aws

Guía para desarrolladores

Amazon Textract

Amazon Textract: Guía para desarrolladores

Table of Contents

¿Qué es Amazon Textract?

Amazon Textract Texact facilita la incorporación de detección y análisis de textos de documentos a sus aplicaciones. Utilizando Amazon Textract, los clientes pueden:

- Detecte textos escritos a mano y escritos a mano en una variedad de documentos, incluidos informes financieros, registros médicos y formularios fiscales.
- Extraiga texto, formularios y tablas de documentos con datos estructurados mediante la API de análisis de documentos de Amazon Textract.
- Procesar facturas y recibos con la API AnalyzeExpense.
- Procesar documentos de identificación, tales como licencias de conducir y pasaportes emitidos por el gobierno de EE. UU., utilizando la API AnalyzeID.

Amazon Textract se basa en la misma tecnología de aprendizaje profundo de eficacia probada y altamente escalable desarrollada por los científicos de visión artificial de Amazon para analizar miles de millones de imágenes y vídeos al día. No necesita experiencia en aprendizaje automático para utilizarlo. Amazon Textract Texact incluye API sencillas y fáciles de usar que pueden analizar archivos de imagen y archivos PDF. Amazon Textract Texact está aprendiendo siempre a partir de nuevos datos y Amazon añade continuamente nuevas características al servicio.

A continuación, se indican algunos casos de uso comunes para usar Amazon Textract:

- Creación de un índice de búsqueda inteligente— Con Amazon Textract puede crear bibliotecas de texto que se detectan en archivos de imagen y PDF.
- Uso de la extracción inteligente de textos para el procesamiento del lenguaje natural (NLP)— Amazon Textract Texact le proporciona control sobre cómo se agrupa el texto como entrada para aplicaciones NLP. Puede extraer texto como palabras y líneas. También agrupa texto por celdas de tabla si el análisis de tablas de documentos de Amazon Textract está habilitado.
- Aceleración de la captura y normalización de datos de diferentes fuentes— Amazon Textract permite la extracción de datos tabulares y de texto de una amplia variedad de documentos, tales como documentos financieros, informes de investigación y notas médicas. Con las API de Amazon Textract Analyze Document, puede extraer de forma fácil y rápida datos no estructurados y estructurados de sus documentos.
- Automatización de la captura de datos desde formularios— Amazon Texact permite extraer datos estructurados de los formularios. Con las API de Amazon Textract Analysis, puede crear

capacidades de extracción en flujos de trabajo empresariales existentes para que los datos de usuario enviados a través de formularios se puedan extraer en un formato utilizable.

Algunos de los beneficios de usar Amazon Textract son:

- Integración de la detección de texto de documentos en sus aplicaciones: Amazon Textract elimina la complejidad de crear funciones de detección de textos en sus aplicaciones al permitir realizar análisis potentes y exactos de textos con una sencilla API. No necesita conocimientos en visión artificial o aprendizaje profundo para usar Amazon Textract detectar textos de documentos. Con las API de Amazon Textract Texact, puede crear fácilmente la detección de texto en cualquier aplicación web, móvil o de un dispositivo conectado.
- Análisis de documentos escalables— Amazon Textract le permite analizar y extraer datos rápidamente de millones de documentos, lo que puede acelerar la toma de decisiones.
- Bajo coste: con Amazon Textract Texact, solo paga por los documentos que analiza. No se requieren pagos mínimos ni compromisos iniciales. Puede empezar con una versión gratuita y disfrutar de un gran ahorro cuando se amplíen sus necesidades con nuestro modelo de precios por niveles de.

Con el procesamiento sincrónico, Amazon Textract analizar documentos de una sola página para aplicaciones en las que la latencia es crítica. Amazon Textract también proporciona operaciones asíncronas para ampliar el soporte a documentos de varias páginas.

Usuarios nuevos Amazon Textract

Si es la primera vez que utiliza Amazon Textract Texact, le recomendamos que lea las siguientes secciones en orden:

1. Funcionamiento de Amazon Textract: esta sección presenta los componentes de Amazon Textract Texact y cómo funcionan juntos para disfrutar de una experiencia integral.
2. Introducción a Amazon Textract: en esta sección va a configurar su cuenta y probar la API de Amazon Textract Texact.

Uso de Amazon Textract con unAWSSDK

Los kits de desarrollo de software (SDK) de AWS están disponibles en muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK for C++	Ejemplos de código de AWS SDK for C++
AWS SDK for Go	Ejemplos de código de AWS SDK for Go
AWS SDK for Java	Ejemplos de código de AWS SDK for Java
AWS SDK for JavaScript	Ejemplos de código de AWS SDK for JavaScript
AWS SDK for .NET	Ejemplos de código de AWS SDK for .NET
AWS SDK for PHP	Ejemplos de código de AWS SDK for PHP
AWS SDK for Python (Boto3)	Ejemplos de código de AWS SDK for Python (Boto3)
AWS SDK for Ruby	Ejemplos de código de AWS SDK for Ruby

Ejemplo de disponibilidad

¿No puede encontrar lo que necesita? Solicite un ejemplo de código mediante el enlace de Provide feedback (Proporcionar comentarios) que se encuentra en la parte inferior de esta página.

Funcionamiento de Amazon Textract

Amazon Textract Texact le permite detectar y analizar texto en documentos de entrada de una o varias páginas (consulteDocumentos de entrada).

Amazon Textract Texact proporciona operaciones para las siguientes acciones.

- Detección de texto únicamente. Para obtener más información, consulteDetección de texto.
- Detección y análisis de relaciones entre texto. Para obtener más información, consulteAnálisis de documentos.
- Detección y análisis de texto en facturas y recibos. Para obtener más información, consulteAnálisis de facturas y recibos.
- Detección y análisis de texto en documentos de identidad gubernamentales. Para obtener más información, consulteAnálisis de documentos de identidad.

Amazon Textract proporciona operaciones sincrónicas para procesar documentos pequeños, de una sola página y con respuestas casi en tiempo real. Para obtener más información, consulte Procesamiento de documentos con operaciones síncronas. Amazon Textract Texact también proporciona operaciones asíncronas que puede utilizar para procesar documentos de mayor tamaño de varias páginas. Las respuestas asíncronas no están en tiempo real. Para obtener más información, consulte Procesamiento de documentos con operaciones asíncronas.

Cuando una operación de Amazon Textract procesa un documento, los resultados se devuelven en una matriz dethe section called “Block”objetos o matriz dethe section called “ExpenseDocument”objects. Ambos objetos contienen información detectada sobre los elementos, incluida su ubicación en el documento y su relación con otros elementos del documento. Para obtener más información, consulte Objetos de respuesta de Amazon Textract. Para ver ejemplos donde se muestra cómo utilizar`Block`objetos, consulteTutoriales.

Temas

- Detección de texto
- Análisis de documentos
- Análisis de facturas y recibos
- Análisis de documentos de identidad
- Documentos de entrada

- Objetos de respuesta de Amazon Textract
- Ubicación del elemento en una página de documento

Detección de texto

Amazon Textract Texact proporciona operaciones sincrónicas y asíncronas que devuelven solo el texto detectado en un documento. Para ambos conjuntos de operaciones, se devuelve la siguiente información en variosthe section called “Block”objects.

- Las líneas y palabras del texto detectado
- Relaciones entre las líneas y las palabras del texto detectado
- Página en la que aparece el texto detectado
- Ubicación de las líneas y palabras del texto en la página del documento

Para obtener más información, consulte the section called “Líneas y palabras de texto”.

Para detectar texto de forma sincrónica, utilice elDetectDocumentTextOperación API y pase un archivo de documento como entrada. La operación devuelve todo el conjunto de resultados. Para obtener más información y un ejemplo, consulte Procesamiento de documentos con operaciones síncronas.

