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
      "Id": "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611273169517517,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994276046753,
            "Y": 0.869536280632019
          },
          {
```

```
            "X": 0.4476994276046753,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
      },
      "Id": "25343360-d906-440a-88b7-92eb89e95949"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99523162841797,
      "Text": "head",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07429949939250946,
          "Height": 0.05485520139336586,
          "Left": 0.49359121918678284,
          "Top": 0.8714361190795898
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.926291286945343
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.926291286945343
          }
        ]
      },
      "Id": "0ef3c194-8322-4575-94f1-82819ee57e3a"
    },
```

```
    {
      "BlockType": "WORD",
      "Confidence": 99.99574279785156,
      "Text": "baker",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1019822508096695,
          "Height": 0.05615599825978279,
          "Left": 0.585354208946228,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
            "X": 0.585354208946228,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.585354208946228,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "d296acd9-3e9a-4985-95f8-f863614f2c46"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9880599975586,
      "Text": "N/A,",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08230073750019073,
          "Height": 0.06588289886713028,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
```

```
        },
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.9381561279296875
          },
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "195cfb5b-ae06-4203-8520-4e4b0a73b5ce"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97914123535156,
      "Text": "current",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12791454792022705,
          "Height": 0.04768490046262741,
          "Left": 0.8387037515640259,
          "Top": 0.8843405842781067
        },
        "Polygon": [
          {
            "X": 0.8387037515640259,
            "Y": 0.8843405842781067
          },
          {
            "X": 0.9666182994842529,
            "Y": 0.8843405842781067
          },
          {
```

```
            "X": 0.9666182994842529,
            "Y": 0.9320254921913147
          },
          {
            "X": 0.8387037515640259,
            "Y": 0.9320254921913147
          }
        ]
      },
      "Id": "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
    }
  ],
  "DetectDocumentTextModelVersion": "1.0",
  "ResponseMetadata": {
    "RequestId": "337129e6-3af7-4014-842b-f6484e82cbf6",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "337129e6-3af7-4014-842b-f6484e82cbf6",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "45675",
      "date": "Mon, 09 Nov 2020 23:54:38 GMT"
    },
    "RetryAttempts": 0
  }
}
}
```


Amazon Textract Textract를 사용하여 문서 텍스트 감지

문서에서 텍스트를 검색하려면DetectDocumentText작업을 수행하고 문서 파일을 입력으로 전달합니다.DetectDocumentText검색된 텍스트의 줄과 단어, 문서의 텍스트 위치 및 감지된 텍스트 간의 관계를 포함하는 JSON 구조를 반환합니다. 자세한 정보는 텍스트 감지을 참조하십시오.

입력 문서를 이미지 바이트 어레이 (base64 인코딩 이미지 바이트) 또는 Amazon S3 객체로 제공할 수 있습니다. 이 절차에서는 S3 버킷에 이미지 파일을 업로드하고 파일 이름을 지정합니다.

문서에서 텍스트를 감지하려면 (API)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.

a. 을 사용하여 IAM 사용자를 생성 또는 업데이트합니다.`AmazonTextractFullAccess`과`AmazonS3ReadOnlyAccess`권한. 자세한 정보는 1단계: AWS 계정 설정 및 IAM 사용자 만들기을 참조하십시오.

b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 정보는 2단계: 설정AWS CLI과 AWSSDK을 참조하십시오.

2. S3 버킷에 문서를 업로드합니다.

   지침은 단원을 참조하십시오.Amazon S3 객체 업로드의Amazon Simple Storage Service.

3. 다음 예제를 사용하여 `DetectDocumentText` 작업을 호출합니다.

   Java

   다음 예제 코드는 검색된 텍스트 줄 주위에 문서와 상자를 표시합니다.

   함수에서 수행`main`의 값을 바꿉니다.`bucket`과`document`Amazon S3 버킷의 이름 및 2단계에서 사용한 문서 이름이 표시됩니다.

```
//Calls DetectDocumentText.
//Loads document from S3 bucket. Displays the document and bounding boxes around
 detected lines/words of text.
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.BoundingBox;
import com.amazonaws.services.textract.model.DetectDocumentTextRequest;
import com.amazonaws.services.textract.model.DetectDocumentTextResult;
import com.amazonaws.services.textract.model.Document;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
```

```
import com.amazonaws.services.textract.model.Relationship;

public class DocumentText extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    DetectDocumentTextResult result;

    public DocumentText(DetectDocumentTextResult documentResult, BufferedImage
 bufImage) throws Exception {
        super();

        result = documentResult; // Results of text detection.
        image = bufImage; // The image containing the document.

    }

    // Draws the image and text bounding box.
    public void paintComponent(Graphics g) {

        int height = image.getHeight(this);
        int width = image.getWidth(this);

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, image.getWidth(this) , image.getHeight(this),
 this);

        // Iterate through blocks and display polygons around lines of detected
 text.
        List<Block> blocks = result.getBlocks();
        for (Block block : blocks) {
            DisplayBlockInfo(block);
            if ((block.getBlockType()).equals("LINE")) {
                ShowPolygon(height, width, block.getGeometry().getPolygon(),
 g2d);
                /*
                  ShowBoundingBox(height, width,
 block.getGeometry().getBoundingBox(), g2d);
                 */
            } else { // its a word, so just show vertical lines.
```

```
                ShowPolygonVerticals(height, width,
block.getGeometry().getPolygon(), g2d);
            }
        }
    }

    // Show bounding box at supplied location.
    private void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox
box, Graphics2D g2d) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(new Color(0, 212, 0));
        g2d.drawRect(Math.round(left), Math.round(top),
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

    }

    // Shows polygon at supplied location
    private void ShowPolygon(int imageHeight, int imageWidth, List<Point>
points, Graphics2D g2d) {

        g2d.setColor(new Color(0, 0, 0));
        Polygon polygon = new Polygon();

        // Construct polygon and display
        for (Point point : points) {
            polygon.addPoint((Math.round(point.getX() * imageWidth)),
                    Math.round(point.getY() * imageHeight));
        }
        g2d.drawPolygon(polygon);
    }

    // Draws only the vertical lines in the supplied polygon.
    private void ShowPolygonVerticals(int imageHeight, int imageWidth,
List<Point> points, Graphics2D g2d) {

        g2d.setColor(new Color(0, 212, 0));
        Object[] parry = points.toArray();
        g2d.setStroke(new BasicStroke(2));
```

```
        g2d.drawLine(Math.round(((Point) parry[0]).getX() * imageWidth),
                Math.round(((Point) parry[0]).getY() * imageHeight),
Math.round(((Point) parry[3]).getX() * imageWidth),
                Math.round(((Point) parry[3]).getY() * imageHeight));

        g2d.setColor(new Color(255, 0, 0));
        g2d.drawLine(Math.round(((Point) parry[1]).getX() * imageWidth),
                Math.round(((Point) parry[1]).getY() * imageHeight),
Math.round(((Point) parry[2]).getX() * imageWidth),
                Math.round(((Point) parry[2]).getY() * imageHeight));

    }
    //Displays information from a block returned by text detection and text
analysis
    private void DisplayBlockInfo(Block block) {
        System.out.println("Block Id : " + block.getId());
        if (block.getText()!=null)
            System.out.println("    Detected text: " + block.getText());
        System.out.println("    Type: " + block.getBlockType());

        if (block.getBlockType().equals("PAGE") !=true) {
            System.out.println("    Confidence: " +
block.getConfidence().toString());
        }
        if(block.getBlockType().equals("CELL"))
        {
            System.out.println("    Cell information:");
            System.out.println("        Column: " + block.getColumnIndex());
            System.out.println("        Row: " + block.getRowIndex());
            System.out.println("        Column span: " + block.getColumnSpan());
            System.out.println("        Row span: " + block.getRowSpan());

        }

        System.out.println("    Relationships");
        List<Relationship> relationships=block.getRelationships();
        if(relationships!=null) {
            for (Relationship relationship : relationships) {
                System.out.println("        Type: " + relationship.getType());
                System.out.println("        IDs: " +
relationship.getIds().toString());
            }
        } else {
            System.out.println("        No related Blocks");
```

```
        }

        System.out.println("    Geometry");
        System.out.println("        Bounding Box: " +
 block.getGeometry().getBoundingBox().toString());
        System.out.println("        Polygon: " +
 block.getGeometry().getPolygon().toString());

        List<String> entityTypes = block.getEntityTypes();

        System.out.println("    Entity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("        Entity Type: " + entityType);
            }
        } else {
            System.out.println("        No entity type");
        }
        if(block.getPage()!=null)
            System.out.println("    Page: " + block.getPage());
        System.out.println();
    }

    public static void main(String arg[]) throws Exception {

        // The S3 bucket and document
        String document = "";
        String bucket = "";


        AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .withEndpointConfiguration(
                        new EndpointConfiguration("https://
s3.amazonaws.com","us-east-1"))
                .build();


        // Get the document from S3
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, document);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        BufferedImage image = ImageIO.read(inputStream);

        // Call DetectDocumentText
```

```
        EndpointConfiguration endpoint = new EndpointConfiguration(
                "https://textract.us-east-1.amazonaws.com", "us-east-1");
        AmazonTextract client = AmazonTextractClientBuilder.standard()
                .withEndpointConfiguration(endpoint).build();


        DetectDocumentTextRequest request = new DetectDocumentTextRequest()
            .withDocument(new Document().withS3Object(new
 S3Object().withName(document).withBucket(bucket)));

        DetectDocumentTextResult result = client.detectDocumentText(request);

        // Create frame and panel.
        JFrame frame = new JFrame("RotateImage");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        DocumentText panel = new DocumentText(result, image);
        panel.setPreferredSize(new Dimension(image.getWidth() ,
 image.getHeight() ));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    }
}
```

AWS CLI

이 AWS CLI 명령은 `detect-document-text` CLI 작업에 대한 JSON 출력을 표시합니다.

다음 값 바꾸기`Bucket`과`Name`Amazon S3 버킷의 이름 및 2단계에서 사용한 문서 이름이 표시됩니다.

```
aws textract detect-document-text \
 --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}'
```

Python

다음 예제 코드는 감지된 텍스트 줄 주위에 문서와 상자를 표시합니다.

함수에서 수행`main`의 값을 바꿉니다.`bucket`과`document`Amazon S3 버킷의 이름 및 2단계에서 사용한 문서 이름이 표시됩니다.

```
#Detects text in a document stored in an S3 bucket. Display polygon box around
 text and angled text
import boto3
import io
from io import BytesIO
import sys

import psutil
import time

import math
from PIL import Image, ImageDraw, ImageFont


# Displays information about a block returned by text detection and text
 analysis
def DisplayBlockInformation(block):
    print('Id: {}'.format(block['Id']))
    if 'Text' in block:
        print('    Detected: ' + block['Text'])
    print('    Type: ' + block['BlockType'])

    if 'Confidence' in block:
        print('    Confidence: ' + "{:.2f}".format(block['Confidence']) + "%")

    if block['BlockType'] == 'CELL':
        print("    Cell information")
        print("        Column: " + str(block['ColumnIndex']))
        print("        Row: " + str(block['RowIndex']))
        print("        ColumnSpan: " + str(block['ColumnSpan']))
        print("        RowSpan: " + str(block['RowSpan']))

    if 'Relationships' in block:
        print('    Relationships: {}'.format(block['Relationships']))
    print('    Geometry: ')
    print('        Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
    print('        Polygon: {}'.format(block['Geometry']['Polygon']))

    if block['BlockType'] == "KEY_VALUE_SET":
        print ('    Entity Type: ' + block['EntityTypes'][0])
    if 'Page' in block:
        print('Page: ' + block['Page'])
    print()
```

```
def process_text_detection(bucket, document):


    #Get the document from S3
    s3_connection = boto3.resource('s3')

    s3_object = s3_connection.Object(bucket,document)
    s3_response = s3_object.get()

    stream = io.BytesIO(s3_response['Body'].read())
    image=Image.open(stream)


    # Detect text in the document

    client = boto3.client('textract')
    #process using image bytes
    #image_binary = stream.getvalue()
    #response = client.detect_document_text(Document={'Bytes': image_binary})

    #process using S3 object
    response = client.detect_document_text(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    #Get the text blocks
    blocks=response['Blocks']
    width, height =image.size
    draw = ImageDraw.Draw(image)
    print ('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:
            print('Type: ' + block['BlockType'])
            if block['BlockType'] != 'PAGE':
                print('Detected: ' + block['Text'])
                print('Confidence: ' + "{:.2f}".format(block['Confidence']) +
 "%")

            print('Id: {}'.format(block['Id']))
            if 'Relationships' in block:
                print('Relationships: {}'.format(block['Relationships']))
            print('Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
            print('Polygon: {}'.format(block['Geometry']['Polygon']))
```

```
            print()
            draw=ImageDraw.Draw(image)
            # Draw WORD - Green -  start of word, red - end of word
            if block['BlockType'] == "WORD":
                draw.line([(width * block['Geometry']['Polygon'][0]['X'],
                height * block['Geometry']['Polygon'][0]['Y']),
                (width * block['Geometry']['Polygon'][3]['X'],
                height * block['Geometry']['Polygon'][3]['Y'])],fill='green',
                width=2)

                draw.line([(width * block['Geometry']['Polygon'][1]['X'],
                height * block['Geometry']['Polygon'][1]['Y']),
                (width * block['Geometry']['Polygon'][2]['X'],
                height * block['Geometry']['Polygon'][2]['Y'])],
                fill='red',
                width=2)


            # Draw box around entire LINE
            if block['BlockType'] == "LINE":
                points=[]

                for polygon in block['Geometry']['Polygon']:
                    points.append((width * polygon['X'], height * polygon['Y']))

                draw.polygon((points), outline='black')

                # Uncomment to draw bounding box
                #box=block['Geometry']['BoundingBox']
                #left = width * box['Left']
                #top = height * box['Top']
                #draw.rectangle([left,top, left + (width * box['Width']), top
 +(height * box['Height'])],outline='black')


    # Display the image
    image.show()
    # display image for 10 seconds


    return len(blocks)

def main():
```

```
    bucket = ''
    document = ''
    block_count=process_text_detection(bucket,document)
    print("Blocks detected: " + str(block_count))

if __name__ == "__main__":
    main()
```

Node.js

다음 Node.js 예제 코드는 검색된 텍스트 줄 주위에 문서와 상자를 표시하여 결과 이미지를 코드를 실행하는 디렉터리로 출력합니다. 그것은 다음을 사용합니다.`image-size`과`images`패키지.

함수에서 수행`main`의 값을 바꿉니다.`bucket`과`document`Amazon S3 버킷의 이름 및 2단계에서 사용한 문서 이름이 표시됩니다. 값 바꾸기`regionConfig`계정이 있는 리전의 이름을 사용합니다.

```
async function main(){

// Import AWS
const AWS = require("aws-sdk")
// Use Image-Size to get
const sizeOf = require('image-size');
// Image tool to draw buffers
const images = require("images");

// Create a canvas and get the context
const { createCanvas } = require('canvas')
const canvas = createCanvas(200, 200)
const ctx = canvas.getContext('2d')

// Set variables
const bucket = 'bucket-name' // the s3 bucket name
const photo  = 'image-name' // the name of file
const regionConfig = 'region'

// Set region if needed
AWS.config.update({region:regionConfig});

// Connect to Textract
```

```
const client = new AWS.Textract();
// Connect to S3 to display image
const s3 = new AWS.S3();

// Define paramaters
const params = {
  Document: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
}

// Function to display image
async function getImage(){
  const imageData =  s3.getObject(
    {
        Bucket: bucket,
        Key: photo
      }

  ).promise();
  return imageData;
}

// get image
var imageData = await getImage()

// Get the height, width of the image
const dimensions = sizeOf(imageData.Body)
const width = dimensions.width
const height = dimensions.height
console.log(imageData.Body)
console.log(width, height)

canvas.width = width;
canvas.height = height;

try{
  // Call API and log response
  const res = await client.detectDocumentText(params).promise();
  var image = images(imageData.Body).size(width, height)
  //console.log the type of block, text, text type, and confidence
```

```
  res.Blocks.forEach(block => {
    console.log(`Block Type: ${block.BlockType}`),
    console.log(`Text: ${block.Text}`)
    console.log(`TextType: ${block.TextType}`)
    console.log(`Confidence: ${block.Confidence}`)

    // Draw box around detected text using polygons
    ctx.strokeStyle = 'rgba(0,0,0,0.5)';
    ctx.beginPath();
    block.Geometry.Polygon.forEach(({X, Y}) =>
    ctx.lineTo(width * X - 10, height * Y - 10)
    );
    ctx.closePath();
    ctx.stroke();
    console.log("-----")
  })

  // render image
  var buffer = canvas.toBuffer("image/png");
  image.draw(images(buffer), 10, 10)
  image.save("output-image.jpg");

} catch (err){
console.error(err);}

}

main()
```

4. 예제를 실행합니다. 파이썬과 Java 예제는 문서 이미지를 표시합니다. 검은색 상자는 감지된 텍스트의 각 줄을 둘러싸고 있습니다. 녹색 세로선은 감지된 단어의 시작입니다. 빨간색 세로선은 감지된 단어의 끝입니다. 이AWS CLI예제는 에 대한 JSON 출력만 표시합니다.`DetectDocumentText`작업을 수행합니다.

Amazon Textract Textract를 사용하여 문서 텍스트 분석

문서의 텍스트를 분석하려면`AnalyzeDocument`작업을 수행하고 문서 파일을 입력으로 전달합니다.`AnalyzeDocument`는 분석된 텍스트를 포함하는 JSON 구조를 반환합니다. 자세한 정보는 문서 분석을 참조하십시오.

입력 문서를 이미지 바이트 어레이 (base64 인코딩 이미지 바이트) 또는 Amazon S3 객체로 제공할 수 있습니다. 이 절차에서는 S3 버킷에 이미지 파일을 업로드하고 파일 이름을 지정합니다.

문서의 텍스트를 분석하려면 (API)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.

   a. 을 사용하여 IAM 사용자를 생성 또는 업데이트합니다.`AmazonTextractFullAccess`과`AmazonS3ReadOnlyAccess`권한. 자세한 정보는 1단계: AWS 계정 설정 및 IAM 사용자 만들기을 참조하십시오.

   b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 정보는 2단계: 설정AWS CLI과 AWSSDK을 참조하십시오.

2. 문서가 포함된 이미지를 S3 버킷에 업로드합니다.

   지침은 단원을 참조하십시오.Amazon S3 객체 업로드의Amazon Simple Storage Service.

3. 다음 예제를 사용하여 `AnalyzeDocument` 작업을 호출합니다.

   Java

   다음 예제 코드는 감지된 항목 주위에 문서와 상자를 표시합니다.

   함수에서 수행`main`의 값을 바꿉니다.`bucket`과`document`2단계에서 사용한 Amazon S3 버킷과 문서 이미지의 이름이 표시됩니다.

```
//Loads document from S3 bucket. Displays the document and polygon around
 detected lines of text.
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.AnalyzeDocumentRequest;
import com.amazonaws.services.textract.model.AnalyzeDocumentResult;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.BoundingBox;
import com.amazonaws.services.textract.model.Document;
```

```
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
import com.amazonaws.services.textract.model.Relationship;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;

public class AnalyzeDocument extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;

    AnalyzeDocumentResult result;

    public AnalyzeDocument(AnalyzeDocumentResult documentResult, BufferedImage
 bufImage) throws Exception {
        super();

        result = documentResult; // Results of text detection.
        image = bufImage; // The image containing the document.

    }

    // Draws the image and text bounding box.
    public void paintComponent(Graphics g) {

        int height = image.getHeight(this);
        int width = image.getWidth(this);

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, image.getWidth(this), image.getHeight(this),
 this);

        // Iterate through blocks and display bounding boxes around everything.

        List<Block> blocks = result.getBlocks();
        for (Block block : blocks) {
            DisplayBlockInfo(block);
            switch(block.getBlockType()) {

            case "KEY_VALUE_SET":
                if (block.getEntityTypes().contains("KEY")){
```

```
                    ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(255,0,0));
                }
                else {  //VALUE
                    ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(0,255,0));
                }
                break;
            case "TABLE":
                ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(0,0,255));
                break;
            case "CELL":
                ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(255,255,0));
                break;
            case "SELECTION_ELEMENT":
                if (block.getSelectionStatus().equals("SELECTED"))
                    ShowSelectedElement(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(0,0,255));
                break;
            default:
                //PAGE, LINE & WORD
                //ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(200,200,0));
            }
        }

         // uncomment to show polygon around all blocks
         //ShowPolygon(height,width,block.getGeometry().getPolygon(),g2d);


    }

    // Show bounding box at supplied location.
    private void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox
box, Graphics2D g2d, Color color) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(color);
        g2d.drawRect(Math.round(left), Math.round(top),
```

```
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

   }
   private void ShowSelectedElement(int imageHeight, int imageWidth,
BoundingBox box, Graphics2D g2d, Color color) {

       float left = imageWidth * box.getLeft();
       float top = imageHeight * box.getTop();

       // Display bounding box.
       g2d.setColor(color);
       g2d.fillRect(Math.round(left), Math.round(top),
               Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

   }

   // Shows polygon at supplied location
   private void ShowPolygon(int imageHeight, int imageWidth, List<Point>
points, Graphics2D g2d) {

       g2d.setColor(new Color(0, 0, 0));
       Polygon polygon = new Polygon();

       // Construct polygon and display
       for (Point point : points) {
           polygon.addPoint((Math.round(point.getX() * imageWidth)),
                   Math.round(point.getY() * imageHeight));
       }
       g2d.drawPolygon(polygon);
   }
   //Displays information from a block returned by text detection and text
analysis
   private void DisplayBlockInfo(Block block) {
       System.out.println("Block Id : " + block.getId());
       if (block.getText()!=null)
           System.out.println("    Detected text: " + block.getText());
       System.out.println("    Type: " + block.getBlockType());

       if (block.getBlockType().equals("PAGE") !=true) {
           System.out.println("    Confidence: " +
block.getConfidence().toString());
       }
```

```
        if(block.getBlockType().equals("CELL"))
        {
            System.out.println("    Cell information:");
            System.out.println("        Column: " + block.getColumnIndex());
            System.out.println("        Row: " + block.getRowIndex());
            System.out.println("        Column span: " + block.getColumnSpan());
            System.out.println("        Row span: " + block.getRowSpan());

        }

        System.out.println("    Relationships");
        List<Relationship> relationships=block.getRelationships();
        if(relationships!=null) {
            for (Relationship relationship : relationships) {
                System.out.println("        Type: " + relationship.getType());
                System.out.println("        IDs: " +
relationship.getIds().toString());
            }
        } else {
            System.out.println("        No related Blocks");
        }

        System.out.println("    Geometry");
        System.out.println("        Bounding Box: " +
block.getGeometry().getBoundingBox().toString());
        System.out.println("        Polygon: " +
block.getGeometry().getPolygon().toString());

        List<String> entityTypes = block.getEntityTypes();

        System.out.println("    Entity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("        Entity Type: " + entityType);
            }
        } else {
            System.out.println("        No entity type");
        }

        if(block.getBlockType().equals("SELECTION_ELEMENT")) {
            System.out.print("    Selection element detected: ");
            if (block.getSelectionStatus().equals("SELECTED")){
                System.out.println("Selected");
            }else {
```

```
                System.out.println(" Not selected");
            }
        }

        if(block.getPage()!=null)
            System.out.println("    Page: " + block.getPage());
        System.out.println();
    }

    public static void main(String arg[]) throws Exception {

        // The S3 bucket and document
        String document = "";
        String bucket = "";

        AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .withEndpointConfiguration(
                        new EndpointConfiguration("https://
s3.amazonaws.com","us-east-1"))
                .build();


        // Get the document from S3
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, document);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        BufferedImage image = ImageIO.read(inputStream);

        // Call AnalyzeDocument
        EndpointConfiguration endpoint = new EndpointConfiguration(
                "https://textract.us-east-1.amazonaws.com", "us-east-1");
        AmazonTextract client = AmazonTextractClientBuilder.standard()
                .withEndpointConfiguration(endpoint).build();


        AnalyzeDocumentRequest request = new AnalyzeDocumentRequest()
                .withFeatureTypes("TABLES","FORMS")
                 .withDocument(new Document().
                        withS3Object(new
 S3Object().withName(document).withBucket(bucket)));


        AnalyzeDocumentResult result = client.analyzeDocument(request);
```

```
        // Create frame and panel.
        JFrame frame = new JFrame("RotateImage");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        AnalyzeDocument panel = new AnalyzeDocument(result, image);
        panel.setPreferredSize(new Dimension(image.getWidth(),
 image.getHeight()));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    }
}
```

AWS CLI

이 AWS CLI 명령은 `detect-document-text` CLI 작업에 대한 JSON 출력을 표시합니다.

다음 값 바꾸기`Bucket`과`Name`Amazon S3 버킷의 이름 및 2단계에서 사용한 문서 이름이 표시됩니다.

```
aws textract analyze-document \
 --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
 --feature-types '["TABLES","FORMS"]'
```

Python

다음 예제 코드는 감지된 항목 주위에 문서와 상자를 표시합니다.

함수에서 수행`main`의 값을 바꿉니다.`bucket`과`document`Amazon S3 버킷의 이름 및 2단계에서 사용한 문서 이름이 표시됩니다.

```
#Analyzes text in a document stored in an S3 bucket. Display polygon box around
 text and angled text
import boto3
import io
from io import BytesIO
import sys

import math
from PIL import Image, ImageDraw, ImageFont
```

```
def ShowBoundingBox(draw,box,width,height,boxColor):

    left = width * box['Left']
    top = height * box['Top']
    draw.rectangle([left,top, left + (width * box['Width']), top +(height *
 box['Height'])],outline=boxColor)

def ShowSelectedElement(draw,box,width,height,boxColor):

    left = width * box['Left']
    top = height * box['Top']
    draw.rectangle([left,top, left + (width * box['Width']), top +(height *
 box['Height'])],fill=boxColor)

# Displays information about a block returned by text detection and text
 analysis
def DisplayBlockInformation(block):
    print('Id: {}'.format(block['Id']))
    if 'Text' in block:
        print('    Detected: ' + block['Text'])
    print('    Type: ' + block['BlockType'])

    if 'Confidence' in block:
        print('    Confidence: ' + "{:.2f}".format(block['Confidence']) + "%")

    if block['BlockType'] == 'CELL':
        print("    Cell information")
        print("        Column:" + str(block['ColumnIndex']))
        print("        Row:" + str(block['RowIndex']))
        print("        Column Span:" + str(block['ColumnSpan']))
        print("        RowSpan:" + str(block['ColumnSpan']))

    if 'Relationships' in block:
        print('    Relationships: {}'.format(block['Relationships']))
    print('    Geometry: ')
    print('        Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
    print('        Polygon: {}'.format(block['Geometry']['Polygon']))

    if block['BlockType'] == "KEY_VALUE_SET":
        print ('    Entity Type: ' + block['EntityTypes'][0])

    if block['BlockType'] == 'SELECTION_ELEMENT':
        print('    Selection element detected: ', end='')
```

```
        if block['SelectionStatus'] =='SELECTED':
            print('Selected')
        else:
            print('Not selected')

    if 'Page' in block:
        print('Page: ' + block['Page'])
    print()

def process_text_analysis(bucket, document):

    #Get the document from S3
    s3_connection = boto3.resource('s3')

    s3_object = s3_connection.Object(bucket,document)
    s3_response = s3_object.get()

    stream = io.BytesIO(s3_response['Body'].read())
    image=Image.open(stream)

    # Analyze the document
    client = boto3.client('textract')

    image_binary = stream.getvalue()
    response = client.analyze_document(Document={'Bytes': image_binary},
        FeatureTypes=["TABLES", "FORMS"])

    ### Alternatively, process using S3 object ###
    #response = client.analyze_document(
    #    Document={'S3Object': {'Bucket': bucket, 'Name': document}},
    #    FeatureTypes=["TABLES", "FORMS"])

    ### To use a local file ###
    # with open("pathToFile", 'rb') as img_file:
        ### To display image using PIL ###
    #    image = Image.open()
        ### Read bytes ###
    #    img_bytes = img_file.read()
    #    response = client.analyze_document(Document={'Bytes': img_bytes},
 FeatureTypes=["TABLES", "FORMS"])


    #Get the text blocks
    blocks=response['Blocks']
```

```
    width, height =image.size
    draw = ImageDraw.Draw(image)
    print ('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:

        DisplayBlockInformation(block)

        draw=ImageDraw.Draw(image)
        if block['BlockType'] == "KEY_VALUE_SET":
            if block['EntityTypes'][0] == "KEY":
                ShowBoundingBox(draw, block['Geometry']
['BoundingBox'],width,height,'red')
            else:
                ShowBoundingBox(draw, block['Geometry']
['BoundingBox'],width,height,'green')

        if block['BlockType'] == 'TABLE':
            ShowBoundingBox(draw, block['Geometry']['BoundingBox'],width,height,
 'blue')

        if block['BlockType'] == 'CELL':
            ShowBoundingBox(draw, block['Geometry']['BoundingBox'],width,height,
 'yellow')
        if block['BlockType'] == 'SELECTION_ELEMENT':
            if block['SelectionStatus'] =='SELECTED':
                ShowSelectedElement(draw, block['Geometry']
['BoundingBox'],width,height, 'blue')

            #uncomment to draw polygon for all Blocks
            #points=[]
            #for polygon in block['Geometry']['Polygon']:
            #    points.append((width * polygon['X'], height * polygon['Y']))
            #draw.polygon((points), outline='blue')

    # Display the image
    image.show()
    return len(blocks)


def main():

    bucket = ''
```

```
    document = ''
    block_count=process_text_analysis(bucket,document)
    print("Blocks detected: " + str(block_count))

if __name__ == "__main__":
    main()
```

Node.js

다음 예제 코드는 감지된 항목 주위에 문서와 상자를 표시합니다.

아래 코드에서 다음 값을 바꿉니다.bucket과photoAmazon S3 버킷의 이름 및 2단계에서 사용한 문서 이름이 표시됩니다. 값 바꾸기region계정과 연결된 리전을 사용합니다.

```
// Import required AWS SDK clients and commands for Node.js
import { AnalyzeDocumentCommand } from  "@aws-sdk/client-textract";
import  { TextractClient } from "@aws-sdk/client-textract";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
// Create SNS service object.
const textractClient = new TextractClient({ region: REGION });

const bucket = 'buckets'
const photo = 'photo'

// Set params
const params = {
    Document: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
    FeatureTypes: ['TABLES', 'FORMS'],
  }

const displayBlockInfo = async (response) => {
    try {
        response.Blocks.forEach(block => {
            console.log(`ID: ${block.Id}`)
            console.log(`Block Type: ${block.BlockType}`)
            if ("Text" in block && block.Text !== undefined){
```

```
                console.log(`Text: ${block.Text}`)
            }
            else{}
            if ("Confidence" in block && block.Confidence !== undefined){
                console.log(`Confidence: ${block.Confidence}`)
            }
            else{}
            if (block.BlockType == 'CELL'){
                console.log("Cell info:")
                console.log(`   Column Index - ${block.ColumnIndex}`)
                console.log(`   Row - ${block.RowIndex}`)
                console.log(`   Column Span - ${block.ColumnSpan}`)
                console.log(`   Row Span - ${block.RowSpan}`)
            }
            if ("Relationships" in block && block.Relationships !== undefined){
                console.log(block.Relationships)
                console.log("Geometry:")
                console.log(`   Bounding Box -
 ${JSON.stringify(block.Geometry.BoundingBox)}`)
                console.log(`   Polygon -
 ${JSON.stringify(block.Geometry.Polygon)}`)
            }
            console.log("-----")
        });
      } catch (err) {
        console.log("Error", err);
      }
}

const analyze_document_text = async () => {
    try {
        const analyzeDoc = new AnalyzeDocumentCommand(params);
        const response = await textractClient.send(analyzeDoc);
        //console.log(response)
        displayBlockInfo(response)
        return response; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

analyze_document_text()
```

4. 예제를 실행합니다. 파이썬과 Java 예제는 다음과 같은 컬러 경계 상자와 함께 문서 이미지를 표시합니다.

   - 빨간색 — 키 블록 객체
   - 녹색 — 밸류 블록 객체
   - 파란색 — TABLE 블록 객체
   - 노란색 — CELL 블록 객체

   선택한 선택 요소는 파란색으로 채워집니다.

   이AWS CLI예제는 에 대한 JSON 출력만 표시합니다.`AnalyzeDocument`작업을 수행합니다.

Amazon Textract Textract를 사용하여 송장 및 영수증 분석

송장 및 영수증 문서를 분석하려면 AnalyzeFreent API를 사용하고 문서 파일을 입력으로 전달합니다.`AnalyzeExpense`는 분석된 텍스트를 포함하는 JSON 구조를 반환하는 동기 작업입니다. 자세한 정보는 송장 및 수금 분석을 참조하십시오.

인보이스 및 영수증을 비동기적으로 분석하려면`StartExpenseAnalysis`입력 문서 파일 처리를 시작하고`GetExpenseAnalysis`결과를 볼 수 있습니다를 사용합니다.

입력 문서를 이미지 바이트 어레이 (base64 인코딩 이미지 바이트) 또는 Amazon S3 객체로 제공할 수 있습니다. 이 절차에서는 S3 버킷에 이미지 파일을 업로드하고 파일 이름을 지정합니다.

인보이스 또는 영수증 (API) 을 분석하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.

   a. 을 사용하여 IAM 사용자를 생성 또는 업데이트합니다.`AmazonTextractFullAccess`과`AmazonS3ReadOnlyAccess`권한. 자세한 정보는 1단계: AWS 계정 설정 및 IAM 사용자 만들기을 참조하십시오.

   b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 정보는 2단계: 설정AWS CLI과 AWSSDK을 참조하십시오.

2. 문서가 포함된 이미지를 S3 버킷에 업로드합니다.

   지침은 단원을 참조하십시오.Amazon S3 객체 업로드의Amazon Simple Storage Service.

3. 다음 예제를 사용하여 `AnalyzeExpense` 작업을 호출합니다.

CLI

```
  aws textract analyze-expense --document '{"S3Object": {"Bucket": "bucket
 name","Name":
"object name"}}'
```

Python

```
import boto3
import io
from PIL import Image, ImageDraw

def draw_bounding_box(key, val, width, height, draw):
    # If a key is Geometry, draw the bounding box info in it
    if "Geometry" in key:
        # Draw bounding box information
        box = val["BoundingBox"]
        left = width * box['Left']
        top = height * box['Top']
        draw.rectangle([left, top, left + (width * box['Width']), top + (height
 * box['Height'])],
                       outline='black')

# Takes a field as an argument and prints out the detected labels and values
def print_labels_and_values(field):
    # Only if labels are detected and returned
    if "LabelDetection" in field:
        print("Summary Label Detection - Confidence: {}".format(
            str(field.get("LabelDetection")["Confidence"])) + ", "
              + "Summary Values: {}".format(str(field.get("LabelDetection")
["Text"])))
        print(field.get("LabelDetection")["Geometry"])
    else:
        print("Label Detection - No labels returned.")
    if "ValueDetection" in field:
        print("Summary Value Detection - Confidence: {}".format(
            str(field.get("ValueDetection")["Confidence"])) + ", "
              + "Summary Values: {}".format(str(field.get("ValueDetection")
["Text"])))
        print(field.get("ValueDetection")["Geometry"])
```

```
    else:
        print("Value Detection - No values returned")

def process_text_detection(bucket, document):
    # Get the document from S3
    s3_connection = boto3.resource('s3')
    s3_object = s3_connection.Object(bucket, document)
    s3_response = s3_object.get()

    # opening binary stream using an in-memory bytes buffer
    stream = io.BytesIO(s3_response['Body'].read())

    # loading stream into image
    image = Image.open(stream)

    # Detect text in the document
    client = boto3.client('textract', region_name="us-east-1")

    # process using S3 object
    response = client.analyze_expense(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    # Set width and height to display image and draw bounding boxes
    # Create drawing object
    width, height = image.size
    draw = ImageDraw.Draw(image)

    for expense_doc in response["ExpenseDocuments"]:
        for line_item_group in expense_doc["LineItemGroups"]:
            for line_items in line_item_group["LineItems"]:
                for expense_fields in line_items["LineItemExpenseFields"]:
                    print_labels_and_values(expense_fields)
                    print()

        print("Summary:")
        for summary_field in expense_doc["SummaryFields"]:
            print_labels_and_values(summary_field)
            print()

        #For draw bounding boxes
        for line_item_group in expense_doc["LineItemGroups"]:
            for line_items in line_item_group["LineItems"]:
                for expense_fields in line_items["LineItemExpenseFields"]:
                    for key, val in expense_fields["ValueDetection"].items():
```

```
                        if "Geometry" in key:
                            draw_bounding_box(key, val, width, height, draw)

        for label in expense_doc["SummaryFields"]:
            if "LabelDetection" in label:
                for key, val in label["LabelDetection"].items():
                    draw_bounding_box(key, val, width, height, draw)

    # Display the image
    image.show()

def main():
    bucket = 'Bucket-Name'
    document = 'Document-Name'
    process_text_detection(bucket, document)

if __name__ == "__main__":
    main()
```

Java

```
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.*;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.textract.TextractClient;
```

```
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseRequest;
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseResponse;
import software.amazon.awssdk.services.textract.model.BoundingBox;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.ExpenseDocument;
import software.amazon.awssdk.services.textract.model.ExpenseField;
import software.amazon.awssdk.services.textract.model.LineItemFields;
import software.amazon.awssdk.services.textract.model.LineItemGroup;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.model.Point;

/**
 *
 * Demo code to parse Textract AnalyzeExpense API
 *
 */
public class TextractAnalyzeExpenseSample extends JPanel {

 private static final long serialVersionUID = 1L;

 BufferedImage image;
 static AnalyzeExpenseResponse result;

 public TextractAnalyzeExpenseSample(AnalyzeExpenseResponse documentResult,
 BufferedImage bufImage) throws Exception {
  super();

  result = documentResult; // Results of analyzeexpense summaryfields and
 lineitemgroups detection.
  image = bufImage; // The image containing the document.

 }

 // Draws the image and text bounding box.
 public void paintComponent(Graphics g) {

  Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

  // Draw the image.
  g2d.drawImage(image, 0, 0, image.getWidth(this), image.getHeight(this), this);

  // Iterate through summaryfields and lineitemgroups and display boundedboxes
 around lines of detected label and value.
  List<ExpenseDocument> expenseDocuments = result.expenseDocuments();
```

```
 for (ExpenseDocument expenseDocument : expenseDocuments) {

  if (expenseDocument.hasSummaryFields()) {
   DisplayAnalyzeExpenseSummaryInfo(expenseDocument);
   List<ExpenseField> summaryfields = expenseDocument.summaryFields();
   for (ExpenseField summaryfield : summaryfields) {

    if (summaryfield.valueDetection() != null) {
     ShowBoundingBox(image.getHeight(this), image.getWidth(this),
       summaryfield.valueDetection().geometry().boundingBox(), g2d, new
Color(0, 0, 0));
    }

    if (summaryfield.labelDetection() != null) {

     ShowBoundingBox(image.getHeight(this), image.getWidth(this),
       summaryfield.labelDetection().geometry().boundingBox(), g2d, new
Color(0, 0, 0));

    }

   }

  }

  if (expenseDocument.hasLineItemGroups()) {
   DisplayAnalyzeExpenseLineItemGroupsInfo(expenseDocument);

   List<LineItemGroup> lineitemgroups = expenseDocument.lineItemGroups();

   for (LineItemGroup lineitemgroup : lineitemgroups) {

    if (lineitemgroup.hasLineItems()) {

     List<LineItemFields> lineItems = lineitemgroup.lineItems();
     for (LineItemFields lineitemfield : lineItems) {

      if (lineitemfield.hasLineItemExpenseFields()) {

       List<ExpenseField> expensefields =
lineitemfield.lineItemExpenseFields();
       for (ExpenseField expensefield : expensefields) {

        if (expensefield.valueDetection() != null) {
```

```
          ShowBoundingBox(image.getHeight(this), image.getWidth(this),
            expensefield.valueDetection().geometry().boundingBox(), g2d,
            new Color(0, 0, 0));
         }

         if (expensefield.labelDetection() != null) {
          ShowBoundingBox(image.getHeight(this), image.getWidth(this),
            expensefield.labelDetection().geometry().boundingBox(), g2d,
            new Color(0, 0, 0));
         }

        }
       }

      }

     }

    }

  }
 }

}

// Show bounding box at supplied location.
private void ShowBoundingBox(float imageHeight, float imageWidth, BoundingBox
box, Graphics2D g2d, Color color) {

 float left = imageWidth * box.left();
 float top = imageHeight * box.top();

 // Display bounding box.
 g2d.setColor(color);
 g2d.drawRect(Math.round(left), Math.round(top), Math.round(imageWidth *
box.width()),
   Math.round(imageHeight * box.height()));

}

private void ShowSelectedElement(float imageHeight, float imageWidth,
BoundingBox box, Graphics2D g2d,
  Color color) {
```

```
 float left = (float) imageWidth * (float) box.left();
 float top = (float) imageHeight * (float) box.top();
 System.out.println(left);
 System.out.println(top);

 // Display bounding box.
 g2d.setColor(color);
 g2d.fillRect(Math.round(left), Math.round(top), Math.round(imageWidth *
box.width()),
   Math.round(imageHeight * box.height()));

}

// Shows polygon at supplied location
private void ShowPolygon(int imageHeight, int imageWidth, List<Point> points,
Graphics2D g2d) {

 g2d.setColor(new Color(0, 0, 0));
 Polygon polygon = new Polygon();

 // Construct polygon and display
 for (Point point : points) {
  polygon.addPoint((Math.round(point.x() * imageWidth)), Math.round(point.y() *
imageHeight));
 }
 g2d.drawPolygon(polygon);
}

private void DisplayAnalyzeExpenseSummaryInfo(ExpenseDocument expensedocument)
{
 System.out.println(" ExpenseId : " + expensedocument.expenseIndex());
 System.out.println("    Expense Summary information:");
 if (expensedocument.hasSummaryFields()) {

  List<ExpenseField> summaryfields = expensedocument.summaryFields();

  for (ExpenseField summaryfield : summaryfields) {

   System.out.println("    Page: " + summaryfield.pageNumber());
   if (summaryfield.type() != null) {

    System.out.println("    Expense Summary Field Type:" +
summaryfield.type().text());
```

```
   }
   if (summaryfield.labelDetection() != null) {

    System.out.println("    Expense Summary Field Label:" +
summaryfield.labelDetection().text());
    System.out.println("    Geometry");
    System.out.println("        Bounding Box: "
      + summaryfield.labelDetection().geometry().boundingBox().toString());
    System.out.println(
      "        Polygon: " +
summaryfield.labelDetection().geometry().polygon().toString());

   }
   if (summaryfield.valueDetection() != null) {
    System.out.println("    Expense Summary Field Value:" +
summaryfield.valueDetection().text());
    System.out.println("    Geometry");
    System.out.println("        Bounding Box: "
      + summaryfield.valueDetection().geometry().boundingBox().toString());
    System.out.println(
      "        Polygon: " +
summaryfield.valueDetection().geometry().polygon().toString());

   }

  }

 }

}

private void DisplayAnalyzeExpenseLineItemGroupsInfo(ExpenseDocument
expensedocument) {

 System.out.println(" ExpenseId : " + expensedocument.expenseIndex());
 System.out.println("    Expense LineItemGroups information:");

 if (expensedocument.hasLineItemGroups()) {

  List<LineItemGroup> lineitemgroups = expensedocument.lineItemGroups();

  for (LineItemGroup lineitemgroup : lineitemgroups) {
```

```
   System.out.println("    Expense LineItemGroupsIndexID :" +
lineitemgroup.lineItemGroupIndex());

   if (lineitemgroup.hasLineItems()) {

    List<LineItemFields> lineItems = lineitemgroup.lineItems();

    for (LineItemFields lineitemfield : lineItems) {

     if (lineitemfield.hasLineItemExpenseFields()) {

      List<ExpenseField> expensefields = lineitemfield.lineItemExpenseFields();
      for (ExpenseField expensefield : expensefields) {

       if (expensefield.type() != null) {
        System.out.println("    Expense LineItem Field Type:" +
expensefield.type().text());

       }

       if (expensefield.valueDetection() != null) {
        System.out.println(
          "    Expense Summary Field Value:" +
expensefield.valueDetection().text());
        System.out.println("    Geometry");
        System.out.println("        Bounding Box: "
          + expensefield.valueDetection().geometry().boundingBox().toString());
        System.out.println("        Polygon: "
          + expensefield.valueDetection().geometry().polygon().toString());

       }

       if (expensefield.labelDetection() != null) {
        System.out.println(
          "    Expense LineItem Field Label:" +
expensefield.labelDetection().text());
        System.out.println("    Geometry");
        System.out.println("        Bounding Box: "
          + expensefield.labelDetection().geometry().boundingBox().toString());
        System.out.println("        Polygon: "
          + expensefield.labelDetection().geometry().polygon().toString());
       }

      }
```

```
      }

     }

    }
   }
  }

 }

 public static void main(String arg[]) throws Exception {

  // Creates a default async client with credentials and AWS Region loaded from
  // the
  // environment


  S3AsyncClient client =
 S3AsyncClient.builder().region(Region.US_EAST_1).build();

  System.out.println("Creating the S3 Client");

  // Start the call to Amazon S3, not blocking to wait for the result
  CompletableFuture<ResponseBytes<GetObjectResponse>> responseFuture =
 client.getObject(
    GetObjectRequest.builder().bucket("textractanalyzeexpense").key("input/
sample-receipt.jpg").build(),
    AsyncResponseTransformer.toBytes());

  System.out.println("Successfully read the object");

  // When future is complete (either successfully or in error), handle the
  // response
  CompletableFuture<ResponseBytes<GetObjectResponse>> operationCompleteFuture =
 responseFuture
    .whenComplete((getObjectResponse, exception) -> {
     if (getObjectResponse != null) {
      // At this point, the file my-file.out has been created with the data
      // from S3; let's just print the object version
      // Convert this into Async call and remove the below block from here and
 put it
      // outside
```

```
      TextractClient textractclient =
 TextractClient.builder().region(Region.US_EAST_1).build();

      AnalyzeExpenseRequest request = AnalyzeExpenseRequest.builder()
        .document(
          Document.builder().s3Object(S3Object.builder().name("YOURObjectName")
            .bucket("YOURBucket").build()).build())
        .build();

      AnalyzeExpenseResponse result = textractclient.analyzeExpense(request);

      System.out.print(result.toString());

      ByteArrayInputStream bais = new
 ByteArrayInputStream(getObjectResponse.asByteArray());
      try {
       BufferedImage image = ImageIO.read(bais);
       System.out.println("Successfully read the image");
       JFrame frame = new JFrame("Expense Image");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       TextractAnalyzeExpense panel = new TextractAnalyzeExpense(result, image);
       panel.setPreferredSize(new Dimension(image.getWidth(),
 image.getHeight()));
       frame.setContentPane(panel);
       frame.pack();
       frame.setVisible(true);
      } catch (IOException e) {
       throw new RuntimeException(e);
      } catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
     } else {
      // Handle the error
      exception.printStackTrace();
     }
    });

  // We could do other work while waiting for the AWS call to complete in
  // the background, but we'll just wait for "whenComplete" to finish instead
  operationCompleteFuture.join();

 }
}
```

Node.Js

```
                    // Import required AWS SDK clients and commands for Node.js
import { AnalyzeExpenseCommand } from  "@aws-sdk/client-textract";
import  { TextractClient } from "@aws-sdk/client-textract";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
// Create SNS service object.
const textractClient = new TextractClient({ region: REGION });

const bucket = 'bucket'
const photo = 'photo'

// Set params
const params = {
    Document: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
  }

const process_text_detection = async () => {
    try {
        const aExpense = new AnalyzeExpenseCommand(params);
        const response = await textractClient.send(aExpense);
        //console.log(response)
        response.ExpenseDocuments.forEach(doc => {
            doc.LineItemGroups.forEach(items => {
                items.LineItems.forEach(fields => {
                    fields.LineItemExpenseFields.forEach(expenseFields =>{
                        console.log(expenseFields)
                    })
                }
                )}
                )
            }
```

```
        )
        return response; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

process_text_detection()
```

4. 그러면 에 대한 JSON 출력을 제공합니다.AnalyzeExpense작업을 수행합니다.

Amazon Textract Textract를 사용하여 자격 증명 문서 분석

ID 문서를 분석하려면 AnalyzeId API를 사용하고 문서 파일을 입력으로 전달합니다.AnalyzeID는 분석된 텍스트를 포함하는 JSON 구조를 반환합니다. 자세한 정보는 ID 문서 분석을 참조하십시오.

입력 문서를 이미지 바이트 어레이 (base64 인코딩 이미지 바이트) 또는 Amazon S3 객체로 제공할 수 있습니다. 이 절차에서는 S3 버킷에 이미지 파일을 업로드하고 파일 이름을 지정합니다.

ID 문서 (API) 를 분석하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.

   a. 을 사용하여 IAM 사용자를 생성 또는 업데이트합니다.AmazonTextractFullAccess과AmazonS3ReadOnlyAccess권한. 자세한 정보는 1단계: AWS 계정 설정 및 IAM 사용자 만들기을 참조하십시오.

   b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 정보는 2단계: 설정AWS CLI과 AWSSDK을 참조하십시오.

2. 문서가 포함된 이미지를 S3 버킷에 업로드합니다.

   지침은 단원을 참조하십시오.Amazon S3 객체 업로드의Amazon Simple Storage Service.

3. 다음 예제를 사용하여 AnalyzeID 작업을 호출합니다.

   CLI

   다음 예제에서는 S3 버킷에서 입력 파일을 가져와AnalyzeID그것에 대한 작업. 아래 코드에서 다음 값을 바꿉니다.##S3 버킷의 이름을 사용하여##을 사용하여 버킷의 파일 이름과 의 값을 사용하여##의 이름과 함께region계정과 연결됩니다.

```
aws textract analyze-id --document-pages '[{"S3Object":
{"Bucket":"bucket","Name":"name"}}]' --region region
```

입력에 다른 S3 객체를 추가하여 운전 면허증의 앞면과 뒷면을 사용하여 API를 호출할 수도 있습니다.

```
aws textract analyze-id --document-pages '[{"S3Object":
{"Bucket":"bucket","Name":"name front"}}, {"S3Object":
{"Bucket":"bucket","Name":"name back"}}]' --region us-east-1
```

Windows 장치에서 CLI에 액세스하는 경우 작은 따옴표 대신 큰따옴표를 사용하고 백슬래시 (예:\) 로 내부 큰따옴표를 이스케이프하여 발생할 수 있는 파서 오류를 해결합니다. 예제는 아래를 참조하십시오.

```
aws textract analyze-id --document-pages "[{\"S3Object\":{\"Bucket\":\"bucket\",
\"Name\":\"name\"}}]" --region region
```

Python

다음 예제에서는 S3 버킷에서 입력 파일을 가져와AnalyzeID이 작업에 대해 검색된 키-값 페어를 반환합니다. 아래 코드에서 다음 값을 바꿉니다.*bucket_name*S3 버킷의 이름을 사용하여*file_name*을 사용하여 버킷의 파일 이름과 의 값을 사용하여##의 이름과 함께region계정과 연결됩니다.

```
import boto3

bucket_name = "bucket-name"
file_name = "file-name"
region = "region-name"

def analyze_id(region, bucket_name, file_name):

    textract_client = boto3.client('textract', region_name=region)
    response = textract_client.analyze_id(DocumentPages=[{"S3Object":
{"Bucket":bucket_name,"Name":file_name}}])
```

```
    for doc_fields in response['IdentityDocuments']:
        for id_field in doc_fields['IdentityDocumentFields']:
            for key, val in id_field.items():
                if "Type" in str(key):
                    print("Type: " + str(val['Text']))
            for key, val in id_field.items():
                if "ValueDetection" in str(key):
                    print("Value Detection: " + str(val['Text']))
            print()

analyze_id(region, bucket_name, file_name)
```

Java

다음 예제에서는 S3 버킷에서 입력 파일을 가져와AnalyzeID작업, 감지 된 데이터를 반환합니다. main 함수에서 다음 값을 바꿉니다.s3bucket과sourceDoc2단계에서 사용한 Amazon S3 버킷과 문서 이미지의 이름이 표시됩니다.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.amazonaws.samples;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.textract.AmazonTextractClient;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.*;
import java.util.ArrayList;
import java.util.List;

public class AnalyzeIdentityDocument {

    public static void main(String[] args) {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <s3bucket><sourceDoc> \n\n" +
                "Where:\n" +
                "    s3bucket - the Amazon S3 bucket where the document is
 located. \n" +
```

```
                "    sourceDoc - the name of the document. \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String s3bucket = "bucket-name"; //args[0];
        String sourceDoc = "sourcedoc-name";  //args[1];
        AmazonTextractClient textractClient = (AmazonTextractClient)
AmazonTextractClientBuilder.standard()
                .withRegion(Regions.US_EAST_1)
                .build();

        getDocDetails(textractClient, s3bucket, sourceDoc);
    }

    public static void getDocDetails(AmazonTextractClient textractClient, String
s3bucket, String sourceDoc ) {

       try {

            S3Object s3 = new S3Object();
            s3.setBucket(s3bucket);
            s3.setName(sourceDoc);

            com.amazonaws.services.textract.model.Document myDoc = new
com.amazonaws.services.textract.model.Document();
            myDoc.setS3Object(s3);

            List<Document> list1 = new ArrayList();
            list1.add(myDoc);

            AnalyzeIDRequest idRequest = new AnalyzeIDRequest();
            idRequest.setDocumentPages(list1);

            AnalyzeIDResult result = textractClient.analyzeID(idRequest);
            List<IdentityDocument> docs =  result.getIdentityDocuments();
            for (IdentityDocument doc: docs) {

                List<IdentityDocumentField>idFields =
doc.getIdentityDocumentFields();
                for (IdentityDocumentField field: idFields) {
```

```
                    System.out.println("Field type is "+
 field.getType().getText());
                    System.out.println("Field value is "+
 field.getValueDetection().getText());
                }
            }

       } catch (Exception e) {
            e.printStackTrace();
       }
    }
}
```

4. 그러면 에 대한 JSON 출력을 제공합니다.AnalyzeID작업을 수행합니다.

비동기 작업을 사용한 문서 처리

Amazon Textract Textract는 PDF 또는 TIFF 형식의 다중 페이지 문서의 텍스트를 감지하고 분석할 수 있습니다. 여기에는 송장 및 영수증이 포함됩니다. 다중 페이지 문서 처리는 비동기식 작업입니다. 문서의 비동기 처리는 대용량 다중 페이지 문서를 처리하는 데 유용합니다. 예를 들어 1,000페이지가 넘는 PDF 파일을 처리하는 데 시간이 걸립니다. PDF 파일을 비동기적으로 처리하면 응용 프로그램이 프로세스가 완료될 때까지 기다리는 동안 다른 작업을 완료할 수 있습니다.

이 단원에서는 Amazon Textract Textract를 사용하여 여러 페이지 또는 단일 페이지 문서의 텍스트를 비동기적으로 감지하고 분석하는 방법에 대해 설명합니다. 다중 페이지 문서는 PDF 또는 TIFF 형식이어야 합니다. 비동기 작업으로 처리된 단일 페이지 문서는 JPEG, PNG, TIFF 또는 PDF 형식일 수 있습니다.

Amazon Textract 비동기식 작업은 다음과 같은 용도로 사용할 수 있습니다.

- 텍스트 감지 — 여러 페이지로 된 문서에서 줄과 단어를 감지할 수 있습니다. 비동기 작업은 다음과 같습니다.StartDocumentTextDetection과GetDocumentTextDetection. 자세한 정보는 텍스트 감지을 참조하십시오.
- 텍스트 분석 — 여러 페이지로 된 문서에서 검색된 텍스트 간의 관계를 식별할 수 있습니다. 비동기 작업은 다음과 같습니다.StartDocumentAnalysis과GetDocumentAnalysis. 자세한 정보는 문서 분석을 참조하십시오.
- 비용 분석 — 여러 페이지 송장 및 영수증에 대한 데이터 관계를 식별할 수 있습니다. Amazon Textract Textract는 여러 페이지로 된 문서의 각 송장 또는 영수증 페이지를 개별 영수증 또는 송장으로 취급합니다. 여러 페이지 문서의 한 페이지에서 다른 페이지로의 컨텍스트를 유지하지 않습니다. 비동기 작업은 다음과 같습니다.StartExpenseAnalysis과GetExpenseAnalysis. 자세한 정보는 송장 및 수금 분석을 참조하십시오.

주제

- Amazon Textract 비동기 작업 호출
- 비동기 작업을 위한 Amazon Textract 구성
- 다중 페이지 문서에서 텍스트 감지 또는 분석
- Amazon Textract TextractResults 알림

Amazon Textract 비동기 작업 호출

Amazon Textract Textract는 PDF 또는 TIFF 형식으로 여러 페이지 문서를 처리하는 데 사용할 수 있는 비동기 API를 제공합니다. 비동기 작업을 사용하여 JPEG, PNG, TIFF 또는 PDF 형식의 단일 페이지 문서를 처리할 수도 있습니다.

이 주제의 정보는 텍스트 검색 작업을 사용하여 Amazon Textract 비동기 작업을 사용하는 방법을 보여줍니다. 동일한 접근 방식이 텍스트 분석 작업에서 작동합니다.the section called “StartDocumentAnalysis”과the section called “GetDocumentAnalysis”. 또한 에서 동일하게 작동합니다.the section called “StartExpenseAnalysis”과the section called “GetExpenseAnalysis”.

문제 해결 예는 다중 페이지 문서에서 텍스트 감지 또는 분석을(를) 참조하십시오.

Amazon Textract S3 버킷에 저장된 문서를 비동기식으로 처리합니다. a를 호출하여 처리를 시작합니다.`Start`다음과 같은 작업StartDocumentTextDetection. 요청 완료 상태가 Amazon Simple Notification Service (Amazon SNS) 주제에 게시됩니다. Amazon SNS 주제에서 완료 상태를 가져오려면 Amazon Simple Queue Service (Amazon SQS) 대기열 또는AWS Lambda함수. 완료 상태가 되면 GetDocumentTextDetection 같은 `Get` 작업을 불러와 요청 결과를 가져옵니다.

작업을 사용하여 Amazon S3 버킷을 지정하지 않는 한 비동기 호출 결과는 기본적으로 Amazon Textract 소유 버킷에 7일 동안 암호화되어 저장됩니다.`OutputConfig`인수.

다음 표에는 Amazon Textract Textract에서 지원하는 다양한 유형의 비동기 처리에 대한 해당 시작 및 Get 작업이 나와 있습니다.

Amazon Textract 비동기 작업을 위한 API 작업 시작/가져오기

처리 유형	API 시작	API 가져오기
텍스트 감지	StartDocumentTextDetection	GetDocumentTextDetection
텍스트 분석	StartDocumentAnalysis	GetDocumentAnalysis
비용 분석	시작익스텐스분석	GetExpense분석

를 사용하는 예제의 경우AWS Lambda함수, 참조Amazon Textract Textract를 사용한 대규모 문서 처리.

다음 그림은 Amazon S3 버킷에 저장된 문서 이미지에서 문서 텍스트를 감지하는 프로세스를 보여줍니다. 이 다이어그램에서 Amazon SQS 대기열은 Amazon SNS 주제의 완료 상태를 가져옵니다.

위 다이어그램에서 표시하는 프로세스는 텍스트 및 송장/영수증을 분석할 때 동일합니다. 호출하여 텍스트 분석을 시작합니다.the section called “StartDocumentAnalysis”전화를 통해 인보이스/영수증 분석을 시작합니다.the section called “StartExpenseAnalysis”호출하여 결과를 얻을 수 있습니다.the section called “GetDocumentAnalysis”또는the section called “GetExpenseAnalysis”각각.

텍스트 감지 시작

을 불러와 Amazon Textract 텍스트 감지 요청을 시작합니다.StartDocumentTextDetection. 다음은 `StartDocumentTextDetection`으로 전달되는 JSON 요청의 예입니다.

```
{
    "DocumentLocation": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "image.pdf"
        }
    },
    "ClientRequestToken": "DocumentDetectionToken",
    "NotificationChannel": {
        "SNSTopicArn": "arn:aws:sns:us-east-1:nnnnnnnnnn:topic",
        "RoleArn": "arn:aws:iam::nnnnnnnnnn:role/roleTopic"
    },
    "JobTag": "Receipt"
}
```

입력 파라미터`DocumentLocation`에서는 문서 파일 이름과 파일을 검색할 수 있는 Amazon S3 버킷을 제공합니다.`NotificationChannel`텍스트 검색 요청이 완료될 때 Amazon Textract가 통지하는 Amazon SNS 주제의 Amazon 리소스 이름 (ARN) 이 포함됩니다. Amazon SNS 주제는 호출하는 Amazon Textract 엔드포인트와 동일한 AWS 리전에 있어야 합니다.`NotificationChannel`또한 Amazon Textract SNS 주제에 게시할 때 사용할 수 있는 역할의 ARN도 있습니다. Amazon Textract Textract는 IAM 서비스 역할을 생성하여 Amazon SNS 주제에 Amazon Textract의 게시 권한을 줍니다. 자세한 정보는 비동기 작업을 위한 Amazon Textract 구성을 참조하십시오.

선택적 입력 파라미터를 지정할 수도 있습니다.`JobTag`이렇게 하면 Amazon SNS 주제에 게시된 완료 상태에서 작업 또는 작업 그룹을 식별할 수 있습니다. 예를 들어 를 사용할 수 있습니다.`JobTag`세금 양식 또는 영수증과 같이 처리되는 문서의 유형을 식별합니다.

분석 작업이 실수로 중복되지 않도록 idempotent 토큰 `ClientRequestToken`을 선택적으로 제공할 수 있습니다. 에 대한 값을 제공하는 경우`ClientRequestToken`,`Start`작업이 동일하게 반환됩니

다.JobId여러 개의 동일한 호출에 대해Start다음과 같은 작업StartDocumentTextDetection. ClientRequestToken 토큰은 수명이 7일입니다. 7일 후에 다시 사용할 수 있습니다. 토큰 수명 기간 동안 토큰을 재사용할 경우 다음과 같은 현상이 생깁니다.

- 동일한 Start 작업과 동일한 입력 파라미터로 토큰을 다시 사용할 경우 동일한 JobId가 반환됩니다. 이 작업은 다시 실행되지 않으며 Amazon Textract TExtract는 등록된 Amazon SNS 주제에 완료 상태를 전송하지 않습니다.
- 동일한 Start 작업에서 약간의 입력 파라미터를 변경하여 토큰을 재사용할 경우 idempotentparametermismatchexception(HTTP 상태 코드: 400) 예외가 표시됩니다.
- 다른 Start 작업에 토큰을 재사용할 경우 작업이 진행됩니다.

사용 가능한 또 다른 선택적 매개 변수는OutputConfig를 사용하면 출력 위치를 조정할 수 있습니다. 기본적으로 Amazon Textract Textract는 결과를 내부적으로 저장하며 Get API 작업에서만 액세스할 수 있습니다. 다음으로 바꿉니다.OutputConfig활성화하면 출력이 전송될 버킷의 이름과 결과를 다운로드할 수 있는 결과의 파일 접두사를 설정할 수 있습니다. 또한 를 설정할 수 있습니다.KMSKeyID고객 관리 키에 대한 매개 변수를 사용하여 출력을 암호화합니다. 이 매개 변수를 설정하지 않으면 Amazon Textract Textract는 다음을 사용하여 서버 측을 암호화합니다.AWS 관리형 키 Amazon S3용

> **Note**
>
> 이 매개 변수를 사용하기 전에 출력 버킷에 대한 PutObject 권한이 있는지 확인하십시오. 또한, 에 대한 암호 해독, ReEncrypt, GenerateDataKey 및 DescribeKey 권한이 있는지 확인합니다.AWS KMS키를 사용하기로 결정한 경우.

StartDocumentTextDetection 작업에 대한 응답은 작업 식별자입니다(JobId). 사용JobId을 사용하여 Amazon Textract가 Amazon SNS 주제에 완료 상태를 게시한 후에 요청을 추적하고 분석 결과를 가져옵니다. 다음은 그 한 예입니다.

```
{"JobId":"270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3"}
```

너무 많은 작업을 동시에 시작하는 경우StartDocumentTextDetection를 올리십시오.LimitExceededException동시 실행 작업의 수가 Amazon Textract 서비스 제한 미만이 될 때까지 예외 (HTTP 상태 코드: 400) 입니다.

다수의 작업으로 인해 LimitExceededException 예외가 발생하는 경우에는 Amazon SQS 대기열을 사용하여 수신되는 요청을 관리하는 것이 좋습니다. 연락처AWSAmazon SQS 대기열로도 평균 동시 요청 수를 관리하지 못하여 계속 수신할 경우에는 SupportLimitExceededException예외.

Amazon Textract 분석 요청의 완료 상태 가져오기

Amazon Textract Textract는 등록된 Amazon SNS 주제로 분석 완료 알림을 보냅니다. 알림에는 JSON 문자열 형태의 작업 완료 상태와 작업 식별자가 포함됩니다. 성공적인 텍스트 감지 요청에는SUCCEEDED상태가 됩니다. 예를 들어, 다음 결과는 텍스트 감지 작업의 성공적인 처리를 보여줍니다.

```
{
    "JobId": "642492aea78a86a40665555dc375ee97bc963f342b29cd05030f19bd8fd1bc5f",
    "Status": "SUCCEEDED",
    "API": "StartDocumentTextDetection",
    "JobTag": "Receipt",
    "Timestamp": 1543599965969,
    "DocumentLocation": {
        "S3ObjectName": "document",
        "S3Bucket": "bucket"
    }
}
```

자세한 정보는 Amazon Textract TextractResults 알림을 참조하십시오.

Amazon Textract가 Amazon SNS 주제에 게시한 상태 정보를 가져오려면 다음 옵션 중 하나를 사용합니다.

- AWS Lambda— 구독할 수 있습니다AWS LambdaAmazon SNS 주제에 기록하는 함수입니다. 이 함수는 Amazon Textract Textract가 Amazon SNS 주제에 요청 완료 사실을 알릴 때 호출됩니다. 서버 측 코드로 텍스트 감지 요청 결과를 처리하도록 하려면 Lambda 함수를 사용합니다. 예를 들어, 서버 측 코드를 사용하여 이미지에 주석을 달거나 감지된 텍스트에 관한 보고서를 생성한 후에 정보를 클라이언트 애플리케이션으로 반환해야 하는 경우가 있을 수 있습니다.
- Amazon SQS— Amazon SQS 대기열에서 Amazon SNS 주제를 구독할 수 있습니다. 그런 다음 Amazon SQS 대기열을 폴링하여 텍스트 검색 요청이 완료될 때 Amazon Textract Textract가 게시한 완료 상태를 검색합니다. 자세한 정보는 다중 페이지 문서에서 텍스트 감지 또는 분석을 참조하십시오. Amazon Textract 작업을 클라이언트 애플리케이션에서만 불러와야 할 경우에는 Amazon SQS 대기열을 사용합니다.

> **Important**
>
> Amazon Textract 을 반복해서 불러와 요청 완료 상태를 가져오는 것은 권장하지 않습니다.`Get`작업을 사용합니다. 이는 Amazon Textract Textract가`Get`요청이 너무 많을 경우 작업을 수행합니다. 여러 문서를 동시에 처리할 경우에는 Amazon Textract Textract가 각 작업의 상태를 개별적으로 확인하기 위해 폴링하는 것보다는 하나의 SQS 대기열을 완료 알림을 모니터링하는 것이 더 효율적이고 간단합니다.

Amazon Textract 텍스트 탐지 결과 얻기

텍스트 탐지 요청의 결과를 가져오려면 먼저 Amazon SNS 주제에서 검색된 완료 상태가 인지 확인합니다.`SUCCEEDED`. 그런 다음 `GetDocumentTextDetection`을 호출하면 이에 의해 `StartDocumentTextDetection`에서 반환된 `JobId` 값이 통과됩니다. 요청 JSON은 다음 예제와 비슷합니다.

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
    "MaxResults": 10,
    "SortBy": "TIMESTAMP"
}
```

`JobId`는 텍스트 감지 작업의 식별자입니다. 텍스트 감지는 대량의 데이터를 생성할 수 있으므로`MaxResults`단일 결과로 반환할 최대 결과 수를 지정합니다.`Get`작업을 사용합니다. 의 기본값`MaxResults`1,000입니다. 1,000보다 큰 값을 지정한 경우에는 1,000개의 결과만 반환됩니다. 작업에서 모든 결과가 반환되지 않을 경우에는 다음 페이지의 페이지 매김 토큰이 반환됩니다. 결과의 다음 페이지를 가져오려면 에서 토큰을 지정하십시오.`NextToken`파라미터.

> **Note**
>
> Amazon Textract Textract는 비동기식 작업의 결과를 7일 동안 보유합니다. 이 후에는 결과를 검색할 수 없습니다.

이`GetDocumentTextDetection`작업 응답 JSON은 다음과 유사합니다. 검색된 총 페이지 수가 반환됩니다.`DocumentMetadata`. 검색된 텍스트는 다음 위치에 반환됩니다.`Blocks`어레이. 에 대한 내용은`Block`객체, 참조텍스트 감지 및 문서 분석 응답 객체.

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "JobStatus": "SUCCEEDED",
    "Blocks": [
        {
            "BlockType": "PAGE",
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Height": 1.0,
                    "Left": 0.0,
                    "Top": 0.0
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.0
                    },
                    {
                        "X": 1.0,
                        "Y": 0.0
                    },
                    {
                        "X": 1.0,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
            },
            "Id": "64533157-c47e-401a-930e-7ca1bb3ac3fa",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "4297834d-dcb1-413b-8908-3b96866ebbb5",
                        "1d85ba24-2877-4d09-b8b2-393833d769e9",
                        "193e9c47-fd87-475a-ba09-3fda210d8784",
                        "bd8aeb62-961b-4b47-b78a-e4ed9eeecd0f"
```

```
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 53.301639556884766,
            "Text": "ellooworio",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.9999999403953552,
                    "Height": 0.5365243554115295,
                    "Left": 0.0,
                    "Top": 0.46347561478614807
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 0.9999999403953552,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 0.9999999403953552,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
            },
            "Id": "4297834d-dcb1-413b-8908-3b96866ebbb5",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "170c3eb9-5155-4bec-8c44-173bba537e70"
                    ]
                }
            ],
```

```
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 89.15632629394531,
            "Text": "He llo,",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33642634749412537,
                    "Height": 0.49159330129623413,
                    "Left": 0.13885067403316498,
                    "Top": 0.17169663310050964
                },
                "Polygon": [
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
                        "Y": 0.6632899641990662
                    },
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.6632899641990662
                    }
                ]
            },
            "Id": "1d85ba24-2877-4d09-b8b2-393833d769e9",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "516ae823-3bab-4f9a-9d74-ad7150d128ab",
                        "6bcf4ea8-bbe8-4686-91be-b98dd63bc6a6"
                    ]
                }
            ],
            "Page": 1
        },
```

```
        {
            "BlockType": "LINE",
            "Confidence": 82.44834899902344,
            "Text": "worlo",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33182239532470703,
                    "Height": 0.3766750991344452,
                    "Left": 0.5091826915740967,
                    "Top": 0.23131252825260162
                },
                "Polygon": [
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.607987642288208
                    },
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.607987642288208
                    }
                ]
            },
            "Id": "193e9c47-fd87-475a-ba09-3fda210d8784",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "ed135c3b-35dd-4085-8f00-26aedab0125f"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 88.50325775146484,
```

```
            "Text": "world",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.35004907846450806,
                    "Height": 0.19635874032974243,
                    "Left": 0.527581512928009,
                    "Top": 0.30100569128990173
                },
                "Polygon": [
                    {
                        "X": 0.527581512928009,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.49736443161964417
                    },
                    {
                        "X": 0.527581512928009,
                        "Y": 0.49736443161964417
                    }
                ]
            },
            "Id": "bd8aeb62-961b-4b47-b78a-e4ed9eeecd0f",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "9e28834d-798e-4a62-8862-a837dfd895a6"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 53.301639556884766,
            "Text": "ellooworio",
            "Geometry": {
                "BoundingBox": {
```

```
                    "Width": 1.0,
                    "Height": 0.5365243554115295,
                    "Left": 0.0,
                    "Top": 0.46347561478614807
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 1.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 1.0,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
            },
            "Id": "170c3eb9-5155-4bec-8c44-173bba537e70",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 88.46246337890625,
            "Text": "He",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.15350718796253204,
                    "Height": 0.29955607652664185,
                    "Left": 0.13885067403316498,
                    "Top": 0.21856294572353363
                },
                "Polygon": [
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.21856294572353363
                    },
                    {
```

```
                    "X": 0.292357861995697,
                    "Y": 0.21856294572353363
                },
                {
                    "X": 0.292357861995697,
                    "Y": 0.5181190371513367
                },
                {
                    "X": 0.13885067403316498,
                    "Y": 0.5181190371513367
                }
            ]
        },
        "Id": "516ae823-3bab-4f9a-9d74-ad7150d128ab",
        "Page": 1
    },
    {
        "BlockType": "WORD",
        "Confidence": 89.8501968383789,
        "Text": "llo,",
        "Geometry": {
            "BoundingBox": {
                "Width": 0.17724157869815826,
                "Height": 0.49159327149391174,
                "Left": 0.2980354428291321,
                "Top": 0.17169663310050964
            },
            "Polygon": [
                {
                    "X": 0.2980354428291321,
                    "Y": 0.17169663310050964
                },
                {
                    "X": 0.47527703642845154,
                    "Y": 0.17169663310050964
                },
                {
                    "X": 0.47527703642845154,
                    "Y": 0.6632899045944214
                },
                {
                    "X": 0.2980354428291321,
                    "Y": 0.6632899045944214
                }
```

```
                ]
            },
            "Id": "6bcf4ea8-bbe8-4686-91be-b98dd63bc6a6",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 82.44834899902344,
            "Text": "worlo",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33182239532470703,
                    "Height": 0.3766750991344452,
                    "Left": 0.5091826915740967,
                    "Top": 0.23131252825260162
                },
                "Polygon": [
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.607987642288208
                    },
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.607987642288208
                    }
                ]
            },
            "Id": "ed135c3b-35dd-4085-8f00-26aedab0125f",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 88.50325775146484,
            "Text": "world",
            "Geometry": {
                "BoundingBox": {
```

```
                    "Width": 0.35004907846450806,
                    "Height": 0.19635874032974243,
                    "Left": 0.527581512928009,
                    "Top": 0.30100569128990173
                },
                "Polygon": [
                    {
                        "X": 0.527581512928009,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.49736443161964417
                    },
                    {
                        "X": 0.527581512928009,
                        "Y": 0.49736443161964417
                    }
                ]
            },
            "Id": "9e28834d-798e-4a62-8862-a837dfd895a6",
            "Page": 1
        }
    ]
}
```


비동기 작업을 위한 Amazon Textract 구성

다음 절차에서는 Amazon Simple Notification Service (Amazon SNS) 주제 및 Amazon Simple Queue Service (Amazon SQS) 대기열과 함께 사용하도록 Amazon Textract를 구성하는 방법을 보여줍니다.

> **Note**
>
> 다음 지침을 사용하여 를 설정하는 경우[다중 페이지 문서에서 텍스트 감지 또는 분석](#)예를 들어 3단계부터 6단계까지 수행할 필요가 없습니다. 예제에는 Amazon SNS 주제 및 Amazon SQS 대기열을 생성하고 구성하기 위한 코드가 포함되어 있습니다.

Amazon Textract 구성하려면

1. 설정AWSAmazon Textract Textract에 액세스할 수 있는 계정 자세한 정보는 1단계: AWS 계정 설정 및 IAM 사용자 만들기을 참조하십시오.

   사용자에게 최소한 다음 권한이 있는지 확인합니다.