> **Note**
>
> La operación de la API de Amazon Rekognition`DetectTextes` diferente de`DetectDocumentText`. Usas`DetectText`para detectar texto en escenas en directo, como carteles o señales de tráfico.

Para detectar texto de forma asíncrona, utiliceStartDocumentTextDetectionpara empezar a procesar un archivo de documento de entrada. Para obtener los resultados, llameGetDocumentTextDetection. Los resultados se devuelven en una o más respuestas de`GetDocumentTextDetection`. Para obtener más información y un ejemplo, consulte Procesamiento de documentos con operaciones asíncronas.

Análisis de documentos

Amazon Textract analiza documentos y formularios en busca de relaciones entre el texto detectado. Las operaciones de análisis de Amazon Textract devuelven 3 categorías de extracción de documentos: texto, formularios y tablas. El análisis de facturas y recibos se gestiona mediante un proceso diferente; para obtener más información, consulteAnálisis de facturas y recibos.

Extracción de texto

Texto sin procesar extraído de un documento. Para obtener más información, consulteLíneas y palabras de texto.

Extraction de formularios

Los datos del formulario están vinculados a elementos de texto extraídos de un documento. Amazon Textract representa los datos de formulario como pares clave-valor. En el siguiente ejemplo, una de las líneas de texto detectadas por Amazon Textract esName: Jane Doe. Amazon Textract Texact también identifica una clave (Name:) y un valor (Jane Doe). Para obtener más información, consulteDatos de formulario (pares clave-valor).

Name: Jane Doe

Address: 123 Any Street, Anytown, Estados Unidos

Fecha de nacimiento: 12-26-09-1980

Los pares clave-valor también se utilizan para representar casillas de verificación o botones de opción (botones de opción) que se extraen de los formularios.

male:☑

Para obtener más información, consulteElementos de selección.

Extraction de tablas

Amazon Texact puede extraer tablas, celdas de tabla y elementos de celdas de tabla y puede programarse para devolver los resultados en un archivo JSON, .csv o un archivo.txt.

Nombre	Dirección
Ana Carolina	Cualquier Ciudad 123

Para obtener más información, consulte Tablas. Los elementos de selección también se pueden extraer de las tablas. Para obtener más información, consulteElementos de selección.

Para los artículos analizados, Amazon Textract devuelve lo siguiente en variosthe section called "Block"objects:

- Las líneas y palabras del texto detectado
- El contenido de los elementos detectados
- Relación entre los elementos detectados
- Página en la que se ha detectado el elemento
- Ubicación del elemento en la página del documento

Puede utilizar operaciones síncronas o asíncronas para analizar el texto de un documento. Para analizar el texto de forma sincrónica, utilice elAnalyzeDocumenty pasar un documento como entrada.`AnalyzeDocument`devuelve todo el conjunto de resultados. Para obtener más información, consulte Análisis del texto del documento con Amazon Textract.

Para detectar texto de forma asíncrona, utiliceStartDocumentAnalysispara empezar a procesar. Para obtener los resultados, llameGetDocumentAnalysis. Los resultados se devuelven en una o más respuestas de`GetDocumentAnalysis`. Para obtener más información y un ejemplo, consulte Detección o análisis de texto en un documento de varias páginas.

Para especificar qué tipo de análisis se va a realizar, puede utilizar el`FeatureTypes`parámetro de entrada de lista. Agregue TABLES a la lista para devolver información sobre las tablas detectadas en el documento de entrada, por ejemplo, celdas de tabla, texto de celda y elementos de selección de celdas. Agregue FORMULARIOS para devolver relaciones de palabras, como pares clave-valor y elementos de selección. Para realizar ambos tipos de análisis, agregue TABLAS y FORMS a`FeatureTypes`.

Todas las líneas y palabras detectadas en el documento se incluyen en la respuesta (incluido el texto no relacionado con el valor de`FeatureTypes`).

Análisis de facturas y recibos

Amazon Textract Texact extrae datos relevantes, como información de contacto, artículos comprados y nombre del proveedor, de casi cualquier factura o recibo sin necesidad de plantillas ni configuración. Las facturas y recibos suelen utilizar varios diseños, lo que dificulta y lleva mucho tiempo extraer datos manualmente a escala. Amazon Textract utiliza ML para comprender el

contexto de las facturas y los recibos y extrae automáticamente datos como la fecha de la factura o la recepción, el número de factura o recibo, los precios del artículo, el importe total y las condiciones de pago para satisfacer las necesidades de su empresa.

Amazon Textract también identifica nombres de proveedores que son críticos para sus flujos de trabajo pero que no se etiquetan explícitamente. Por ejemplo, Amazon Textract puede encontrar el nombre del proveedor en un recibo aunque solo se indique dentro de un logotipo en la parte superior de la página sin una combinación explícita de par clave-valor. Amazon Textract Texact también le facilita la consolidación de los insumos de diversos recibos y facturas que utilizan palabras diferentes para el mismo concepto. Por ejemplo, Amazon Textract asigna relaciones entre nombres de campo en distintos documentos, como el número de cliente, el número de cliente y el ID de cuenta, lo que genera taxonomía estándar como`INVOICE_RECEIPT_ID`. En este caso, Amazon Textract los datos de forma coherente en distintos tipos de documentos. Los campos que no se alinean con la taxonomía estándar se clasifican como`OTHER`.

A continuación se muestra la lista de los campos estándar que AnalyzeExpense admite actualmente:

- Nombre del proveedor:`VENDOR_NAME`
- Total:`TOTAL`
- Dirección del destinatario:`RECEIVER_ADDRESS`
- Fecha de factura/recepción:`INVOICE_RECEIPT_DATE`
- ID de factura/recibo:`INVOICE_RECEIPT_ID`
- Condiciones de pago:`PAYMENT_TERMS`
- Subtotal:`SUBTOTAL`
- Fecha de vencimiento:`DUE_DATE`
- Tax:`TAX`
- ID del contribuyente de facturas (SSN/ITIN o EIN):`TAX_PAYER_ID`
- Nombre del objeto:`ITEM_NAME`
- Precio del artículo:`PRICE`
- Cantidad de artículo:`QUANTITY`

La API AnalyzeExpense devuelve los siguientes elementos de una página de documento determinada:

- El número de recibos o facturas de una página representada como`ExpenseIndex`

- El nombre estandarizado de los campos individuales representados como`Type`
- El nombre real del campo tal como aparece en el documento, representado como`LabelDetection`
- El valor del campo correspondiente representado como`ValueDetection`
- El número de páginas del documento presentado representadas como`Pages`
- El número de página en el que se detectó el campo, el valor o las líneas de pedido, representado como`PageNumber`
- La geometría, que incluye el cuadro delimitador y la ubicación de las coordenadas del campo, el valor o los elementos de línea individuales de la página, representada como`Geometry`
- Puntuación de confianza asociada a cada dato detectado en el documento, representado como`Confidence`
- La fila completa de líneas de pedido individuales compradas, representadas como`EXPENSE_ROW`

A continuación se muestra una parte de la salida de la API de un recibo procesado por AnalyzeExpense que muestra el Total: 55,64$ en el documento extraído como campo estándar`TOTAL`, texto real del documento como «Total», Puntuación de confianza de «97,1», Número de página «1», El valor total como «55,64$» y el cuadro delimitador y las coordenadas del polígono:


```
{
    "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
}
```

Puede utilizar operaciones síncronas para analizar una factura o un recibo. Para analizar estos documentos, utilice la operación AnalyzeExpense y le pasa un recibo o factura.`AnalyzeExpense`devuelve todo el conjunto de resultados. Para obtener más información, consulte Análisis de facturas y recibos con Amazon Textract.