   - AmazonTextractFullAccess
   - AmazonS3ReadOnlyAccess
   - AmazonSNSFullAccess
   - AmazonSQSFullAccess
2. 필요한 AWS SDK를 설치하고 구성합니다. 자세한 정보는 2단계: 설정AWS CLI과AWSSDK을 참조하십시오.
3. Amazon SNS 주제 생성. 주제 이름 앞에 추가아마존 추출물. 주제 Amazon Resource Name(ARN)을 기록합니다. 주제가 와 동일한 리전에 있어야 합니다.AWSAWS 계정에서 사용 중인 엔드포인트입니다.
4. Amazon SQS 표준 대기열 생성를 사용하여Amazon SQS 콘솔. 대기열 ARN을 기록합니다.
5. 3단계에서 만든 주제에 대한 대기열을 구독합니다.
6. Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한 부여.
7. Amazon SNS 주제에 Amazon Textract 액세스 권한을 부여할 IAM 서비스 역할을 생성합니다. 서비스 역할의 Amazon Resource Name(ARN)을 적어둡니다. 자세한 정보는 Amazon SNS 주제에 대한 Amazon Textract 액세스 권한 부여을 참조하십시오.
8. 다음 인라인 정책을 추가합니다.1단계에서 생성한 IAM 사용자에게 전송합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MySid",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "Service role ARN from step 7"
        }
    ]
}
```

인라인 정책에 이름을 지정합니다.

9. 이제 에서 예시를 실행할 수 있습니다.다중 페이지 문서에서 텍스트 감지 또는 분석.

Amazon SNS 주제에 대한 Amazon Textract 액세스 권한 부여

Amazon Textract Textract는 비동기 작업이 완료되면 Amazon SNS 주제에 메시지를 보낼 수 있는 권한이 필요합니다. Amazon SNS 주제에 대한 Amazon Textract 액세스 권한을 부여할 IAM 서비스 역할을 사용합니다.

Amazon SNS 주제를 생성할 때 주제 이름 앞에 다음을 추가해야 합니다.**`AmazonTextract`**—예:**`AmazonTextractMyTopicName`**.

1. IAM 콘솔(https://console.aws.amazon.com/iam)에 로그인합니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성(Create role)을 선택합니다.
4. Select type of trusted entity(신뢰할 수 있는 엔터티 유형 선택)에서 AWS service(AWS 서비스)를 선택합니다.
5. 용이 역할을 사용할 서비스 선택, 선택Textract.
6. [다음: 권한(Next: Permissions)]을 선택합니다.
7. 다음을 확인합니다.AmazonTextractServiceRole정책이 연결된 정책 목록에 포함되었습니다. 목록에 정책을 표시하려면 정책 이름의 일부를필터 정책.
8. [다음: 권한(Next: Tags)]를 선택합니다.
9. 태그는 추가하지 않아도 되므로다음: Review)]를 선택합니다.
10. Review(검토) 섹션에서 Role name(역할 이름)에 역할의 이름을 입력합니다(예: `TextractRole`). In역할 설명을 사용하여 역할에 대한 설명을 업데이트한 후역할 생성.
11. 새 역할을 선택하여 역할의 세부 정보 페이지를 엽니다.
12. Summary(요약)에서 Role ARN(역할 ARN) 값을 복사해 저장합니다.
13. 신뢰 관계를 선택합니다.
14. 선택신뢰 관계 편집신뢰 정책이 다음과 같이 표시되는지 확인합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "textract.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

15. 신뢰 정책 업데이트(Update Trust Policy)를 선택합니다.

다중 페이지 문서에서 텍스트 감지 또는 분석

이 절차에서는 Amazon Textract 탐지 작업, Amazon S3 버킷에 저장된 문서, Amazon SNS 주제 및 Amazon SQS 대기열을 사용하여 다중 페이지 문서의 텍스트를 감지하거나 분석하는 방법을 보여줍니다. 다중 페이지 문서 처리는 비동기식 작업입니다. 자세한 정보는 Amazon Textract 비동기 작업 호출을 참조하십시오.

코드에서 수행할 처리 유형 (텍스트 검색, 텍스트 분석 또는 비용 분석) 을 선택할 수 있습니다.

처리 결과는 다음과 같은 배열로 반환됩니다.the section called “Block”객체: 를 사용하는 처리 유형에 따라 다릅니다.

여러 페이지로 된 문서에서 텍스트를 감지하거나 분석하려면 다음을 수행합니다.

1. Amazon SNS 주제 및 Amazon SQS 대기열을 생성합니다.
2. 주제에 대한 대기열을 구독합니다.
3. 주제에 대기열로 메시지를 전송할 권한을 부여합니다.
4. 문서 처리를 시작합니다. 선택한 분석 유형에 적합한 연산을 사용합니다.
   - StartDocumentTextDetection텍스트 감지 작업용
   - StartDocumentAnalysis텍스트 분석 작업용
   - StartExpenseAnalysis비용 분석 작업에 사용됩니다.
5. Amazon SQS 대기열에서 완료 상태를 가져옵니다. 예제 코드는 작업 식별자를 추적합니다 (`JobId`) 에 의해 반환됩니다.`Start`작업을 사용합니다. 완료 상태에서 읽은 작업 식별자와 일치하는 결과만 가져옵니다. 이는 다른 애플리케이션에서 동일한 대기열과 주제를 사용할 경우에 중요합니다. 간소화를 위해 이 예제에서는 일치하지 않는 작업을 삭제합니다. 삭제된 작업을 Amazon SQS 배달 못한 편지 대기열에 추가하여 나중에 검사해 보십시오.

6. 선택한 분석 유형에 적합한 작업을 호출하여 처리 결과를 가져오고 표시합니다.
   - GetDocumentTextDetection텍스트 감지 작업용
   - GetDocumentAnalysis텍스트 분석 작업용
   - GetExpenseAnalysis비용 분석 작업에 사용됩니다.
7. Amazon SNS 주제 및 Amazon SQS 대기열을 삭제합니다.

비동기식 작업 수행

이 절차의 예제 코드는 Java, Python 및AWS CLI. 시작하기 전에 적절한 설치AWSSDK. 자세한 정보는 2단계: 설정AWS CLI과AWSSDK을 참조하십시오.

여러 페이지로 된 문서에서 텍스트를 감지하거나 분석하려면

1. Amazon Textract에 대한 사용자 액세스를 구성하고 Amazon SNS에 대한 Amazon Textract 액세스를 구성합니다. 자세한 정보는 비동기 작업을 위한 Amazon Textract 구성을 참조하십시오. 이 절차를 완료하려면 PDF 형식의 여러 페이지로 된 문서가 필요합니다. 예제 코드에서 Amazon SNS 주제 및 Amazon SQS 대기열을 생성하고 구성하기 때문에 3-6단계를 건너뜁니다. 만약에 컴플tCLI 예제에서는 SQS 대기열을 설정할 필요가 없습니다.
2. Amazon S3 버킷에 PDF 또는 TIFF 형식의 다중 페이지 문서 파일을 업로드합니다. (JPEG, PNG, TIFF 또는 PDF 형식의 단일 페이지 문서도 처리할 수 있습니다.)

   지침은 단원을 참조하십시오.Amazon S3 객체 업로드의Amazon Simple Storage Service.
3. 다음을 사용합니다.AWS SDK for Java, SDK for Python (Boto3) 또는AWS CLI여러 페이지로 된 문서에서 텍스트를 감지하거나 텍스트를 분석하는 코드입니다. 에서`main`함수:
   - 값 바꾸기`roleArn`에서 저장한 IAM 역할의 ARN을 사용합니다.Amazon SNS 주제에 대한 Amazon Textract 액세스 권한 부여.
   - 다음 값 바꾸기`bucket`과`document`2단계에서 지정한 버킷 및 문서 파일 이름을 사용합니다.
   - 의 값을 바꿉니다.`type`의 입력 매개 변수`ProcessDocument`를 원하는 처리 유형을 사용할 수 있는 함수를 사용합니다. 사용`ProcessType.DETECTION`텍스트를 감지합니다. 사용`ProcessType.ANALYSIS`텍스트를 분석합니다.
   - 파이썬 예제의 경우 다음 값을 바꿉니다.`region_name`클라이언트가 운영하고 있는 리전을 사용합니다.

   이AWS CLI예를 들어 다음을 수행합니다.

- 전화할 때StartDocumentTextDetection의 값을 바꿉니다.bucket-name를 S3 버킷의 이름으로 바꿉니다.file-name2단계에서 지정한 파일 이름을 사용합니다. 대체하여 버킷의 리전 지정region-name해당 지역의 이름을 사용합니다. CLI 예제에서는 SQS를 사용하지 않습니다.
- 전화할 때GetDocumentTextDetection바꾸다job-id-number와 함께job-id에서 반환한StartDocumentTextDetection. 대체하여 버킷의 리전 지정region-name해당 지역의 이름을 사용합니다.

Java

```
package com.amazonaws.samples;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.amazonaws.auth.policy.Condition;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.QueueAttributeName;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.DocumentLocation;
import com.amazonaws.services.textract.model.DocumentMetadata;
import com.amazonaws.services.textract.model.GetDocumentAnalysisRequest;
import com.amazonaws.services.textract.model.GetDocumentAnalysisResult;
import com.amazonaws.services.textract.model.GetDocumentTextDetectionRequest;
```

```
import com.amazonaws.services.textract.model.GetDocumentTextDetectionResult;
import com.amazonaws.services.textract.model.NotificationChannel;
import com.amazonaws.services.textract.model.Relationship;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.StartDocumentAnalysisRequest;
import com.amazonaws.services.textract.model.StartDocumentAnalysisResult;
import com.amazonaws.services.textract.model.StartDocumentTextDetectionRequest;
import com.amazonaws.services.textract.model.StartDocumentTextDetectionResult;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;;
public class DocumentProcessor {

    private static String sqsQueueName=null;
    private static String snsTopicName=null;
    private static String snsTopicArn = null;
    private static String roleArn= null;
    private static String sqsQueueUrl = null;
    private static String sqsQueueArn = null;
    private static String startJobId = null;
    private static String bucket = null;
    private static String document = null;
    private static AmazonSQS sqs=null;
    private static AmazonSNS sns=null;
    private static AmazonTextract textract = null;

    public enum ProcessType {
        DETECTION,ANALYSIS
    }

    public static void main(String[] args) throws Exception {

        String document = "document";
        String bucket = "bucket";
        String roleArn="role";

        sns = AmazonSNSClientBuilder.defaultClient();
        sqs= AmazonSQSClientBuilder.defaultClient();
        textract=AmazonTextractClientBuilder.defaultClient();

        CreateTopicandQueue();
        ProcessDocument(bucket,document,roleArn,ProcessType.DETECTION);
        DeleteTopicandQueue();
        System.out.println("Done!");
```

```
   }
   // Creates an SNS topic and SQS queue. The queue is subscribed to the
topic.
   static void CreateTopicandQueue()
   {
       //create a new SNS topic
       snsTopicName="AmazonTextractTopic" +
Long.toString(System.currentTimeMillis());
       CreateTopicRequest createTopicRequest = new
CreateTopicRequest(snsTopicName);
       CreateTopicResult createTopicResult =
sns.createTopic(createTopicRequest);
       snsTopicArn=createTopicResult.getTopicArn();

       //Create a new SQS Queue
       sqsQueueName="AmazonTextractQueue" +
Long.toString(System.currentTimeMillis());
       final CreateQueueRequest createQueueRequest = new
CreateQueueRequest(sqsQueueName);
       sqsQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();
       sqsQueueArn = sqs.getQueueAttributes(sqsQueueUrl,
Arrays.asList("QueueArn")).getAttributes().get("QueueArn");

       //Subscribe SQS queue to SNS topic
       String sqsSubscriptionArn = sns.subscribe(snsTopicArn, "sqs",
sqsQueueArn).getSubscriptionArn();

       // Authorize queue
         Policy policy = new Policy().withStatements(
                 new Statement(Effect.Allow)
                 .withPrincipals(Principal.AllUsers)
                 .withActions(SQSActions.SendMessage)
                 .withResources(new Resource(sqsQueueArn))
                 .withConditions(new
Condition().withType("ArnEquals").withConditionKey("aws:SourceArn").withValues(snsTopic
                 );


         Map queueAttributes = new HashMap();
         queueAttributes.put(QueueAttributeName.Policy.toString(),
policy.toJson());
         sqs.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueUrl,
queueAttributes));
```

```
         System.out.println("Topic arn: " + snsTopicArn);
         System.out.println("Queue arn: " + sqsQueueArn);
         System.out.println("Queue url: " + sqsQueueUrl);
         System.out.println("Queue sub arn: " + sqsSubscriptionArn );
     }
   static void DeleteTopicandQueue()
   {
       if (sqs !=null) {
           sqs.deleteQueue(sqsQueueUrl);
           System.out.println("SQS queue deleted");
       }

       if (sns!=null) {
           sns.deleteTopic(snsTopicArn);
           System.out.println("SNS topic deleted");
       }
   }

   //Starts the processing of the input document.
   static void ProcessDocument(String inBucket, String inDocument, String
inRoleArn, ProcessType type) throws Exception
   {
       bucket=inBucket;
       document=inDocument;
       roleArn=inRoleArn;

       switch(type)
       {
           case DETECTION:
               StartDocumentTextDetection(bucket, document);
               System.out.println("Processing type: Detection");
               break;
           case ANALYSIS:
               StartDocumentAnalysis(bucket,document);
               System.out.println("Processing type: Analysis");
               break;
           default:
               System.out.println("Invalid processing type. Choose Detection or
Analysis");
               throw new Exception("Invalid processing type");

       }
```

```
        System.out.println("Waiting for job: " + startJobId);
        //Poll queue for messages
        List<Message> messages=null;
        int dotLine=0;
        boolean jobFound=false;

        //loop until the job status is published. Ignore other messages in
queue.
        do{
            messages = sqs.receiveMessage(sqsQueueUrl).getMessages();
            if (dotLine++<40){
                System.out.print(".");
            }else{
                System.out.println();
                dotLine=0;
            }

            if (!messages.isEmpty()) {
                //Loop through messages received.
                for (Message message: messages) {
                    String notification = message.getBody();

                    // Get status and job id from notification.
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found was " + operationJobId);
                    // Found job. Get the results and display.
                    if(operationJobId.asText().equals(startJobId)){
                        jobFound=true;
                        System.out.println("Job id: " + operationJobId );
                        System.out.println("Status : " +
operationStatus.toString());
                        if (operationStatus.asText().equals("SUCCEEDED")){
                            switch(type)
                            {
                                case DETECTION:
                                    GetDocumentTextDetectionResults();
```

```
                                    break;
                                case ANALYSIS:
                                    GetDocumentAnalysisResults();
                                    break;
                                default:
                                    System.out.println("Invalid processing type.
Choose Detection or Analysis");
                                    throw new Exception("Invalid processing
type");

                            }
                        }
                        else{
                            System.out.println("Document analysis failed");
                        }


sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }

                    else{
                        System.out.println("Job received was not job " +
startJobId);
                        //Delete unknown message. Consider moving message to
dead letter queue

sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }
                }
            }
            else {
                Thread.sleep(5000);
            }
        } while (!jobFound);

        System.out.println("Finished processing document");
    }

    private static void StartDocumentTextDetection(String bucket, String
document) throws Exception{

        //Create notification channel
        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
```

```
                .withRoleArn(roleArn);

        StartDocumentTextDetectionRequest req = new
StartDocumentTextDetectionRequest()
                .withDocumentLocation(new DocumentLocation()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
                        .withName(document)))
                .withJobTag("DetectingText")
                .withNotificationChannel(channel);

        StartDocumentTextDetectionResult startDocumentTextDetectionResult =
textract.startDocumentTextDetection(req);
        startJobId=startDocumentTextDetectionResult.getJobId();
    }

 //Gets the results of processing started by StartDocumentTextDetection
    private static void GetDocumentTextDetectionResults() throws Exception{
        int maxResults=1000;
        String paginationToken=null;
        GetDocumentTextDetectionResult response=null;
        Boolean finished=false;

        while (finished==false)
        {
            GetDocumentTextDetectionRequest documentTextDetectionRequest= new
GetDocumentTextDetectionRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);
            response =
textract.getDocumentTextDetection(documentTextDetectionRequest);
            DocumentMetadata documentMetaData=response.getDocumentMetadata();

            System.out.println("Pages: " +
documentMetaData.getPages().toString());

            //Show blocks information
            List<Block> blocks= response.getBlocks();
            for (Block block : blocks) {
                DisplayBlockInfo(block);
            }
            paginationToken=response.getNextToken();
            if (paginationToken==null)
```

```
                finished=true;

        }

    }

    private static void StartDocumentAnalysis(String bucket, String document)
throws Exception{
        //Create notification channel
        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
                .withRoleArn(roleArn);

        StartDocumentAnalysisRequest req = new StartDocumentAnalysisRequest()
                .withFeatureTypes("TABLES","FORMS")
                .withDocumentLocation(new DocumentLocation()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
                        .withName(document)))
                .withJobTag("AnalyzingText")
                .withNotificationChannel(channel);

        StartDocumentAnalysisResult startDocumentAnalysisResult =
textract.startDocumentAnalysis(req);
        startJobId=startDocumentAnalysisResult.getJobId();
    }
    //Gets the results of processing started by StartDocumentAnalysis
    private static void GetDocumentAnalysisResults() throws Exception{

        int maxResults=1000;
        String paginationToken=null;
        GetDocumentAnalysisResult response=null;
        Boolean finished=false;

        //loops until pagination token is null
        while (finished==false)
        {
            GetDocumentAnalysisRequest documentAnalysisRequest= new
GetDocumentAnalysisRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);

            response = textract.getDocumentAnalysis(documentAnalysisRequest);
```

```
            DocumentMetadata documentMetaData=response.getDocumentMetadata();

            System.out.println("Pages: " +
documentMetaData.getPages().toString());

            //Show blocks, confidence and detection times
            List<Block> blocks= response.getBlocks();

            for (Block block : blocks) {
                DisplayBlockInfo(block);
            }
            paginationToken=response.getNextToken();
            if (paginationToken==null)
                finished=true;
        }

    }
    //Displays Block information for text detection and text analysis
    private static void DisplayBlockInfo(Block block) {
        System.out.println("Block Id : " + block.getId());
        if (block.getText()!=null)
            System.out.println("\tDetected text: " + block.getText());
        System.out.println("\tType: " + block.getBlockType());

        if (block.getBlockType().equals("PAGE") !=true) {
            System.out.println("\tConfidence: " +
block.getConfidence().toString());
        }
        if(block.getBlockType().equals("CELL"))
        {
            System.out.println("\tCell information:");
            System.out.println("\t\tColumn: " + block.getColumnIndex());
            System.out.println("\t\tRow: " + block.getRowIndex());
            System.out.println("\t\tColumn span: " + block.getColumnSpan());
            System.out.println("\t\tRow span: " + block.getRowSpan());

        }

        System.out.println("\tRelationships");
        List<Relationship> relationships=block.getRelationships();
        if(relationships!=null) {
            for (Relationship relationship : relationships) {
                System.out.println("\t\tType: " + relationship.getType());
```

```
                System.out.println("\t\tIDs: " +
 relationship.getIds().toString());
            }
        } else {
            System.out.println("\t\tNo related Blocks");
        }

        System.out.println("\tGeometry");
        System.out.println("\t\tBounding Box: " +
 block.getGeometry().getBoundingBox().toString());
        System.out.println("\t\tPolygon: " +
 block.getGeometry().getPolygon().toString());

        List<String> entityTypes = block.getEntityTypes();

        System.out.println("\tEntity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("\t\tEntity Type: " + entityType);
            }
        } else {
            System.out.println("\t\tNo entity type");
        }

        if(block.getBlockType().equals("SELECTION_ELEMENT")) {
            System.out.print("    Selection element detected: ");
            if (block.getSelectionStatus().equals("SELECTED")){
                System.out.println("Selected");
            }else {
                System.out.println(" Not selected");
            }
        }
        if(block.getPage()!=null)
            System.out.println("\tPage: " + block.getPage());
        System.out.println();
    }
}
```

AWS CLI

이AWS CLI명령은 지정된 문서에서 텍스트의 비동기식 감지를 시작합니다. a를 반환합니다.`job-id`탐지 결과를 검색하는 데 사용할 수 있습니다.

```
aws textract start-document-text-detection --document-location
"{\"S3Object\":{\"Bucket\":\"bucket-name\",\"Name\":\"file-name\"}}" --
region region-name
```

이AWS CLI명령은 와 함께 제공된 경우 Amazon Textract 비동기 작업에 대한 결과를 반환합니다.job-id.

```
aws textract get-document-text-detection --region region-name --job-id job-id-
number
```

Windows 장치에서 CLI에 액세스하는 경우 작은 따옴표 대신 큰따옴표를 사용하고 백슬래시(예:\) 로 내부 큰따옴표를 이스케이프하여 발생할 수 있는 파서 오류를 해결합니다. 예시는 아래를 참조하십시오.

```
aws textract start-document-text-detection --document-location "{\"S3Object\":
{\"Bucket\":\"bucket\",\"Name\":\"document\"}}" --region region-name
```

Python

```
import boto3
import json
import sys
import time


class ProcessType:
    DETECTION = 1
    ANALYSIS = 2


class DocumentProcessor:
    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
```

```
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
        self.document = document
        self.region_name = region

        self.textract = boto3.client('textract', region_name=self.region_name)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')

    def ProcessDocument(self, type):
        jobFound = False

        self.processType = type
        validType = False

        # Determine which type of processing to perform
        if self.processType == ProcessType.DETECTION:
            response = self.textract.start_document_text_detection(
                DocumentLocation={'S3Object': {'Bucket': self.bucket, 'Name':
self.document}},
                NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})
            print('Processing type: Detection')
            validType = True

        if self.processType == ProcessType.ANALYSIS:
            response = self.textract.start_document_analysis(
                DocumentLocation={'S3Object': {'Bucket': self.bucket, 'Name':
self.document}},
                FeatureTypes=["TABLES", "FORMS"],
                NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})
            print('Processing type: Analysis')
            validType = True

        if validType == False:
            print("Invalid processing type. Choose Detection or Analysis.")
            return

        print('Start Job Id: ' + response['JobId'])
        dotLine = 0
```

```
        while jobFound == False:
            sqsResponse = self.sqs.receive_message(QueueUrl=self.sqsQueueUrl,
MessageAttributeNames=['ALL'],
                                                   MaxNumberOfMessages=10)

            if sqsResponse:

                if 'Messages' not in sqsResponse:
                    if dotLine < 40:
                        print('.', end='')
                        dotLine = dotLine + 1
                    else:
                        print()
                        dotLine = 0
                    sys.stdout.flush()
                    time.sleep(5)
                    continue

                for message in sqsResponse['Messages']:
                    notification = json.loads(message['Body'])
                    textMessage = json.loads(notification['Message'])
                    print(textMessage['JobId'])
                    print(textMessage['Status'])
                    if str(textMessage['JobId']) == response['JobId']:
                        print('Matching Job Found:' + textMessage['JobId'])
                        jobFound = True
                        self.GetResults(textMessage['JobId'])
                        self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])
                    else:
                        print("Job didn't match:" +
                              str(textMessage['JobId']) + ' : ' +
str(response['JobId']))
                    # Delete the unknown message. Consider sending to dead
letter queue
                    self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])

        print('Done!')

    def CreateTopicandQueue(self):
```

```
        millis = str(int(round(time.time() * 1000)))

        # Create SNS topic
        snsTopicName = "AmazonTextractTopic" + millis

        topicResponse = self.sns.create_topic(Name=snsTopicName)
        self.snsTopicArn = topicResponse['TopicArn']

        # create SQS queue
        sqsQueueName = "AmazonTextractQueue" + millis
        self.sqs.create_queue(QueueName=sqsQueueName)
        self.sqsQueueUrl = self.sqs.get_queue_url(QueueName=sqsQueueName)
['QueueUrl']

        attribs = self.sqs.get_queue_attributes(QueueUrl=self.sqsQueueUrl,
                                                AttributeNames=['QueueArn'])
['Attributes']

        sqsQueueArn = attribs['QueueArn']

        # Subscribe SQS queue to SNS topic
        self.sns.subscribe(
            TopicArn=self.snsTopicArn,
            Protocol='sqs',
            Endpoint=sqsQueueArn)

        # Authorize SNS to write SQS queue
        policy = """{{
  "Version":"2012-10-17",
  "Statement":[
    {{
      "Sid":"MyPolicy",
      "Effect":"Allow",
      "Principal" : {{"AWS" : "*"}},
      "Action":"SQS:SendMessage",
      "Resource": "{}",
      "Condition":{{
        "ArnEquals":{{
          "aws:SourceArn": "{}"
        }}
      }}
    }}
  ]
}}""".format(sqsQueueArn, self.snsTopicArn)
```

```
        response = self.sqs.set_queue_attributes(
            QueueUrl=self.sqsQueueUrl,
            Attributes={
                'Policy': policy
            })

    def DeleteTopicandQueue(self):
        self.sqs.delete_queue(QueueUrl=self.sqsQueueUrl)
        self.sns.delete_topic(TopicArn=self.snsTopicArn)

    # Display information about a block
    def DisplayBlockInfo(self, block):

        print("Block Id: " + block['Id'])
        print("Type: " + block['BlockType'])
        if 'EntityTypes' in block:
            print('EntityTypes: {}'.format(block['EntityTypes']))

        if 'Text' in block:
            print("Text: " + block['Text'])

        if block['BlockType'] != 'PAGE':
            print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

        print('Page: {}'.format(block['Page']))

        if block['BlockType'] == 'CELL':
            print('Cell Information')
            print('\tColumn: {} '.format(block['ColumnIndex']))
            print('\tRow: {}'.format(block['RowIndex']))
            print('\tColumn span: {} '.format(block['ColumnSpan']))
            print('\tRow span: {}'.format(block['RowSpan']))

            if 'Relationships' in block:
                print('\tRelationships: {}'.format(block['Relationships']))

        print('Geometry')
        print('\tBounding Box: {}'.format(block['Geometry']['BoundingBox']))
        print('\tPolygon: {}'.format(block['Geometry']['Polygon']))

        if block['BlockType'] == 'SELECTION_ELEMENT':
            print('    Selection element detected: ', end='')
            if block['SelectionStatus'] == 'SELECTED':
```

```
                print('Selected')
            else:
                print('Not selected')

    def GetResults(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:

            response = None

            if self.processType == ProcessType.ANALYSIS:
                if paginationToken == None:
                    response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults)
                else:
                    response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

            if self.processType == ProcessType.DETECTION:
                if paginationToken == None:
                    response =
self.textract.get_document_text_detection(JobId=jobId,

MaxResults=maxResults)
                else:
                    response =
self.textract.get_document_text_detection(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

            blocks = response['Blocks']
            print('Detected Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))

            # Display block information
```

```
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True

    def GetResultsDocumentAnalysis(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:

            response = None
            if paginationToken == None:
                response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults)
            else:
                response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

                # Get the text blocks
            blocks = response['Blocks']
            print('Analyzed Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))
            # Display block information
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()

                if 'NextToken' in response:
                    paginationToken = response['NextToken']
                else:
                    finished = True
```

```
def main():
    roleArn = ''
    bucket = ''
    document = ''
    region_name = ''

    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.CreateTopicandQueue()
    analyzer.ProcessDocument(ProcessType.DETECTION)
    analyzer.DeleteTopicandQueue()


if __name__ == "__main__":
    main()
```

Node.JS

이 예제에서는 의 값을 바꿉니다.`roleArn`에서 저장한 IAM 역할의 ARN을 사용합니다.Amazon SNS 주제에 대한 Amazon Textract 액세스 권한 부여. 다음 값 바꾸기`bucket`과`document`위의 2단계에서 지정한 버킷 및 문서 파일 이름을 사용합니다. 값 바꾸기`processType`입력 문서에 사용하려는 처리 유형을 사용합니다. 마지막으로, 의 값을 바꿉니다.`REGION`클라이언트가 운영하고 있는 리전을 사용합니다.

```
 // snippet-start:[sqs.JavaScript.queues.createQueueV3]
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand, GetQueueAttributesCommand, GetQueueUrlCommand,
    SetQueueAttributesCommand, DeleteQueueCommand, ReceiveMessageCommand,
 DeleteMessageCommand } from  "@aws-sdk/client-sqs";
  import {CreateTopicCommand, SubscribeCommand, DeleteTopicCommand } from "@aws-
sdk/client-sns";
  import  { SQSClient } from "@aws-sdk/client-sqs";
  import  { SNSClient } from "@aws-sdk/client-sns";
  import  { TextractClient, StartDocumentTextDetectionCommand,
 StartDocumentAnalysisCommand, GetDocumentAnalysisCommand,
 GetDocumentTextDetectionCommand, DocumentMetadata } from "@aws-sdk/client-
textract";
  import { stdout } from "process";

  // Set the AWS Region.
  const REGION = "us-east-1"; //e.g. "us-east-1"
  // Create SNS service object.
```

```
 const sqsClient = new SQSClient({ region: REGION });
 const snsClient = new SNSClient({ region: REGION });
 const textractClient = new TextractClient({ region: REGION });

 // Set bucket and video variables
 const bucket = "bucket-name";

 const documentName = "document-name";
 const roleArn = "role-arn"
 const processType = "DETECTION"
 var startJobId = ""

 var ts = Date.now();
 const snsTopicName = "AmazonTextractExample" + ts;
 const snsTopicParams = {Name: snsTopicName}
 const sqsQueueName = "AmazonTextractQueue-" + ts;

 // Set the parameters
 const sqsParams = {
   QueueName: sqsQueueName, //SQS_QUEUE_URL
   Attributes: {
     DelaySeconds: "60", // Number of seconds delay.
     MessageRetentionPeriod: "86400", // Number of seconds delay.
   },
 };

 // Process a document based on operation type
 const processDocumment = async (type, bucket, videoName, roleArn, sqsQueueUrl,
snsTopicArn) =>
   {
   try
   {
       // Set job found and success status to false initially
     var jobFound = false
     var succeeded = false
     var dotLine = 0
     var processType = type
     var validType = false

     if (processType == "DETECTION"){
       var response = await textractClient.send(new
StartDocumentTextDetectionCommand({DocumentLocation:{S3Object:{Bucket:bucket,
Name:videoName}},
         NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}))
```

```
        console.log("Processing type: Detection")
        validType = true
      }

      if (processType == "ANALYSIS"){
        var response = await textractClient.send(new
StartDocumentAnalysisCommand({DocumentLocation:{S3Object:{Bucket:bucket,
Name:videoName}},
          NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}))
        console.log("Processing type: Analysis")
        validType = true
      }

      if (validType == false){
          console.log("Invalid processing type. Choose Detection or Analysis.")
          return
      }
    // while not found, continue to poll for response
    console.log(`Start Job ID: ${response.JobId}`)
    while (jobFound == false){
      var sqsReceivedResponse = await sqsClient.send(new
ReceiveMessageCommand({QueueUrl:sqsQueueUrl,
        MaxNumberOfMessages:'ALL', MaxNumberOfMessages:10}));
      if (sqsReceivedResponse){
        var responseString = JSON.stringify(sqsReceivedResponse)
        if (!responseString.includes('Body')){
          if (dotLine < 40) {
            console.log('.')
            dotLine = dotLine + 1
          }else {
            console.log('')
            dotLine = 0
          };
          stdout.write('', () => {
            console.log('');
          });
          await new Promise(resolve => setTimeout(resolve, 5000));
          continue
        }
      }

        // Once job found, log Job ID and return true if status is succeeded
        for (var message of sqsReceivedResponse.Messages){
            console.log("Retrieved messages:")
```

```
            var notification = JSON.parse(message.Body)
            var rekMessage = JSON.parse(notification.Message)
            var messageJobId = rekMessage.JobId
            if (String(rekMessage.JobId).includes(String(startJobId))){
                console.log('Matching job found:')
                console.log(rekMessage.JobId)
                jobFound = true
                // GET RESUlTS FUNCTION HERE
                var operationResults = await GetResults(processType,
rekMessage.JobId)
                //GET RESULTS FUMCTION HERE
                console.log(rekMessage.Status)
            if (String(rekMessage.Status).includes(String("SUCCEEDED"))){
                succeeded = true
                console.log("Job processing succeeded.")
                var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
            }
            }else{
            console.log("Provided Job ID did not match returned ID.")
            var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
            }
       }

   console.log("Done!")
   }
   }catch (err) {
       console.log("Error", err);
     }
 }

 // Create the SNS topic and SQS Queue
 const createTopicandQueue = async () => {
   try {
     // Create SNS topic
     const topicResponse = await snsClient.send(new
CreateTopicCommand(snsTopicParams));
     const topicArn = topicResponse.TopicArn
     console.log("Success", topicResponse);
     // Create SQS Queue
```

```
     const sqsResponse = await sqsClient.send(new
CreateQueueCommand(sqsParams));
     console.log("Success", sqsResponse);
     const sqsQueueCommand = await sqsClient.send(new
GetQueueUrlCommand({QueueName: sqsQueueName}))
     const sqsQueueUrl = sqsQueueCommand.QueueUrl
     const attribsResponse = await sqsClient.send(new
GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
['QueueArn']}))
     const attribs = attribsResponse.Attributes
     console.log(attribs)
     const queueArn = attribs.QueueArn
     // subscribe SQS queue to SNS topic
     const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
topicArn, Protocol:'sqs', Endpoint: queueArn}))
     const policy = {
       Version: "2012-10-17",
       Statement: [
         {
           Sid: "MyPolicy",
           Effect: "Allow",
           Principal: {AWS: "*"},
           Action: "SQS:SendMessage",
           Resource: queueArn,
           Condition: {
             ArnEquals: {
               'aws:SourceArn': topicArn
             }
           }
         }
       ]
     };

     const response = sqsClient.send(new SetQueueAttributesCommand({QueueUrl:
sqsQueueUrl, Attributes: {Policy: JSON.stringify(policy)}}))
     console.log(response)
     console.log(sqsQueueUrl, topicArn)
     return [sqsQueueUrl, topicArn]

   } catch (err) {
     console.log("Error", err);

   }
 }
```

```
 const deleteTopicAndQueue = async (sqsQueueUrlArg, snsTopicArnArg) => {
   const deleteQueue = await sqsClient.send(new DeleteQueueCommand({QueueUrl:
sqsQueueUrlArg}));
   const deleteTopic = await snsClient.send(new DeleteTopicCommand({TopicArn:
snsTopicArnArg}));
   console.log("Successfully deleted.")
 }

 const displayBlockInfo = async (block) => {
   console.log(`Block ID: ${block.Id}`)
   console.log(`Block Type: ${block.BlockType}`)
   if (String(block).includes(String("EntityTypes"))){
       console.log(`EntityTypes: ${block.EntityTypes}`)
   }
   if (String(block).includes(String("Text"))){
       console.log(`EntityTypes: ${block.Text}`)
   }
   if (!String(block.BlockType).includes('PAGE')){
       console.log(`Confidence: ${block.Confidence}`)
   }
   console.log(`Page: ${block.Page}`)
   if (String(block.BlockType).includes("CELL")){
       console.log("Cell Information")
       console.log(`Column: ${block.ColumnIndex}`)
       console.log(`Row: ${block.RowIndex}`)
       console.log(`Column Span: ${block.ColumnSpan}`)
       console.log(`Row Span: ${block.RowSpan}`)
       if (String(block).includes("Relationships")){
           console.log(`Relationships: ${block.Relationships}`)
       }
   }

   console.log("Geometry")
   console.log(`Bounding Box: ${JSON.stringify(block.Geometry.BoundingBox)}`)
   console.log(`Polygon: ${JSON.stringify(block.Geometry.Polygon)}`)

   if (String(block.BlockType).includes('SELECTION_ELEMENT')){
     console.log('Selection Element detected:')
     if (String(block.SelectionStatus).includes('SELECTED')){
       console.log('Selected')
     } else {
       console.log('Not Selected')
     }
```

```
    }
  }

  const GetResults = async (processType, JobID) => {

    var maxResults = 1000
    var paginationToken = null
    var finished = false

    while (finished == false){
      var response = null
      if (processType == 'ANALYSIS'){
        if (paginationToken == null){
          response = textractClient.send(new
GetDocumentAnalysisCommand({JobId:JobID, MaxResults:maxResults}))

        }else{
          response = textractClient.send(new
GetDocumentAnalysisCommand({JobId:JobID, MaxResults:maxResults,
NextToken:paginationToken}))
        }
      }

      if(processType == 'DETECTION'){
        if (paginationToken == null){
          response = textractClient.send(new
GetDocumentTextDetectionCommand({JobId:JobID, MaxResults:maxResults}))

        }else{
          response = textractClient.send(new
GetDocumentTextDetectionCommand({JobId:JobID, MaxResults:maxResults,
NextToken:paginationToken}))
        }
      }

      await new Promise(resolve => setTimeout(resolve, 5000));
      console.log("Detected Documented Text")
      console.log(response)
      //console.log(Object.keys(response))
      console.log(typeof(response))
      var blocks = (await response).Blocks
      console.log(blocks)
      console.log(typeof(blocks))
```

```
      var docMetadata = (await response).DocumentMetadata
      var blockString = JSON.stringify(blocks)
      var parsed = JSON.parse(JSON.stringify(blocks))
      console.log(Object.keys(blocks))
      console.log(`Pages: ${docMetadata.Pages}`)
      blocks.forEach((block)=> {
        displayBlockInfo(block)
        console.log()
        console.log()
      })

      //console.log(blocks[0].BlockType)
      //console.log(blocks[1].BlockType)


      if(String(response).includes("NextToken")){
        paginationToken = response.NextToken
      }else{
        finished = true
      }
    }

  }


  // DELETE TOPIC AND QUEUE
  const main = async () => {
    var sqsAndTopic = await createTopicandQueue();
    var process = await processDocumment(processType, bucket, documentName,
 roleArn, sqsAndTopic[0], sqsAndTopic[1])
    var deleteResults = await deleteTopicAndQueue(sqsAndTopic[0],
 sqsAndTopic[1])
  }

main()
```

4. 코드를 실행합니다. 이 작업은 마치는 데 시간이 걸릴 수 있습니다. 작업이 끝나면 탐지되거나 분석된 텍스트의 블록 목록이 표시됩니다.

Amazon Textract TextractResults 알림

Amazon Textract는 완료 상태를 포함해 Amazon Textract 분석 요청의 결과를 Amazon Simple Notification Service (Amazon SNS) 주제에 게시합니다. Amazon SNS 주제에서 알림을 가져오려면 Amazon SQS 대기열 또는AWS Lambda함수. 자세한 정보는 Amazon Textract 비동기 작업 호출을 참조하십시오. 문제 해결 예는 다중 페이지 문서에서 텍스트 감지 또는 분석을(를) 참조하십시오.

결과의 JSON 형식은 다음과 같습니다.

```
{
  "JobId": "String",
  "Status": "String",
  "API": "String",
  "JobTag": "String",
  "Timestamp": Number,
  "DocumentLocation": {
    "S3ObjectName": "String",
    "S3Bucket": "String"
  }
}
```

이 표에서는 Amazon Textract 응답 내의 다양한 파라미터에 대해 설명합니다.

파라미터	설명
JobId	Amazon Textract Textract가 작업에 할당하는 고유 식별자입니다. 에서 반환되는 작업 식별자와 일치합니다.`Start`다음과 같은 작업StartDocumentTextDetection.
상태	작업의 상태입니다. 유효한 값은 성공, 실패 또는 오류입니다.
API	입력 문서를 분석할 때 사용되는 Amazon Textract 작업 (예:)StartDocumentTextDetection또는StartDocumentAnalysis.
JobTag	작업에 대한 사용자 지정 식별자입니다. 지정합니다.`JobTag`에 대한 통화에서`Start`다음과 같은 작업StartDocumentTextDetection.

파라미터	설명
타임스탬프	작업이 완료된 시점을 나타내는 Unix 타임스탬프로 밀리초 단위로 반환됩니다.
문서위치	처리된 문서에 대한 세부 정보입니다. 파일 이름과 파일이 저장된 Amazon S3 버킷이 포함되어 있습니다.

스로틀된 통화 및 끊어진 연결 처리

초당 최대 트랜잭션 수 (TPS) 를 초과하여 서비스가 애플리케이션을 스로틀링하거나 연결이 끊어지면 Amazon Textract 작업이 실패할 수 있습니다. 예를 들어, 단기간에 Amazon Textract 작업에 너무 많은 전화를 걸면 호출이 제한되고`ProvisionedThroughputExceededException`작업 응답에 오류가 있습니다. Amazon Textract TPS 할당량에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract 할당량.

작업을 자동으로 다시 시도하여 제한 및 끊어진 연결을 관리할 수 있습니다. 를 포함하여 재시도 횟수를 지정할 수 있습니다.`Config`Amazon Textract 클라이언트를 생성할 때 매개 변수입니다. 재시도 횟수 5개를 사용하는 것이 좋습니다. 이AWSSDK는 실패하고 예외를 throw하기 전에 지정된 횟수만큼 작업을 재시도합니다. 자세한 내용은 AWS의 오류 재시도 및 지수 백오프 단원을 참조하십시오.

> **Note**
>
> 자동 재시도는 동기식 및 비동기 작업 모두에서 작동합니다. 자동 재시도를 지정하기 전에 최신 버전의 AWS SDK를 사용해야 합니다. 자세한 정보는 2단계: 설정AWS CLI과AWSSDK을 참조하십시오.

다음 예제는 여러 문서를 처리할 때 Amazon Textract 작업을 자동으로 재시도하는 방법을 보여 줍니다.

사전 조건

- 아직 설정하지 않았다면 다음과 같이 하십시오.
  a. 을 사용하여 IAM 사용자를 생성하거나 업데이트합니다.`AmazonTextractFullAccess`과`AmazonS3ReadOnlyAccess`권한. 자세한 정보는 1단계: AWS 계정 설정 및 IAM 사용자 만들기을 참조하십시오.
  b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 정보는 2단계: 설정AWS CLI과AWSSDK을 참조하십시오.

자동으로 작업을 재시도하려면

1. 동기식 예제를 실행하기 위해 S3 버킷에 여러 문서 이미지를 업로드합니다. S3 버킷에 여러 페이지 문서를 업로드하고`StartDocumentTextDetection`비동기 예제를 실행하십시오.

지침은 단원을 참조하십시오.Amazon S3 객체 업로드의Amazon Simple Storage Service.

2. 다음 예제는Config매개 변수를 사용하여 작업을 자동으로 다시 시도합니다. 동기식 예제는DetectDocumentText작업, 비동기 예제는GetDocumentTextDetection작업.

   Sync Example

   다음 예제를 사용하여DetectDocumentTextAmazon S3 버킷의 문서에 대한 작업을 수행합니다. Inmain에서 값을 변경합니다.bucketS3 버킷에 업로드합니다. 의 값을 변경하려면documents2단계에서 업로드한 문서 이미지의 이름을 지정합니다.