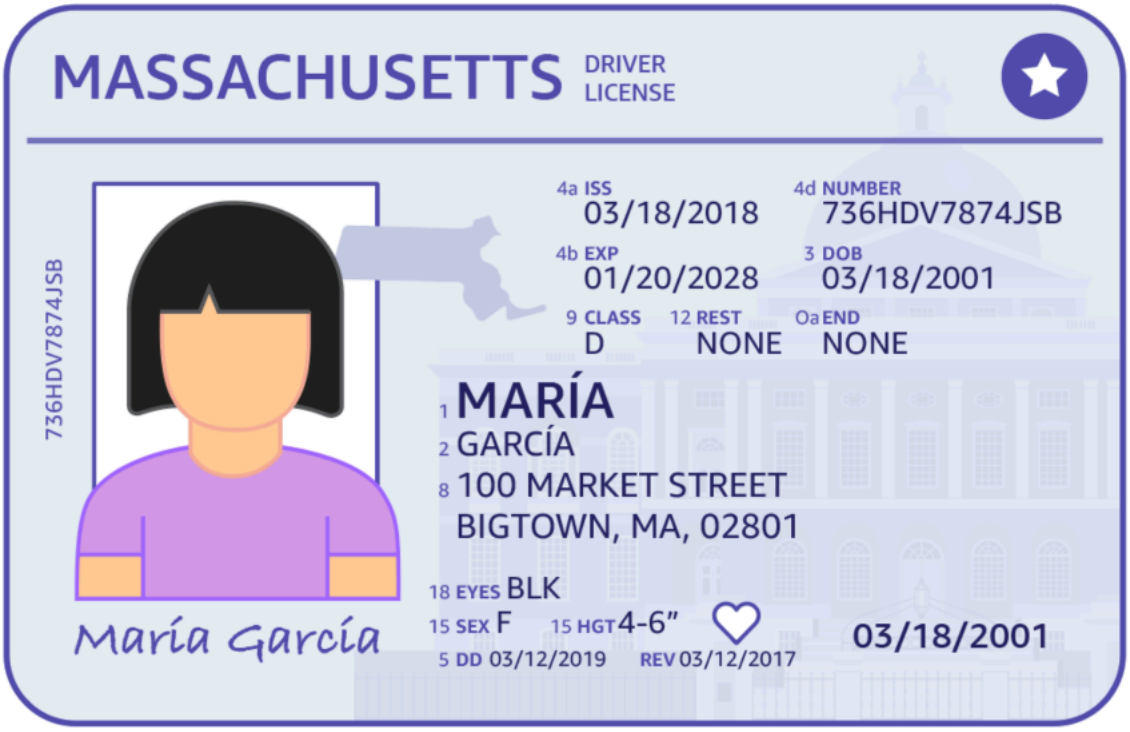
Para analizar facturas y recibos de forma asíncrona, utiliceStartExpenseAnalysispara empezar a procesar un archivo de documento de entrada. Para obtener los resultados, llameGetExpenseAnalysis. Los resultados de una llamada dada aStartExpenseAnalysisson devueltas por`GetExpenseAnalysis`. Para obtener más información y un ejemplo, consulte Procesamiento de documentos con operaciones asíncronas.

Análisis de documentos de identidad

Amazon Textract puede extraer información relevante de pasaportes, licencias de conducir y otra documentación de identidad emitida por el gobierno de EE. UU. mediante la API AnalyzeID. Con Analyze ID, las empresas pueden extraer información rápida y precisa de identificadores como licencias de conducir estadounidenses, ID de estado y pasaportes que tienen una plantilla o formato diferentes. La API AnalyzeID devuelve dos categorías de tipos de datos:

- Pares clave-valor disponibles en ID, como fecha de nacimiento, fecha de emisión, número de ID, clase y restricciones.
- Campos implícitos del documento que pueden no tener claves explícitas asociadas a ellos, como Nombre, Dirección y Emitido por.

Los nombres clave están estandarizados dentro de la respuesta. Por ejemplo, si su licencia de conducir indica el número de licencia (número de licencia) y el pasaporte dice No de pasaporte, la respuesta de Analizar ID devolverá la clave estandarizada como «ID de documento» junto con la clave sin procesar (por ejemplo, número de licencia). Esta estandarización permite a los clientes combinar fácilmente información en muchos ID que utilizan términos diferentes para el mismo concepto.



Analizar ID devuelve información de las estructuras denominadas`IdentityDocumentFields`. Estos son`JSO`Nestructuras que contienen dos fragmentos de información: el tipo normalizado y el valor asociado al tipo. Ambos también tienen un puntaje de confianza. Para obtener más información, consulte Objetos de respuesta de la documentación.

Puede utilizar operaciones sincrónicas para analizar una licencia de conducir o pasaporte. Para analizar estos documentos, utilice la operación AnalyzeID y le pasa un documento de identidad.`AnalyzeID`devuelve todo el conjunto de resultados. Para obtener más información, consulte Análisis de la documentación de identidad con Amazon Textract.

> **Note**
>
> Algunos documentos de identidad, como las licencias de conducir, tienen dos caras. Puede pasar las imágenes frontal y posterior de las licencias de conducir como imágenes separadas dentro de la misma solicitud de API de Analyze ID.

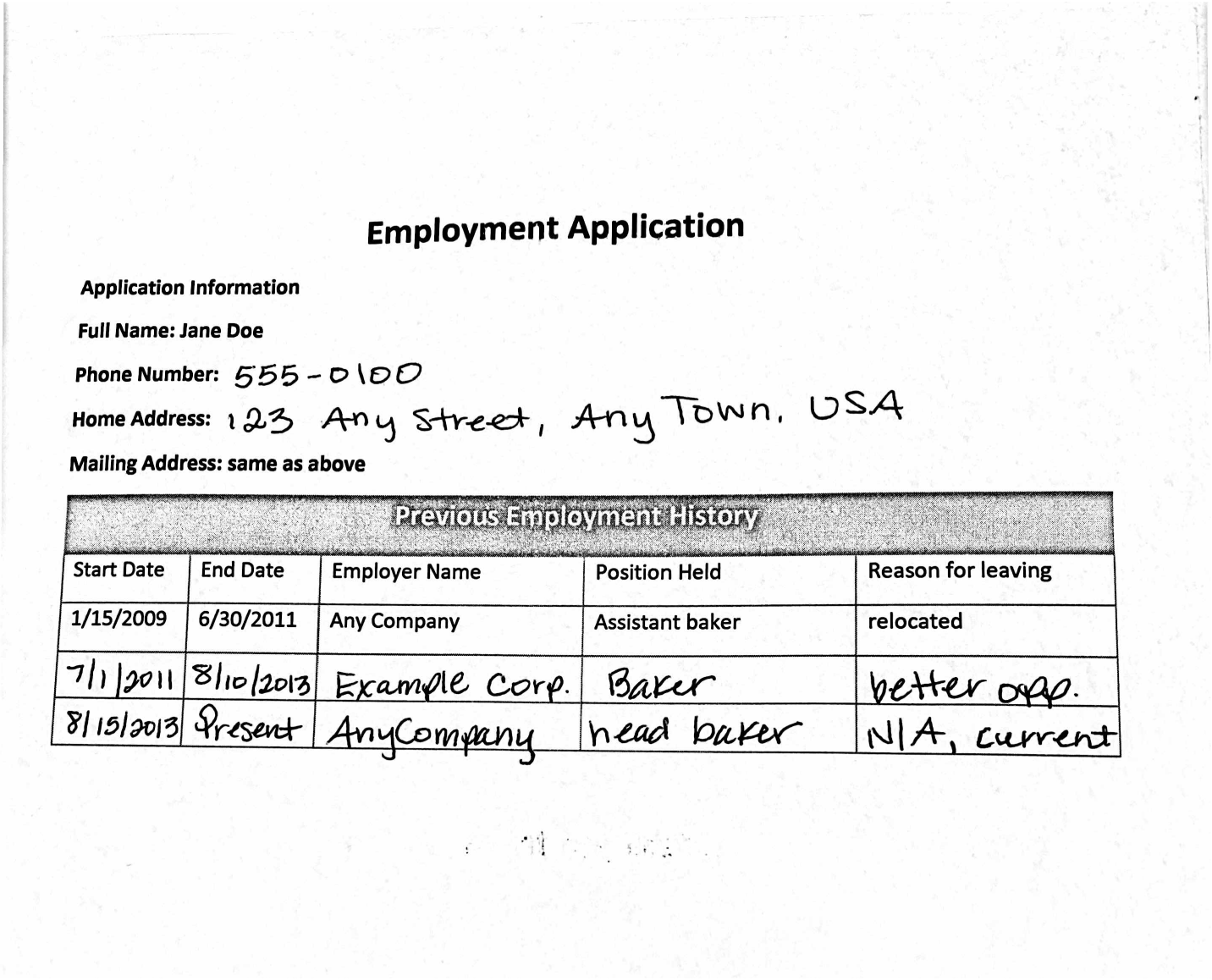
Documentos de entrada

Una entrada adecuada para una operación de Amazon Textract Texact es un documento de una o varias páginas. Algunos ejemplos son un documento legal, un formulario, un documento de identidad

o una carta. Un formulario es un documento con preguntas o solicitudes para que un usuario proporcione respuestas. Algunos ejemplos son un formulario de registro de pacientes, un formulario fiscal o un formulario de reclamación de seguro.

Un documento puede estar en formato JPEG, PNG, PDF o TIFF. Con los archivos en formato PDF y TIFF, puede procesar documentos de varias páginas. Para obtener información sobre cómo Amazon Textract representa los documentos como`Block`objetos, consulte[Objetos de respuesta de detección de texto y análisis de documentos](#).

A continuación se muestra el ejemplo de documento de entrada aceptable.



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

Para obtener información sobre los límites de documentos, consulte[Límites máximos de Amazon Textract](#).

Para las operaciones síncronas de Amazon Texact, puede utilizar documentos de entrada almacenados en un bucket de Amazon S3 o pasar bytes de imagen codificados en base64. Para obtener más información, consulte Llamar a operaciones sincrónicas de Amazon Textract. Para operaciones asíncronas, debe proporcionar documentos de entrada en un bucket de Amazon S3. Para obtener más información, consulte Llamar a operaciones asíncronas de Amazon Textract.

Objetos de respuesta de Amazon Textract

Las operaciones de Amazon Textract devuelven distintos tipos de objetos en función de las operaciones ejecutadas. Para detectar texto y analizar un documento genérico, la operación devuelve un objeto Block. Para analizar una factura o un recibo, la operación devuelve un objeto ExpenseDocuments. Para analizar la documentación de identidad, la operación devuelve un objeto IdentityDocumentFields. Para obtener más información acerca de estos objetos de respuesta, consulte las siguientes secciones:

Temas

- Objetos de respuesta de detección de texto y análisis de documentos
- Objetos de respuesta de factura y recepción
- Objetos de respuesta de la documentación

Objetos de respuesta de detección de texto y análisis de documentos

Cuando Amazon Textract procesa un documento, crea una lista deBlockobjetos del texto detectado o analizado. Cada bloque contiene información sobre un artículo detectado, dónde se encuentra y la confianza que Amazon Textract tiene en la exactitud del procesamiento.

Un documento se compone de los siguientes tipos de`Block`objects.

- Páginas
- Líneas y palabras de texto
- Datos de formulario (pares clave-valor)
- Tablas y celdas
- Elementos de selección

El contenido de un bloque depende de la operación a la que llame. Si llama a una de las operaciones de detección de texto, se devuelven las páginas, las líneas y las palabras del texto detectado. Para

obtener más información, consulte Detección de texto. Si llama a una de las operaciones de análisis de documentos, se devuelve información sobre las páginas detectadas, los pares clave-valor, las tablas, los elementos de selección y el texto. Para obtener más información, consulte Análisis de documentos.

Algunos `Block` campos de objetos son comunes a ambos tipos de procesamiento. Por ejemplo, cada bloque tiene un identificador único.

Para ver ejemplos donde se muestra cómo usar `Block` objetos, consulte Tutoriales.

Diseño de documento

Amazon Textract Texact devuelve una representación de un documento como una lista de diferentes tipos de `Block` objetos vinculados en una relación padre a hijo o en un par clave-valor. También se devuelven metadatos que proporcionan el número de páginas de un documento. A continuación se muestra el JSON para un típico `Block` objeto de tipo `PAGE`.

```
{
    "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
    }
}
```

Un documento se elabora a partir de uno o variosPAGEbloques. Cada página contiene una lista de bloques secundarios para los elementos principales detectados en la página, como líneas de texto y tablas. Para obtener más información, consulte Páginas.

Puede determinar el tipo de`Block`objeto inspeccionando el`BlockType`.

UNA`Block`contiene una lista de`Block`objetos del`Relationships`, que es una matriz de`Relationship`objects. UNA`Relationships`array es del tipo CHILD o del tipo VALUE. Se utiliza una matriz de tipo CHILD para enumerar los elementos secundarios del bloque actual. Por ejemplo, si el bloque actual es del tipo LINE,`Relationships`contiene una lista de identificadores de los bloques WORD que componen la línea de texto. Una matriz de tipo VALUE se utiliza para contener pares clave-valor. Puede determinar el tipo de relación inspeccionando el`Type`del`Relationship`object.

Los bloques secundarios no tienen información sobre sus objetos Bloquear padre.

Para ver ejemplos que muestran`Block`información, consulte Procesamiento de documentos con operaciones síncronas.

Confianza

Las operaciones de Amazon Textract Texact devuelven el porcentaje de confianza que Amazon Textract Texact tiene respecto a la precisión del artículo detectado. Para obtener la confianza, utilice el`Confidence`del`Block`object. Un valor más elevado indica una mayor confianza. Según el escenario, las detecciones con poca confianza podrían necesitar la confirmación visual de un humano.

Geometry

Las operaciones de Amazon Textract, a excepción del análisis de identidad, devuelven información de ubicación sobre la ubicación de los artículos detectados en una página de documento. Para obtener la ubicación, utilice la`Geometry`del`Block`object. Para obtener más información, consulteUbicación del elemento en una página de documento

Páginas

Un documento consta de una o más páginas. UNAthe section called “Block”objeto de tipo`PAGE`existe para cada página del documento. UNA`PAGE`objeto bloque contiene una lista de los ID secundarios de las líneas de texto, pares clave-valor y tablas que se detectan en la página del documento.

El JSON para un`PAGE`El bloque tiene un aspecto similar al siguiente.

```
{

    "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
},
```

Si utilizas operaciones asíncronas con un documento de varias páginas en formato PDF, puedes determinar la página en la que se encuentra un bloque inspeccionando la`Page`del`Block`object. Una imagen escaneada (una imagen en formato JPEG, PNG, PDF o TIFF) se considera un documento de una sola página, incluso si hay más de una página de documento en la imagen. Las operaciones asíncronas siempre devuelven un`Page`valor de 1 para imágenes escaneadas.

El número total de páginas se devuelve en la`Pages`field de`DocumentMetadata`.`DocumentMetadata`se devuelve con cada lista de`Block`objetos devueltos por una operación de Amazon Textract Texact.

Líneas y palabras de texto

El texto detectado devuelto por las operaciones de Amazon Textract Texact se devuelve en una lista dethe section called “Block”objects. Estos objetos representan líneas de texto o palabras textuales que se detectan en una página de documento. El texto siguiente muestra dos líneas de texto hechas de varias palabras.