```
import boto3
from botocore.client import Config
# Documents

def process_multiple_documents(bucket, documents):

    config = Config(retries = dict(max_attempts = 5))

    # Amazon Textract client
    textract = boto3.client('textract', config=config)

    for documentName in documents:

        print("\nProcessing:
 {}\n=========================================".format(documentName))

        # Call Amazon Textract
        response = textract.detect_document_text(
            Document={
                'S3Object': {
                    'Bucket': bucket,
                    'Name': documentName
                }
            })

        # Print detected text
        for item in response["Blocks"]:
            if item["BlockType"] == "LINE":
                print ('\033[94m' +  item["Text"] + '\033[0m')


def main():
```

```
    bucket = ""
    documents = ["document-image-1.png",
    "document-image-2.png", "document-image-3.png",
    "document-image-4.png", "document-image-5.png" ]
    process_multiple_documents(bucket, documents)



if __name__ == "__main__":
    main()
```

Async Example

다음 예제를 사용하여 GetDocumentTextDetection 작업을 호출합니다. 이미 전화를 걸었다고 가정합니다.StartDocumentTextDetection을 Amazon S3 버킷의 문서에 업로드하고JobId. Inmain에서 값을 변경합니다.bucketS3 버킷에 업로드하고roleArnTextract 역할에 할당된 Arn에. 의 값도 변경해야 합니다.document을 Amazon S3 버킷의 여러 페이지 문서 이름으로 바꿉니다. 마지막으로, 의 값을 대체합니다.region_name을 지역 이름으로 바꿉니다.GetResults의 이름을 가진 함수jobId.

```
import boto3
from botocore.client import Config

class DocumentProcessor:
    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
        self.document = document
        self.region_name = region
        self.config = Config(retries = dict(max_attempts = 5))
```

```
        self.textract = boto3.client('textract', region_name=self.region_name,
 config=self.config)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')

# Display information about a block
    def DisplayBlockInfo(self, block):

        print("Block Id: " + block['Id'])
        print("Type: " + block['BlockType'])
        if 'EntityTypes' in block:
            print('EntityTypes: {}'.format(block['EntityTypes']))

        if 'Text' in block:
            print("Text: " + block['Text'])

        if block['BlockType'] != 'PAGE':
            print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

        print('Page: {}'.format(block['Page']))

        if block['BlockType'] == 'CELL':
            print('Cell Information')
            print('\tColumn: {} '.format(block['ColumnIndex']))
            print('\tRow: {}'.format(block['RowIndex']))
            print('\tColumn span: {} '.format(block['ColumnSpan']))
            print('\tRow span: {}'.format(block['RowSpan']))

            if 'Relationships' in block:
                print('\tRelationships: {}'.format(block['Relationships']))

        print('Geometry')
        print('\tBounding Box: {}'.format(block['Geometry']['BoundingBox']))
        print('\tPolygon: {}'.format(block['Geometry']['Polygon']))

        if block['BlockType'] == 'SELECTION_ELEMENT':
            print('    Selection element detected: ', end='')
            if block['SelectionStatus'] == 'SELECTED':
                print('Selected')
            else:
                print('Not selected')

    def GetResults(self, jobId):
        maxResults = 1000
```

```
        paginationToken = None
        finished = False

        while finished == False:

            response = None

            if paginationToken == None:
                response =
 self.textract.get_document_text_detection(JobId=jobId,

 MaxResults=maxResults)
            else:
                response =
 self.textract.get_document_text_detection(JobId=jobId,

 MaxResults=maxResults,

 NextToken=paginationToken)

            blocks = response['Blocks']
            print('Detected Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))

            # Display block information
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True

def main():
    roleArn = 'role-arn'
    bucket = 'bucket-name'
    document = 'document-name'
    region_name = 'region-name'
    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.GetResults("job-id")

if __name__ == "__main__":
```

```
    main()
```

Amazon Textract Textract에 대한 모범 사례

Amazon Textract Textract는 기계 학습을 사용하여 문서를 사람처럼 읽습니다. 문서에서 텍스트, 표 및 양식을 추출합니다. 문서에서 최적의 결과를 얻으려면 다음 모범 사례를 사용합니다.

최적의 입력 문서 제공

다음은 더 나은 결과를 위해 입력 문서를 최적화할 수 있는 몇 가지 방법의 목록입니다.

- 문서 텍스트가 Amazon Textract Textract가 지원하는 언어로 되어 있는지 확인합니다. 현재 Amazon Textract Textract는 영어, 스페인어, 독일어, 이탈리아어, 프랑스어 및 포르투갈어를 지원합니다.
- 이상적으로는 최소 150DPI의 고품질 이미지를 제공합니다.
- 문서가 이미 Amazon Textract Textract가 지원하는 파일 형식 (PDF, TIFF, JPEG 및 PNG) 중 하나인 경우 문서를 Amazon Textract Textract에 업로드하기 전에 변환하거나 다운샘플링하지 마십시오.

문서의 테이블에서 텍스트를 추출할 때 최상의 결과를 얻으려면 다음을 확인하십시오.

- 문서의 테이블은 페이지의 주변 요소와 시각적으로 구분됩니다. 예를 들어 테이블이 이미지나 복잡한 패턴에 중첩되지 않습니다.
- 표 안의 텍스트는 직립입니다. 예를 들어 텍스트는 페이지의 다른 텍스트를 기준으로 회전하지 않습니다.

테이블에서 텍스트를 추출할 때 다음과 같은 경우에 일관되지 않은 결과가 표시될 수 있습니다.

- 여러 열에 걸쳐 있는 병합된 테이블 셀입니다.
- 동일한 테이블의 다른 부분과 다른 셀, 행 또는 열이 있는 테이블입니다.

를 사용하는 것이 좋습니다.텍스트 감지해결 방법으로 수행할 수 있습니다.

신뢰도 점수 사용

Amazon Textract API 작업에서 반환한 신뢰도 점수와 사용 사례의 민감도를 고려해야 합니다. 신뢰도 점수는 주어진 예측이 올바른지에 대해 알 수 있는 확률을 나타내는 0에서 100 사이의 숫자입니다. 이를 통해 결과 사용 방법에 대해 정보에 입각한 결정을 내릴 수 있습니다.

탐지 오류 (오탐지) 에 민감한 응용 프로그램에서는 최소 신뢰 점수 임계값을 적용합니다. 응용 프로그램은 더 높은 수준의 인간 조사가 필요하므로 해당 임계값 또는 플래그 상황 이하의 결과를 버려야 합니다.

최적 임계값은 애플리케이션에 따라 다릅니다. 필기 메모를 문서화하는 것과 같은 보관 목적의 경우 50% 까지 낮을 수 있습니다. 재무 의사 결정과 관련된 비즈니스 프로세스에는 90% 이상의 임계값이 필요할 수 있습니다.

인적 검토 사용 고려

또한 사용자 검토를 워크플로우에 통합하는 것도 고려해 보십시오. 이는 재무 결정과 관련된 비즈니스 프로세스와 같은 민감한 애플리케이션에 특히 중요합니다.

자습서

the section called “Block”Amazon Textract 작업에서 반환되는 객체에는 다음과 같은 텍스트 감지 및 텍스트 분석 작업의 결과가 포함됩니다.the section called “AnalyzeDocument”. 다음 Python 자습서에서는 Block 객체를 사용할 수 있는 몇 가지 방법을 보여줍니다. 예를 들어 테이블 정보를 CSV (쉼표로 구분된 값) 파일로 내보낼 수 있습니다.

자습서에서는 모든 결과를 반환하는 동기식 Amazon Textract 작업을 사용합니다. 다음과 같은 비동기 작업을 사용하려는 경우the section called “StartDocumentAnalysis”를 사용하여 여러 개의 반환된 배치를 수용하도록 예제 코드를 변경해야 합니다.`Block`객체. 비동기 작업 예제를 사용하려면 에 제공된 지침을 따랐는지 확인하십시오.비동기 작업을 위한 Amazon Textract 구성.

Amazon Textract Textract를 사용하는 다른 방법을 보여주는 예제는 단원을 참조하십시오.추가 코드 샘플.

주제

- 사전 조건
- 양식 문서에서 키-값 쌍 추출
- 테이블을 CSV 파일로 내보내기
- 생성AWS Lambda기능
- 추가 코드 샘플

사전 조건

이 단원의 예제를 실행하기 전에 먼저 환경을 구성해야 합니다.

환경을 구성하려면

1. 를 IAM 사용자를 생성 또는 업데이트합니다.`AmazonTextractFullAccess`권한. 자세한 정보는 1단계: AWS 계정 설정 및 IAM 사용자 만들기을 참조하십시오.
2. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 정보는 2단계: 설정AWS CLI과 AWSSDK을 참조하십시오.

양식 문서에서 키-값 쌍 추출

다음 Python 예제는 양식 문서에서 키-값 쌍을 추출하는 방법을 보여줍니다.the section called "Block"지도에 저장된 객체입니다. 에 대한 호출에서 블록 객체가 반환됩니다.the section called "AnalyzeDocument". 자세한 정보는 양식 데이터 (키-값 쌍)을 참조하십시오.

다음 함수를 사용합니다.

- `get_kv_map`— 통화AnalyzeDocument키 및 VALUE BLOCK 객체를 맵에 저장합니다.
- `get_kv_relationship`과`find_value_block`— 맵에서 키-값 관계를 구성합니다.

양식 문서에서 키-값 쌍을 추출하려면

1. 환경을 구성합니다. 자세한 정보는 사전 조건을 참조하십시오.
2. 다음 예제 코드를 이라는 파일에 저장합니다.textract_python_kv_parser.py.

```
import boto3
import sys
import re
import json


def get_kv_map(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)

    # process using image bytes
    client = boto3.client('textract')
    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['FORMS'])

    # Get the text blocks
    blocks=response['Blocks']


    # get key and value maps
    key_map = {}
    value_map = {}
```

```
    block_map = {}
    for block in blocks:
        block_id = block['Id']
        block_map[block_id] = block
        if block['BlockType'] == "KEY_VALUE_SET":
            if 'KEY' in block['EntityTypes']:
                key_map[block_id] = block
            else:
                value_map[block_id] = block

    return key_map, value_map, block_map


def get_kv_relationship(key_map, value_map, block_map):
    kvs = {}
    for block_id, key_block in key_map.items():
        value_block = find_value_block(key_block, value_map)
        key = get_text(key_block, block_map)
        val = get_text(value_block, block_map)
        kvs[key] = val
    return kvs


def find_value_block(key_block, value_map):
    for relationship in key_block['Relationships']:
        if relationship['Type'] == 'VALUE':
            for value_id in relationship['Ids']:
                value_block = value_map[value_id]
    return value_block


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] == 'SELECTED':
                            text += 'X '
```

```
    return text


def print_kvs(kvs):
    for key, value in kvs.items():
        print(key, ":", value)


def search_value(kvs, search_key):
    for key, value in kvs.items():
        if re.search(search_key, key, re.IGNORECASE):
            return value

def main(file_name):

    key_map, value_map, block_map = get_kv_map(file_name)

    # Get Key Value relationship
    kvs = get_kv_relationship(key_map, value_map, block_map)
    print("\n\n== FOUND KEY : VALUE pairs ===\n")
    print_kvs(kvs)

    # Start searching a key value
    while input('\n Do you want to search a value for a key? (enter "n" for exit)
 ') != 'n':
        search_key = input('\n Enter a search key:')
        print('The value is:', search_value(kvs, search_key))

if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. 명령 프롬프트에서 다음 명령을 입력합니다. Replacefile분석할 문서 이미지 파일을 사용합니다.

```
textract_python_kv_parser.py file
```

4. 메시지가 표시되면 입력 문서에 있는 키를 입력합니다. 코드가 키를 감지하면 키 값이 표시됩니다.

테이블을 CSV 파일로 내보내기

이 Python 예제는 문서의 이미지에서 테이블을 CSV (쉼표로 구분된 값) 파일로 내보내는 방법을 보여 줍니다.

동기식 문서 분석의 예에서는 호출에서 테이블 정보를 수집합니다.the section called "AnalyzeDocument". 비동기 문서 분석의 예제는 다음을 호출합니다.the section called "StartDocumentAnalysis"그런 다음 다음 결과를 반환합니다.the section called "GetDocumentAnalysis"같이`Block`객체.

테이블 정보는 다음과 같이 반환됩니다.the section called "Block"호출에서 받는 객체the section called "AnalyzeDocument". 자세한 정보는 테이블을 참조하십시오. 이`Block`객체는 테이블 데이터를 CSV 파일로 내보내는 데 사용되는 맵 구조에 저장됩니다.

Synchronous

이 예제에서는 다음과 같은 함수를 사용합니다.

- `get_table_csv_results`— 통화AnalyzeDocument을 (를) 클릭하고 문서에서 검색된 테이블 맵을 작성합니다. 감지된 모든 테이블의 CSV 표현을 만듭니다.
- `generate_table_csv`— 개별 테이블에 대한 CSV 파일을 생성합니다.
- `get_rows_columns_map`— 맵에서 행과 열을 가져옵니다.
- `get_text`— 셀에서 텍스트를 가져옵니다.

테이블을 CSV 파일로 내보내려면

1. 환경을 구성합니다. 자세한 정보는 사전 조건을 참조하십시오.
2. 다음 예제 코드를 이라는 파일에 저장합니다.textract_python_table_parser.py.

```
import webbrowser, os
import json
import boto3
import io
from io import BytesIO
import sys
from pprint import pprint


def get_rows_columns_map(table_result, blocks_map):
```

```
    rows = {}
    for relationship in table_result['Relationships']:
        if relationship['Type'] == 'CHILD':
            for child_id in relationship['Ids']:
                cell = blocks_map[child_id]
                if cell['BlockType'] == 'CELL':
                    row_index = cell['RowIndex']
                    col_index = cell['ColumnIndex']
                    if row_index not in rows:
                        # create new row
                        rows[row_index] = {}

                    # get the text value
                    rows[row_index][col_index] = get_text(cell, blocks_map)
    return rows


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] =='SELECTED':
                            text +=  'X '
    return text


def get_table_csv_results(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)

    # process using image bytes
    # get the results
    client = boto3.client('textract')
```

```
    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['TABLES'])

    # Get the text blocks
    blocks=response['Blocks']
    pprint(blocks)

    blocks_map = {}
    table_blocks = []
    for block in blocks:
        blocks_map[block['Id']] = block
        if block['BlockType'] == "TABLE":
            table_blocks.append(block)

    if len(table_blocks) <= 0:
        return "<b> NO Table FOUND </b>"

    csv = ''
    for index, table in enumerate(table_blocks):
        csv += generate_table_csv(table, blocks_map, index +1)
        csv += '\n\n'

    return csv

def generate_table_csv(table_result, blocks_map, table_index):
    rows = get_rows_columns_map(table_result, blocks_map)

    table_id = 'Table_' + str(table_index)

    # get cells.
    csv = 'Table: {0}\n\n'.format(table_id)

    for row_index, cols in rows.items():

        for col_index, text in cols.items():
            csv += '{}'.format(text) + ","
        csv += '\n'

    csv += '\n\n\n'
    return csv

def main(file_name):
    table_csv = get_table_csv_results(file_name)
```

```
    output_file = 'output.csv'

    # replace content
    with open(output_file, "wt") as fout:
        fout.write(table_csv)

    # show the results
    print('CSV OUTPUT FILE: ', output_file)


if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. 명령 프롬프트에서 다음 명령을 입력합니다. Replace`file`분석할 문서 이미지 파일의 이름을 사용합니다.

```
python textract_python_table_parser.py file
```

예제를 실행하면 CSV 출력이 라는 파일에 저장됩니다.`output.csv`.

Asynchronous

이 예에서는 두 개의 서로 다른 스크립트를 make를 사용합니다. 첫 번째 스크립트는 다음과 같이 문서를 비동기적으로 분석하는 프로세스를 시작합니다.`StartDocumentAnalysis`를 가져옵니다.`Block`다음에 의해 반환되는 정보`GetDocumentAnalysis`. 두 번째 스크립트는 반환된 스크립트를 사용합니다.`Block`각 페이지에 대한 정보는 데이터를 테이블로 포맷하고 테이블을 CSV 파일에 저장합니다.

테이블을 CSV 파일로 내보내려면

1. 환경을 구성합니다. 자세한 정보는 사전 조건을 참조하십시오.
2. 참조 에 제공된 지침을 따랐는지 확인합니다.비동기 작업을 위한 Amazon Textract 구성. 이 페이지에 설명된 프로세스를 통해 비동기 작업의 완료 상태에 대한 메시지를 보내고 받을 수 있습니다.
3. 다음 코드 예제에서 다음 값을 바꿉니다.`roleArn`Arn이 2단계에서 생성한 역할에 할당됩니다. 값 바꾸기`bucket`문서가 포함된 S3 버킷의 이름입니다. 값 바꾸기`document`을 S3 버킷에 있는 문서의 이름으로 값 바꾸기`region_name`을 버킷 리전 이름으로 바꿉니다.

   다음 예제 코드를 이라는 파일에 저장합니다.start_doc_analysis_for_table_extraction.py..

```
import boto3
import time

class DocumentProcessor:

    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
        self.document = document
        self.region_name = region

        self.textract = boto3.client('textract', region_name=self.region_name)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')

    def ProcessDocument(self):

        jobFound = False

        response =
 self.textract.start_document_analysis(DocumentLocation={'S3Object': {'Bucket':
 self.bucket, 'Name': self.document}},
                FeatureTypes=["TABLES", "FORMS"],
 NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})
        print('Processing type: Analysis')

        print('Start Job Id: ' + response['JobId'])

        print('Done!')

    def CreateTopicandQueue(self):
```

```
        millis = str(int(round(time.time() * 1000)))

        # Create SNS topic
        snsTopicName = "AmazonTextractTopic" + millis

        topicResponse = self.sns.create_topic(Name=snsTopicName)
        self.snsTopicArn = topicResponse['TopicArn']

        # create SQS queue
        sqsQueueName = "AmazonTextractQueue" + millis
        self.sqs.create_queue(QueueName=sqsQueueName)
        self.sqsQueueUrl = self.sqs.get_queue_url(QueueName=sqsQueueName)
['QueueUrl']

        attribs = self.sqs.get_queue_attributes(QueueUrl=self.sqsQueueUrl,
                                                AttributeNames=['QueueArn'])
['Attributes']

        sqsQueueArn = attribs['QueueArn']

        # Subscribe SQS queue to SNS topic
        self.sns.subscribe(TopicArn=self.snsTopicArn, Protocol='sqs',
 Endpoint=sqsQueueArn)

        # Authorize SNS to write SQS queue
        policy = """{{
      "Version":"2012-10-17",
      "Statement":[
        {{
          "Sid":"MyPolicy",
          "Effect":"Allow",
          "Principal" : {{"AWS" : "*"}},
          "Action":"SQS:SendMessage",
          "Resource": "{}",
          "Condition":{{
            "ArnEquals":{{
              "aws:SourceArn": "{}"
            }}
          }}
        }}
      ]
    }}""".format(sqsQueueArn, self.snsTopicArn)
```

```
        response = self.sqs.set_queue_attributes(
            QueueUrl=self.sqsQueueUrl,
            Attributes={
                'Policy': policy
            })

def main():
    roleArn = 'role-arn'
    bucket = 'bucket-name'
    document = 'document-name'
    region_name = 'region-name'

    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.CreateTopicandQueue()
    analyzer.ProcessDocument()

if __name__ == "__main__":
    main()
```

4. 코드를 실행합니다. 코드는 JobId 인쇄합니다. 이 JobId 복사합니다.
5. 작업 처리가 완료될 때까지 기다렸다가 작업이 완료된 후 다음 코드를 라는 파일에 복사합니다.get_doc_analysis_for_table_extraction.py. 값 바꾸기`jobId`이전에 복사한 Job ID를 사용합니다. 값 바꾸기`region_name`Textract 역할과 연결된 리전의 이름을 사용합니다. 값 바꾸기`file_name`를 CSV로 제공하려는 이름으로

```
import boto3
from pprint import pprint

jobId = 'job-id'
region_name = 'region-name'
file_name = "output-file-name.csv"

textract = boto3.client('textract', region_name=region_name)

# Display information about a block
def DisplayBlockInfo(block):
    print("Block Id: " + block['Id'])
    print("Type: " + block['BlockType'])
    if 'EntityTypes' in block:
        print('EntityTypes: {}'.format(block['EntityTypes']))

    if 'Text' in block:
        print("Text: " + block['Text'])
```

```
    if block['BlockType'] != 'PAGE':
        print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

def GetResults(jobId, file_name):
    maxResults = 1000
    paginationToken = None
    finished = False

    while finished == False:

        response = None

        if paginationToken == None:
            response = textract.get_document_analysis(JobId=jobId,
 MaxResults=maxResults)
        else:
            response = textract.get_document_analysis(JobId=jobId,
 MaxResults=maxResults,

 NextToken=paginationToken)

        blocks = response['Blocks']
        table_csv = get_table_csv_results(blocks)
        output_file = file_name
        # replace content
        with open(output_file, "at") as fout:
            fout.write(table_csv)
        # show the results
        print('Detected Document Text')
        print('Pages: {}'.format(response['DocumentMetadata']['Pages']))
        print('OUTPUT TO CSV FILE: ', output_file)

        # Display block information
        for block in blocks:
            DisplayBlockInfo(block)
            print()
            print()

        if 'NextToken' in response:
            paginationToken = response['NextToken']
        else:
            finished = True
```

```
def get_rows_columns_map(table_result, blocks_map):
    rows = {}
    for relationship in table_result['Relationships']:
        if relationship['Type'] == 'CHILD':
            for child_id in relationship['Ids']:
                try:
                    cell = blocks_map[child_id]
                    if cell['BlockType'] == 'CELL':
                        row_index = cell['RowIndex']
                        col_index = cell['ColumnIndex']
                        if row_index not in rows:
                            # create new row
                            rows[row_index] = {}

                        # get the text value
                        rows[row_index][col_index] = get_text(cell, blocks_map)
                except KeyError:
                    print("Error extracting Table data - {}:".format(KeyError))
                    pass
    return rows


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    try:
                        word = blocks_map[child_id]
                        if word['BlockType'] == 'WORD':
                            text += word['Text'] + ' '
                        if word['BlockType'] == 'SELECTION_ELEMENT':
                            if word['SelectionStatus'] == 'SELECTED':
                                text += 'X '
                    except KeyError:
                        print("Error extracting Table data -
 {}:".format(KeyError))

    return text


def get_table_csv_results(blocks):
```

```
    pprint(blocks)

    blocks_map = {}
    table_blocks = []
    for block in blocks:
        blocks_map[block['Id']] = block
        if block['BlockType'] == "TABLE":
            table_blocks.append(block)

    if len(table_blocks) <= 0:
        return "<b> NO Table FOUND </b>"

    csv = ''
    for index, table in enumerate(table_blocks):
        csv += generate_table_csv(table, blocks_map, index + 1)
        csv += '\n\n'

    return csv


def generate_table_csv(table_result, blocks_map, table_index):
    rows = get_rows_columns_map(table_result, blocks_map)

    table_id = 'Table_' + str(table_index)

    # get cells.
    csv = 'Table: {0}\n\n'.format(table_id)

    for row_index, cols in rows.items():

        for col_index, text in cols.items():
            csv += '{}'.format(text) + ","
        csv += '\n'

    csv += '\n\n\n'
    return csv

response_blocks = GetResults(jobId, file_name)
```

6. 코드를 실행합니다.

   결과를 얻은 후에는 관련 SNS 및 SQS 리소스를 삭제해야 합니다. 그렇지 않으면 해당 리소스에 대한 요금이 발생할 수 있습니다.

생성AWS Lambda기능

에서 Amazon Textract API 작업을 호출할 수 있습니다.AWS Lambda함수. 다음 지침에서는 Python 에서 호출하는 Lambda 함수를 생성하는 방법을 보여줍니다.the section called “DetectDocumentText”. 는 목록을 반환합니다.the section called “Block”객체. 이 예제를 실행하려면 PNG 또는 JPEG 형식의 문서를 포함하는 Amazon S3 버킷이 필요합니다. 함수를 생성하려면 콘솔을 사용합니다.

Lambda 함수를 사용하여 대규모로 문서를 처리하는 예제는 단원을 참조하십시오.Amazon Textract Textract를 사용한 대규모 문서 처리.

Lambda 함수에서 DetectDocumentText 작업을 호출하려면

1단계: Lambda 배포 패키지 생성

1. 명령 창을 엽니다.
2. 다음 명령을 입력하여 최신 버전의AWSSDK.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2단계: Lambda 함수 생성

1. AWS Management Console에 로그인하고 AWS Lambdahttps://console.aws.amazon.com/lambda/에서 콘솔을 엽니다.
2. 함수 생성(Create function)을 선택합니다.
3. 다음을 지정합니다.
   - [새로 작성(Author from scratch)]을 선택합니다.
   - 함수 이름에 이름을 입력합니다.
   - 용런타임, 선택Python 3.7또는Python 3.6.
   - 용실행 역할 선택 또는 생성, 선택기본 Lambda 권한을 가진 새 역할을 생성.
4. 선택함수 생성를 Lambda 함수를 생성합니다.
5. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
6. 탐색 창에서 다음을 선택합니다.Roles.

7. 리소스 목록에서 Lambda가 생성한 IAM 역할을 선택합니다. 역할 이름은 Lambda 함수의 이름으로 시작합니다.
8. 를 선택합니다.권한탭을 선택한 다음정책 연결.
9. 아마존텍스추적전체 액세스 및 AmazonS3ReadOnlyAccess 정책을 선택합니다.
10. Select정책 연결.

자세한 내용은 단원을 참조하십시오.콘솔을 사용하여 Lambda 함수 생성

3단계: 계층 생성 및 추가

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 탐색 창에서 계층을 선택합니다.
3. 계층 생성을 선택합니다.
4. 용이름에서 이름을 입력합니다.
5. 설명에 설명을 입력합니다.
6. 용코드 입력 유형, 선택.zip 파일 업로드를 선택합니다.업로드.
7. 대화 상자에서 만든 zip 파일 (boto3-layer.zip) 을 선택합니다.1단계: Lambda 배포 패키지 생성.
8. 용호환되는 런타임에서 선택한 런타임의 버전을 선택합니다.2단계: Lambda 함수 생성.
9. 선택생성계층을 생성하려면
10. 탐색 창 메뉴 아이콘을 선택합니다.
11. 탐색 창에서 함수를 선택합니다.
12. 리소스 목록에서 에서 생성한 함수를 선택합니다.2단계: Lambda 함수 생성.
13. 선택Configuration그리고디자이너섹션, 선택계층(Lambda 함수 이름 아래).
14. 에서계층섹션, 선택계층 추가.
15. 선택런타임 호환 레이어 목록에서 선택.
16. In호환되는 계층를 선택합니다.이름과Version을 3단계에서 생성한 계층입니다.
17. 추가(Add)를 선택합니다.

4단계: 함수에 python 코드를 추가합니다.

1. In디자이너함수를 선택합니다.
2. 함수 코드 편집기에서 다음을 파일에 추가합니다.lambda_function.py. 의 값 변경`bucket`과`document`버킷과 문서로 이동합니다.

```
import json
import boto3

def lambda_handler(event, context):

    bucket="bucket"
    document="document"
    client = boto3.client('textract')



    #process using S3 object
    response = client.detect_document_text(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    #Get the text blocks
    blocks=response['Blocks']

    return {
        'statusCode': 200,
        'body': json.dumps(blocks)
    }
```

3. 선택Save를 Lambda 함수를 저장합니다.

5단계: Lambda 테스트

1. Select테스트.
2. 값을 입력합니다.이벤트 이름.
3. 생성(Create)을 선택합니다.
4. 출력, 목록the section called “Block”객체가 실행 결과 창에 나타납니다.

만약AWS Lambda함수가 시간 초과 오류를 반환합니다. Amazon Textract API 작업 호출이 원인일 수 있습니다. 시간 초과 기간 연장에 대한 자세한 내용은AWS Lambda함수, 를 참조하십시오.AWS Lambda 함수 구성.

코드에서 Lambda 함수를 호출하는 방법에 대한 자세한 내용은 단원을 참조하십시오.호출AWS Lambda함수.

추가 코드 샘플

다음 표는 Amazon Textract 코드 예제에 대한 링크를 제공합니다.

예	설명
Amazon Textract 코드 샘플	Amazon Textract Textract를 사용할 수 있는 다양한 방법을 보여줍니다.
Amazon Textract Textract를 사용한 대규모 문서 처리	대규모로 문서를 처리하는 서버리스 참조 아키텍처를 보여 줍니다.
Amazon Textract 구문 분석기	를 구문 분석하는 방법을 보여 줍니다.the section called “Block”Amazon Textract 작업에서 반환된 객체입니다.
Amazon Textract 설명서 코드 예제	이 가이드에 사용된 코드 예제입니다.
텍트랙터	Amazon Textract 출력을 여러 형식으로 변환하는 방법을 보여 줍니다.
Amazon Textract Textract를 사용하여 검색 가능한 PDF 문서 생성	JPG/PNG 형식 이미지 및 스캔한 PDF 문서와 같은 다양한 유형의 입력 문서에서 검색 가능한 PDF 문서를 만드는 방법을 보여 줍니다.

Amazon Textract의 코드 예제

다음 코드 예제에서는 Amazon Textract Textract를AWS소프트웨어 개발 키트 (SDK).

예제는 다음과 같은 범주로 나뉩니다.

작업

개별 서비스 함수를 호출하는 방법을 보여주는 코드 발췌.

교차 서비스 예제

여러 곳에서 작동하는 샘플 애플리케이션AWS서비스.

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예제

- Amazon Textract Textract에 사용되는 조치
  - Amazon Textract Textract를 사용하여 문서 분석AWSSDK
  - Amazon Textract Textract를 사용하여 문서의 텍스트 감지AWSSDK
  - 다음을 사용하여 Amazon Textract 문서 분석 작업에 대한 데이터 가져오기AWSSDK
  - Amazon Textract Textract를 사용하여 문서의 비동기 분석을 시작합니다.AWSSDK
  - Amazon Textract Textract를 사용하여 비동기 텍스트 감지 시작AWSSDK
- Amazon Textract Textract의 교차 서비스 예제
  - Amazon Textract 탐색기 애플리케이션 생성
  - 다음을 사용하여 이미지에서 추출한 텍스트의 엔티티 감지AWSSDK

Amazon Textract Textract에 사용되는 조치

다음 코드 예제에서는 를 사용하여 개별 Amazon Textract 작업을 수행하는 방법을 보여줍니다.AWSSDK. 이러한 발췌문은 Amazon Textract API를 호출하며 별도로 실행되지 않습니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

다음 예에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 단원을 참조하십시오.Amazon Textract TextractAPI 참조.

예제

- Amazon Textract Textract를 사용하여 문서 분석AWSSDK
- Amazon Textract Textract를 사용하여 문서의 텍스트 감지AWSSDK
- 다음을 사용하여 Amazon Textract 문서 분석 작업에 대한 데이터 가져오기AWSSDK
- Amazon Textract Textract를 사용하여 문서의 비동기 분석을 시작합니다.AWSSDK
- Amazon Textract Textract를 사용하여 비동기 텍스트 감지 시작AWSSDK

Amazon Textract Textract를 사용하여 문서 분석AWSSDK

다음 코드 예제에서는 Amazon Textract Textract를 사용하여 문서를 분석하는 방법을 보여줍니다.

Java

SDK for Java 2.x

```
   public static void analyzeDoc(TextractClient textractClient, String
sourceDoc) {

       try {
           InputStream sourceStream = new FileInputStream(new File(sourceDoc));
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           // Get the input Document object as bytes
           Document myDoc = Document.builder()
                   .bytes(sourceBytes)
                   .build();

           List<FeatureType> featureTypes = new ArrayList<FeatureType>();
           featureTypes.add(FeatureType.FORMS);
           featureTypes.add(FeatureType.TABLES);

           AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
                   .featureTypes(featureTypes)
                   .document(myDoc)
                   .build();
```

```
            AnalyzeDocumentResponse analyzeDocument =
 textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is "
 +block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.AnalyzeDocument에서AWS SDK for Java 2.xAPI 참조.

Python

SDK for Python(Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def analyze_file(
```

```
            self, feature_types, *, document_file_name=None,
document_bytes=None):
        """
        Detects text and additional elements, such as forms or tables, in a local
image
        file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to
detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={'Bytes': document_bytes}, FeatureTypes=feature_types)
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.AnalyzeDocument에서AWSPython용 API 참조 SDK (Boto3).

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Textract Textract를 사용하여 문서의 텍스트 감지AWSSDK

다음 코드 예제에서는 Amazon Textract Textract를 사용하여 문서에서 텍스트를 감지하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

입력 문서에서 텍스트를 감지합니다.

```
   public static void detectDocText(TextractClient textractClient,String
sourceDoc) {

       try {

           InputStream sourceStream = new FileInputStream(new File(sourceDoc));
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           // Get the input Document object as bytes
           Document myDoc = Document.builder()
                   .bytes(sourceBytes)
                   .build();

           DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                   .document(myDoc)
                   .build();

           // Invoke the Detect operation
           DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);

           List<Block> docInfo = textResponse.blocks();

           Iterator<Block> blockIterator = docInfo.iterator();

           while(blockIterator.hasNext()) {
               Block block = blockIterator.next();
               System.out.println("The block type is "
+block.blockType().toString());
           }
```

```
            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is "
+documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Amazon S3 버킷에 있는 문서에서 텍스트를 감지합니다.

```
    public static void detectDocTextS3 (TextractClient textractClient, String
bucketName, String docName) {

        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            // Create a DetectDocumentTextRequest object
            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the detectDocumentText method
            DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);

            List<Block> docInfo = textResponse.blocks();

            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
```

```
                Block block = blockIterator.next();
                System.out.println("The block type is "
 +block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is "
 +documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.DetectDocumentText에서AWS SDK for Java 2.xAPI 참조.

Python

SDK for Python(Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.
```

```
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={'Bytes': document_bytes})
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.DetectDocumentText에서AWSPython용 API 참조 SDK (Boto3).

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

다음을 사용하여 Amazon Textract 문서 분석 작업에 대한 데이터 가져오기 AWSSDK

다음 코드 예제에서는 Amazon Textract 문서 분석 작업에 대한 데이터를 가져오는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
```

```
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
            response = self.textract_client.get_document_analysis(
                JobId=job_id)
            job_status = response['JobStatus']
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
            return response
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.GetDocumentAnalysis에서AWSPython용 API 참조 SDK (Boto3).

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Textract Textract를 사용하여 문서의 비동기 분석을 시작합니다.AWSSDK

다음 코드 예제에서는 Amazon Textract Textract를 사용하여 문서의 비동기 분석을 시작하는 방법을 보여줍니다.