Esto es texto.

En dos líneas separadas.

El texto detectado se devuelve en el`Text`de un`Block`object. La`BlockType`determina si el texto es una línea de texto (LINE) o una palabra (WORD). UNAWORDes uno o varios caracteres en alfabeto latino básico ISO que no están separados por espacios. UNALÍNEAes una cadena de palabras contiguas y delimitadas por tabulaciones.

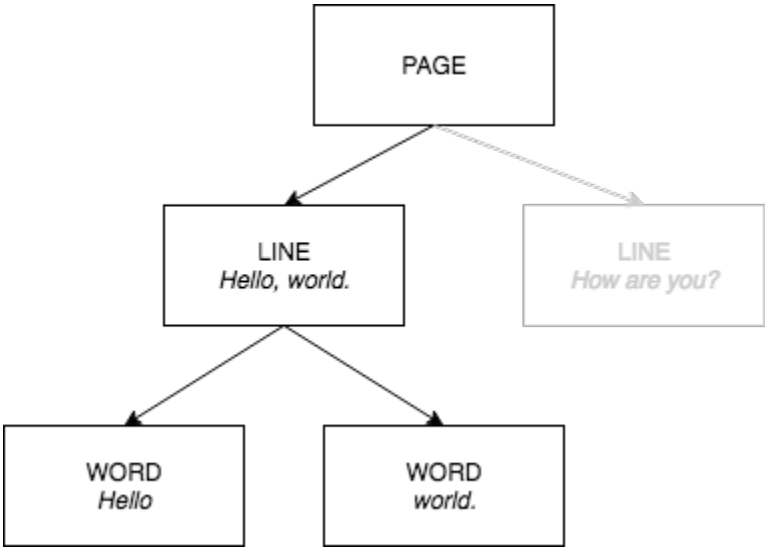
Además, Amazon Textract Texact determinará si un fragmento de texto se ha escrito a mano o se imprimió utilizando el`TextTypes`. Estos devuelven como ESCRITURA A MANO e IMPRESOS respectivamente.

El otro`Block`las propiedades son comunes a todos los tipos de bloques, como la información de ID, confianza y geometría. Para obtener más información, consulte the section called “Objetos de respuesta de detección de texto y análisis de documentos”.

Para detectar solo líneas y palabras, puede utilizarDetectDocumentTextoStartDocumentTextDetection. Para obtener más información, consulte Detección de texto. Para obtener el texto detectado (líneas y palabras) e información sobre cómo se relaciona con otras partes del documento, como tablas, puede utilizarAnalyzeDocumentoStartDocumentAnalysis. Para obtener más información, consulte Análisis de documentos.

PAGE,LINE, yWORDlos bloques están relacionados entre sí en una relación de padre a hijo. UNAPAGEblock es el principal de todosLINEobjetos de bloque en una página de documento. Debido a que una LINE puede tener una o más palabras, laRelationshipsarray para un bloque LINE almacena los ID de los bloques WORD secundarios que componen la línea de texto.

En el siguiente diagrama, se muestra cómo la líneaHello, world.del textoHello, world. ¿Cómo estás?está representado porBlockobjects.



A continuación se muestra la salida JSON deDetectDocumentTextcuando la fraseHello, world. ¿Cómo estás?se detecta. El primer ejemplo es el JSON de la página del documento. Tenga en cuenta cómo los ID CHILD le permiten navegar por el documento.

```
{
    "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"
},
```

El siguiente es el JSON de los bloques LINE que componen la línea «Hola, Mundo»:

```
{
    "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"
},
```

El siguiente es el JSON del bloque WORD de la palabraHello,:

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

El JSON final es el bloque WORD de la palabraworld.:

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

Datos de formulario (pares clave-valor)

Amazon Texact puede extraer datos de formulario de documentos en pares clave-valor. Por ejemplo, en el siguiente texto, Amazon Textract puede identificar una clave (Name:) y un valor (Ana Carolina).

Name: Ana Carolina

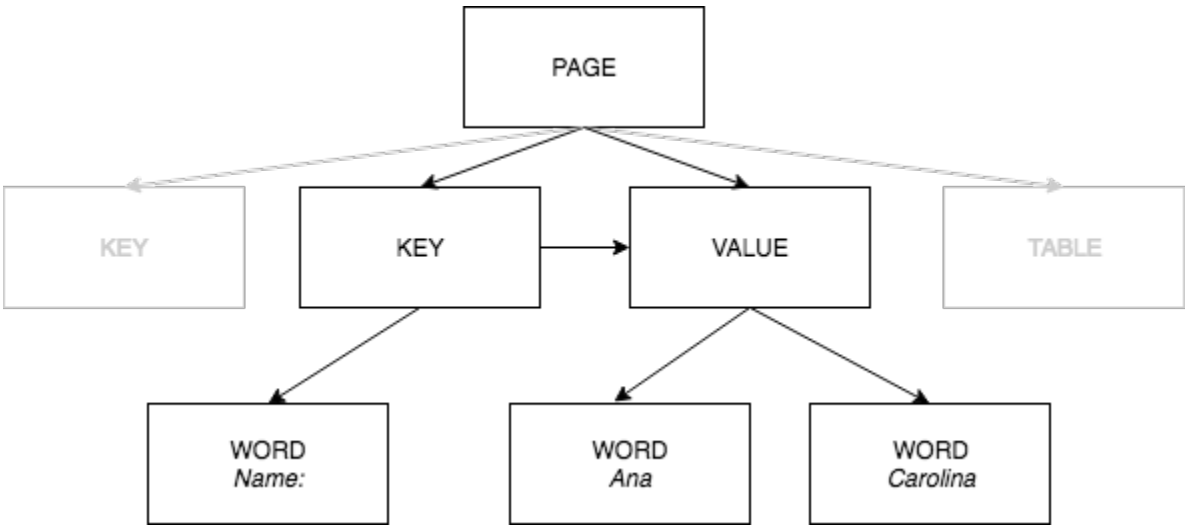
Los pares clave-valor detectados se devuelven comoBlockobjetos en las respuestas deAnalyzeDocumentyGetDocumentAnalysis. Puede utilizar el`FeatureTypes`parámetro de entrada para recuperar información sobre pares clave-valor, tablas o ambos. Solo para pares clave-valor, utilice el valor`FORMS`. Para ver un ejemplo, consulte Extracción de pares clave-valor de un documento de formulario. Para obtener información general sobre cómo un documento está representado por`Block`objetos, consulteObjetos de respuesta de detección de texto y análisis de documentos.

Los objetos de bloque con el tipo KEY_VALUE_SET son los contenedores de los objetos KEY o VALUE Block que almacenan información sobre los elementos de texto vinculados detectados en un documento. Puede utilizar el`EntityType`para determinar si un bloque es KEY o VALUE.

- UNACLAVEobjeto contiene información sobre la clave del texto vinculado. Por ejemplo,Name:. Un bloque KEY tiene dos listas de relaciones. Una relación de tipo VALUE es una lista que contiene el ID del bloque VALUE asociado a la clave. Una relación de tipo CHILD es una lista de ID de los bloques WORD que componen el texto de la clave.
- UNAVALUEobjeto contiene información sobre el texto asociado a una clave. En el modelo de ejemplo anterior,Ana Carolinaes el valor de la claveName:. Un bloque VALUE tiene una relación con una lista de bloques CHILD que identifican bloques WORD. Cada bloque WORD contiene una de las palabras que componen el texto del valor. UNAVALUEEl objeto también puede contener información sobre los elementos seleccionados. Para obtener más información, consulte Elementos de selección.

Cada instancia de un KEY_VALUE_SET`Blocks`es un elemento secundario de la PAGE`Block`objeto que corresponde a la página actual.

En el siguiente diagrama se muestra cómo el par clave-valorName: Ana Carolinaestá representado por`Block`objects.



En los siguientes ejemplos se muestra cómo el par clave-valorName: Ana Carolinaestá representado por JSON.

El bloque PAGE tiene bloques CHILD de tipoKEY_VALUE_SETpara cada bloque KEY y VALUE detectados en el documento.

```
{
    "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
},
```

El siguiente JSON muestra que el bloque KEY (52be1777-53f7-42f6-a7cf-6d09bdc15a30) tiene una relación con el bloque VALUE (7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c). También tiene un bloque CHILD para el bloque WORD (c734fca6-c4c4-415c-b6c1-30f7510b72ee) que contiene el texto de la clave (Name:).

```
{
    "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
},
```

El siguiente JSON muestra que el bloque VALUE 7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c tiene una lista SECUNDARIA de ID para los bloques WORD que componen el texto del valor (AnayCarolina).

```
{
    "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
}
```

El siguiente JSON muestra el`Block`objetos para las palabrasName:,Ana, yCarolina.

```
{
    "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"
},
```

Tablas

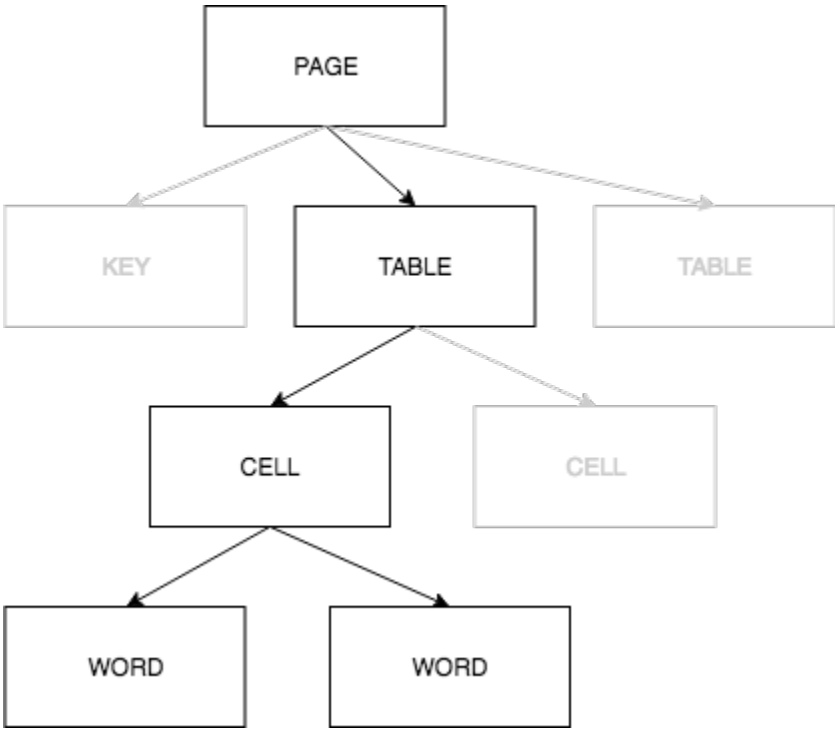
Amazon Texact puede extraer tablas y celdas de una tabla. Por ejemplo, cuando se detecta la siguiente tabla en un formulario, Amazon Textract Texact detecta una tabla con cuatro celdas.

Nombre	Dirección
Ana Carolina	Cualquier Ciudad 123

Las tablas detectadas se devuelven comoBlockobjetos en las respuestas deAnalyzeDocumentyGetDocumentAnalysis. Puede utilizar el`FeatureTypes`parámetro de entrada

para recuperar información sobre pares clave-valor, tablas o ambos. Solo para tablas, utilice el valor`TABLES`. Para ver un ejemplo, consulte Exportación de tablas a un archivo CSV. Para obtener información general sobre cómo un documento está representado por`Block`objetos, consulteObjetos de respuesta de detección de texto y análisis de documentos.

En el siguiente diagrama se muestra cómo una sola celda de una tabla está representada por:`Block`objects.



Una celda contiene`WORD`bloques para palabras detectadas, y`SELECTION_ELEMENT`bloques para elementos de selección, tales como casillas de verificación.

El siguiente es JSON parcial para la tabla anterior, que tiene cuatro celdas.

El objeto PAGE Block tiene una lista de ID de bloque CHILD para el bloque TABLE y cada LÍNEA de texto detectada.

```
{
    "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"
},
```

El bloque TABLE incluye una lista de identificadores secundarios de las celdas de la tabla. Un bloque TABLE también incluye información de geometría para la ubicación de la tabla del documento. El siguiente JSON muestra que la tabla tiene cuatro celdas, que se enumeran en el`Ids`matriz.

```
{
    "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"
},
```

El tipo de bloque de las celdas de la tabla es CELL. La`Block`objeto de cada celda incluye información sobre la ubicación de la celda en comparación con otras celdas de la tabla. También incluye información de geometría para la ubicación de la celda del documento. En el modelo de ejemplo anterior,`505e9581-0d1c-42fb-a214-6ff736822e8c`es el ID secundario de la celda que contiene la palabraNombre. El ejemplo siguiente es la información de la celda.

```
{
    "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"
},
```

Cada celda tiene una ubicación en una tabla, y la primera celda es 1,1. En el ejemplo anterior, la celda con el valorNombrese encuentra en la fila 1, columna 1. La celda con el valorCualquier Ciudad 123se encuentra en la fila 2, columna 2. Un objeto de bloque de celdas contiene esta información en el`RowIndex`y`ColumnIndex`. La lista secundaria contiene los ID de los objetos WORD Block que contienen el texto que se encuentra dentro de la celda. Las palabras de la lista están en el orden en que se detectan, desde la parte superior izquierda de la celda hasta la parte inferior derecha de la celda. En el ejemplo anterior, la celda tiene un ID secundario con el valor e9108c8e-0167-4482-989e-8b6cd3c3653e. El siguiente resultado es para el bloque WORD con el valor de ID e9108c8e-0167-4482-989e-8b6cd3c3653e:

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

Elementos de selección

Amazon Textract puede detectar elementos de selección, tales como botones de opción (botones de opción) y casillas de verificación de una página de documento. Los elementos de selección se pueden detectar endata del formularioy enTablas de. Por ejemplo, cuando se detecta la siguiente tabla en un formulario, Amazon Texact detecta las casillas de verificación de las celdas de la tabla.

	Concordar	Neutral	Discrepar
Buen servicio	☑	☐	☐
Fácil de usar	☐	☑	☐
Precio justo	☑	☐	☐

Los elementos de selección detectados se devuelven comoBlockobjetos en las respuestas deAnalyzeDocumentyGetDocumentAnalysis.

> **Note**
>
> Puede utilizar el`FeatureTypes`parámetro de entrada para recuperar información sobre pares clave-valor, tablas o ambos. Por ejemplo, si filtra por tablas, la respuesta incluye los elementos de selección que se detectan en las tablas. Los elementos de selección detectados en pares clave-valor no se incluyen en la respuesta.

La información sobre un elemento de selección está contenida en un`Block`objeto de tipo`SELECTION_ELEMENT`. Para determinar el estado de un elemento seleccionable, utilice la`SelectionStatus`del`SELECTION_ELEMENT`bloque. El estado puede serSELECCIONADASoNOT_SELECTED. Por ejemplo, el valor de`SelectionStatus`para la imagen anterior esSELECCIONADAS.

UNA`SELECTION_ELEMENT` `Block`se asocia a un par clave-valor o a una celda de tabla. UNA`SELECTION_ELEMENT` `Block`contiene información de cuadro delimitador de un elemento de selección en el`Geometry`. UNA`SELECTION_ELEMENT` `Block`objeto no es hijo de un`PAGE` `Block`object.