Java

SDK for Java 2.x

```
   public static String startDocAnalysisS3 (TextractClient textractClient,
String bucketName, String docName) {

       try {

           List<FeatureType> myList = new ArrayList<FeatureType>();
           myList.add(FeatureType.TABLES);
           myList.add(FeatureType.FORMS);

           S3Object s3Object = S3Object.builder()
                   .bucket(bucketName)
                   .name(docName)
                   .build();

           DocumentLocation location = DocumentLocation.builder()
                   .s3Object(s3Object)
                   .build();

           StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                   .documentLocation(location)
                   .featureTypes(myList)
                   .build();

           StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

           // Get the job ID
           String jobId = response.jobId();
           return jobId;

       } catch (TextractException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "" ;
    }

    private static String getJobResults(TextractClient textractClient, String
jobId) {

        boolean finished = false;
        int index = 0 ;
        String status = "" ;

       try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                    .jobId(jobId)
                    .maxResults(1000)
                    .build();

            GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++ ;
        }
        return status;

       } catch( InterruptedException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
       return "";
    }
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.StartDocumentAnalysis에서AWS SDK for Java 2.xAPI 참조.

Python

SDK for Python(Boto3)

비동기 작업을 시작하여 문서를 분석합니다.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_analysis_job(
            self, bucket_name, document_file_name, feature_types, sns_topic_arn,
            sns_role_arn):
        """
        Starts an asynchronous job to detect text and additional elements, such
 as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
 publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
 image.
        :param document_file_name: The name of the document image stored in
 Amazon S3.
        :param feature_types: The types of additional document features to
 detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
 topic
                              where job completion notification is published.
```

```
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
(IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name':
document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn},
                FeatureTypes=feature_types)
            job_id = response['JobId']
            logger.info(
                "Started text analysis job %s on %s.", job_id,
document_file_name)
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.StartDocumentAnalysis에서AWSPython용 API 참조 SDK (Boto3).

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Textract Textract를 사용하여 비동기 텍스트 감지 시작AWSSDK

다음 코드 예제에서는 Amazon Textract Textract를 사용하여 문서에서 비동기 텍스트 탐지를 시작하는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

문서의 텍스트를 감지하는 비동기 작업을 시작합니다.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_detection_job(
            self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn):
        """
        Starts an asynchronous job to detect text elements in an image stored in
 an
        Amazon S3 bucket. Textract publishes a notification to the specified
 Amazon SNS
        topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
 image.
        :param document_file_name: The name of the document image stored in
 Amazon S3.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
 topic
                              where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
 (IAM)
                             role that can be assumed by Textract and grants
 permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
```

```
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name':
document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn})
            job_id = response['JobId']
            logger.info(
                "Started text detection job %s on %s.", job_id,
document_file_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- GitHub에서 지침과 추가 코드를 확인해 보세요.
- API 자세한 내용은 단원을 참조하십시오.StartDocumentTextDetection에서AWSPython용 API 참조 SDK (Boto3).

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Textract Textract의 교차 서비스 예제

다음 샘플 애플리케이션에서AWSAmazon Textract 다른 SDK를 결합하는 SDKAWS서비스. 각 예제에는 GitHub 링크가 포함되어 있습니다. 여기서 애플리케이션을 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

예제

- Amazon Textract 탐색기 애플리케이션 생성
- 다음을 사용하여 이미지에서 추출한 텍스트의 엔티티 감지AWSSDK

Amazon Textract 탐색기 애플리케이션 생성

다음 코드 예제에서는 대화형 애플리케이션을 통해 Amazon Textract 출력을 탐색하는 방법을 보여줍니다.

JavaScript

SDK for JavaScript V3

Amazon Textract를 사용하여 문서 이미지에서 데이터를 추출하고 대화형 웹 페이지에 표시하는 React 애플리케이션을 AWS SDK for JavaScript로 구축하는 방법을 보여줍니다. 이 예제는 웹 브라우저에서 실행되며 자격 증명을 위해 인증된 Amazon Cognito 자격 증명이 필요합니다. 이 애플리케이션은 스토리지로 Amazon Simple Storage Service(Amazon S3)를 사용하고 알림을 위해 Amazon Simple Notification Service(Amazon SNS) 주제를 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 폴링합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Cognito 자격 증명
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python(Boto3)

AWS SDK for Python (Boto3)을 Amazon Textract와 함께 사용하여 문서 이미지의 텍스트, 양식 및 테이블 요소를 감지하는 방법을 보여줍니다. 입력 이미지와 Amazon Textract 출력은 탐지된 요소를 탐색할 수 있는 Tkinter 애플리케이션에 표시됩니다.

- 문서 이미지를 Amazon Textract에 제출하고 감지된 요소의 출력을 탐색합니다.
- Amazon Textract로 직접, 또는 Amazon Simple Storage Service(Amazon S3) 버킷을 통해 이미지를 제출합니다.
- 비동기식 API를 사용하여 작업이 완료되면 Amazon Simple Notification Service(Amazon SNS) 주제에 알림을 게시하는 작업을 시작합니다.
- Amazon Simple Queue Service(Amazon SQS) 대기열에서 작업 완료 메시지를 폴링하고 결과를 표시합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

다음을 사용하여 이미지에서 추출한 텍스트의 엔티티 감지AWSSDK

다음 코드 예제에서는 Amazon Comprehend를 사용하여 Amazon S3에 저장된 이미지에서 Amazon Textract에서 추출한 텍스트의 엔티티를 감지하는 방법을 보여 줍니다.

Python

SDK for Python(Boto3)

를 사용하는 방법을 보여 줍니다.AWS SDK for Python (Boto3)Jupyter 노트북에서 이미지에서 추출된 텍스트의 엔티티를 감지합니다. 이 예제에서는 Amazon Textract를 사용하여 Amazon Simple Storage Service (Amazon S3) 와 Amazon Comprehend에 저장된 이미지에서 텍스트를 추출하여 추출된 텍스트의 엔터티를 감지합니다.

이 예제는 Jupyter 노트북이며 노트북을 호스팅할 수 있는 환경에서 실행되어야 합니다. Amazon SageMaker SageMaker를 사용하여 예제를 실행하는 방법에 대한 지침은 의 지침을 참조하세요.텍스트 추적 및 이해노트북. IPynb.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Amazon S3
- Amazon Textract

의 전체 목록은 단원을 참조하십시오.AWSSDK 개발자 가이드 및 코드 예제는 다음을 참조하십시오.와 함께 Amazon Textract 추출을 사용하여AWSSDK. 이 항목에는 시작에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Augmented AI AI를 사용하여 Amazon Textract 출력에 인간 검토 추가

Amazon Augmented AI (Amazon A2I) 는 ML 분석을 검토할 수 있는 워크플로를 손쉽게 구축할 수 있는 기계 학습 (ML) 서비스입니다.

Amazon Textract Textract는 Amazon A2I와 통합됩니다. 이 도구를 사용하여 신뢰도가 낮은 문서 분석 결과를 사람 검토자에게 라우팅할 수 있습니다.

Amazon Textract 추출을 사용할 수 있습니다.`AnalyzeDocument`양식 및 Amazon A2I 콘솔에서 데이터를 추출하는 API입니다. Amazon A2I가 검토자에게 예측을 라우팅하는 조건을 지정할 수 있습니다. 중요한 양식 키의 신뢰 임계값을 기반으로 조건을 설정합니다. 예를 들어, 키가 있는 경우 사람 검토자에게 문서를 보낼 수 있습니다.이름또는 관련 값제인 Doe신뢰도가 낮아 감지되었습니다.

주제

- Amazon A2I의 핵심 개념
- Amazon A2I 사용 시작하기

Amazon A2I의 핵심 개념

다음 용어를 검토하여 Amazon A2I의 핵심 개념을 숙지하십시오.

인적 검토 활성화 조건

Amazon A2I를 사용할 수 있습니다.활성화 조건검토를 위해 문서를 사람에게 보내는 시점과 작업자가 검토하도록 요청하는 양식 내용을 지정합니다.

예를 들어 다음과 같이 중요한 키가 낮은 신뢰도로 감지되면 Amazon Textract Textract가 양식을 Amazon A2I로 라우팅하도록 활성화 조건을 설정할 수 있습니다.전화번호. 이 예에서는 인간 검토자에게전화번호Amazon Textract Textract에서 감지된 필드 및 값

다음 활성화 조건을 사용하여 검토를 위해 양식이 사람에게 전송되는 시기를 지정할 수 있습니다.

- 양식 키 신뢰도 점수를 기반으로 특정 양식 키에 대한 인적 검토를 트리거합니다. 인적 검토자에게 이러한 양식 키와 관련 값을 검토하라는 메시지가 나타납니다.
- 특정 양식 키가 누락될 때 인적 검토를 트리거합니다. 인간 검토자는 이러한 양식 키 및 관련 값을 식별하도록 요청받습니다.

- Amazon Textract Textract에서 식별된 모든 양식 키에 대해 인적 검토를 트리거합니다.
- 검토를 위해 인적 작업자에게 무작위로 양식 샘플을 보냅니다. Amazon 검토자에게 Amazon Textract Textract에서 탐지된 모든 양식 키와 값을 검토하라는 메시지가 나타납니다.

활성화 조건이 양식 키 신뢰도 점수에 따라 달라지는 경우 두 가지 유형의 예측 신뢰도 임계값을 사용하여 인적 검토를 트리거할 수 있습니다.

- 신원 자료— 양식 내에서 탐지된 키-값 페어에 대한 신뢰도 점수입니다.
- 검증 신뢰— 양식에서 키-값 페어에 포함된 텍스트의 신뢰도 점수입니다.

신뢰 임계값을 지정하는 경우 Amazon A2I는 임계값 내에 있는 예측만 사람 검토자에게 라우팅합니다. 언제든지 이러한 임계값을 조정하여 정확성과 비용 효율성 간의 적절한 균형을 달성할 수 있습니다. 이를 통해 감사를 구현하여 예측 정확도를 정기적으로 모니터링할 수 있습니다.

> **Note**
>
> Amazon A2I 사용자 지정 작업 유형을 사용하여 검토를 위해 사람에게 문서가 전송되는 조건을 추가로 사용자 지정할 수 있습니다. 이 작업 유형을 사용하면 응용 프로그램에서 직접 검토할 조건을 지정할 수 있습니다. 자세한 정보는 단원을 참조하십시오.사용자 지정 작업 유형과 함께 Amazon Augmented AI 사용Amazon SageMaker 개발자 안내문의

인간 검토 워크플로우 (흐름 정의)

다음을 사용합니다.휴먼 리뷰 워크플로우를 a라고도 합니다.흐름 정의를 사용하여 사용자 검토 워크플로우를 만드는 데 사용되는 리소스를 지정하고 활성화 조건을 지정합니다.

지정하는 리소스는 다음과 같습니다.

- Amazon A2I API 작업을 호출할 수 있는 권한이 있는 IAM 역할
- 인적 검토의 출력을 저장하려는 Amazon S3 버킷
- 인적작업 팀
- A작업자 작업 템플릿여기에는 근로자가 검토 작업을 완료하는 데 도움이 되는 지침과 예가 포함되어

또한 Humer Review 워크플로우를 사용하여 활성화 조건을 지정할 수도 있습니다. 자세한 정보는 인적 검토 활성화 조건을 참조하십시오.

단일 인적 검토 워크플로를 사용하여 여러 항목을 생성할 수 있습니다.휴먼 루프.

SageMaker 콘솔이나 SageMaker API를 사용하여 인적 검토 워크플로를 생성할 수 있습니다. 자세한 내용은 인적 검토 워크플로 생성 섹션을 참조하세요.

작업자 태스크 템플릿

다음을 사용합니다.작업자 작업 템플릿사용자 검토 작업에 사용되는 작업자 UI를 생성합니다.

작업자 UI에 문서와 작업자 지침이 표시됩니다. 또한 작업자가 작업을 완료하는 데 사용하는 도구를 제공합니다.

인적 검토 워크플로를 생성할 때 SageMaker 콘솔을 사용하여 작업자 작업 템플릿을 구성할 수 있습니다. 자세한 내용은 인적 검토 워크플로 생성 섹션을 참조하세요.

작업 팀

A작업 팀는 인간 검토 작업을 보낼 인력 그룹입니다.

인적 검토 워크플로를 생성할 때 단일 작업 팀을 지정합니다.

Amazon A2I를 사용하면 조직 내에서 검토자 풀을 사용할 수 있습니다. 또한 Amazon Mechanical Turk를 통해 이미 기계 학습 작업을 수행하고 있는 500,000개 이상의 독립 계약자로 구성된 인력에 액세스할 수 있습니다. 품질 및 보안 절차 준수를 위해 AWS에서 사전 검열을 거친 공급업체 인력도 사용하는 또 다른 방법입니다.

또한 Amazon A2I는 검토자에게 검토 작업을 완료하는 데 필요한 모든 지침과 도구로 구성된 웹 인터페이스를 제공합니다.

각 인력 유형 (프라이빗 인력, 공급업체 및 Mechanical Turk) 에 대해 여러 작업 팀을 생성할 수 있습니다. 각 작업 팀은 여러 사람의 검토 워크플로에서 사용할 수 있습니다. 인력 및 작업 팀을 생성하는 방법에 대한 자세한 내용은 단원을 참조하십시오.작업 인력 생성 및 관리Amazon SageMaker 개발자 안내 문의

> **Important**
>
> 다시 시도하려면이리현재 Amazon Augmented AI AI를 다루는 규정 준수 프로그램을 확인할 수 있습니다. Amazon Augmented AI를 다른 AWS 서비스 (예: Amazon Rekognition 및 Amazon Textract) 와 함께 사용하는 경우 Amazon Augmented AI는 해당 다른 서비스와 동일한 규정 준수 프로그램의 범위에 포함되지 않습니다. Amazon Augmented AI를 사용하는 방법

(서비스에서 고객 데이터를 처리 또는 저장하는 방법 및 데이터 환경의 규정 준수에 미치는 영향) 을 이해하는 것은 물론 Amazon Augmented AI를 사용하는 방법에 대한 책임이 있습니다. AWS 계정 팀과 워크로드 목표 및 목적을 논의해 보시기 바랍니다. AWS 계정 팀은 서비스가 귀하가 제안한 사용 사례 및 아키텍처에 적합한지 여부를 평가하는 데 도움을 드릴 수 있습니다.
현재 Amazon Augmented AI AI는 공개 및 공급업체 인력 사례를 제외하고 PCI를 준수합니다. 아마존 Augmented AI HIPAA 규정 준수에 대한 자세한 내용을 보려면이리.

휴먼 루프

인적 루프를 사용하여 인적 검토 작업을 생성합니다.

인간 검토 작업을 인간 작업자 팀의 작업자에게 할당합니다. 활성화 조건에서 지정한 입력 문서에서 Amazon Textract Textract에서 감지한 키-값 쌍을 검토하라는 메시지가 작업자에게 표시됩니다.

예를 들어, 한 그림이 사과가 들어 있다는 확신이 50% 에서 60% 사이라고 가정해 봅시다. 이는 인적 검토를 위한 신뢰 임계값 내에 속하며 작업자에게 전송됩니다. 작업자는 문서에 사과를 검사하여 사과를 포함하거나 포함하지 않는 것으로 표시합니다. 그런 다음 Amazon A2I가 문서를 워크플로로 다시 보냅니다.

Amazon Textract 호출할 때`AnalyzeDocument`인간 검토 워크플로 (흐름 정의) 및 휴먼 루프 이름을 지정하면 휴먼 검토 워크플로에 지정된 활성화 조건이 충족되면 인간 검토 작업이 만들어집니다. 이러한 작업은 사용자 검토 워크플로우에서 지정한 리소스를 사용하여 만들어집니다.

Amazon A2I 사용 시작하기

다음 단계는 Amazon A2I를 Amazon Textract 단일 페이지 문서 분석 작업에 통합하는 데 도움이 됩니다. 다음을 수행합니다.

1. Amazon A2I 콘솔 (신규 사용자에게 권장) 또는 Amazon A2I API를 사용하여 사람 검토 워크플로를 생성합니다.
2. 양식을 분석하고 필요한 경우 사람의 검토를 포함하려면`AnalyzeDocument`작업을 수행하고 인적 검토 워크플로우의 Amazon 리소스 이름 (ARN) 을 지정합니다. 응답은 인적 검토가 필요한지 여부를 알려줍니다.
3. Amazon A2I 콘솔과 API를 사용하여 인적 루프를 모니터링합니다.
4. 결과가 전송되는 Amazon S3 버킷에서 사람 검토 결과를 검토합니다.

SageMaker 노트북 인스턴스를 설정하고 예제 노트북을 사용하려면 다음을 참조하십시오.Amazon Textract 및 Augmented AI AI를 사용하는 엔드 투 엔드 데모의Amazon SageMaker 개발자 안내서

> **Note**
>
> 이 단원에서는 Amazon A2I, Amazon Textract 작업 유형에 대한 인간 검토 워크플로를 생성하는 방법에 대해 설명합니다. Amazon A2I 및 Amazon Textract 통합을 추가로 사용자 정의하려면 Amazon A2I 사용자 지정 작업 유형을 사용할 수 있습니다. 이 옵션을 사용하면 사용자 지정 작업자 작업 템플릿을 제공하고 응용 프로그램에서 직접 검토를 위해 문서를 보내는 조건을 지정할 수 있습니다. 자세한 정보는 단원을 참조하십시오.사용자 지정 작업 유형과 함께 Amazon Augmented AI 사용의Amazon SageMaker 개발자 안내서.

주제

- 인적 검토 워크플로 생성
- 문서 분석
- 모니터 인적 루프
- 출력 데이터 및 작업자 지표 보기

인적 검토 워크플로 생성

Amazon A2I 콘솔 (신규 사용자에게 권장) 또는 Amazon A2I를 사용하여 사람 검토 워크플로를 생성할 수 있습니다.`CreateFlowDefinition`작업.

주제

- 인적 검토 워크플로 생성(콘솔)
- 인적 검토 워크플로(API) 생성

인적 검토 워크플로 생성(콘솔)

Amazon S3 자체 문서를 사용하여 이 예제를 완료하거나 다운로드할 수 있습니다.이 샘플 문서Amazon S3 버킷에 넣습니다.

S3 버킷이 동일한지 확인합니다.AWSAmazon Textract Textract를 사용 중인 리전입니다. 버킷을 만들려면 단원을 참조하십시오.버킷 만들기의Amazon Simple Storage Service 콘솔.

> **Note**
>
> Amazon A2I 콘솔은 SageMaker 콘솔에 내장되어 있습니다. 콘솔을 사용하려면 SageMaker 콘솔에 액세스하고 작업 팀을 생성할 권한이 있어야 합니다. 시작하려면AmazonSageMakerFullAccessSageMaker에서 대부분의 작업을 수행하는 데 필요한 모든 권한을 포함하는 IAM 관리형 정책입니다. 자세한 내용은 단원을 참조하십시오.Amazon SageMaker SageMaker용 Identity and Access Management의Amazon SageMaker 개발자 안내서.

주제

1단계: 작업 팀 생성 (콘솔)

먼저 Amazon A2I 콘솔에서 작업 팀을 만들고 작업자로 자신을 추가하여 작업자 포털에서 사람 검토 작업을 미리 볼 수 있습니다. 작업 팀 구성원은 자신에게 할당된 다른 작업과 문서를 볼 수 있습니다.

작업자 이메일을 사용하여 프라이빗 작업 인력을 생성하려면 (Console)

1. 에서 SageMaker 콘솔을 엽니다.https://console.aws.amazon.com/sagemaker/.
2. 탐색 창의Ground Truth, 선택인력을 레이블 지정.
3. 프라이빗을 선택한 다음 Create private team(프라이빗 팀 생성)을 선택합니다.
4. Invite new workers by email(이메일로 새 작업자 초대)을 선택합니다.
5. 이 예에서는 작업자 포털을 미리 볼 수 있도록 하려는 다른 사람의 이메일 주소와 이메일 주소를 입력합니다. 쉼표로 구분된 최대 50개의 이메일 주소 목록을이메일 주소상자.
6. 조직 이름과 연락처 이메일을 입력합니다.
7. Create private team(프라이빗 팀 생성)을 선택합니다.

비공개 작업 팀에 자신을 추가하면 다음 주소로 이메일을 받게 됩니다.no-reply@verificationemail.com로그인 정보가 포함되어 있습니다. 이 이메일의 링크를 사용하여 암호를 재설정하고 작업자 포털에 로그인합니다. 전화한 후 사람의 검토 작업이 표시되는 곳입니다.`AnalyzeDocument`.

2단계: 인적 검토 워크플로 생성(콘솔)

이 단계에서는 Amazon Textract 인간 검토 워크플로를 생성합니다.

인적 검토 워크플로를 생성하려면 (콘솔)

1. 에서 Amazon A2I 콘솔을 엽니다.https://console.aws.amazon.com/a2i에 액세스하려면휴먼 리뷰 워크플로우페이지.
2. 선택휴먼 리뷰 워크플로우 생성.
3. 용이름에서 워크플로 이름을 입력합니다.
4. 용S3 버킷Amazon A2I에서 인적 검토 작업의 결과를 저장하도록 할 버킷을 선택합니다. 버킷을 선택하지 않은 경우 버킷 이름을 입력하도록 버킷의 이름을 입력합니다.
5. UnderIAM 역할를 선택합니다.새 역할 생성. 제목과 함께 창이 나타납니다.IAM 역할 생성. 이 창을 사용하여 이 역할이 액세스할 수 있도록 하려는 Amazon S3 버킷을 지정할 수 있습니다. 선택하지 않은 경우S3 버킷에서 4단계에서 지정한 출력 버킷과 입력 문서가 포함된 버킷을 지정합니다.
6. 용작업 유형, 선택Textract - 키-값 페어 추출.
7. InAmazon Textract 양식 추출 - 인간 검토 호출 조건에서 활성화 조건을 지정합니다. 작업자 포털에서 작업자 작업을 미리 볼 수 있도록 사용자 검토를 트리거하려면 문서에서 하나 이상의 키에 대해 높은 신뢰도 점수 임계값을 설정하는 것이 좋습니다.

   이 연습에서 제공된 샘플 문서를 사용한 경우 다음과 같이 활성화 조건을 지정합니다.

   a. 선택양식 키 신뢰도 점수나 특정 양식 키가 누락된 경우 특정 양식 키에 대한 인적 검토를 트리거합니다.
   b. 용키 이름입니다.를 입력합니다.**`Mail Address`**.
   c. 다음을 설정합니다.신원 자료사이의 임계값0과99.
   d. 다음을 설정합니다.검증 신뢰사이의 임계값0과99.
   e. 선택Amazon Textract Textract에서 식별된 모든 양식 키에 대해 인적 검토를 트리거합니다.
   f. 용**`identification confidence`**에서 0과 90 사이의 숫자를 선택합니다.
   g. 용**`qualification confidence`**에서 0과 90 사이의 숫자를 선택합니다.

   그러면 Amazon Textract Textract가 99보다 작은 신뢰 점수를 반환하는 경우 사람의 검토가 트리거됩니다.메일 주소및 해당 값 또는 문서에서 감지된 키-값 쌍에 대해 90보다 작은 신뢰 점수를 반환하는 경우
8. Under작업자 작업 템플릿 생성를 선택합니다.기본 템플릿에서 만들기.

9. 용템플릿 이름에 설명 이름을 입력합니다.
10. 용[Task description]를 사용하려면 다음과 유사한 항목을 추가합니다.

```
Read the instructions and review the document.
```

11. 용작업자고르다프라이빗.
12. 메뉴에서 생성한 비공개 팀을 선택합니다.
13. [생성(Create)]을 선택합니다.

인간 검토 워크플로우가 생성되면 이 워크플로우가 의 테이블에 나타납니다.휴먼 리뷰 워크플로우페이지. 를 사용합니다.상태입니다활성 상태워크플로 ARN을 복사하고 저장합니다.

인적 검토 워크플로(API) 생성

인적 검토 워크플로를 생성하거나흐름 정의Amazon A2I를 사용하여`CreateFlowDefinition`작업.

이 예에서는 Amazon S3 자체 문서를 사용하거나 다운로드할 수 있습니다.이 샘플 문서S3 버킷에 저장합니다.

Amazon S3 버킷이 동일한지 확인합니다.AWS통화에 사용할 리전`AnalyzeDocument`. 버킷을 생성하려면 의 지침을 따르십시오.버킷 만들기의Amazon Simple Storage Service 콘솔.

사전 요구사항

Amazon A2I API를 사용하여 사람 검토 워크플로를 생성하려면 다음 사전 요구 사항을 완료해야 합니다.

- Amazon A2I 및 Amazon Textract API 작업을 모두 호출할 수 있는 권한이 있는 IAM 역할을 구성합니다. 시작하려면 AWS 정책, AmazonAugmentedAIFullAccess 및 AmazonTextractFullAccess IAM 역할에 연결할 수 있습니다. 나중에 필요하므로 IAM 역할 Amazon 리소스 이름 (ARN) 을 기록해 두십시오.

  Amazon Textract Textract를 사용할 때 보다 세부적인 권한은 단원을 참조하십시오Amazon Textract Textract자격 증명 기반 정책 예제. Amazon A2I의 경우 단원을 참조하십시오.Amazon Augmented AI AI의 권한 및 보안의Amazon SageMaker 개발자 안내서.
- 비공개 작업 팀을 만들고 작업팀 ARN을 기록합니다. Amazon A2I의 새 사용자인 경우 의 지침을 따르십시오.1단계: 작업 팀 생성 (콘솔).
- 작업자 작업 템플릿을 생성합니다. 의 지침을 따르십시오.작업자 작업 템플릿 생성을 눌러 Amazon A2I 콘솔을 사용하여 템플릿을 생성합니다. 템플릿을 만들 때 다음을 선택합니다.텍스트 형식 추출...

에 대한템플릿 유형. 템플릿에서 바꾸기s3_arn문서의 Amazon S3 ARN을 사용합니다. 추가 작업자 지침 추가<full-instructions header="Instructions"></full-instructions>.

템플릿을 미리 보려면 IAM 역할에 에서 설명하는 권한이 있는지 확인합니다.작업자 작업 템플릿 미리 보기 사용.

템플릿을 생성한 후 작업자 작업 템플릿 ARN을 기록합니다.

에서 생성한 리소스를 사용합니다.사전 요구사항를 구성합니다.CreateFlowDefinition요청. 이 요청에서 활성화 조건을 JSON 형식으로 지정할 수도 있습니다. 활성화 조건을 구성하는 방법에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract Textract와 함께 인적 루프 활성화 조건 JSON 스키마 사용.

인적 검토 워크플로 생성 (Python용 AWS SDK (Boto3)

이 예제를 사용하려면*red*사양과 리소스가 포함된 텍스트.

먼저 다음 코드를 사용하여 활성화 조건을 JSON 객체로 인코딩합니다. 그러면 Amazon Textract Textract가 99보다 작은 신뢰 점수를 반환하는 경우 사람의 검토가 트리거됩니다.메일 주소및 해당 값 또는 문서에서 감지된 키-값 쌍에 대해 90보다 작은 신뢰 점수를 반환하는 경우 이 예에 제공된 샘플 문서를 사용하는 경우 이러한 활성화 조건은 사람의 검토 작업을 만듭니다.

```
import json

humanLoopActivationConditions = json.dumps("{
                "Conditions": [
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "Mail Address",
                            "KeyValueBlockConfidenceLessThan": 99,
                            "WordBlockConfidenceLessThan": 99
                        }
                    },
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "*",
                            "KeyValueBlockConfidenceLessThan": 90,
                            "WordBlockConfidenceLessThan": 90
                        }
```

```
                    }
                ]
            }"
)
```

사용HumanLoopActivationConditions를 구성하려면create_flow_definition요청. 다음 예제에서는 SDK for Python (Boto3) 를 사용하여create_flow_definitionus-west-2 AWS 리전에서. 개인 작업 팀을 사용하여 지정합니다.

```
response = client.create_flow_definition(
    FlowDefinitionName='string',
    HumanLoopRequestSource={
         'AwsManagedHumanLoopRequestSource': "AWS/Textract/AnalyzeDocument/Forms/V1"
    },
    HumanLoopActivationConfig={
        'HumanLoopActivationConditionsConfig': {
            'HumanLoopActivationConditions': humanLoopActivationConditions
        }
    },
    HumanLoopConfig={
        'WorkteamArn': "arn:aws:sagemaker:us-west-2:111122223333:workteam/private-
crowd/work-team-name",
        'HumanTaskUiArn': "arn:aws:sagemaker:us-west-2:111122223333:human-task-
ui/worker-task-template-name",
        'TaskTitle': "Add a task title",
        'TaskDescription': "Describe your task",
        'TaskCount': 1,
        'TaskAvailabilityLifetimeInSeconds': 3600,
        'TaskTimeLimitInSeconds': 86400,
        'TaskKeywords': ["Document Review", "Content Review"]
        }
    },
    OutputConfig={
        'S3OutputPath': "s3://DOC-EXAMPLE-BUCKET/prefix/",
    },
    RoleArn="arn:aws:iam::111122223333:role/role-name"
)
```

문서 분석

Amazon A2I를 Amazon Textract 문서 분석 워크플로에 통합하려면 다음을 구성합니다.HumanLoopConfig의AnalyzeDocument작업.

InHumanLoopConfig에서 인적 검토 워크플로 (흐름 정의) ARN을 지정합니다.FlowDefinitionArn인간 루프에 이름을 지정하십시오.HumanLoopName.

Analyze the Document (AWS SDK for Python (Boto3))

다음 예제에서는 SDK for Python (Boto3) 를 사용하여analyze_documentus-west-2. 교체하기### ####리소스가 포함된 텍스트 자세한 내용은 단원을 참조하십시오.분석_문서의AWSPython (Boto) API 참조 용 SDK.

```
client.analyze_document(Document={'S3Object': {"Bucket": "DOC-EXAMPLE-BUCKET",
 "Name": "document-name.png"}},
        HumanLoopConfig={"FlowDefinitionArn":"arn:aws:sagemaker:us-
west-2:111122223333:flow-definition/flow-definition-name",
                         "HumanLoopName":"human-loop-name",
                         "DataAttributes":{"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent",]}},
        FeatureTypes=["FORMS"])
```

Analyze the Document (AWS CLI)

다음 예제에서는 를 사용합니다.AWS호출할 CLIanalyze_document. 이 예에는 와 호환됩니다.AWSCLI 버전 2. 첫 번째는 약식 구문이고 두 번째 구문은 JSON 구문입니다. 자세한 내용은 단원을 참조하십시오.분석 문서의AWS CLI명령 참조.

```
aws textract analyze-document \
     --document '{"S3Object":{"Bucket":"bucket_name","Name":"file_name"}}' \
     --human-loop-config
 HumanLoopName="test",FlowDefinitionArn="arn:aws:sagemaker:eu-west-1:xyz:flow-
definition/
hl_name",DataAttributes='{ContentClassifiers=["FreeOfPersonallyIdentifiableInformation","Fre
     --feature-types '["FORMS"]'
```

```
aws textract analyze-document \
     --document '{"S3Object":{"Bucket":"bucket_name","Name":"file_name"}}' \
     --human-loop-config \
          '{"HumanLoopName":"test","FlowDefinitionArn":"arn:aws:sagemaker:eu-
west-1:xyz:flow-definition/hl_name","DataAttributes": {"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation","FreeOfAdultContent"]}}' \
```

```
    --feature-types '["FORMS"]'
```

> **Note**
>
> 코드 처리 문제가 발생할 수 있으므로 —human-loop-config 매개 변수에서 흰색 공백을 사용하지 마십시오.

이 요청에 대한 응답은 다음을 포함합니다.휴먼루프활성화출력는 인간 루프가 만들어졌는지 여부와 그랬다면 그 이유를 나타냅니다. 휴먼 루프가 생성된 경우 이 객체에는HumanLoopArn.

및 를 사용하는 예제에 대한 자세한 내용은AnalyzeDocument작업, 참조Amazon Textract Textract를 사용하여 문서 텍스트 분석.

모니터 인적 루프

Amazon A2I 콘솔 및 API를 사용하여 오류 발생 시 휴먼 루프에 대한 세부 정보를 보고 활성 휴먼 루프를 중지할 수 있습니다.

인적 루프 세부 정보 보기

Amazon A2I 콘솔에서 다음을 사용하여 휴먼 루프 상태를 볼 수 있습니다.Amazon A2I 런타임 API.

휴먼 루프에 대한 세부 정보를 찾으려면 (콘솔)

1. 에서 Amazon A2I 콘솔을 엽니다.https://console.aws.amazon.com/a2i에 액세스하려면휴먼 리뷰 워크플로우페이지.
2. 구성에 사용한 인적 검토 워크플로를 선택합니다.HumanLoopConfig에서AnalyzeDocument.
3. 에서휴먼 루프섹션에서 세부 정보를 보려는 인적 루프를 선택합니다.

휴먼 루프 (API) 에 대한 세부 정보를 찾으려면

Amazon A2I 사용DescribeHumanLoop작업. 호출할 때 사용한 휴먼 루프 이름 지정AnalyzeDocument.

Python용 다음 SDK (Boto3) 예제 호출describe_human_loop.

```
response = client.describe_human_loop(HumanLoopName="human-loop-name")
```

인적 루프 중지

휴먼 루프가 시작된 후에는 Amazon A2I 콘솔과 API를 사용하여 중지할 수 있습니다.

휴먼 루프를 중지하려면 (콘솔)

1. 에서 Amazon A2I 콘솔을 엽니다.https://console.aws.amazon.com/a2i에 액세스하려면휴먼 리뷰 워크플로우페이지.
2. 구성에 사용한 인적 검토 워크플로를 선택합니다.`HumanLoopConfig`의`AnalyzeDocument`작업.
3. 에서휴먼 루프섹션에서 중지할 인적 루프를 선택합니다.
4. 중지를 선택합니다.

휴먼 루프 (API) 를 중지하려면

Amazon A2I 사용`StopHumanLoop`작업. 호출할 때 사용한 인적 루프의 이름을 지정합니다.`AnalyzeDocument`.

Python용 다음 SDK (Boto3) 예제 호출`stop_human_loop`.

```
response = client.stop_human_loop(HumanLoopName="human-loop-name")
```

출력 데이터 및 작업자 지표 보기

작업자가 사람 검토 작업을 완료하면 Amazon A2I는 사용자가 사용자 검토 워크플로에서 지정한 Amazon S3 버킷에 출력 데이터를 저장합니다.

비공개 인력을 사용하는 경우 출력 데이터에 개별 작업자 활동을 추적하는 데 사용할 수 있는 작업자 메타데이터가 포함됩니다.

Amazon S3 출력 데이터 찾기

Amazon A2I는 휴먼 검토 워크플로우를 사용하여 생성된 휴먼 루프의 출력 데이터를 저장하는 파일 이름의 접두사로 사용자 검토 워크플로 이름을 사용합니다.

휴먼 루프 출력의 경로는 다음과 같은 패턴을 사용합니다.*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*연도와 함께 휴먼 루프 생성 날짜를 나타냅니다 (YYYY), 월 (MM), 하루 (DD) 및 생성 시간 (시간 포함)hh), 분 (mm), 두 번째 (ss).

```
s3://output-bucket-specified-in-flow-definition/flow-definition-
name/YYYY/MM/DD/hh/mm/ss/human-loop-name/output.json
```

휴먼 루프의 출력을 보려면 Amazon A2I 콘솔을 사용하십시오.

휴먼 루프 출력을 보려면

1. 에서 Amazon A2I 콘솔을 엽니다.https://console.aws.amazon.com/a2i에 액세스하려면휴먼 리뷰 워크플로우페이지.
2. 구성에 사용하는 인적 검토 워크플로를 선택합니다.HumanLoopConfig에서AnalyzeDocument.
3. 에서휴먼 루프섹션에서 결과를 검토할 휴먼 루프를 선택합니다.
4. Under출력 위치에서 출력 데이터에 대한 링크를 선택합니다.

비공개 근로자 활동 추적

개인 검토 태스크에 개인 인력을 사용하는 경우 결과 데이터에는 검토를 완료한 근로자에 대한 다음 정보가 포함됩니다.

- 이 workerId.
- workerMetadata에서:
  - identityProviderType— 프라이빗 작업 인력을 관리하는 데 사용되는 서비스입니다.
  - issuer— 이 인적 검토 태스크에 할당된 작업 팀과 연결된 Amazon Cognito 사용자 풀 또는 OIDC 자격 증명 공급자 (IdP) 발급자입니다.
  - sub— 작업자를 참조하는 고유 식별자입니다. Amazon Cognito를 사용하여 인력을 생성한 경우 Amazon Cognito를 사용하여 이 ID를 사용하여 이 작업자 (예: 이름 또는 사용자 이름) 에 대한 세부 정보를 검색할 수 있습니다. 자세한 방법은 단원을 참조하십시오.사용자 계정 관리 및 검색에서Amazon Cognito 개발자 안내서.

다음은 Amazon Cognito를 사용하여 비공개 인력을 만들었는지 확인할 수 있는 결과물의 예입니다.

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Cognito",
                    "issuer": "https://cognito-idp.aws-region.amazonaws.com/aws-
region_123456789",
```

```
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

다음은 자체 OIDC IdP를 사용하여 비공개 인력을 만들었는지 확인할 수 있는 결과물의 예입니다.

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Oidc",
                    "issuer": "https://example-oidc-ipd.com/adfs",
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

프라이빗 작업 인력을 사용하는 방법에 대한 자세한 내용은 단원을 참조하십시오.프라이빗 작업 인력 사용의Amazon SageMaker 개발자 안내서.

Amazon Textract의 보안

AWS에서 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

다음 주제를 확인하여 Amazon Textract 리소스를 보호하는 방법을 알아보십시오.

주제

Amazon Textract Textract의 데이터 보호

Amazon Textract 다음을 준수합니다.AWS 공동 책임 모델여기에는 데이터 보호에 대한 규정 및 지침이 포함됩니다.AWS는 모든 것을 실행하는 글로벌 인프라를 보호할 책임이 있습니다.AWS서비스.AWS는 고객 콘텐츠 및 개인 데이터를 처리하기 위한 보안 구성 컨트롤을 포함하여 이 인프라에서 호스팅되는 데이터의 제어를 유지합니다.AWS고객APN데이터 관리자 또는 데이터 프로세서 역할을 하는 파트너는 에 입력한 모든 개인 데이터에 대한 책임이 있습니다.AWS클라우드.

데이터를 보호하려면 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management(IAM)을 사용해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여할 수 있습니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정마다 멀티 팩터 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다.
- AWS CloudTrail으로 API 및 사용자 활동 로깅을 설정합니다.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 보안 컨트롤 기본값과 함께 사용합니다.

- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마세요. 여기에는 Amazon Textract Textract나 기타로 작업하는 경우가 포함됩니다.AWS콘솔을 사용하는 서비스, API,AWS CLI또는AWSSDK. Amazon Textract 또는 기타 서비스에 입력하는 모든 데이터를 진단 로그에 포함할 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시키지 마세요.

데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

Amazon Textract의 암호화

데이터 암호화는 전송 중이거나 상주 중인 데이터를 보호하는 것을 의미합니다. Amazon S3 관리형 키를 사용하여 데이터를 보호할 수 있습니다.AWS KMS key전송 중 표준 전송 계층 보안과 함께 휴면.

저장 데이터 암호화

Amazon Textract Textract에서 데이터를 암호화하는 기본 방법은 서버 측 암호화입니다. Amazon S3 버킷에서 전달된 입력 문서는 Amazon S3 의해 암호화되고 사용자가 액세스할 때 해독됩니다. 요청을 인증하기만 하면 액세스 권한을 갖게 되며, 객체의 암호화 여부와 관계없이 액세스 방식에는 차이가 없습니다. 예를 들어, 미리 서명된 URL을 사용하여 객체를 공유하는 경우, 해당 URL은 암호화된 객체와 암호화되지 않은 객체에 동일하게 작동합니다. 또한 버킷의 객체를 나열할 때`List`API는 암호화 여부와 관계없이 전체 객체의 목록을 반환합니다.

Amazon Textract TExtract는 상호 배타적인 서버 측 암호화 방법을 사용합니다.

Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE-S3)

Amazon S3 관리형 키 (SSE-S3) 를 사용하여 서버 측 암호화 () 를 사용하면 각 객체는 고유한 키로 암호화됩니다. 또한 추가 보안 조치로 주기적으로 바뀌는 마스터 키를 사용하여 키 자체를 암호화합니다. Amazon S3 서버 측 암호화는 가장 강력한 블록 암호 중 하나인 256비트 Advanced Encryption Standard(AES-256)를 사용하여 데이터를 암호화합니다. 더 자세한 내용은 Amazon S3-관리 암호화 키(SSE-S3)와 함께 서버측 암호화를 사용하여 보호하기를 참조하십시오.

AWS Key Management Service (SSE-KMS) 에 저장된 KMS 키를 사용한 서버 측 암호화

AWS Key Management Service에 저장된 KMS 키를 사용한 서버 측 암호화 (SSE-KMS) 는 SSE-S3 와 유사하지만 이 서비스 사용 시 몇 가지 추가적인 이점이 있고 비용이 발생합니다. Amazon S3의 객체

에 대한 무단 액세스에 대응하여 추가적인 보호를 제공하는 -KMS 키를 사용하려면 별도의 권한이 필요합니다. SSE-KMS도 KMS 키가 사용된 때와 사용 주체를 표시하는 감사 추적 기능을 제공합니다. 또한 KMS 키를 생성 및 관리하거나 사용할 수 있습니다.AWS 관리형 키이는 사용자, 서비스 및 리전에 고유한 역할을 합니다. 자세한 내용은 단원을 참조하십시오.서버 측 암호화를 사용하여 데이터 보호 AWS Key Management Service (SSE-KMS) 에 저장된 KMS 키를 사용합니다.

전송 중 데이터 암호화

전송 중인 데이터의 경우 Amazon Textract Textract는 전송 계층 보안 (TLS) 를 사용하여 서비스와 에이전트 간에 전송된 데이터를 암호화합니다. 또한 Amazon Textract Textract는 VPC 엔드포인트를 사용하여 Amazon Textract Textract가 문서를 처리할 때 사용되는 다양한 마이크로서비스 간에 데이터를 전송합니다.

인터네트워크 트래픽 개인 정보

Amazon Textract Textract는 HTTPS 엔드포인트를 통해서만 통신합니다. HTTPS 엔드포인트는 Amazon Textract Textract에서 지원하는 모든 리전에서 지원됩니다.

Amazon Textract Textract의 Identity and Access Management

AWS Identity and Access Management(IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 누구가 될 수 있는지 제어인증되었습니다(로그인) 및인정 받은Amazon Textract 리소스를 사용할 수 있는 권한이 있습니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

대상

사용 방식AWS Identity and Access Management(IAM) 는 Amazon Textract 에서 수행하는 작업에 따라 달라집니다.

서비스 사용자— Amazon Textract 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 Amazon Textract 기능을 사용하여 작업을 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon Textract Textract의 기능에 액세스할 수 없는 경우 단원을 참조하십시오.Amazon Textract Textract자격 증명 및 액세스 문제 해결.

서비스 관리자— 회사에서 Amazon Textract 리소스를 책임지고 있다면 Amazon Textract에 대한 완전한 액세스 권한이 있을 것입니다. 서비스 관리자는 직원이 액세스해야 하는 Amazon Textract 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 Amazon Textract 에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 단원을 참조하십시오.Amazon Textract Textract가 IAM으로 작업하는 방식.

IAM 관리자— IAM 관리자라면 Amazon Textract Textract에 대한 액세스 권한 관리 정책을 작성하는 방법에 대해 자세히 알아보려고 할 수 있습니다. IAM에서 사용할 수 있는 Amazon Textract 자격 증명 기반 정책 예제를 보려면 단원을 참조하십시오.Amazon Textract Textract자격 증명 기반 정책 예제.

자격 증명을 통한 인증

인증은 ID 자격 증명을 사용하여 AWS에 로그인하는 방식입니다. AWS Management Console을 사용한 로그인에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자 또는 루트 사용자로 AWS Management Console에 로그인하기를 참조하세요.

AWS 계정 루트 사용자로, IAM 사용자로, 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다. 회사의 Single Sign-On(SSO) 인증을 사용하거나 Google 또는 Facebook을 사용하여 로그인할 수도 있습니다. 이러한 경우 관리자는 이전에 IAM 역할을 사용하여 자격 증명 연동을 설정한 것입니다. 다른 회사의 자격 증명을 사용하여 AWS에 액세스하면 간접적으로 역할을 가정하는 것입니다.

AWS Management Console에 직접 로그인하려면 루트 사용자 이메일 주소 또는 IAM 사용자 이름과 암호를 사용하세요. 루트 사용자 또는 IAM 사용자 액세스 키를 사용하여 프로그래밍 방식으로 AWS에 액세스할 수 있습니다. AWS는 자격 증명을 사용하여 암호화 방식으로 요청에 서명할 수 있는 SDK 및 명령줄 도구를 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 이렇게 하려면 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 사용합니다. 요청 인증에 대한 자세한 내용은 AWS 일반 참조의 서명 버전 4 서명 프로세스를 참조하세요.

사용하는 인증 방법에 상관 없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS는 멀티 팩터 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

AWS 계정을 처음 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 SSO(Single Sign-In) ID로 시작합니다. 이 자격 증명은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업, 심지어 관리 작업의 경우에도 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 대신, IAM 사용자를 처음 생성할 때만 루트 사용자를 사용하는 모범 사례를 준수하세요. 그런 다음 루트 사용자 자격 증명을 안전하게 보관하고 몇 가지 계정 및 서비스 관리 태스크를 수행할 때만 사용합니다.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 자격 증명입니다. IAM 사용자에게는 사용자 이름과 암호 또는 액세스 키 세트와 같은 장기 자격 증명이 있을 수 있습니다. 액세스 키를 생성하는 방법은 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요. IAM 사용자의 액세스 키를 생성할 때는 키 페어를 보고 안전하게 저장해야 합니다. 향후에 보안 액세스 키를 복구할 수 없습니다. 그 대신 새 액세스 키 페어를 생성해야 합니다.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 귀하는 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 자격 증명만 제공합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 태스크를 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

임시 자격 증명이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 임시 IAM 사용자 권한 - IAM 사용자는 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.

- 페더레이션 사용자 액세스 – IAM 사용자를 생성하는 대신 AWS Directory Service의 기존 자격 증명, 엔터프라이즈 사용자 디렉터리 또는 웹 자격 증명 공급자를 사용할 수 있습니다. 이 사용자를 페더레이션 사용자라고 합니다. AWS에서는 자격 증명 공급자를 통해 액세스가 요청되면 페더레이션 사용자에게 역할을 할당합니다. 연합된 사용자에 대한 자세한 내용은 IAM 사용 설명서의 연합된 사용자 및 역할을 참조하세요.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 교차 계정 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 역할을 (프록시로 사용하는 대신) 리소스에 정책을 직접 연결할 수 있습니다. 교차 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.
- 교차 서비스 액세스 – 일부 AWS 서비스는 다른 AWS 서비스의 기능을 사용합니다. 예를 들어 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 태스크를 수행할 수 있습니다.
  - 보안 주체 권한 - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 정책은 보안 주체에게 권한을 부여합니다. 일부 서비스를 사용할 때는 다른 서비스에서 다른 태스크를 트리거하는 태스크를 수행할 수 있습니다. 이 경우 두 태스크를 모두 수행할 수 있는 권한이 있어야 합니다. 작업에 정책에서 추가 종속 작업이 필요한지 여부를 확인하려면 단원을 참조하십시오.Amazon Textract에 사용되는 작업, 리소스 및 조건 키의서비스 승인 참조.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 수임하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
  - 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 태스크를 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 IAM 계정에 표시되고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행 중인 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 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 자격 증명을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용하여 액세스 관리

정책을 생성하고 IAM 자격 증명 또는 AWS 리소스에 연결하여 AWS에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. 루트 사용자 또는 IAM 사용자로 로그인하거나 IAM 역할을 수임할 수 있습니다. 그런 다음 요청을 수행하면 AWS는 관련 자격 증명 기반 또는 리소스 기반 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로서 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

모든 IAM 개체(사용자 또는 역할)는 처음에는 권한이 없습니다. 다시 말해, 기본적으로 사용자는 아무 작업도 수행할 수 없으며, 자신의 암호를 변경할 수도 없습니다. 사용자에게 태스크를 수행할 권한을 부여하기 위해 관리자는 사용자에게 권한 정책을 연결해야 합니다. 또한 관리자는 의도한 권한을 가지고 있는 그룹에 사용자를 추가할 수 있습니다. 관리자가 그룹에 권한을 부여하면 그룹의 모든 사용자가 해당 권한을 받습니다.

IAM 정책은 태스크를 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

자격 증명 기반 정책

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 에 속한 다수의 사용자, 그룹 및 역할에게 독립적으로 추가할 수 있는 정책입니다.AWS 계정. 관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 제어할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연합된 사용자 또는 AWS 서비스가 포함될 수 있습니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

ACL(액세스 제어 목록)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 WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대한 자세한 내용은 단원을 참조하십시오.ACL(액세스 제어 목록) 개요의Amazon Simple Service 개발자 가이드.

기타 정책 유형

AWS는 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 유형은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 엔터티에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations는 기업이 소유하는 여러 개의 AWS 계정을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자를 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.

- 세션 정책 – 세션 정책은 역할 또는 연합된 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon Textract Textract가 IAM으로 작업하는 방식

IAM을 사용하여 Amazon Textract Textract에 대한 액세스를 관리하기 전에 Amazon Textract Textract에서 사용할 수 있는 IAM 기능을 이해해야 합니다. Amazon Textract Textract와 기타 방법에 대한 상위 수준에서 보려면AWSIAM으로 작업하는 서비스는 단원을 참조하십시오.AWSIAM으로 작업하는 서비스의IAM 사용 설명서.

주제

- Amazon Textract Textract의 자격 증명 기반 정책
- Amazon Textract Textract의 리소스 기반 정책
- Amazon Textract Textract태그를 기반으로 권한 부여
- Amazon Textract IAM 역할

Amazon Textract Textract의 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스 및 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. Amazon Textract Textract는 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWS API 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함시킵니다.

Amazon Textract Textract의 비동기 작업에는 시작 작업과 가져오기 작업에 대한 두 가지 작업 권한이 부여되어야 합니다. 또한 Amazon S3 버킷을 사용하여 문서를 전달하는 경우 계정에 읽기 액세스 권한을 부여해야 합니다.

Amazon Textract Textract에서 모든 정책 조치는 다음과 같이 시작합니다. textract:. 예를 들어 Amazon Textract를 사용하여 Amazon Textract 작업을 실행할 수 있는 권한을 부여하는 경우AnalyzeDocument작업, 당신은 다음을 포함합니다.textract:AnalyzeDocument그들의 정책에서의 행동. 정책 설명에는 Action 또는 NotAction 요소가 반드시 추가되어야 합니다. Amazon Textract Textract는 이 서비스로 수행할 수 있는 태스크를 설명하는 고유한 작업 집합을 정의합니다.

명령문 하나에 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "textract:action1",
      "textract:action2"
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "textract:Describe*"
```

Amazon Textract Textract의 작업 목록에 대해서는 단원을 참조하십시오.Amazon Textract에서 정의한 작업의IAM 사용 설명서.

리소스

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 태스크를 수행 할 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 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우 와일드카드(*)를 사용하여 명령문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon Textract TExtract는 정책에서 리소스 ARN 지정을 지원하지 않습니다.

조건 키

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 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 선택 사항입니다. 같음이나 미만 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 AWS는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 `OR` 태스크를 사용하여 조건을 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리표시자 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.

AWS는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon Textract Textract는 서비스별 조건 키를 제공하지 않지만, 일부 전역 조건 키 사용을 지원합니다. 전체 목록을 보려면AWS전역 조건 키, 를 참조하십시오.AWS전역 조건 컨텍스트 키의IAM 사용 설명서.

예제

Amazon Textract Textract자격 증명 기반 정책의 예를 보려면 단원을 참조하십시오.Amazon Textract Textract자격 증명 기반 정책 예제.

Amazon Textract Textract의 리소스 기반 정책

Amazon Textract Textract는 리소스 기반 정책을 지원하지 않습니다.

Amazon Textract Textract태그를 기반으로 권한 부여

Amazon Textract TExtract는 리소스 태그 지정 또는 태그 기반 액세스 제어를 지원하지 않습니다.

Amazon Textract IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 내 엔터티입니다.

Amazon Textract 임시 자격 증명 사용

임시 자격 증명을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken 같은 AWS STS API 태스크를 호출하여 임시 보안 자격 증명을 가져옵니다.

Amazon Textract Textract는 임시 자격 증명 사용을 지원합니다.

서비스 연결 역할

서비스 연결 역할을 사용하면 AWS 제품이 다른 서비스의 리소스에 액세스하여 사용자 대신 태스크를 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Amazon Textract Textract는 서비스 연결 역할을 지원하지 않습니다.

> **Note**
>
> Amazon Textract Textract는 서비스 연결 역할을 지원하지 않으므로 AWS 서비스 주체를 지원하지 않습니다. 서비스 주체에 대한 자세한 내용은 단원을 참조하십시오.AWS 서비스 보안 주체의IAM 사용 설명서

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 서비스 역할을 수임할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 태스크를 완료할 수 있습니다. 서비스 역할은 IAM 계정에 나타나고, 해당 계정이 소유합니다. 즉, IAM 관리자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

Amazon Textract Textract는 서비스 역할을 지원합니다.

Amazon Textract Textract자격 증명 기반 정책 예제

기본적으로 IAM 사용자 및 역할은 Amazon Textract 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS CLI 또는 AWS API를 사용해 태스크를 수행할 수 없습니다. IAM 관리자는 지정된 리소스에서 특정 API 태스크를 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하세요.

주제

- 정책 모범 사례
- 사용자가 자신이 권한을 볼 수 있도록 허용
- Amazon Textract Textract에서 동기식 작업에 대한 액세스 권한 부여
- Amazon Textract Textract에서 비동기 작업에 대한 액세스 권한 부여

정책 모범 사례

자격 증명 기반 정책은 매우 강력합니다. 이러한 정책은 계정에서 사용자가 Amazon Textract 리소스를 생성, 액세스 또는 삭제할 수 있는지를 결정합니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- 사용 시작하기AWS관리형 정책— Amazon Textract Textract를 빠르게 사용하려면 다음을 사용하십시오.AWS직원에게 필요한 권한을 부여하는 관리형 정책입니다. 이 정책은 이미 계정에서 사용할 수 있으며 AWS에 의해 유지 관리 및 업데이트됩니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책으로 권한 사용 시작하기를 참조하세요.
- 최소 권한 부여 - 사용자 지정 정책을 생성할 때 태스크를 수행하는 데 필요한 권한만 부여합니다. 최소한의 권한 조합으로 시작하여 필요에 따라 추가 권한을 부여합니다. 처음부터 권한을 많이 부여한 후 나중에 줄이는 방법보다 이 방법이 안전합니다. 자세한 내용은 IAM 사용 설명서의 최소 권한 부여를 참조하세요.
- 중요한 작업에 대해 MFA 활성화 - 보안을 강화하기 위해 IAM 사용자가 중요한 리소스 또는 API 작업에 액세스할 때 멀티 팩터 인증(MFA)을 사용하도록 합니다. 자세한 내용은 IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.

- 보안 강화를 위해 정책 조건 사용 – 실제로 가능한 경우 자격 증명 기반 정책이 리소스에 대한 액세스를 허용하는 조건을 정의합니다. 예를 들어 요청을 할 수 있는 IP 주소의 범위를 지정하도록 조건을 작성할 수 있습니다. 지정된 날짜 또는 시간 범위 내에서만 요청을 허용하거나, SSL 또는 MFA를 사용해야 하는 조건을 작성할 수도 있습니다. 자세한 내용은 단원을 참조하십시오.IAM JSON 정책 요소: Condition의IAM 사용 설명서.

사용자가 자신이 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI나 AWS API를 사용하여 프로그래밍 방식으로 이 태스크를 완료할 수 있는 권한이 포함됩니다.

```
{
    "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": "*"
        }
```

```
    ]
}
```

Amazon Textract Textract에서 동기식 작업에 대한 액세스 권한 부여

이 예제 정책은 Amazon Textract Textract의 동기식 작업에 대한 액세스 권한을 사용자의 IAM 사용자에게 부여합니다.AWS계정.

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument"
            ],
            "Resource": "*"
        }
    ]
```

Amazon Textract Textract에서 비동기 작업에 대한 액세스 권한 부여

다음 예제 정책은 IAM 사용자에게AWSAmazon Textract Textract에서 사용되는 모든 비동기 작업에 대한 계정 액세스

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:StartDocumentTextDetection",
                "textract:StartDocumentAnalysis",
                "textract:GetDocumentTextDetection",
                "textract:GetDocumentAnalysis"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Textract Textract자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amazon Textract 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon Textract Textract에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행할 권한이 없음
- 액세스 키를 보기를 원함
- 관리자인데, 다른 사용자가 Amazon Textract Textract에 액세스할 수 있게 허용하기를 원함
- 내 외부의 사람들을 허용하기를 원함AWS내 Amazon Textract 리소스에 액세스할 수 있는 계정

Amazon Textract Textract에서 작업을 수행할 권한이 없음

AWS Management Console에서 태스크를 수행할 권한이 없다는 메시지가 나타나는 경우 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 암호를 제공한 사람입니다.

다음 예제 오류는 다음과 같은 경우 발생합니다.`mateojackson`IAM 사용자가 실행을 시도합니다.`DetectDocumentText`테스트 이미지에는 있지만`textract:DetectDocumentText`권한.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 textract:DetectDocumentText on resource: textimage.png
```

이 경우 Mateo는 `textimage.png` 작업을 사용하여 `textract:DetectDocumentText` 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행할 권한이 없음

`iam:PassRole` 태스크를 수행할 권한이 없다는 오류가 수신되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 암호를 제공한 사람입니다. 역할을 Amazon Textract 에 전달할 수 있도록 정책을 업데이트하라고 관리자에게 요청합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신, 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 될 때 발생합니다.`marymajor`는 콘솔을 사용하여 Amazon Textract Textract에서 작업을 수행하려고 합니다. 하지만 태스크를 수행하려면 서비스에 서비스 역할

이 부여한 권한이 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 Mary는 `iam:PassRole` 태스크를 수행하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

액세스 키를 보기를 원함

IAM 사용자 액세스 키를 생성한 후에는 언제든지 액세스 키 ID를 볼 수 있습니다. 하지만 보안 액세스 키는 다시 볼 수 없습니다. 보안 액세스 키를 잃어버린 경우 새로운 액세스 키 페어를 생성해야 합니다.

액세스 키는 액세스 키 ID(예: `AKIAIOSFODNN7EXAMPLE`)와 보안 액세스 키(예: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)의 두 가지 부분으로 구성됩니다. 사용자 이름 및 암호와 같이 액세스 키 ID와 보안 액세스 키를 함께 사용하여 요청을 인증해야 합니다. 사용자 이름과 암호를 관리하는 것처럼 안전하게 액세스 키를 관리합니다.

> **Important**
>
> 정식 사용자 ID를 찾는 데 도움이 되더라도 액세스 키를 타사에 제공하지 마시기 바랍니다. 이로 인해 다른 사람에게 계정에 대한 영구 액세스를 제공하게 될 수 있습니다.

액세스 키 페어를 생성할 때는 액세스 키 ID와 보안 액세스 키를 안전한 위치에 저장하라는 메시지가 나타납니다. 보안 액세스 키는 생성할 때만 사용할 수 있습니다. 하지만 보안 액세스 키를 잃어버린 경우 새로운 액세스 키를 IAM 사용자에게 추가해야 합니다. 최대 두 개의 액세스 키를 가질 수 있습니다. 이미 두 개가 있는 경우 새로 생성하려면 먼저 키 페어 하나를 삭제해야 합니다. 지침을 보려면 IAM 사용 설명서의 액세스 키 관리 단원을 참조하세요.

관리자인데, 다른 사용자가 Amazon Textract Textract에 액세스할 수 있게 허용하기를 원함

다른 사용자가 Amazon Textract Textract에 액세스하도록 허용하려면 액세스 권한이 필요한 사용자 또는 애플리케이션에 대한 IAM 엔터티 (사용자 또는 역할) 를 생성해야 합니다. 다른 사용자들은 해당 엔터티에 대한 자격 증명을 사용해 AWS에 액세스합니다. 그런 다음 Amazon Textract Textract에서 올바른 권한을 부여하는 정책을 엔터티에 연결해야 합니다.

바로 시작하려면 IAM 사용 설명서의 첫 번째 IAM 위임 사용자 및 그룹 생성을 참조하세요.

내 외부의 사람들을 허용하기를 원함AWS내 Amazon Textract 리소스에 액세스할 수 있는 계정

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스하는 데 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 ACL(액세스 제어 목록)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- Amazon Textract Textract가 이러한 기능을 지원하는지 여부를 알아보려면 을 참조하십시오.Amazon Textract Textract가 IAM으로 작업하는 방식.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- 자격 증명 연동을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 연동)을 참조하세요.
- 교차 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

로깅 및 모니터링

Amazon Textract 추출을 모니터링하려면 Amazon CloudWatch를 사용하십시오. 이 단원에서는 Amazon Textract Textract에 대한 모니터링을 설정하는 방법에 대해 설명합니다. 또한 Amazon Textract 지표에 대한 참조 콘텐츠도 제공합니다.

주제

- Amazon Textract 모니터링
- Amazon Textract의 CloudWatch 지표

Amazon Textract 모니터링

CloudWatch를 사용하면 계정에 대한 개별 Amazon Textract 작업 측정치 또는 전역 Amazon Textract 측정치를 얻을 수 있습니다. 측정치를 사용하여 Amazon Texttract 기반 솔루션의 상태를 추적하고, 하나 이상의 측정치가 정의된 임계값을 벗어나면 이를 알리도록 경보를 설정할 수 있습니다. 예를 들

어 발생한 서버 오류 수에 대한 측정치를 볼 수 있습니다. 또한 특정 Amazon Textract 작업이 성공한 횟수에 대한 측정치도 볼 수 있습니다. 지표를 보려면 를 사용하여Amazon CloudWatch,AWS CLI, 또는CloudWatch API.

Amazon Textract Textract에 CloudWatch 지표 사용

측정치를 사용하려면 다음 정보를 지정해야 합니다.

- 측정치 차원 또는 차원 없음. 차원은 지표를 고유하게 식별하는 데 도움이 되는 이름-값 페어입니다. Amazon Textract Textract에는 하나의 차원이 있습니다.작업. 이는 특정 작업에 대한 측정치를 제공합니다. 차원을 지정하지 않으면 측정치의 범위가 계정 내의 모든 Amazon Textract 작업으로 지정됩니다.
- `UserErrorCount`와 같은 지표 이름.

을 사용하여 Amazon Textract Textract에 대한 모니터링 데이터를 얻을 수 있습니다.AWS Management Console,AWS CLI또는 CloudWatch API. Amazon AWS 소프트웨어 개발 키트 (SDK) 또는 CloudWatch API 도구 중 하나를 통해 CloudWatch API를 사용할 수도 있습니다. 콘솔에는 CloudWatch API의 원시 데이터를 기초로 일련의 그래프가 표시됩니다. 필요에 따라 콘솔에 표시되거나 API에서 가져온 그래프를 사용하는 것이 더 나을 수 있습니다.

다음은 몇 가지 일반적인 지표 사용 사례입니다. 모든 사용 사례를 망라한 것은 아니지만 시작하는 데 참고가 될 것입니다.

방법	관련 지표
내 애플리케이션이 초당 최대 요청 수에 도달했는지 여부를 어떻게 알 수 있습니까?	`ThrottledCount` 측정치의 `Sum` 통계를 모니터링합니다.
요청 오류는 어떻게 모니터링할 수 있습니까?	`UserErrorCount` 측정치의 `Sum` 통계를 사용합니다.
총 요청 수를 찾으려면 어떻게 해야 합니까?	`ResponseTime` 측정치의 `SampleCount` 통계를 사용합니다. 여기에는 오류를 초래하는 모든 요청이 포함됩니다. 성공한 작업 호출만 보려면 `SuccessfulRequestCount` 측정치를 사용합니다.

방법	관련 지표
Amazon Textract Textract의 작업 호출의 지연 시간은 어떻게 모니터링할 수 있습니까?	`ResponseTime` 측정치를 사용합니다.

CloudWatch를 사용하여 Amazon Textract Textract를 모니터링하려면 적절한 CloudWatch 권한이 있어야 합니다. 자세한 내용은 Amazon CloudWatch에 대한 인증 및 액세스 제어를 참조하십시오.

Amazon Textract Textract용 지표

다음 예제에서는 CloudWatch 콘솔인AWS CLI을 사용하여 CloudWatch API를 사용할 수 있습니다.

지표를 보려면(콘솔)

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 선택지표를 선택합니다.모든 지표탭을 선택한 다음Amazon Textract.
3. 선택작업에 의한 작업[] 를 선택한 다음 측정치를 선택합니다.

   예를 들어, 다음을 선택합니다.StartDocumentAnalysis비동기 문서 분석이 시작된 횟수를 측정하는 메트릭입니다.
4. 날짜 범위 값을 선택합니다. 측정치 개수는 그래프에 표시됩니다.

성공에 대한 지표를 보려면**`StartDocumentAnalysis`**일정 기간 동안 이루어진 작업 호출 (CLI)

- AWS CLI를 열고 다음 명령을 입력합니다.

```
aws cloudwatch get-metric-statistics \
    --metric-name SuccessfulRequestCount \
    --start-time 2019-02-01T00:00:00Z \
    --period 3600 \
    --end-time 2019-03-01T00:00:00Z \
    --namespace AWS/Textract \
    --dimensions Name=Operation,Value=StartDocumentAnalysis \
    --statistics Sum
```

  이 예제는 일정 기간 동안 성공적으로 이루어진 `StartDocumentAnalysis` 작업 호출을 보여 줍니다. 자세한 내용은 get-metric-statistics를 참조하십시오.

측정치에 액세스하려면 (CloudWatch API)

- 을 호출합니다..`GetMetricStatistics` 자세한 내용은 단원을 참조하십시오.Amazon CloudWatch API 레퍼런스.

경보 만들기

경보 때문에 상태가 변경되면 Amazon Simple Notification Service (Amazon SNS) 메시지를 보내는 CloudWatch 경보를 생성할 수 있습니다. 경보는 지정한 기간 동안 단일 지표를 감시합니다. 경보는 기간 수에 대한 주어진 임계값과 지표 값을 비교하여 하나 이상의 작업을 수행합니다. 이 작업은 Amazon SNS 주제 또는 Auto Scaling 정책에 전송되는 알림입니다.

경보는 지속적인 상태 변경에 대해서만 작업을 호출합니다. CloudWatch 경보는 특정 상태에 있다고 해서 작업을 호출하지는 않습니다. 상태가 변경되어 지정된 기간 동안 유지되어야 합니다.

경보를 설정하려면(콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [] 를 선택합니다.경보을 선택하고경보 생성. 이 열립니다.Create Alarm 마법사.
3. 지표 선택(Select metric)을 선택하세요.
4. 에서모든 지표탭, 선택Textract.
5. 선택작업에 의한 작업[] 를 선택한 다음 측정치를 선택합니다.

   예를 들어, 다음을 선택합니다.StartDocumentAnalysis을 눌러 최대 비동기 문서 분석 작업에 대한 경보를 설정합니다.
6. 그래프로 표시된 지표(Graphed metrics) 탭을 선택합니다.
7. Statistic(통계)에서 Sum(합계)를 선택합니다.
8. 지표 선택(Select metric)을 선택하세요.
9. [Name]과 [Description]을 입력합니다. [Whenever]에서 [>=]를 선택하고 원하는 최대값을 입력합니다.
10. 경보 상태에 도달할 때 CloudWatch에서 이메일을 보내도록 하려면이 알람 때마다:, 선택상태는 알람입니다.. 기존 Amazon SNS 주제에 경보를 전송하려면다음 주소로 알림 보내기:에서 기존 SNS 주제를 선택합니다. 새 이메일 구독 목록에 대한 이름 및 이메일 주소를 설정하려면새 목록. CloudWatch는 목록을 저장하고 필드에 표시하므로 나중에 경보를 설정하는 데 사용할 수 있습니다.

> **Note**
>
> 사용하는 경우새 목록새 Amazon SNS 주제를 생성하려면, 의도한 수신자가 알림을 받기 전에 이메일 주소를 확인해야 합니다. Amazon SNS 경보가 경보 상태에 진입할 때만 이메일을 전송합니다. 이러한 경보 상태 변경이 이메일 주소 확인 전에 발생할 경우, 의도된 수신자는 알림을 받지 못합니다.

11. 경보 생성(Create Alarm)을 선택합니다.

경보를 설정하려면(AWS CLI)

- AWS CLI를 열고 다음 명령을 입력합니다. 의 값을 변경합니다.`alarm-actions`파라미터는 이전에 만든 Amazon SNS 주제를 참조합니다.

```
aws cloudwatch put-metric-alarm \
    --alarm-name StartDocumentAnalysisUserErrors \
    --alarm-description "Alarm when more than 10 StartDocumentAnalysys user errors
 occur within 5 minutes" \
    --metric-name UserErrorCount \
    --namespace AWS/Textract \
    --statistic Sum \
    --period 300 \
    --threshold 10 \
    --comparison-operator GreaterThanThreshold \
    --evaluation-periods 1 \
    --unit Count \
    --dimensions Name=Operation,Value=StartDocumentAnalysis \
    --alarm-actions arn:aws:sns:us-east-1:111111111111:alarmtopic
```

  이 예제는 에 대한 호출에 대해 10분 이내에 사용자 오류가 발생하는 경우의 경보를 생성하는 방법을 보여줍니다.`StartDocumentAnalysis`. 자세한 내용은 put-metric-alarm을 참조하십시오.

경보를 설정하려면 (CloudWatch API)

- 을 호출합니다..`PutMetricAlarm` 자세한 내용은 단원을 참조하십시오.Amazon CloudWatch API 레퍼런스.

Amazon Textract의 CloudWatch 지표

이 단원에는 Amazon CloudWatch 측정치 및 에 대한 정보가 나와 있습니다.작업Amazon Textract Textract에 사용할 수 있는 차원입니다.

Amazon Textract 콘솔에서 Amazon Textract 측정치의 집계 확인도 볼 수 있습니다.

Amazon Textract의 CloudWatch 지표

다음 표에는 Amazon Textract Textract측정치가 요약되어 있습니다.

지표	설명
SuccessfulRequestCount	성공한 요청 수. 성공적 요청의 응답 코드 범위는 200 - 299입니다. 단위: 개수 유효한 통계: `Sum,Average`
ThrottledCount	조정된 요청 수. Amazon Textract Textract는 계정에 대해 설정된 초당 트랜잭션 한도 이상의 요청이 수신되면 요청을 제한합니다. 계정에 대해 설정된 한도가 자주 초과되면 한도 증가를 요청할 수 있습니다. 증가를 요청하려면 AWS 서비스 한도를 참조하십시오. 단위: 개수 유효한 통계: `Sum,Average`
ResponseTime	Amazon Textract 이 응답을 계산하는 시간 (밀리초). 단위: 1. `Data Samples` 통계 개수 2. `Average` 통계의 밀리초 유효한 통계: `Data Samples,Average`

지표	설명
	Note 이ResponseTime 메트릭은 Amazon Textract 메트릭 창에 포함되지 않습니다.
ServerErrorCount	서버 오류 수. 서버 오류의 응답 코드 범위는 500 - 599입니다. 단위: 개수 유효한 통계: Sum,Average
UserErrorCount	사용자 오류 수 (잘못된 파라미터, 잘못된 이미지, 권한 없음 등). 사용자 오류의 응답 코드 범위는 400~499입니다. 단위: 개수 유효한 통계: Sum,Average

Amazon Textract Textract용 CloudWatch 차원

작업별 측정치를 검색하려면 AWS/Textract 네임스페이스를 사용하고 operation 차원을 제공합니다. 차원에 대한 자세한 내용은 단원을 참조하십시오.차원의Amazon CloudWatch 사용 설명서.

을 사용하여 Amazon Textract API 호출 로깅AWS CloudTrail

Amazon Textract Textract는 다음과 통합되어 있습니다.AWS CloudTrail, 사용자, 역할 또는 에서 수행한 작업의 기록을 제공하는 서비스AWSAmazon Textract Textract에서 서비스를 제공합니다. CloudTrail은 Amazon Textract Textract에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Amazon Textract 콘솔로부터의 호출과 Amazon Textract API 작업에 대한 코드 호출이 포함됩니다.

추적을 생성하면 Amazon Textract에 대한 이벤트를 비롯하여 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록(Event history)에서 최신 이벤트를 볼 수 있습니다. CloudTrail이 수집한 정보를 사용하여 Amazon Textract Textract에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Amazon Textract 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. Amazon Textract Textract에서 활동이 발생하면 해당 활동이 다른 활동과 함께 CloudTrail 이벤트에 기록됩니다.AWS서비스 이벤트이벤트 기록. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

에서 이벤트를 지속적으로 기록하려면AWSAmazon Textract Textract의 이벤트를 포함한 계정은 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 기타를 구성할 수 있습니다.AWSCloudTrail 로그에 수집된 이벤트 데이터를 좀 더 분석하고 작업하는 서비스입니다. 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

모든 Amazon Textract 작업은 CloudTrail에서 로깅되며 에서 문서화됩니다API 참조. 예를 들어 `DetectDocumentText`, `AnalyzeDocument` 및 `GetDocumentText` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management(IAM) 사용자 자격 증명으로 했는지.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

로깅되지 않은 요청 매개 변수 및 응답 필드

개인 정보 보호를 위해 특정 요청 매개 변수 및 응답 필드는 기록되지 않습니다 (예: 요청 이미지 바이트 또는 응답 경계 상자 정보). 요청 파라미터에 제공된 Amazon S3 버킷 이름과 파일 이름은 CloudTrail 로그 항목에 제공됩니다. CloudTrail 로그에는 요청에서 전달된 이미지 바이트에 대한 정보가 제공되지 않습니다. 다음 표에는 각 Amazon Textract 작업에 대해 기록되지 않은 입력 파라미터와 응답 파라미터가 나와 있습니다.

작업	요청 파라미터	응답 필드
AnalyzeDocument	`Bytes`	모두
DetectDocumentText	`Bytes`	모두
StartDocumentAnalysis	None	None
GetDocumentAnalysis	None	모두
StartDocumentTextDetection	None	None
GetDocumentTextDetection	None	모두

Amazon Textract Textract로그 파일 항목 이해

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 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예제는 `AnalyzeDocument` 작업을 보여주는 CloudTrail 로그 항목입니다. 입력에 대한 이미지 바이트`document`및 분석 결과 (`responseElements`) 가 기록되지 않았습니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/janedoe",
        "accountId": "111111111111",
```

```
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "userName": "janedoe"
    },
    "eventTime": "2019-04-03T23:56:31Z",
    "eventSource": "textract.amazonaws.com",
    "eventName": "AnalyzeDocument",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "198.51.100.0",
    "userAgent": "",
    "requestParameters": {
        "document": {},
        "featureTypes": [
            "TABLES"
        ]
    },
    "responseElements": null,
    "requestID": "e387676b-d1f0-4ea7-85d6-f5a344052dce",
    "eventID": "c5db79ce-e4ea-4401-8517-784481d559f7",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

다음 예제는 의 CloudTrail 로그 항목을 표시합니다.StartDocumentAnalysis작업을 수행합니다. 로그 항목에는 에 Amazon S3 버킷 이름 및 이미지 파일 이름이 포함됩니다.documentLocation. 로그에는 작업 응답도 포함됩니다.

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111111111111:user/janedoe",
                "accountId": "11111111111",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "janedoe"
            },
            "eventTime": "2019-04-04T01:42:24Z",
            "eventSource": "textract.amazonaws.com",
            "eventName": "StartDocumentAnalysis",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "198.51.100.0",
```

```
            "userAgent": "",
            "requestParameters": {
                "documentLocation": {
                    "s3Object": {
                        "bucket": "bucket",
                        "name": "document.png"
                    }
                },
                "featureTypes": [
                    "TABLES"
                ]
            },
            "responseElements": {
                "jobId":
 "f3c718b444fa603d5d625ab967008f4b620d4650c9db8ca1cae01ef7efe51373"
            },
            "requestID": "9ae352e8-9de1-41ad-b77b-85aa348c2e82",
            "eventID": "f741bca0-c3cb-4805-82ea-baf76439deef",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111111111111"
        }

    ]
}
```

Amazon Textract Textract에 사용되는 규정 준수 확인

타사 감사자는 다중 감사자의 일환으로 Amazon Textract Textract의 보안 및 규정 준수를 평가합니다.AWS규정 준수 프로그램. 여기에는 HIPAA, SOC, ISO 및 PCI가 포함됩니다.

> **Note**
>
> PCI DSS 규정 준수가 적용되는 Textract 서비스를 통해 데이터를 처리하는 경우 AWS Support에 연락하고 제공된 절차에 따라 계정을 거부해야 합니다.

특정 규정 준수 프로그램의 범위 내에 있는 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하십시오. 일반적인 내용은 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하십시오.

Amazon Textract Textract를 사용할 때 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 결정됩니다.AWS는 규정 준수에 도움이 되도록 다음 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수 기술 백서 아키텍팅 - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 모음입니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 – AWS Config 서비스는 내부 사례, 산업 지침 및 규제에 대한 리소스 구성의 준수 상태를 평가합니다.
- AWS Security Hub— 이AWS서비스는 내에서 보안 상태에 대한 포괄적인 보기를 제공합니다.AWS. 보안 허브를 사용하면 보안 업계 표준 및 모범 사례를 준수하는지 확인할 수 있습니다.

Amazon Textract Textract의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

> **Note**
>
> GDPR (일반 데이터 보호 규정) 으로 인해 지역 간 데이터 전송은 허용되지 않습니다.

Amazon Textract Textract의 인프라 보안

관리형 서비스인 Amazon Textract Textract는AWS에 설명된 글로벌 네트워크 보안 절차Amazon Web Services: 보안 프로세스 개요백서.

사용하는 경우AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 Amazon Textract 에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과

같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

Amazon Textract Textract의 구성 및 취약성 분석

구성 및 IT 제어는 AWS와 고객 간의 공동 책임입니다. 자세한 내용은 AWS 공동 책임 모델을 참조하세요.

Amazon Textract Textract와 인터페이스 VPC 엔드포인트 (AWS PrivateLink)

를 생성하여 VPC와 Amazon Textract 간에 프라이빗 연결을 설정할 수 있습니다.인터페이스 VPC 엔드포인트. 인터페이스 엔드포인트는 다음으로 구동됩니다.AWS PrivateLink는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 Amazon Textract API에 비공개로 액세스할 수 있도록 지원하는 기술입니다. VPC 인스턴스는 Amazon Textract API와 통신하는 데 퍼블릭 IP 주소를 필요로 하지 않습니다. VPC와 Amazon Textract 간의 트래픽은 AWS 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다.

자세한 내용은 단원을 참조하십시오.인터페이스 VPC 엔드포인트(AWS PrivateLink)의Amazon VPC User Guide.

Amazon Textract TextractVPC 엔드포인트에 대한 고려 사항

Amazon Textract Textract에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 검토해야 합니다.인터페이스 엔드포인트 속성 및 제한의Amazon VPC User Guide.

Amazon Textract TExtract는 VPC 모든 API 작업에 대한 호출 수행을 지원합니다.

Amazon Textract Textract용 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔이나 을 사용하여 Amazon Textract 서비스에 대한 VPC 엔드포인트를 생성할 수 있습니다.AWS Command Line Interface(AWS CLI). 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 사용하여 Amazon Textract 용 VPC 엔드포인트를 생성합니다.

- com.아마조나우.##.textract - 대부분의 Amazon Textract 작업을 위한 엔드포인트를 생성합니다.
- com.아마조나우.##.textract-fips - 미국 연방 정보 처리 표준 (FIPS) Publication 140-2 US 정부 표준을 준수하는 Amazon Textract Textract에 대한 엔드포인트를 생성합니다.

엔드포인트에 프라이빗 DNS를 사용하도록 설정하는 경우 리전에 대한 기본 DNS 이름 (예:) 을 사용하여 Amazon Textract Textract에 API 요청을 수행할 수 있습니다.`textract.us-east-1.amazonaws.com`.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

Amazon Textract Textract에 대한 VPC 엔드포인트 정책 생성

Amazon Textract Textract에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 태스크를 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업입니다.
- 태스크를 수행할 있는 리소스.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

예: Amazon Textract Textract작업에 대한 VPC 엔드포인트 정책

다음은 Amazon Textract Textract에 대한 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 Amazon Textract 작업에 부여합니다.

이 예제 정책은 작업에만 액세스하도록 허용합니다.`DetectDocumentText`과`AnalyzeDocument`. 사용자는 여전히 VPC 엔드포인트 외부에서 Amazon Textract 작업을 호출할 수 있습니다.

```
"Statement":[
   {
      "Principal":"*",
      "Effect":"Allow",
```

```
         "Action":[
            "textract:DetectDocumentText",
            "textract:AnalyzeDocument",
         ],
         "Resource":"*"
      }
   ]
}
```

API 참조

이 단원에서는 Amazon Textract API 작업에 대한 설명서를 제공합니다.

주제

작업

다음 작업이 지원됩니다.

- AnalyzeDocument
- AnalyzeExpense
- AnalyzeID
- DetectDocumentText
- GetDocumentAnalysis
- GetDocumentTextDetection
- GetExpenseAnalysis
- StartDocumentAnalysis
- StartDocumentTextDetection
- StartExpenseAnalysis

AnalyzeDocument

검색된 항목 간의 관계에 대한 입력 문서를 분석합니다.

반환되는 정보의 유형은 다음과 같습니다.

- 양식 데이터 (키-값 페어) 관련 정보는 2로 반환됩니다.Block객체, 각 유형KEY_VALUE_SET: 키Block객체 및 값Block객체입니다. 예,이름: 아나 실바 캐롤라이나에는 키와 값이 포함되어 있습니다. 이름:열쇠입니다. 아나 실바 캐롤라이나이 값입니다.
- 테이블 및 테이블 셀 데이터입니다. 테이블Block객체에는 탐지된 테이블에 대한 정보가 들어 있습니다. 셀Block테이블의 각 셀에 대해 객체가 반환됩니다.
- 줄 및 텍스트 단어. 줄 바꿈Block객체가 하나 이상의 WORD가 포함되어 있습니다.Block객체입니다. 문서에서 감지된 모든 줄과 단어가 반환됩니다 (값과 관계가 없는 텍스트 포함)FeatureTypes).

확인란 및 옵션 버튼 (라디오 버튼) 과 같은 선택 요소는 양식 데이터 및 테이블에서 감지할 수 있습니다. 셀렉션_엘리먼트Block객체에는 선택 상태를 포함하여 선택 요소에 대한 정보가 들어 있습니다.

다음을 지정하여 수행할 분석 유형을 선택할 수 있습니다.FeatureTypes나열.

출력은 다음 목록에 반환됩니다.Block객체입니다.

AnalyzeDocument는 동기식 작업입니다. 문서를 비동기적으로 분석하려면StartDocumentAnalysis.

자세한 내용은 단원을 참조하십시오.문서 분석.

요청 구문

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "FeatureTypes": [ "string" ],
   "HumanLoopConfig": {
      "DataAttributes": {
         "ContentClassifiers": [ "string" ]
```

```
        },
        "FlowDefinitionArn": "string",
        "HumanLoopName": "string"
    }
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

Document

입력 문서인 base64로 인코딩된 바이트 또는 Amazon S3 객체입니다. AWS CLI를 사용하여 Amazon Textract 작업을 호출하는 경우 이미지 바이트를 전달할 수 없습니다. 문서는 JPEG, PNG, PDF 또는 TIFF 형식의 이미지여야 합니다.

AWS SDK를 사용하여 Amazon Textract Textract를 호출하는 경우 다음을 사용하여 전달된 이미지 바이트를 base64로 인코딩할 필요가 없습니다.`Bytes`필드.

유형: Document 객체

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

FeatureTypes

수행할 분석 유형의 목록입니다. 목록에 TABLES를 추가하여 입력 문서에서 검색된 테이블에 대한 정보를 반환합니다. FORMS를 추가하여 검색된 양식 데이터를 반환합니다. 두 가지 유형의 분석을 모두 수행하려면 TABLES 및 FORMS를`FeatureTypes`. 문서에서 감지된 모든 줄과 단어가 응답에 포함됩니다 (값과 관련이 없는 텍스트 포함)`FeatureTypes`).

Type: 문자열 배열

유효한 값: `TABLES | FORMS`

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

HumanLoopConfig

문서 분석을 위한 루프 워크플로우에서 사람에 대한 구성을 설정합니다.

유형: HumanLoopConfig 객체

: 필수 아니요

응답 구문

```
{
   "AnalyzeDocumentModelVersion": "string",
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DocumentMetadata": {
```

```
      "Pages": number
   },
   "HumanLoopActivationOutput": {
      "HumanLoopActivationConditionsEvaluationResults": "string",
      "HumanLoopActivationReasons": [ "string" ],
      "HumanLoopArn": "string"
   }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

AnalyzeDocumentModelVersion

문서를 분석하는 데 사용된 모델 버전입니다.