Datos de formulario (pares clave-valor)

Un par clave-valor se utiliza para representar un elemento de selección detectado en un formulario. La`KEY`bloque contiene el texto del elemento de selección. La`VALUE`contiene el bloque SELECTION_ELEMENT. En el siguiente diagrama, se muestran cómo los elementos de selección están representados por:the section called “Block”objects.

Para obtener más información sobre los pares clave-valor, consulteDatos de formulario (pares clave-valor).

El siguiente fragmento JSON muestra la clave de un par clave-valor que contiene un elemento de selección (male ☑). El ID secundario (Id bd14cfd5-9005-498b-a7f3-45ceb171f0ff) es el ID del bloque WORD que contiene el texto del elemento de selección (masculino). El ID del valor (Id. 24aaac7f-fcce-49c7-a4f0-3688b05586d4) es el ID delVALUEbloque que contiene elSELECTION_ELEMENTobjeto de bloque.

```
{
    "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"
}
```

El siguiente fragmento JSON es el bloque WORD de la palabraMasculino. El bloque WORD también tiene un bloque LINE principal.

```
{
    "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"
},
```

El bloque VALUE tiene un hijo (Id. f2f5e8cd-e73a-4e99-a095-053acd3b6bfb) que es el bloque SELECTION_ELEMENT.

```
{
    "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"
},
}
```

El siguiente JSON es el bloque SELECTION_ELEMENT. El valor de`SelectionStatus`indica que la casilla de verificación está activada.

```
{
    "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"
}
```

Celdas de tabla

Amazon Textract puede detectar elementos de selección dentro de una celda de tabla. Por ejemplo, las celdas de la tabla siguiente tienen casillas de verificación.

	Concordar	Neutral	Discrepar
Buen servicio	☑	☐	☐
Fácil de usar	☐	☑	☐
Precio justo	☑	☐	☐

UNA`CELL`puede contener un bloque secundario`SELECTION_ELEMENT`objetos para elementos de selección, así como secundarios`WORD`bloques para texto detectado.

Para obtener más información sobre las tablas, consulte Tablas.

La TABLA`Block`objeto de la tabla anterior tiene un aspecto similar a este.

```
{
    "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"
}
```

CeldaBLOCKobject (Id. c63ad40d-5a14-4646-a8df-2d4304213dbc) para la celda que contiene la casilla de verificaciónBuen servicioparece ser el siguiente. Incluye a un niño`Block`(Id = 26d122fd-c5f4-4b53-92c4-0ae92730ee1e) que es el`SELECTION_ELEMENT` `Block`para la casilla de verificación.

```
{
    "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"
}
```

SELECTION_ELEMENT`Block`para la casilla de verificación es el siguiente. El valor de`SelectionStatus`indica que la casilla de verificación está activada.

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

Objetos de respuesta de factura y recepción

Cuando envía una factura o un recibo a la API AnalyzeExpense, devuelve una serie de objetos ExpenseDocuments. Cada documento de gastos se divide aún más en`LineItemGroups`y`SummaryFields`. La mayoría de las facturas y recibos contienen información como el nombre del proveedor, el número de recibo, la fecha de recepción o el importe total. AnalyzeExpense devuelve esta información en`SummaryFields`. Los recibos y facturas también contienen detalles sobre los artículos comprados. La API AnalyzeExpense devuelve esta información en`LineItemGroups`. La`ExpenseIndex`identifica de forma exclusiva el gasto y asocia el correspondiente`SummaryFields`y`LineItemGroups`detectado en ese gasto.

El nivel de datos más granular en la respuesta de AnalyzeExpense consiste en`Type`,`ValueDetection`, y`LabelDetection`(Opcional). Las entidades individuales son:

- Tipo: Hace referencia a qué tipo de información se detecta a alto nivel.
- Detección de etiquetas: hace referencia a la etiqueta de un valor asociado dentro del texto del documento.`LabelDetection`es opcional y solo se devuelve si la etiqueta está escrita.
- Detección de valor: hace referencia al valor de la etiqueta o el tipo devuelto.

La API AnalyzeExpense también detecta`ITEM`,`QUANTITY`, y`PRICE`dentro de las líneas de pedido como campos normalizados. Si hay otro texto en una línea de pedido en la imagen del recibo, como SKU o descripción detallada, se incluirá en el JSON como`EXPENSE_ROW`como se muestra en el siguiente ejemplo:

```
            {
                                "Type": {
                                    "Text": "EXPENSE_ROW",
                                    "Confidence": 99.95216369628906
                                },
```

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

El ejemplo anterior muestra cómo la API AnalyzeExpense devuelve toda la fila de un recibo que contiene información de línea de pedido de 5 plátanos vendidos por 2,5 USD.

Tipo

A continuación se muestra el tipo estándar o normalizado del par clave-valor:

```
                {
                     "PageNumber": 1,
                     "Type": {
                         "Text": "VENDOR_NAME",
                         "Confidence": 70.0
                     },
                     "ValueDetection": {
                         "Geometry": { ... },
                         "Text": "AMAZON",
                         "Confidence": 87.89806365966797
                     }
                 }
```

El recibo no incluía el «Nombre del proveedor» en la lista explícita. Sin embargo, la API de análisis de gastos reconoció el documento como un recibo y clasificó el valor «AMAZON» como Tipo`VENDOR_NAME`.

Detección de etiquetas

A continuación se muestra un ejemplo de texto tal como se muestra en una página de documento de cliente:

```
                {
```

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

El documento de ejemplo contenía «CASHIER Mina». La API de análisis de gastos extrajo el valor tal cual y lo devuelve en`LabelDetection`. Para valores implícitos como «Nombre del proveedor», donde la «clave» no se muestra explícitamente en el recibo,`LabelDetection`no se incluirá en el elemento AnalyzeExpense. En tales casos, la API AnalyzeExpense no devuelve`LabelDetection`.

Detección de valor

A continuación se muestra el «valor» del par clave-valor.

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

```
                    }
                }
```

En el ejemplo, el documento contenía «CASHIER Mina». La API AnalyzeExpense detectó el valor del Cajero como Mina y lo devolvió en`ValueDetection`.

Objetos de respuesta de la documentación

Cuando envía un documento de identidad a la API AnalyzeID, devuelve una serie de`IdentityDocumentField`objects. Cada uno de estos objetos contiene`Type`, y`Value`.`Type`registra el campo normalizado que Amazon Textract Texact detecta y`Value`registra el texto asociado al campo normalizado.

A continuación se muestra un ejemplo de`IdentityDocumentField`, abreviado para mayor brevedad.