Type: String

Blocks

에 의해 감지되고 분석되는 항목AnalyzeDocument.

Type: 어레이Block사물

DocumentMetadata

분석된 문서에 대한 메타데이터입니다. 페이지 수를 예로 들 수 있습니다.

유형: DocumentMetadata 객체

HumanLoopActivationOutput

루프 평가에서 사람의 결과를 표시합니다.

유형: HumanLoopActivationOutput 객체

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

BadDocumentException

Amazon Textract Textract는 문서를 읽을 수 없습니다. Amazon Textract Textract의 문서 한도에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

HTTP 상태 코드: 400

DocumentTooLargeException

문서가 너무 크기 때문에 처리할 수 없습니다. 동기 작업의 최대 문서 크기입니다. 10MB 비동기 작업의 최대 문서 크기는 PDF 파일의 경우 500MB입니다.

HTTP 상태 코드: 400

HumanLoopQuotaExceededException

사용 가능한 루프 워크플로의 최대 개수를 초과했음을 나타냅니다.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는`InvalidParameterException`예외가 발생하지 않을 때`S3Object`또는`Bytes`값은 다음에 제공됩니다.`Document`요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract TExtract가 요청에서 지정된 S3 객체에 액세스할 수 없습니다. 자세한 내용은Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract가 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

UnsupportedDocumentException

입력 문서의 형식은 지원되지 않습니다. 작업 문서는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다.

HTTP 상태 코드: 400

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

AnalyzeExpense

AnalyzeExpense텍스트 간의 재정적으로 관련된 관계에 대한 입력 문서를 동기적으로 분석합니다.

정보는 다음과 같이 반환됩니다.ExpenseDocuments다음과 같이 분리됩니다.

- LineItemGroups- 다음을 포함하는 데이터 세트LineItems구매한 품목과 영수증에 가격 등 텍스트 줄에 대한 정보를 저장합니다.
- SummaryFields- 헤더 정보 또는 공급업체 이름과 같은 영수증에 다른 모든 정보가 들어 있습니다.

요청 구문

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

Document

입력 문서 (바이트 또는 S3 객체) 입니다.

를 사용하여 Amazon Textract API 작업에 이미지 바이트를 전달합니다.Bytes속성입니다. 예를 들어 를 사용합니다.Bytes로컬 파일 시스템에서 로드된 문서를 전달하는 속성입니다. 를 사용하여 전달된 이미지 바이트Bytes속성입니다 base64로 인코딩해야 합니다. AWS SDK를 사용하여 Amazon Textract API 작업을 호출하는 경우 코드가 문서 파일 바이트를 인코딩하지 않아도 될 수 있습니다.

다음을 사용하여 S3 버킷에 저장된 이미지를 Amazon Textract API 작업으로 전달합니다.S3Object속성입니다. S3 버킷에 저장된 문서는 base64로 인코딩할 필요가 없습니다.

S3 객체가 있는 S3 버킷의 AWS 리전과 Amazon Textract 작업에 사용하는 AWS 리전이 일치해야 합니다.

AWS CLI를 사용하여 Amazon Textract 작업을 호출하는 경우 바이트 속성을 사용하여 이미지 바이트를 전달하는 작업은 지원되지 않습니다. 먼저 문서를 Amazon S3 버킷에 업로드한 다음 S3Object 속성을 사용하여 작업을 호출해야 합니다.

Amazon Textract Textract가 S3 객체를 처리하려면 사용자에게 S3 객체에 액세스할 수 있는 권한이 있어야 합니다.

유형: Document 객체

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

응답 구문

```
{
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
               "LineItems": [
                  {
                     "LineItemExpenseFields": [
                        {
                           "LabelDetection": {
                              "Confidence": number,
                              "Geometry": {
                                 "BoundingBox": {
                                    "Height": number,
                                    "Left": number,
                                    "Top": number,
                                    "Width": number
                                 },
                                 "Polygon": [
                                    {
```

```
                                    "X": number,
                                    "Y": number
                                }
                            ]
                        },
                        "Text": "string"
                    },
                    "PageNumber": number,
                    "Type": {
                        "Confidence": number,
                        "Text": "string"
                    },
                    "ValueDetection": {
                        "Confidence": number,
                        "Geometry": {
                            "BoundingBox": {
                                "Height": number,
                                "Left": number,
                                "Top": number,
                                "Width": number
                            },
                            "Polygon": [
                                {
                                    "X": number,
                                    "Y": number
                                }
                            ]
                        },
                        "Text": "string"
                    }
                }
            ]
        }
    ]
}
],
"SummaryFields": [
    {
        "LabelDetection": {
            "Confidence": number,
            "Geometry": {
                "BoundingBox": {
                    "Height": number,
                    "Left": number,
```

```
                        "Top": number,
                        "Width": number
                     },
                     "Polygon": [
                        {
                           "X": number,
                           "Y": number
                        }
                     ]
                  },
                  "Text": "string"
               },
               "PageNumber": number,
               "Type": {
                  "Confidence": number,
                  "Text": "string"
               },
               "ValueDetection": {
                  "Confidence": number,
                  "Geometry": {
                     "BoundingBox": {
                        "Height": number,
                        "Left": number,
                        "Top": number,
                        "Width": number
                     },
                     "Polygon": [
                        {
                           "X": number,
                           "Y": number
                        }
                     ]
                  },
                  "Text": "string"
               }
            }
         ]
      }
   ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

DocumentMetadata

입력 문서에 대한 정보입니다.

유형: DocumentMetadata 객체

ExpenseDocuments

Amazon Textract Textract에서 감지한 비용입니다.

Type: 배열ExpenseDocument사물

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

BadDocumentException

Amazon Textract Textract는 문서를 읽을 수 없습니다. Amazon Textract Textract의 문서 한도에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

HTTP 상태 코드: 400

DocumentTooLargeException

문서가 너무 크기 때문에 처리할 수 없습니다. 동기 작업의 최대 문서 크기입니다. 10MB 비동기 작업의 최대 문서 크기는 PDF 파일의 경우 500MB입니다.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는InvalidParameterException예외가 발생하지 않을 때S3Object또는Bytes값은 다음 위치에 제공됩니다.Document요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract Textract가 요청에서 지정된 S3 객체에 액세스할 수 없습니다. 자세한 내용은Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract가 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

UnsupportedDocumentException

출력 문서의 형식은 지원되지 않습니다. 작업 문서는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다.

HTTP 상태 코드: 400

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

AnalyzeID

관련 정보에 대한 ID 문서를 분석합니다. 이 정보는 추출되어 다음과 같이 반환됩니다.`IdentityDocumentFields`추출된 텍스트의 정규화된 필드와 값을 모두 기록합니다. 다른 Amazon Textract 작업과 달리`AnalyzeID`는 지오메트리 데이터를 반환하지 않습니다.

요청 구문

```
{
   "DocumentPages": [
      {
         "Bytes": blob,
         "S3Object": {
            "Bucket": "string",
            "Name": "string",
            "Version": "string"
         }
      }
   ]
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

DocumentPages

애널리제이드에 전달되는 문서입니다.

Type: 배열Document사물

어레이 멤버: 최소 항목 수는 1개입니다. 최대 항목 수는 2개입니다.

필수 항목: 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

응답 구문

```
{
   "AnalyzeIDModelVersion": "string",
   "DocumentMetadata": {
```

```
      "Pages": number
   },
   "IdentityDocuments": [
      {
         "DocumentIndex": number,
         "IdentityDocumentFields": [
            {
               "Type": {
                  "Confidence": number,
                  "NormalizedValue": {
                     "Value": "string",
                     "ValueType": "string"
                  },
                  "Text": "string"
               },
               "ValueDetection": {
                  "Confidence": number,
                  "NormalizedValue": {
                     "Value": "string",
                     "ValueType": "string"
                  },
                  "Text": "string"
               }
            }
         ]
      }
   ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

AnalyzeIDModelVersion

문서를 처리하는 데 사용되는 애널리틱스 아이덴티티 API의 버전입니다.

Type: String

DocumentMetadata

입력 문서에 대한 정보입니다.

유형: DocumentMetadata 객체

IdentityDocuments

AnalyzeID에서 처리한 문서 목록입니다. 목록에 있는 위치를 나타내는 번호와 문서에 대한 응답 구조를 포함합니다.

Type: 배열IdentityDocument사물

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

BadDocumentException

Amazon Textract Textract는 문서를 읽을 수 없습니다. Amazon Textract Textract의 문서 한도에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

HTTP 상태 코드: 400

DocumentTooLargeException

문서가 너무 크기 때문에 처리할 수 없습니다. 동기 작업의 최대 문서 크기입니다. 10MB 비동기 작업의 최대 문서 크기는 PDF 파일의 경우 500MB입니다.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는`InvalidParameterException`예외가 발생하지 않을 때`S3Object`또는`Bytes`값은 다음 위치에 제공됩니다.`Document`요청 파라미터를 지정합니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract Textract가 요청에서 지정된 S3 객체에 액세스할 수 없습니다. 자세한 내용은Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract가 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

UnsupportedDocumentException

입력 문서의 형식은 지원되지 않습니다. 작업 문서는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다.

HTTP 상태 코드: 400

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK

- AWS SDK for Ruby V3

DetectDocumentText

입력 문서에서 텍스트를 감지합니다. Amazon Textract Textract는 텍스트 줄과 텍스트 한 줄을 구성하는 단어를 감지할 수 있습니다. 입력 문서는 JPEG, PNG, PDF 또는 TIFF 형식의 이미지여야 합니다.`DetectDocumentText`검색된 텍스트를 다음 배열로 반환합니다.Block객체.

각 문서 페이지에는 연결된 페이지가 있습니다.`Block`페이지 유형입니다. 각 페이지`Block`객체가 LINE의 상위`Block`페이지에서 감지된 텍스트 행을 나타내는 개체입니다. 줄 바꿈`Block`object는 선을 구성하는 각 단어의 부모입니다. 단어는 다음과 같이 표시됩니다.`Block`WORD 유형의 객체입니다.

`DetectDocumentText`은 동기식 작업입니다. 문서를 비동기적으로 분석하려면StartDocumentTextDetection.

자세한 내용은 단원을 참조하십시오.문서 텍스트 감지.

요청 구문

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

Document

입력 문서인 base64로 인코딩된 바이트 또는 Amazon S3 객체입니다. AWS CLI를 사용하여 Amazon Textract 작업을 호출하는 경우 이미지 바이트를 전달할 수 없습니다. 이 문서는 JPEG 또는 PNG 형식의 이미지여야 합니다.

AWS SDK를 사용하여 Amazon Textract Textract를 호출하는 경우 다음을 사용하여 전달된 이미지 바이트를 base64로 인코딩할 필요가 없습니다.`Bytes`필드.

유형: Document 객체

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

응답 구문

```
{
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
```

```
    ],
    "DetectDocumentTextModelVersion": "string",
    "DocumentMetadata": {
       "Pages": number
    }
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

Blocks

의 어레이Block문서에서 검색된 텍스트를 포함하는 개체입니다.

Type: 배열Block사물

DetectDocumentTextModelVersion

Type: String

DocumentMetadata

문서에 대한 메타데이터입니다. 이 문서에는 문서에서 검색된 페이지 수가 포함됩니다.

유형: DocumentMetadata 객체

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

BadDocumentException

Amazon Textract Textract는 문서를 읽을 수 없습니다. Amazon Textract Textract의 문서 한도에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

HTTP 상태 코드: 400

DocumentTooLargeException

문서가 너무 크기 때문에 처리할 수 없습니다. 동기 작업의 최대 문서 크기입니다. 10MB입니다. 비동기 작업의 최대 문서 크기는 PDF 파일의 경우 500MB입니다.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는`InvalidParameterException`예외가 발생하지 않을 때`S3Object`또는`Bytes`값은 다음 항목에 제공됩니다.`Document`요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract 이 요청에서 지정된 S3 객체에 액세스할 수 없습니다. 자세한 내용은Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract이 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

UnsupportedDocumentException

출력 문서의 형식은 지원되지 않습니다. 작업 문서는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다.

HTTP 상태 코드: 400

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

GetDocumentAnalysis

문서의 텍스트를 분석하는 Amazon Textract 비동기 작업의 결과를 가져옵니다.

다음을 호출하여 비동기 텍스트 분석을 시작합니다.StartDocumentAnalysis를 반환하는 작업 식별자 (JobId). 텍스트 분석 작업이 완료되면 Amazon Textract Textract는 초기 호출에 등록된 Amazon Simple Notification Service (Amazon SNS) 주제에 완료 상태를 게시합니다.StartDocumentAnalysis. 텍스트 감지 작업의 결과를 얻으려면 먼저 Amazon SNS 주제에 게시된 상태 값이 다음과 같은지 확인하십시오.SUCCEEDED. 그렇다면 전화하십시오.GetDocumentAnalysis작업 식별자를 전달하고 (JobId) 초기 통화부터StartDocumentAnalysis.

GetDocumentAnalysis의 배열을 반환합니다.Block객체. 다음 정보 유형이 반환됩니다.

- 양식 데이터 (키-값 쌍). 관련 정보는 2개로 반환됩니다.Block객체, 각 유형KEY_VALUE_SET: 키Block객체 및 값Block객체. 예,이름: 아나 실바 캐롤라이나에는 키와 값이 포함되어 있습니다. 이름:열쇠입니다. 아나 실바 캐롤라이나이 값입니다.
- 테이블 및 테이블 셀 데이터입니다. 테이블Block객체에는 탐지된 테이블에 대한 정보가 들어 있습니다. 셀Block테이블의 각 셀에 대해 객체가 반환됩니다.
- 줄 및 텍스트 단어. 줄 바꿈Block객체가 하나 이상의 WORD가 포함되어 있습니다.Block객체. 문서에서 감지된 모든 줄과 단어가 반환됩니다 (텍스트 포함).StartDocumentAnalysis FeatureTypes입력 매개 변수).

확인란 및 옵션 버튼 (라디오 버튼) 과 같은 선택 요소는 양식 데이터 및 테이블에서 감지할 수 있습니다. 셀렉션_엘리먼트Block객체에는 선택 상태를 포함하여 선택 요소에 대한 정보가 들어 있습니다.

사용MaxResults파라미터를 사용하여 반환되는 블록 수를 제한합니다. 에 지정된 것보다 많은 결과가 있는 경우MaxResults, 의 가치NextToken작업 응답에는 다음 결과 집합을 가져오기 위한 페이지 지정 토큰이 포함되어 있습니다. 결과의 다음 페이지를 가져오려면GetDocumentAnalysis를 채우고NextToken이전 호출에서 반환된 토큰 값이 있는 요청 매개 변수GetDocumentAnalysis.

자세한 내용은 단원을 참조하십시오.문서 텍스트 분석.

요청 구문

```
{
   "JobId": "string",
   "MaxResults": number,
```

```
    "NextToken": "string"
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

JobId

텍스트 감지 작업의 고유 식별자입니다. 이JobId에서 반환됩니다.StartDocumentAnalysis. AJobId값은 7일 동안만 유효합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: ^[a-zA-Z0-9-_]+$

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

MaxResults

페이지가 지정된 호출당 반환할 최대 결과 수입니다. 지정할 수 있는 가장 큰 값은 1,000입니다. 1,000보다 큰 값을 지정한 경우 최대 1,000개의 결과가 반환됩니다. 기본값은 1000입니다.

Type: 정수

유효 범위: 최소값 1.

: 필수 아니요

NextToken

이전 응답이 불완전한 경우 (검색할 블록이 더 많기 때문에) Amazon Textract Textract는 응답에 페이지 매김 토큰을 반환합니다. 이 페이지 매김 토큰을 사용하여 다음 블록 세트를 검색할 수 있습니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 255.

Pattern: .*\S.*

: 필수 아니요

응답 구문

```
{
   "AnalyzeDocumentModelVersion": "string",
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DocumentMetadata": {
```

```
      "Pages": number
   },
   "JobStatus": "string",
   "NextToken": "string",
   "StatusMessage": "string",
   "Warnings": [
      {
         "ErrorCode": "string",
         "Pages": [ number ]
      }
   ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

AnalyzeDocumentModelVersion

Type: String

Blocks

텍스트 분석 작업의 결과입니다.

Type: 배열Block사물

DocumentMetadata

Amazon Textract Textract가 처리한 문서에 대한 정보입니다.DocumentMetadata는 Amazon Textract 비디오 작업에서 페이지 매겨진 응답의 모든 페이지에서 반환됩니다.

유형: DocumentMetadata 객체

JobStatus

텍스트 검색 작업의 현재 상태입니다.

Type: String

유효한 값: IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

응답이 잘린 경우 Amazon Textract Textract는 이 토큰을 반환합니다. 이 토큰을 후속 요청에서 사용하여 다음 텍스트 검색 결과 집합을 가져올 수 있습니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 255.

Pattern: `.*\S.*`

StatusMessage

검색 작업을 완료할 수 없는 경우 반환합니다. 발생한 오류에 대한 설명이 들어 있습니다.

Type: String

Warnings

문서 분석 작업 중에 발생한 경고 목록입니다.

Type: 배열Warning사물

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidJobIdException

잘못된 작업 식별자가 에 전달되었습니다.GetDocumentAnalysis또는GetDocumentAnalysis.

HTTP 상태 코드: 400

InvalidKMSKeyException

KMS 키를 입력한 상태에서 암호 해독 권한이 없거나 KMS 키가 잘못 입력되었음을 나타냅니다.

HTTP 상태 코드: 400

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서는InvalidParameterException예외가 발생하지 않을 때S3Object또는Bytes값은 다음 위치에 제공됩니다.Document요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract 이 요청에서 지정된 S3 객체에 액세스할 수 없습니다. 자세한 내용은Amazon S3 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract 이 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3

- AWSPython용 SDK
- AWS SDK for Ruby V3

GetDocumentTextDetection

문서에서 텍스트를 감지하는 Amazon Textract 비동기 작업에 대한 결과를 가져옵니다. Amazon Textract Textract는 텍스트 줄과 텍스트 한 줄을 구성하는 단어를 감지할 수 있습니다.

를 호출하여 비동기 텍스트 감지를 시작합니다.StartDocumentTextDetection를 반환하는 작업 식별자 (JobId). 텍스트 검색 작업이 완료되면 Amazon Textract Textract는 초기 호출에 등록된 Amazon Simple Notification Service (Amazon SNS) 주제에 완료 상태를 게시합니다.StartDocumentTextDetection. 텍스트 감지 작업의 결과를 얻으려면 먼저 Amazon SNS 주제에 게시된 상태 값이 다음과 같은지 확인하십시오.SUCCEEDED. 그렇다면 전화하십시오.GetDocumentTextDetection작업 식별자를 전달하고 (JobId) 초기 통화부터StartDocumentTextDetection.

GetDocumentTextDetection의 배열을 반환합니다.Block객체.

각 문서 페이지는 연결된 상태로Block페이지 유형입니다. 각 페이지Block객체가 LINE에 있는 상위Block페이지에서 감지된 텍스트 행을 나타내는 개체입니다. 줄 바꿈Blockobject는 선을 구성하는 각 단어의 부모입니다. 단어는 다음과 같이 표시됩니다.BlockWORD 유형의 객체입니다.

MaxResulent 파라미터를 사용하여 반환되는 블록 수를 제한합니다. 에 지정된 것보다 많은 결과가 있는 경우MaxResults, 의 가치NextToken작업 응답에 다음 결과 집합을 가져오기 위한 페이지 매김 토큰이 포함되어 있습니다. 결과의 다음 페이지를 가져오려면GetDocumentTextDetection를 채우고NextToken이전 호출에서 반환된 토큰 값이 있는 요청 매개 변수GetDocumentTextDetection.

자세한 내용은 단원을 참조하십시오.문서 텍스트 감지.

요청 구문

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

JobId

텍스트 감지 작업의 고유 식별자입니다. 이JobId에서 반환됩니다.StartDocumentTextDetection. AJobId값은 7일 동안만 유효합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: ^[a-zA-Z0-9-_]+$

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

MaxResults

페이지가 지정된 호출당 반환할 결과의 최대 수입니다. 지정할 수 있는 가장 큰 값은 1,000입니다. 1,000보다 큰 값을 지정한 경우 최대 1,000개의 결과가 반환됩니다. 기본값은 1000입니다.

Type: 정수

유효 범위: 최소값 1.

: 필수 아니요

NextToken

이전 응답이 불완전한 경우 (검색할 블록이 더 많기 때문에) Amazon Textract Textract는 응답에 페이지 매김 토큰을 반환합니다. 이 페이지 매김 토큰을 사용하여 다음 블록 세트를 검색할 수 있습니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 255.

Pattern: .*\S.*

: 필수 아니요

응답 구문

```
{
   "Blocks": [
      {
         "BlockType": "string",
```

```
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DetectDocumentTextModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
   "JobStatus": "string",
   "NextToken": "string",
   "StatusMessage": "string",
   "Warnings": [
      {
         "ErrorCode": "string",
         "Pages": [ number ]
```

```
      }
   ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

Blocks

텍스트 감지 작업의 결과입니다.

Type: 배열Block사물

DetectDocumentTextModelVersion

Type: String

DocumentMetadata

Amazon Textract Textract가 처리한 문서에 대한 정보입니다.DocumentMetadata는 Amazon Textract 비디오 작업에서 페이지 매겨진 응답의 모든 페이지에서 반환됩니다.

유형: DocumentMetadata 객체

JobStatus

텍스트 검색 작업의 현재 상태입니다.

Type: String

유효한 값: IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

응답이 잘린 경우 Amazon Textract Textract는 이 토큰을 반환합니다. 후속 요청에서 이 토큰을 사용하여 다음 텍스트 감지 결과 집합을 가져올 수 있습니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 255.

Pattern: .*\S.*

StatusMessage

검색 작업을 완료할 수 없는 경우 반환합니다. 발생한 오류에 대한 설명이 들어 있습니다.

Type: String

Warnings

문서에 대한 텍스트 감지 작업 중에 발생한 경고 목록입니다.

Type: 배열Warning사물

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidJobIdException

잘못된 작업 식별자가 에 전달되었습니다.GetDocumentAnalysis또는GetDocumentAnalysis.

HTTP 상태 코드: 400

InvalidKMSKeyException

KMS 키를 입력한 상태에서 암호 해독 권한이 없거나 KMS 키가 잘못 입력되었음을 나타냅니다.

HTTP 상태 코드: 400

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는`InvalidParameterException`예외가 발생하지 않을 때`S3Object`또는`Bytes`값은 다음 위치에 제공됩니다.`Document`요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract TExtract이 요청에서 지정된 S3 객체에 액세스할 수 없습니다. 자세한 내용은Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract이 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

GetExpenseAnalysis

송장 및 영수증을 분석하는 Amazon Textract 비동기 작업에 대한 결과를 가져옵니다. Amazon Textract Textract는 입력 송장 및 영수증에서 연락처 정보, 구매한 품목 및 공급업체 이름을 찾습니다.

다음을 호출하여 비동기 인보이스/영수증 분석을 시작합니다.StartExpenseAnalysis를 반환하는 작업 식별자 (JobId). Amazon Textract는 송장/영수증 분석이 완료되면 Amazon Simple Notification Service (Amazon SNS) 주제에 완료 상태를 게시합니다. 이 주제는 에 대한 초기 호출에 등록해야 합니다.StartExpenseAnalysis. 송장/영수증 분석 작업의 결과를 가져오려면 먼저 Amazon SNS 주제에 게시된 상태 값이 인지 확인합니다.SUCCEEDED. 그렇다면 전화하십시오.GetExpenseAnalysis작업 식별자를 전달하고 (JobId) 초기 통화부터StartExpenseAnalysis.

MaxResults 파라미터를 사용하여 반환되는 블록 수를 제한합니다. 에 지정된 것보다 많은 결과가 있는 경우MaxResults, 의 가치NextToken작업 응답에는 다음 결과 집합을 가져오기 위한 페이지 매김 토큰이 포함되어 있습니다. 결과의 다음 페이지를 가져오려면GetExpenseAnalysis을 채우고NextToken이전 호출에서 반환된 토큰 값이 있는 요청 매개 변수GetExpenseAnalysis.

자세한 내용은 단원을 참조하십시오.송장 및 수금 분석.

요청 구문

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

JobId

텍스트 검색 작업의 고유 식별자입니다. 이JobId에서 반환됩니다.StartExpenseAnalysis. AJobId값은 7일 동안만 유효합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: ^[a-zA-Z0-9-_]+$

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

MaxResults

페이지가 지정된 호출당 반환할 결과의 최대 수입니다. 지정할 수 있는 가장 큰 값은 20입니다. 20보다 큰 값을 지정한 경우 최대 20개의 결과가 반환됩니다. 기본값은 20입니다.

Type: 정수

유효 범위: 최소값은 1이고,

: 필수 아니요

NextToken

이전 응답이 불완전한 경우 (검색할 블록이 더 많기 때문에) Amazon Textract Textract는 응답에 페이지 매김 토큰을 반환합니다. 이 페이지 매김 토큰을 사용하여 다음 블록 세트를 검색할 수 있습니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 255.

Pattern: `.*\S.*`

: 필수 아니요

응답 구문

```
{
   "AnalyzeExpenseModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
               "LineItems": [
                  {
                     "LineItemExpenseFields": [
```

```
                    {
                       "LabelDetection": {
                          "Confidence": number,
                          "Geometry": {
                             "BoundingBox": {
                                "Height": number,
                                "Left": number,
                                "Top": number,
                                "Width": number
                             },
                             "Polygon": [
                                {
                                   "X": number,
                                   "Y": number
                                }
                             ]
                          },
                          "Text": "string"
                       },
                       "PageNumber": number,
                       "Type": {
                          "Confidence": number,
                          "Text": "string"
                       },
                       "ValueDetection": {
                          "Confidence": number,
                          "Geometry": {
                             "BoundingBox": {
                                "Height": number,
                                "Left": number,
                                "Top": number,
                                "Width": number
                             },
                             "Polygon": [
                                {
                                   "X": number,
                                   "Y": number
                                }
                             ]
                          },
                          "Text": "string"
                       }
                    }
                 ]
```

```
                }
            ]
        }
    ],
    "SummaryFields": [
        {
            "LabelDetection": {
                "Confidence": number,
                "Geometry": {
                    "BoundingBox": {
                        "Height": number,
                        "Left": number,
                        "Top": number,
                        "Width": number
                    },
                    "Polygon": [
                        {
                            "X": number,
                            "Y": number
                        }
                    ]
                },
                "Text": "string"
            },
            "PageNumber": number,
            "Type": {
                "Confidence": number,
                "Text": "string"
            },
            "ValueDetection": {
                "Confidence": number,
                "Geometry": {
                    "BoundingBox": {
                        "Height": number,
                        "Left": number,
                        "Top": number,
                        "Width": number
                    },
                    "Polygon": [
                        {
                            "X": number,
                            "Y": number
                        }
                    ]
```

```
                    },
                    "Text": "string"
                 }
              }
           ]
        }
    ],
    "JobStatus": "string",
    "NextToken": "string",
    "StatusMessage": "string",
    "Warnings": [
       {
          "ErrorCode": "string",
          "Pages": [ number ]
       }
    ]
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

AnalyzeExpenseModelVersion

분석의 현재 모델 버전입니다.

Type: String

DocumentMetadata

Amazon Textract Textract가 처리한 문서에 대한 정보입니다.DocumentMetadata는 Amazon Textract 작업에서 페이지 매겨진 응답의 모든 페이지에서 반환됩니다.

유형: DocumentMetadata 객체

ExpenseDocuments

Amazon Textract Textract에서 감지한 비용입니다.

Type: 배열ExpenseDocument사물

JobStatus

텍스트 검색 작업의 현재 상태입니다.

Type: String

유효한 값: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

응답이 잘린 경우 Amazon Textract Textract는 이 토큰을 반환합니다. 이 토큰을 후속 요청에서 사용하여 다음 텍스트 검색 결과 집합을 가져올 수 있습니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 255.

Pattern: `.*\S.*`

StatusMessage

검색 작업을 완료할 수 없는 경우 반환합니다. 발생한 오류에 대한 설명이 들어 있습니다.

Type: String

Warnings

문서에 대한 텍스트 감지 작업 중에 발생한 경고 목록입니다.

Type: 배열Warning사물

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidJobIdException

잘못된 작업 식별자가 에 전달되었습니다.GetDocumentAnalysis또는GetDocumentAnalysis.

HTTP 상태 코드: 400

InvalidKMSKeyException

KMS 키를 입력한 상태에서 암호 해독 권한이 없거나 KMS 키가 잘못 입력되었음을 나타냅니다.

HTTP 상태 코드: 400

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는`InvalidParameterException`예외가 발생하지 않을 때`S3Object`또는`Bytes`값은 다음 위치에 제공됩니다.`Document`요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract 이 요청에서 지정된 S3 객체에 액세스할 수 없습니다. 자세한 내용은Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract가 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

StartDocumentAnalysis

키-값 페어, 테이블 및 선택 요소와 같은 감지된 항목 간의 관계에 대한 입력 문서의 비동기식 분석을 시작합니다.

StartDocumentAnalysis에서는 JPEG, PNG, TIFF 및 PDF 형식의 문서에서 텍스트를 분석할 수 있습니다. Amazon S3 버킷에 저장됩니다. 사용DocumentLocation을 눌러 문서의 버킷 이름과 파일 이름을 지정합니다.

StartDocumentAnalysis작업 식별자를 반환합니다 (JobId) 을 사용하여 작업 결과를 얻습니다. 텍스트 분석이 완료되면 Amazon Textract Textract가 에서 지정한 Amazon Simple Notification Service (Amazon SNS) 주제에 완료 상태를 게시합니다.NotificationChannel. 텍스트 분석 작업의 결과를 얻으려면 먼저 Amazon SNS 주제에 게시된 상태 값이 다음과 같은지 확인하십시오.SUCCEEDED. 그렇다면 전화하십시오.GetDocumentAnalysis작업 식별자를 전달하고 (JobId) 초기 통화부터StartDocumentAnalysis.

자세한 내용은 단원을 참조하십시오.문서 텍스트 분석.

요청 구문

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "FeatureTypes": [ "string" ],
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

ClientRequestToken

시작 요청을 식별하는 데 사용하는 멱등성 토큰입니다. 동일한 토큰을 여러 개 사용하여 사용하는 경우StartDocumentAnalysis요청, 동일JobId이 반환됩니다. 사용ClientRequestToken같은 작업이 실수로 두 번 이상 시작되지 않도록 합니다. 자세한 내용은 단원을 참조하십시오.Amazon Textract 비동기 작업 호출.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: ^[a-zA-Z0-9-_]+$

: 필수 아니요

DocumentLocation

처리할 문서의 위치입니다.

유형: DocumentLocation 객체

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

FeatureTypes

수행할 분석 유형의 목록입니다. 목록에 TABLES를 추가하여 입력 문서에서 검색된 테이블에 대한 정보를 반환합니다. FORMS를 추가하여 검색된 양식 데이터를 반환합니다. 두 가지 유형의 분석을 모두 수행하려면 TABLES 및 FORMS를FeatureTypes. 문서에서 감지된 모든 줄과 단어가 응답에 포함됩니다 (값과 관련이 없는 텍스트 포함)FeatureTypes).

Type: 문자열 배열

유효한 값: TABLES | FORMS

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

JobTag

지정한 식별자로, Amazon SNS 주제에 게시된 완료 알림에 포함됩니다. 예를 들어 를 사용할 수 있습니다.JobTag완료 통지가 해당하는 문서 유형 (예: 세금 양식 또는 영수증) 을 식별합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: `[a-zA-Z0-9_.\-:]+`

: 필수 아니요

KMSKeyId

추론 결과를 암호화하는 데 사용되는 KMS 키입니다. 키 ID 또는 키 별칭 형식일 수 있습니다. KMS 키가 제공되면 고객 버킷에 있는 객체의 서버 측 암호화에 KMS 키가 사용됩니다. 이 매개 변수를 활성화하지 않으면 결과는 SSE-S3 를 사용하여 서버 측에서 암호화됩니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 2048.

Pattern: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

: 필수 아니요

NotificationChannel

Amazon Textract가 작업의 완료 상태를 게시하도록 하려는 Amazon SNS 주제 ARN입니다.

유형: NotificationChannel 객체

: 필수 아니요

OutputConfig

출력이 고객 정의 버킷으로 이동할지 여부를 설정합니다. 기본적으로 Amazon Textract Textract는 GetDocumentAnalysis 작업에서 액세스할 수 있도록 결과를 내부적으로 저장합니다.

유형: OutputConfig 객체

: 필수 아니요

응답 구문

```
{
   "JobId": "string"
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

JobId

문서 텍스트 감지 작업의 식별자입니다. 사용JobId후속 호출에서 작업을 식별합니다.GetDocumentAnalysis. AJobId값은 7일 동안만 유효합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: ^[a-zA-Z0-9-_]+$

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

BadDocumentException

Amazon Textract Textract는 문서를 읽을 수 없습니다. Amazon Textract Textract의 문서 한도에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

HTTP 상태 코드: 400

DocumentTooLargeException

문서가 너무 크기 때문에 처리할 수 없습니다. 동기 작업의 최대 문서 크기입니다. 10MB 비동기 작업의 최대 문서 크기는 PDF 파일의 경우 500MB입니다.

HTTP 상태 코드: 400

IdempotentParameterMismatchException

AClientRequestTokeninput 파라미터가 작업과 함께 재사용되었지만 하나 이상의 다른 입력 파라미터가 이전의 작업 호출과 다릅니다.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidKMSKeyException

KMS 키를 입력한 상태에서 암호 해독 권한이 없거나 KMS 키가 잘못 입력되었음을 나타냅니다.

HTTP 상태 코드: 400

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는`InvalidParameterException`예외가 발생하지 않을 때`S3Object`또는`Bytes`값은 다음 위치에 제공됩니다.`Document`요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract TExtract가 요청에서 지정된 S3 객체에 액세스할 수 없습니다.Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

LimitExceededException

Amazon Textract 서비스 제한을 초과했습니다. 예를 들어 너무 많은 비동기 작업을 동시에 시작하는 경우 시작 작업을 호출합니다 (`StartDocumentTextDetection`예를 들어) 을 발생시키면 동시 실행 작업의 수가 Amazon Textract 서비스 제한 미만이 될 때까지 (HTTP 상태 코드: 400) 이 발생합니다.

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract가 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

UnsupportedDocumentException

출력 문서의 형식은 지원되지 않습니다. 작업 문서는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다.

HTTP 상태 코드: 400

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

StartDocumentTextDetection

문서에서 텍스트의 비동기 감지를 시작합니다. Amazon Textract Textract는 텍스트 줄과 텍스트 한 줄을 구성하는 단어를 감지할 수 있습니다.

`StartDocumentTextDetection`에서는 JPEG, PNG, TIFF 및 PDF 형식의 문서에서 텍스트를 분석할 수 있습니다. 문서는 Amazon S3 버킷에 저장됩니다. 사용DocumentLocation을 눌러 문서의 버킷 이름과 파일 이름을 지정합니다.

`StartTextDetection`작업 식별자를 반환합니다 (`JobId`) 을 사용하여 작업 결과를 얻습니다. 텍스트 감지가 완료되면 Amazon Textract Textract가 에서 지정한 Amazon Simple Notification Service (Amazon SNS) 주제에 완료 상태를 게시합니다.`NotificationChannel`. 텍스트 감지 작업의 결과를 얻으려면 먼저 Amazon SNS 주제에 게시된 상태 값이 다음과 같은지 확인하십시오.`SUCCEEDED`. 그렇다면 전화하십시오.GetDocumentTextDetection작업 식별자를 전달하고 (`JobId`) 초기 호출부터`StartDocumentTextDetection`.

자세한 내용은 단원을 참조하십시오.문서 텍스트 감지.

요청 구문

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

ClientRequestToken

시작 요청을 식별하는 데 사용되는 멱등 토큰입니다. 동일한 토큰을 여러 개 사용하여 사용하는 경우`StartDocumentTextDetection`요청, 동일`JobId`이 반환됩니다. 사용`ClientRequestToken`같은 작업이 실수로 두 번 이상 시작되지 않도록 합니다. 자세한 내용은 단원을 참조하십시오.Amazon Textract 비동기 작업 호출.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: `^[a-zA-Z0-9-_]+$`

: 필수 아니요

DocumentLocation

처리할 문서의 위치입니다.

유형: DocumentLocation 객체

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

JobTag

지정한 식별자로, Amazon SNS 주제에 게시된 완료 알림에 포함됩니다. 예를 들어 를 사용할 수 있습니다.`JobTag`완료 통지가 해당하는 문서 유형 (예: 세금 양식 또는 영수증) 을 식별합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: `[a-zA-Z0-9_.\-:]+`

: 필수 아니요

KMSKeyId

추론 결과를 암호화하는 데 사용되는 KMS 키입니다. 키 ID 또는 키 별칭 형식일 수 있습니다. KMS 키가 제공되면 고객 버킷에 있는 객체의 서버 측 암호화에 KMS 키가 사용됩니다. 이 매개 변수를 활성화하지 않으면 결과는 SSE-S3 를 사용하여 서버 측에서 암호화됩니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 2048.

Pattern: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

: 필수 아니요

NotificationChannel

Amazon Textract가 작업의 완료 상태를 게시하도록 하려는 Amazon SNS 주제 ARN입니다.

유형: NotificationChannel 객체

: 필수 아니요

OutputConfig

출력이 고객 정의 버킷으로 이동할지 여부를 설정합니다. 기본적으로 Amazon Textract Textract는 GetDocumentTextDetection 작업을 통해 액세스할 수 있도록 결과를 내부적으로 저장합니다.

유형: OutputConfig 객체

: 필수 아니요

응답 구문

```
{
   "JobId": "string"
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

JobId

문서에 대한 텍스트 검색 작업의 식별자입니다. 사용`JobId`후속 호출에서 작업을 식별합니다.`GetDocumentTextDetection`. A`JobId`값은 7일 동안만 유효합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: `^[a-zA-Z0-9-_]+$`

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

BadDocumentException

Amazon Textract Textract는 문서를 읽을 수 없습니다. Amazon Textract Textract의 문서 한도에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

HTTP 상태 코드: 400

DocumentTooLargeException

문서가 너무 크기 때문에 처리할 수 없습니다. 동기 작업의 최대 문서 크기 10MB. 비동기 작업의 최대 문서 크기는 PDF 파일의 경우 500MB입니다.

HTTP 상태 코드: 400

IdempotentParameterMismatchException

`AClientRequestToken`입력 파라미터가 작업에서 재사용되었지만 하나 이상의 다른 입력 파라미터가 이전의 작업 호출과 다릅니다.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidKMSKeyException

KMS 키를 입력한 상태에서 암호 해독 권한이 없거나 KMS 키가 잘못 입력되었음을 나타냅니다.

HTTP 상태 코드: 400

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는InvalidParameterException예외가 발생하지 않을 때S3Object또는Bytes값은 다음 위치에 제공됩니다.Document요청 파라미터입니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract Textract가 요청에서 지정된 S3 객체에 액세스할 수 없습니다.Amazon S3 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

LimitExceededException

Amazon Textract Service 제한을 초과했습니다. 예를 들어 너무 많은 비동기 작업을 동시에 시작하면 시작 작업을 호출합니다 (StartDocumentTextDetection예를 들어) 을 발생시킬 경우에는 동시 실행 작업의 수가 Amazon Textract Service 제한 미만이 될 때까지 제한 Excepeption 예외 (HTTP 상태 코드: 400) 가 발생합니다.

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract가 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

UnsupportedDocumentException

입력 문서의 형식은 지원되지 않습니다. 작업 문서는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다.

HTTP 상태 코드: 400

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

StartExpenseAnalysis

연락처 정보, 구매한 항목 및 공급업체 이름과 같은 데이터에 대한 송장 또는 영수증의 비동기 분석을 시작합니다.

StartExpenseAnalysis에서는 JPEG, PNG 및 PDF 형식의 문서에서 텍스트를 분석할 수 있습니다. 문서는 Amazon S3 버킷에 저장되어야 합니다. 사용DocumentLocation파라미터를 사용하여 S3 버킷 이름과 해당 버킷에 있는 문서의 이름을 지정합니다.

StartExpenseAnalysis작업 식별자를 반환합니다 (JobId) 귀하가 제공할 것GetExpenseAnalysis을 사용하여 작업 결과를 검색합니다. 입력 송장/영수증 분석이 완료되면 Amazon Textract Textract는 완료 상태를 셀러가 제공하는 아마존 Simple Notification Service (Amazon SNS) 주제에 게시합니다.NotificationChannel. 송장 및 영수증 분석 작업의 결과를 얻으려면 Amazon SNS 주제에 게시된 상태 값이 다음과 같은지 확인하십시오.SUCCEEDED. 그렇다면 전화하십시오.GetExpenseAnalysis작업 식별자를 전달하고 (JobId) 귀하의 전화가 반환되었습니다.StartExpenseAnalysis.

자세한 내용은 단원을 참조하십시오.송장 및 수금 분석.

요청 구문

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

요청 파라미터

요청은 JSON 형식의 다음 데이터를 받습니다.

ClientRequestToken

시작 요청을 식별하는 데 사용되는 멱등성 토큰입니다. 동일한 토큰을 여러 개 사용하여 사용하는 경우`StartDocumentTextDetection`요청, 동일`JobId`가 반환됩니다. 사용`ClientRequestToken`같은 작업이 실수로 두 번 이상 시작되지 않도록 합니다. 자세한 내용은 단원을 참조하십시오.Amazon Textract 비동기 작업 호출

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: `^[a-zA-Z0-9-_]+$`

: 필수 아니요

DocumentLocation

처리할 문서의 위치입니다.

유형: DocumentLocation 객체

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

JobTag

지정한 식별자로, Amazon SNS 주제에 게시된 완료 알림에 포함됩니다. 예를 들어 를 사용할 수 있습니다.`JobTag`완료 통지가 해당하는 문서 유형 (예: 세금 양식 또는 영수증) 을 식별합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: `[a-zA-Z0-9_.\-:]+`

: 필수 아니요

KMSKeyId

추론 결과를 암호화하는 데 사용되는 KMS 키입니다. 키 ID 또는 키 별칭 형식일 수 있습니다. KMS 키가 제공되면 고객 버킷에 있는 객체의 서버 측 암호화에 KMS 키가 사용됩니다. 이 매개 변수를 활성화하지 않으면 결과는 SSE-S3 를 사용하여 서버 측에서 암호화됩니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 2048.

Pattern: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

: 필수 아니요

NotificationChannel

Amazon Textract가 작업의 완료 상태를 게시하도록 하려는 Amazon SNS 주제 ARN입니다.

유형: NotificationChannel 객체

: 필수 아니요

OutputConfig

출력이 고객 정의 버킷으로 이동할지 여부를 설정합니다. 기본적으로 Amazon Textract Textract는 다음에 액세스할 수 있도록 결과를 내부적으로 저장합니다.`GetExpenseAnalysis`작업.

유형: OutputConfig 객체

: 필수 아니요

응답 구문

```
{
   "JobId": "string"
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 다시 전송합니다.

다음 데이터는 서비스에 의해 JSON 형식으로 반환됩니다.

JobId

텍스트 검색 작업의 고유 식별자입니다. 이`JobId`에서 반환됩니다.`StartExpenseAnalysis`. A`JobId`값은 7일 동안만 유효합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 64.

Pattern: `^[a-zA-Z0-9-_]+$`

오류

AccessDeniedException

작업을 수행할 권한이 없습니다. 권한 있는 사용자 또는 IAM 역할의 Amazon 리소스 이름(ARN)을 사용하여 작업을 수행하십시오.

HTTP 상태 코드: 400

BadDocumentException

Amazon Textract Textract는 문서를 읽을 수 없습니다. Amazon Textract Textract의 문서 한도에 대한 자세한 내용은 단원을 참조하십시오.Amazon Textract TEXTRACT에서의 하드 제한.

HTTP 상태 코드: 400

DocumentTooLargeException

문서가 너무 크기 때문에 처리할 수 없습니다. 동기 작업의 최대 문서 크기입니다. 10MB입니다. 비동기 작업의 최대 문서 크기는 PDF 파일의 경우 500MB입니다.

HTTP 상태 코드: 400

IdempotentParameterMismatchException

`AClientRequestToken`입력 파라미터가 작업에서 재사용되었지만 하나 이상의 다른 입력 파라미터가 이전의 작업 호출과 다릅니다.

HTTP 상태 코드: 400

InternalServerError

Amazon Textract Textract에 서비스 문제가 발생했습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

InvalidKMSKeyException

KMS 키를 입력한 상태에서 암호 해독 권한이 없거나 KMS 키가 잘못 입력되었음을 나타냅니다.

HTTP 상태 코드: 400

InvalidParameterException

입력 파라미터가 제약 조건을 위반했습니다. 예를 들어, 동기 작업에서 는`InvalidParameterException`예외가 발생하지 않을 때`S3Object`또는`Bytes`값은 다음 위치에 제공됩니다.`Document`요청 파라미터를 지정합니다. 파라미터를 확인한 다음 API 작업을 다시 호출하십시오.

HTTP 상태 코드: 400

InvalidS3ObjectException

Amazon Textract Textract가 요청에서 지정된 S3 객체에 액세스할 수 없습니다.Amazon S3 대한 액세스 구성문제 해결 정보는 를 참조하십시오.Amazon S3 문제 해결

HTTP 상태 코드: 400

LimitExceededException

Amazon Textract 서비스 한도를 초과했습니다. 예를 들어 너무 많은 비동기 작업을 동시에 시작하면 시작 작업을 호출합니다 (`StartDocumentTextDetection`예를 들어) 을 발생시킬 경우에는 동시 실행 작업의 수가 Amazon Textract 서비스 제한 미만이 될 때까지 예외 (HTTP 상태 코드: 400) 가 발생합니다.

HTTP 상태 코드: 400

ProvisionedThroughputExceededException

요청의 수가 처리량 한도를 초과했습니다. 이 한도를 늘려야 하는 경우 Amazon Textract Textract에 문의하십시오.

HTTP 상태 코드: 400

ThrottlingException

Amazon Textract Textract가 요청을 일시적으로 처리할 수 없습니다. 호출을 다시 시도하십시오.

HTTP 상태 코드: 500

UnsupportedDocumentException

출력 문서의 형식은 지원되지 않습니다. 작업 문서는 PNG, JPEG, PDF 또는 TIFF 형식일 수 있습니다.

HTTP 상태 코드: 400

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWSPython용 SDK
- AWS SDK for Ruby V3

데이터 유형

다음 데이터 유형이 지원됩니다.

- AnalyzeIDDetections
- Block
- BoundingBox
- Document
- DocumentLocation
- DocumentMetadata
- ExpenseDetection
- ExpenseDocument
- ExpenseField
- ExpenseType
- Geometry
- HumanLoopActivationOutput
- HumanLoopConfig
- HumanLoopDataAttributes

- IdentityDocument
- IdentityDocumentField
- LineItemFields
- LineItemGroup
- NormalizedValue
- NotificationChannel
- OutputConfig
- Point
- Relationship
- S3Object
- Warning

AnalyzeIDDetections

AnalyzeID 작업에서 감지된 정보를 포함하는 데 사용됩니다.

목차

Confidence

감지된 텍스트 의 신뢰 점수입니다.

Type: 부동 소수점

유효한 범위: 최소값 0. 최대 값은 100입니다.

필수 사항: 아니요

NormalizedValue

날짜에 대해서만 반환되며, 감지된 값의 유형과 더 기계 읽기 쉬운 방식으로 작성된 날짜를 반환합니다.

유형: NormalizedValue 객체

필수 사항: 아니요

Text

정규화된 필드 또는 연관된 값의 텍스트입니다.

Type: String

필수 사항: 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

Block

ABlock는 서로 가까운 픽셀 그룹 내의 문서에서 인식되는 항목을 나타냅니다. 에서 반환되는 정보Block객체는 작업 유형에 따라 다릅니다. 문서에 대한 텍스트 감지 (예:DetectDocumentText) 를 선택하면 검색된 단어와 텍스트 줄에 대한 정보를 얻을 수 있습니다. 텍스트 분석 (예:AnalyzeDocument) 을 사용하면 문서에서 검색된 필드, 테이블 및 선택 요소에 대한 정보도 얻을 수 있습니다.

의 어레이Block객체는 동기 및 비동기 작업 모두에 의해 반환됩니다. 동기식 작업 (예:DetectDocumentText의 배열Block객체는 전체 결과 집합입니다. 비동기 작업 (예:GetDocumentAnalysis를 선택하면 하나 이상의 응답을 통해 배열이 반환됩니다.

자세한 내용은 단원을 참조하십시오.Amazon Textract 작동 방식.

목차

BlockType

인식되는 텍스트 항목의 유형입니다. 텍스트 감지 작업에서 다음 유형이 반환됩니다.

- 페이지- LINE 목록이 포함되어 있습니다.Block문서 페이지에서 감지된 개체입니다.
- 단어- 문서 페이지에서 감지된 단어입니다. 단어는 공백으로 구분되지 않은 하나 이상의 ISO 기본 라틴 스크립트 문자입니다.
- 선- 문서 페이지에서 감지되는 탭으로 구분된 연속 단어 문자열입니다.

텍스트 분석 작업에서는 다음 유형이 반환됩니다.

- 페이지- 하위 목록이 포함되어 있습니다.Block문서 페이지에서 감지된 개체입니다.
- 키_값_세트- 키 및 가치 저장Block문서 페이지에서 감지된 링크된 텍스트의 개체입니다. 사용EntityTypeKEY_VALUE_SET 객체가 키인지 확인하는 필드Block객체 또는 값Block객체입니다.
- 단어- 문서 페이지에서 감지된 단어입니다. 단어는 공백으로 구분되지 않은 하나 이상의 ISO 기본 라틴 스크립트 문자입니다.
- 선- 문서 페이지에서 감지되는 탭으로 구분된 연속 단어 문자열입니다.
- 표- 문서 페이지에서 감지된 테이블입니다. 테이블은 두 개 이상의 행이나 열이 있는 그리드 기반 정보로, 셀 범위는 한 행과 각각 하나의 열로 구성됩니다.
- 세포- 감지된 테이블 내의 셀입니다. 셀은 셀의 텍스트를 포함하는 블록의 상위입니다.
- 선택_요소- 옵션 버튼 (라디오 버튼) 또는 문서 페이지에서 감지된 확인란과 같은 선택 요소 다음 값을 사용합니다.SelectionStatus선택 요소의 상태를 확인할 수 있습니다.

Type: String

유효한 값: KEY_VALUE_SET | PAGE | LINE | WORD | TABLE | CELL | SELECTION_ELEMENT

: 필수 아니요

ColumnIndex

테이블 셀이 나타나는 열입니다. 첫 번째 열 위치는 1입니다.ColumnIndex는 반환되지 않습니다.DetectDocumentText과GetDocumentTextDetection.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

ColumnSpan

테이블 셀이 걸쳐있는 열 수입니다. 현재 이 값은 스팬된 열 수가 1 보다 큰 경우에도 항상 1입니다.ColumnSpan는 반환되지 않습니다.DetectDocumentText과GetDocumentTextDetection.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

Confidence

인식된 텍스트의 정확도와 인식된 텍스트 주위의 지오메트리 포인트의 정확성에 대해 Amazon Textract Textract의 신뢰도 점수입니다.

Type: 부동 소수점

유효한 범위: 최소값 0. 최대값 100입니다.

: 필수 아니요

EntityTypes

엔터티의 유형입니다. 다음을 반환할 수 있습니다.

- 키- 문서의 필드에 대한 식별자입니다.
- 값- 필드 텍스트입니다.

EntityTypes는 반환되지 않습니다.DetectDocumentText과GetDocumentTextDetection.

Type: 문자열 배열

유효한 값: KEY | VALUE

: 필수 아니요

Geometry

이미지에서 인식된 텍스트의 위치입니다. 여기에는 텍스트를 둘러싸는 축 정렬된 거친 경계 상자와 보다 정확한 공간 정보를 위한 미세한 다각형이 포함되어 있습니다.

유형: Geometry 객체

: 필수 아니요

Id

인식된 텍스트의 식별자입니다. 식별자는 단일 작업에 대해서만 고유합니다.

Type: String

Pattern: .*\S.*

: 필수 아니요

Page

블록이 감지된 페이지입니다.Page는 비동기 작업에 의해 반환됩니다. 1보다 큰 페이지 값은 PDF 또는 TIFF 형식의 여러 페이지 문서에 대해서만 반환됩니다. 스캔한 이미지 (JPEG/PNG) 는 여러 문서 페이지가 포함되어 있더라도 단일 페이지 문서로 간주됩니다. 의 가치Page는 항상 1입니다. 동기식 작업이 반환되지 않음Page모든 입력 문서는 단일 페이지 문서로 간주되기 때문입니다.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

Relationships

현재 블록의 하위 블록 목록입니다. 예를 들어 LINE 객체에는 텍스트 줄의 일부인 각 WORD 블록에 대한 하위 블록이 있습니다. 현재 블록에 하위 블록이 없는 경우와 같이 존재하지 않는 관계에 대한 관계 객체가 목록에 없습니다. 목록 크기는 다음과 같습니다.

- 0 - 블록에 하위 블록이 없습니다.
- 1 - 블록에 하위 블록이 있습니다.

Type: 배열Relationship사물

: 필수 아니요

RowIndex

테이블 셀이 있는 행입니다. 첫 번째 행 위치는 1입니다.RowIndex는 반환되지 않습니다.DetectDocumentText과GetDocumentTextDetection.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

RowSpan

테이블 셀의 행 수입니다. 현재 이 값은 스팬된 행 수가 1보다 크더라도 항상 1입니다.RowSpan는 반환되지 않습니다.DetectDocumentText과GetDocumentTextDetection.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

SelectionStatus

옵션 버튼이나 확인란과 같은 선택 요소의 선택 상태입니다.

Type: String

유효한 값: SELECTED | NOT_SELECTED

: 필수 아니요

Text

Amazon Textract Textract에서 인식하는 단어 또는 텍스트 행입니다.

Type: String

: 필수 아니요

TextType

Amazon Textract Textract가 감지한 텍스트의 종류입니다. 필기 텍스트와 인쇄된 텍스트를 확인할 수 있습니다.

Type: String

유효한 값: `HANDWRITING | PRINTED`

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

BoundingBox

검색된 페이지, 텍스트, 키-값 쌍, 표, 표 셀 또는 문서 페이지의 선택 요소 주위의 경계 상자입니다. 이`left`(x-좌표) 및`top`(y 좌표) 는 경계 상자의 상단 및 왼쪽을 나타내는 좌표입니다. 이미지의 좌측 상단 모서리는 원점 (0,0) 입니다.

이`top`과`left`반환되는 값은 전체 문서 페이지 크기의 비율입니다. 예를 들어 입력 이미지가 700x200 픽셀이고, 경계 상자의 상단-좌측 좌표가 350x50픽셀일 경우 API는 `left` 값 0.5(350/700)와 `top` 값 0.25(50/200)를 반환합니다.

이`width`과`height`값은 전체 문서 페이지 차원에 대한 비율로서 경계 상자의 크기를 나타냅니다. 예를 들어 문서 페이지 크기가 700 x 200픽셀이고 테두리 상자 너비가 70픽셀인 경우 반환되는 너비는 0.1입니다.

목차

Height

높이 전체 문서 페이지 높이에 대한 비율로서 경계 상자의 높이입니다.

Type: 부동 소수점

: 필수 아니요

Left

좌측 전체 문서 페이지 너비에 대한 비율로서 경계 상자의 좌측 좌표입니다.

Type: 부동 소수점

: 필수 아니요

Top

상단 전체 문서 페이지 높이에 대한 비율로서 경계 상자의 상단 좌표입니다.

Type: 부동 소수점

: 필수 아니요

Width

너비 전체 문서 페이지 너비에 대한 비율로서 경계 상자의 너비입니다.

Type: 부동 소수점

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

Document

입력 문서 (바이트 또는 S3 객체) 입니다.

를 사용하여 Amazon Textract API 작업에 이미지 바이트를 전달합니다.`Bytes`속성입니다. 예를 들어 다음을 사용합니다.`Bytes`로컬 파일 시스템에서 로드된 문서를 전달하는 속성입니다. 를 사용하여 전달된 이미지 바이트`Bytes`속성은 base64로 인코딩해야 합니다. AWS SDK를 사용하여 Amazon Textract API 작업을 호출하는 경우 코드가 문서 파일 바이트를 인코딩하지 않아도 될 수 있습니다.

다음을 사용하여 S3 버킷에 저장된 이미지를 Amazon Textract API 작업으로 전달합니다.`S3Object`속성입니다. S3 버킷에 저장된 문서는 base64로 인코딩할 필요가 없습니다.

S3 객체가 있는 S3 버킷의 AWS 리전과 Amazon Textract 작업에 사용하는 AWS 리전과 일치해야 합니다.

AWS CLI를 사용하여 Amazon Textract 작업을 호출하는 경우 Bytes를 사용하여 이미지 바이트를 전달하는 작업은 지원되지 않습니다. 먼저 문서를 Amazon S3 버킷에 업로드한 다음 S3Object 속성을 사용하여 작업을 호출해야 합니다.

Amazon Textract Textract가 S3 객체를 처리하려면 사용자에게 S3 객체에 액세스할 수 있는 권한이 있어야 합니다.

목차

Bytes

base64로 인코딩된 문서 바이트의 blob입니다. Blob 단위로 제공되는 문서의 최대 크기는 5MB입니다. 문서 바이트는 PNG 또는 JPEG 형식이어야 합니다.

AWS SDK를 사용하여 Amazon Textract Textract를 호출하는 경우 다음을 사용하여 전달된 이미지 바이트를 base64로 인코딩할 필요가 없습니다.`Bytes`필드.

Type: Base64 인코딩 이진수 데이터 객체

길이 제약 조건: 최소 길이는 1이고, 최대 길이 10485760.

: 필수 아니요

S3Object

S3 객체를 문서 소스로 식별합니다. S3 버킷에 저장된 문서의 최대 크기는 5MB입니다.

유형: S3Object 객체

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

DocumentLocation

처리할 문서가 저장된 Amazon S3 버킷입니다. 다음과 같은 비동기 작업에서 사용됩니다.StartDocumentTextDetection.

입력 문서는 JPEG 또는 PNG 형식의 이미지 파일일 수 있습니다. PDF 형식의 파일일 수도 있습니다.

목차

S3Object

입력 문서가 저장된 Amazon S3 버킷입니다.

유형: S3Object 객체

요구 사항: 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

DocumentMetadata

입력 문서에 대한 정보입니다.

목차

Pages

문서에서 검색된 페이지 수입니다.

Type: 정수

유효한 범위: 최소값 0.

필수 사항: 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

ExpenseDetection

Amazon Textract Textract에서 감지한 값 또는 라벨에 대한 정보를 저장하는 데 사용되는 객체입니다.

목차

Confidence

감지에 대한 신뢰도 (백분율)

Type: 부동 소수점

유효한 범위: 최소값 0. 최대값 100입니다.

: 필수 아니요

Geometry

검색된 페이지, 텍스트, 키-값 페어, 테이블, 테이블 셀 및 선택 요소와 같은 문서 페이지에서 항목이 있는 위치에 대한 정보.

유형: Geometry 객체

: 필수 아니요

Text

Amazon Textract Textract에서 인식하는 단어 또는 텍스트 줄

Type: String

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

ExpenseDocument

애널리즈피스가 반환한 모든 정보를 담고 있는 구조

목차

ExpenseIndex

문서의 어떤 인보이스 또는 영수증을 나타냅니다. 따라서 첫 번째 문서는 1이고 두 번째 2입니다.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

LineItemGroups

문서의 각 테이블에서 감지된 정보 (`LineItems`.

Type: 배열LineItemGroup사물

: 필수 아니요

SummaryFields

Amazon Textract Textract에서 테이블 외부에서 발견된 모든 정보입니다.

Type: 배열ExpenseField사물

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

ExpenseField

탐지된 정보의 분석, 카테고리 유형, 레이블 감지 및 값 감지로 구분됩니다.

목차

LabelDetection

탐지된 요소의 명시적으로 명시된 레이블입니다.

유형: ExpenseDetection 객체

: 필수 아니요

PageNumber

값이 감지된 페이지 번호입니다.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

Type

탐지된 요소의 암시적 레이블입니다. 명시적 요소에 대한 레이블 탐지와 함께 표시됩니다.

유형: ExpenseType 객체

: 필수 아니요

ValueDetection

탐지된 요소의 값입니다. 명시적 및 암시적 요소에 존재합니다.

유형: ExpenseDetection 객체

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++

- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

ExpenseType

Amazon Textract Textract에서 감지한 유형에 대한 정보를 저장하는 데 사용되는 객체입니다.

목차

Confidence

정확성에 대한 신뢰도를 백분율로 나타냅니다.

Type: 부동 소수점

유효한 범위: 최소값 0. 최대값 100.

필수: 아니요

Text

Amazon Textract Textract에서 감지한 단어 또는 텍스트 행입니다.

Type: String

필수: 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

Geometry

검색된 페이지, 텍스트, 키-값 페어, 테이블, 테이블 셀 및 선택 요소와 같은 항목이 문서 페이지에 있는 위치에 대한 정보입니다.

목차

BoundingBox

문서 페이지에서 인식된 항목의 위치를 축으로 정렬한 거친 표현입니다.

유형: BoundingBox 객체

: 필수 아니요

Polygon

경계 상자 내에서 인식된 항목 주위에 세밀한 다각형이 있습니다.

Type: 배열Point사물

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

HumanLoopActivationOutput

루프 평가에서 사람의 결과를 표시합니다. HumanLooPARN이 없으면 입력이 인간 검토를 트리거하지 않았습니다.

목차

HumanLoopActivationConditionsEvaluationResults

인간 검토를 활성화한 조건을 포함하여 조건 평가 결과를 표시합니다.

Type: String

길이 제약 조건: 최대 길이 10,240입니다.

: 필수 사항 아니요

HumanLoopActivationReasons

사람의 검토가 필요한지 여기와 이유를 보여줍니다.

Type: 문자열 배열

배열 멤버: 최소 항목 수는 1개입니다.

: 필수 사항 아니요

HumanLoopArn

생성된 HumanLoop Amazon 리소스 이름 (ARN) 입니다.

Type: String

길이 제약 조건: 최대 길이 256.

: 필수 사항 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go

- AWSSDK for Java V2
- AWS SDK for Ruby V3

HumanLoopConfig

조건 중 하나가 충족되면 문서를 보낼 사람 검토 워크플로를 설정합니다. 검토하기 전에 이미지의 특정 속성을 설정할 수도 있습니다.

목차

DataAttributes

입력 데이터의 속성을 설정합니다.

유형: HumanLoopDataAttributes 객체

: 필수 아니요

FlowDefinitionArn

흐름 정의의 Amazon 리소스 이름 (ARN) 입니다.

Type: String

길이 제약 조건: 최대 길이 256.

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

HumanLoopName

이 이미지에 사용되는 인적 워크플로의 이름입니다. 리전 내에서 고유해야 합니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이 63.

Pattern: `^[a-z0-9](-*[a-z0-9])*`

: 필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++

- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

HumanLoopDataAttributes

이미지의 속성을 설정할 수 있습니다. 현재 이미지를 개인 식별 정보 및 성인 콘텐츠가 없는 것으로 선언할 수 있습니다.

목차

ContentClassifiers

입력 이미지에 개인 식별 정보 또는 성인 콘텐츠가 없다는 것을 설정합니다.

Type: 문자열 배열

어레이 멤버: 최대 항목 수는 256개입니다.

유효한 값: `FreeOfPersonallyIdentifiableInformation | FreeOfAdultContent`

필수: 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

IdentityDocument

AnalyzeID 작업에서 처리된 각 문서를 나열하는 구조입니다.

목차

DocumentIndex

IdentityDocument 목록에서 문서의 배치를 나타냅니다. 첫 번째 문서는 1로 표시되고 두 번째 문서는 2로 표시됩니다.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

IdentityDocumentFields

ID 문서에서 추출한 정보를 기록하는 데 사용되는 구조입니다. 정규화된 필드와 추출된 텍스트의 값을 모두 포함합니다.

Type: 배열IdentityDocumentField사물

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

IdentityDocumentField

추출된 정보의 정규화된 유형과 이와 연관된 텍스트를 모두 포함하는 구조체입니다. 이들은 각각 유형 및 값으로 추출됩니다.

목차

Type

AnalyzeID 작업에서 감지된 정보를 포함하는 데 사용됩니다.

유형: AnalyzeIDDetections 객체

: 필수 아니요

ValueDetection

AnalyzeID 작업에서 감지된 정보를 포함하는 데 사용됩니다.

유형: AnalyzeIDDetections 객체

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

LineItemFields

문서 테이블에 있는 여러 줄에 대한 정보를 담고 있는 구조입니다.

목차

LineItemExpenseFields

테이블에서 감지된 라인의 정보를 표시하는 데 사용되는 ExpenseFields.

Type: 배열ExpenseField사물

필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

LineItemGroup

LineItems를 포함하는 테이블 그룹화이며, 각 테이블은 테이블로 식별됩니다.`LineItemGroupIndex`.

목차

LineItemGroupIndex

문서에서 특정 테이블을 식별하는 데 사용되는 번호입니다. 첫 번째 테이블에는 라인 항목 그룹 인덱스 1, 두 번째 2 등이 있습니다.

Type: 정수

유효한 범위: 최소값 0.

: 필수 아니요

LineItems

테이블의 특정 행에 대한 정보 분석입니다.

Type: 배열LineItemFields사물

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

NormalizedValue

문서의 날짜와 관련된 정보 (값 유형 및 값 포함) 를 포함합니다.

목차

Value

연월-일수:분:초로 작성된 날짜의 값입니다.

Type: String

: 필수 아니요

ValueType

탐지된 값의 정규화된 유형입니다. 이 경우 DATE입니다.

Type: String

유효한 값: `DATE`

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

NotificationChannel

Amazon Textract가 다음과 같은 비동기 문서 작업의 완료 상태를 게시할 Amazon Simple Notification Service (Amazon SNS) 주제를 생성합니다.StartDocumentTextDetection.

목차

RoleArn

Amazon Textract에 Amazon SNS 주제에 대한 게시 권한을 부여하는 IAM 역할의 Amazon 리소스 이름 (ARN) 입니다.

Type: String

길이 제약 조건: 최소 길이 20. 최대 길이 2048.

Pattern: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

SNSTopicArn

Amazon Textract가 완료 상태를 게시하는 Amazon SNS 주제입니다.

Type: String

길이 제약 조건: 최소 길이 20. 최대 길이는 1,024입니다.

Pattern: `(^arn:([a-z\d-]+):sns:[a-zA-Z\d-]{1,20}:\w{12}:.+$)`

필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2

- AWS SDK for Ruby V3

OutputConfig

출력이 사용자 생성 버킷으로 이동할지 여부를 설정합니다. 버킷의 이름과 출력 파일의 접두사를 설정하는 데 사용됩니다.

`OutputConfig`는 출력 위치를 조정할 수 있는 선택적 매개 변수입니다. 기본적으로 Amazon Textract Textract는 결과를 내부적으로 저장하며 Get API 작업에서만 액세스할 수 있습니다. OutputConfig를 활성화하면 출력이 전송될 버킷의 이름과 결과를 다운로드할 수 있는 결과의 파일 접두사를 설정할 수 있습니다. 또한 다음을 설정할 수 있습니다.`KMSKeyID`CMK (고객 마스터 키) 를 사용하여 출력을 암호화합니다. 이 매개 변수를 설정하지 않으면 Amazon Textract Textract는 Amazon S3 S3용 AWS 관리형 CMK를 사용하여 서버 측을 암호화합니다.

Amazon Textract Textract에서 문서를 처리하려면 고객 콘텐츠의 암호 해독이 필요합니다. AI 서비스 옵트아웃 정책에 따라 계정이 옵트아웃된 경우, 서비스에서 고객 콘텐츠를 처리한 후 암호화되지 않은 모든 고객 콘텐츠가 즉시 영구적으로 삭제됩니다. 출력의 사본은 Amazon Textract Textract에 의해 보존되지 않습니다. 자세한 선택 해제 방법은 단원을 참조하십시오.AI 서비스 옵트아웃 정책 관리.

데이터 프라이버시에 대한 자세한 내용은 단원을 참조하십시오.데이터 프라이버시 FAQ.

목차

S3Bucket

출력된 버킷의 이름.

Type: String

길이 제약 조건: 최소 길이는 3이고, 최대 길이 255.

Pattern: `[0-9A-Za-z\.\-_]*`

필수 프로세스는 페이지 쓰기 후 세그먼트화된 가장 오래전에 사용된(SLRU) 데이터가 내구성 있는 스토리지에 도달할 때까지 기다리고 있습니다.

S3Prefix

출력을 저장할 객체 키의 접두사입니다. 활성화되지 않은 경우 접두사는 “textract_output”이 됩니다.

Type: String

길이 제약 조건: 최소 길이는 1이고, 최대 길이는 1,024입니다.

Pattern: `.*\S.*`

필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

Point

문서 페이지에 있는 점의 X 및 Y 좌표입니다. 반환되는 X 및 Y 값은 전체 문서 페이지 크기의 비율입니다. 예를 들어 입력 문서가 700 x 200이고 작업이 X=0.5이고 Y=0.25를 반환하는 경우 점은 문서 페이지의 (350,50) 픽셀 좌표에 있습니다.

의 어레이`Point`객체입니다.`Polygon`이 (가) 의 일부로 반환됩니다.Geometry에서 반환되는 객체Block객체입니다. A`Polygon`object는 감지된 텍스트, 키-값 쌍, 표, 표 셀 또는 선택 요소 주위의 세밀한 다각형을 나타냅니다.

목차

X

a 상의 점에 대한 X 좌표 값입니다.`Polygon`.

Type: 부동 소수점

: 필수 아니요

Y

a 상의 점에 대한 Y 좌표의 값입니다.`Polygon`.

Type: 부동 소수점

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

Relationship

블록이 서로 어떻게 관련되어 있는지에 대한 정보입니다. ABlock개체에 0 이상 포함Relation리스트에 있는 객체,Relationships. 자세한 정보는 Block을 참조하십시오.

이Type요소는 모든 블록에 대한 관계 유형을 제공합니다.IDs어레이.

목차

Ids

관련 블록에 대한 ID 배열입니다. 에서 관계의 유형을 볼 수 있습니다.Type요소.

Type: 문자열 배열

Pattern: `.*\S.*`

필수 사항: 아니요

Type

ID 배열의 블록이 현재 블록과 갖는 관계 유형입니다. 관계는 다음과 같습니다.VALUE또는CHILD. VALUE 유형의 관계는 키-값 쌍의 KEY와 연결된 VALUE 블록의 ID를 포함하는 목록입니다. CHALD 유형의 관계는 행의 경우 WORD 블록을 식별하는 ID 목록입니다. 테이블의 경우 셀 블록과 선택 요소의 경우 WORD 블록을 식별합니다.

Type: String

유효한 값: `VALUE | CHILD | COMPLEX_FEATURES`

필수 사항: 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

S3Object

문서를 식별하는 S3 버킷 이름 및 파일 이름입니다.

문서가 포함된 S3 버킷의 AWS 리전은 Amazon Textract 작업에 사용하는 리전과 일치해야 합니다.

Amazon Textract Textract가 S3 버킷의 파일을 처리하려면 사용자에게 S3 버킷과 파일에 액세스할 수 있는 권한이 있어야 합니다.

목차

Bucket

S3 버킷의 이름 파일 이름에는 # 문자가 유효하지 않습니다.

Type: String

길이 제약: 최소 길이는 3이고, 최대 길이 255.

Pattern: `[0-9A-Za-z\.\-_]*`

: 필수 아니요

Name

입력 문서의 파일 이름입니다. 동기 작업은 JPEG 또는 PNG 형식의 이미지 파일을 사용할 수 있습니다. 비동기 작업은 PDF 및 TIFF 형식 파일도 지원합니다.

Type: String

길이 제약: 최소 길이는 1이고, 최대 길이는 1,024입니다.

Pattern: `.*\S.*`

: 필수 아니요

Version

버킷의 버전 관리가 활성화된 경우 객체 버전을 지정할 수 있습니다.

Type: String

길이 제약: 최소 길이는 1이고, 최대 길이는 1,024입니다.

Pattern: `.*\S.*`

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

Warning

비동기 텍스트 분석 중 발생한 문제에 대한 경고 (StartDocumentAnalysis) 또는 비동기 문서 텍스트 감지 (StartDocumentTextDetection).

목차

ErrorCode

경고에 대한 오류 코드입니다.

Type: String

: 필수 아니요

Pages

경고가 적용되는 페이지의 목록입니다.

Type: 정수 배열

유효한 범위: 최소값 0.

: 필수 아니요

참고 항목

이 API를 언어별 AWS SDK 중 하나로 사용하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby V3

Amazon Textract TEXTRACT에서의 하드 제한

다음은 Amazon Textract Textract의 하드 제한 목록입니다. 변경 불가. 위치 제한 및 변경 가능한 제한에 대한 자세한 내용은Amazon Textract 엔드포인트 및 할당량. 변경 가능한 제한에 대한 자세한 내용은 AWS 서비스 제한을 참조하십시오. 제한을 변경하려면 Create Case(사례 생성)을 참조하십시오.

Amazon Textract

한도	설명
허용된 파일 형식	작업은 JPEG, PNG, PDF 및 TIFF 파일을 지원합니다. (PDF 내에서 JPEG 2000으로 인코딩된 이미지가 지원됩니다)..
파일 크기 및 페이지 수 제한	동기 작업의 경우 JPEG, PNG, PDF 및 TIFF 파일의 크기 제한이 10MB입니다. PDF 및 TIFF 파일도 1페이지로 제한됩니다. 비동기 작업의 경우 JPEG 및 PNG 파일의 크기 제한이 10MB입니다. PDF 및 TIFF 파일에는 500MB의 제한이 있습니다. PDF 및 TIFF 파일의 제한은 3,000페이지입니다.
PDF 관련 제한	최대 높이와 너비는 40인치와 2880포인트입니다. PDF는 암호로 보호될 수 없습니다. PDF에는 JPEG 2000 형식의 이미지가 포함될 수 있습니다.
문서 회전 및 이미지 크기	Amazon Textract Textract는 모든 평면 내 문서 회전을 지원합니다 (예: 45도 평면 내 회전). Amazon Textract Textract는 해상도가 모든면에서 10000 픽셀 이하인 이미지를 지원합니다.
텍스트 정렬	텍스트는 문서 내에서 가로로 정렬된 텍스트일 수 있습니다. Amazon Textract Textract는 문서 내에서 세로 텍스트 정렬을 지원하지 않습니다.
언어	Amazon Textract Textract는 영어, 프랑스어, 독일어, 이탈리아어, 포르투갈어 및 스페인어 텍스트 감지를 지원합니다. Amazon Textract Textract는 출력에서 감지된 언어를 반환하지 않습니다.

한도	설명
문자 크기	감지할 텍스트의 최소 높이는 15픽셀입니다. 150 DPI에서 이 글꼴은 8 포인트 글꼴과 같습니다.
문자 형식	Amazon Textract Textract는 손글씨 및 인쇄된 문자 인식을 모두 지원합니다.
문자	Amazon Textract TEXTRACT에서는 다음 문자를 감지합니다. • a-z • A-Z • 0~9 • ä Ö Ö ü ü ü é é é ü ü ü ü ü é é é é í ó ú ü ü ü ü ü ñ ò ã • ! “# $% '& () * +, -./:; =? @ [\] ^ _ `{|} ~ > <° € £ ¥ # ß ¡€ £ ¥ # Ø ©™ § ¹ ² ³ '
애널리제이ID 특정 제한	AnalyzeID는 미국 여권 및 미국 운전 면허증만 지원합니다.

Amazon Textract Textract용 문서 기록

다음 표에서는 의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.Amazon Textract 개발자 안내서. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- 최신 설명서 업데이트: 2019년 5월 29일

update-history-change	update-history-description	update-history-date
코드 예제 통합AWS문서 SDK 코드 예제 GitHub 리포지토리	이제 Amazon Textract 가이드에 추가 코드 예제가 포함되어 있습니다. 이전 예제 섹션의 이름을 튜토리얼로 변경했습니다.	2022년 1월 30일
분석비용 추가된 내용	Amazon Textract Textract는 이제 AnalyzeFreents API를 사용하여 송장 및 영수증 문서의 분석을 지원합니다. 이 기능은 아시아 태평양 (뭄바이), 아시아 태평양 (서울), 아시아 태평양 (싱가포르), 아시아 태평양 (시드니), 캐나다 (중부), 유럽 (프랑크푸르트), 유럽 (아일랜드), 유럽 (런던), 미국 동부 (버지니아 북부), 미국 동부 (캘리포니아 북부) 리전에서 사용 가능합니다.	2021년 7월 26일
Augmented AI Support	Amazon Textract Textract는 이제 인간 검토를 구현하기 위해 Amazon Augmented AI를 지원합니다.	2019년 12월 3일

새로운 서비스 및 가이드	Amazon Textract Textract를 이제 일반 용도로 사용할 수 있습니다.	2019년 5월 29일
선택 요소 Support	이제 Amazon Textract Textract에서 선택 요소 (라디오 버튼 및 확인란) 를 감지할 수 있습니다.	2019년 4월 24일
Amazon Textract Textract의 출시	이 설명서는 Amazon Textract Textract용 설명서의 첫 번째 릴리스입니다.	2018년 11월 28일

AWS 용어집

최신 AWS 용어는 AWS 일반 참조서의 AWS 용어집을 참조하세요.

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