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "IdentityDocumentFields": [
        {
            "Type": {
                "Text": "first name"
            },
            "ValueDetection": {
                "Text": "jennifer",
                "Confidence": 99.99908447265625
            }
        },
        {
            "Type": {
                "Text": "last name"
            },
            "ValueDetection": {
                "Text": "sample",
                "Confidence": 99.99758911132812
            }
        },
```

Estos son dos ejemplos de IdentityDocumentFields cortados de una respuesta más larga. Hay una separación entre el tipo detectado y el valor de ese tipo. Aquí, es el nombre y el apellido respectivamente. Esta estructura se repite con toda la información contenida. Si un tipo no se reconoce como campo normalizado, aparecerá como «otro».

A continuación se presenta una lista de campos normalizados para licencias de conducir:

- name (Nombre)
- apellido
- segundo nombre
- sufijo
- city in address (ciudad)
- código postal en la dirección
- state in address (estado)
- condado
- número de documento
- fecha de vencimiento
- fecha de nacimiento
- name (nombre)
- fecha de emisión
- class
- restricciones
- avales
- id type
- veterano
- address

A continuación se presenta una lista de campos normalizados para pasaportes estadounidenses:

- name (Nombre)
- apellido
- segundo nombre

- número de documento
- fecha de vencimiento
- fecha de nacimiento
- lugar de nacimiento
- fecha de emisión
- id type

Ubicación del elemento en una página de documento

Las operaciones de Amazon Textract devuelven la ubicación y la geometría de los artículos encontrados en una página de documento.DetectDocumentTextyGetDocumentTextDetectiondevuelve la ubicación y la geometría de líneas y palabras, mientras queAnalyzeDocumentyGetDocumentAnalysisDevuelve la ubicación y la geometría de pares clave-valor, tablas, celdas y elementos de selección.

Para determinar dónde está un elemento en una página de documento, utilice el cuadro delimitador (Geometry) información devuelta por la operación Amazon Textract Texact en unBlockobject. La`Geometry`objeto contiene dos tipos de ubicación e información geométrica para elementos detectados:

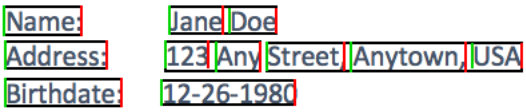
- Un eje alineadoBoundingBoxobjeto que contiene la coordenada superior izquierda y la anchura y la altura del elemento.
- Objeto poligonal que describe el contorno del elemento, especificado como una matriz dePointobjetos que contienenX(eje horizontal) yYcoordenadas de página de documento (eje vertical) de cada punto.

El JSON para un`Block`El objeto tiene un aspecto similar al siguiente. Tenga en cuenta la`BoundingBox`y`Polygon`.

```
{
    "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"
},
```


Puede utilizar información de geometría para dibujar cuadros delimitadores alrededor de los elementos detectados. Para un ejemplo que utiliza`BoundingBox`y`Polygon`información para dibujar cuadros alrededor de líneas y líneas verticales al principio y al final de cada palabra, consulte[Detección de texto de documento con Amazon Textract](). El ejemplo de resultado es similar al siguiente.



Bounding Box

Un cuadro delimitador (`BoundingBox`) presenta las siguientes propiedades:

- Height: la altura del cuadro delimitador expresada proporcionalmente respecto a la altura total de página del documento.

- Left: coordenada X del punto superior izquierdo del cuadro delimitador expresada proporcionalmente respecto a la anchura total de página del documento.
- Top: La coordenada Y del punto superior izquierdo del cuadro delimitador expresada proporcionalmente respecto a la altura total de la página del documento.
- Width: anchura del cuadro delimitador expresada proporcionalmente respecto a la anchura total de página del documento.

Cada propiedad de BoundingBox tiene un valor comprendido entre 0 y 1. El valor es una ratio del ancho total de la imagen (se aplica a`LeftyWidth`) o altura (se aplica a`HeightyTop`). Por ejemplo, si la imagen de entrada es de 700 x 200 píxeles y la coordenada superior izquierda del cuadro delimitador es de (350,50) píxeles, la API devuelve un`Left`valor de 0,5 (350/700) y un`Top`valor de 0,25 (50/200).

En el siguiente diagrama se muestra el rango de una página de documento que cubre cada propiedad BoundingBox.

Para mostrar el cuadro delimitador con su ubicación y tamaño correctos, debe multiplicar los valores de BoundingBox por la anchura o altura de la página del documento (según el valor que desee) para obtener los valores en píxeles. Los valores en píxeles se utilizan para mostrar el cuadro delimitador. Un ejemplo es utilizar una página de documento de 608 píxeles de ancho x 588 píxeles de alto y los siguientes valores de cuadro delimitador para el texto analizado:

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

La ubicación del cuadro delimitador de texto en píxeles se calcula de la siguiente manera:

```
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`

Puede utilizar estos valores para mostrar un cuadro delimitador enmarcan el texto analizado. En los siguientes ejemplos de Java y Python se muestra cómo mostrar un cuadro delimitador.

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

En este ejemplo de Python se incluye el`response`devuelto por elDetectDocumentTextOperación de la API.

```
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)
```

Polygon (Polígono)

El polígono devuelto por`AnalyzeDocument`es una matriz de[Point](#)objects. Cada`Point`tiene una coordenada X e Y para una ubicación específica de la página del documento. Al igual que las coordenadas de BoundingBox, las coordenadas poligonales se normalizan a la anchura y la altura del documento y están entre 0 y 1.

Puede utilizar puntos de la matriz de polígonos para mostrar un cuadro delimitador de grano fino alrededor de un`Block`object. La posición de cada punto poligonal en la página del documento se calcula utilizando la misma técnica utilizada para`BoundingBoxes`. Multiplique la coordenada X por el ancho de página del documento y multiplique la coordenada Y por el alto de página del documento.

En el ejemplo siguiente se muestra cómo mostrar las líneas verticales de un polígono.

```
   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));

    }
```

Introducción a Amazon Textract

En esta sección se proporcionan temas para empezar a utilizar Amazon Textract. Si es la primera vez que utiliza Amazon Textract, le recomendamos que consulte primero los conceptos y la terminología enFuncionamiento de Amazon Textract.

Puede probar la API utilizando la demostración de la consola de Amazon Textract Texact. Para obtener más información, consultehttps://console.aws.amazon.com/textract/.

Temas

- Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM
- Paso 2: Configurar laAWS CLIyAWSSDK de
- Paso 3: Introducción al uso de laAWS CLIyAWSSDK API

Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM

Antes de utilizar Amazon Textract por primera vez, realice las siguientes tareas:

1. Inscripción en AWS.
2. Creación de un usuario de IAM.

Inscripción en AWS

Cuando se inscribe en Amazon Web Services (AWS), la cuenta de AWS se registra automáticamente en todos los servicios publicados en AWS. Solo se le cobrará por los servicios que utilice.

Con Amazon Textract, paga solo por los recursos que usa. Para obtener más información acerca de las tarifas de uso de Amazon Textract, consultePrecios de Amazon Textract. Si es cliente nuevo de AWS, puede comenzar a utilizar Amazon Textract gratuitamente. Para obtener más información, consulte Capa gratuita de AWS.

Si ya dispone de una cuenta de AWS, pase a la siguiente tarea. Si todavía no dispone de una cuenta de AWS, realice los pasos del siguiente procedimiento para crear una.

Para crear una cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones en línea.

   Parte del procedimiento de inscripción consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

Anote su ID de cuenta de AWS porque lo necesitará en la siguiente tarea.

Creación de un usuario de IAM

Para tener acceso a los servicios de AWS, como Amazon Textract, debe proporcionar credenciales. Esto es así para que el servicio puede determinar si usted tiene permisos para obtener acceso a los recursos que son propiedad de tal servicio. La consola requiere que especifique la contraseña. Puede crear claves de acceso para su cuenta de AWS para tener acceso a la AWS CLI o API. Sin embargo, no es recomendable que acceda AWS con las credenciales de su cuenta de AWS. Le recomendamos que utilice en su lugar:

- UsarAWS Identity and Access Management(IAM) para crear un usuario de IAM.
- Añada el usuario a un grupo de IAM con permisos administrativos.

A continuación, podrá obtener acceso a AWS mediante una dirección URL especial y esas credenciales de usuario de IAM.

Si se ha inscrito en AWS pero no ha creado un usuario de IAM, puede crear uno mediante la consola de IAM. Siga el procedimiento para crear un usuario de IAM en su cuenta.

Para crear un usuario de IAM e iniciar sesión en la consola

1. Cree un usuario de IAM con permisos de administrador en su cuenta de AWS. Para obtener instrucciones, consulteCreación del primer grupo de usuarios y administradores de IAMen laIAM User Guide.
2. Como usuario de IAM, puede iniciar sesión en la AWS Management Console con una URL especial. Para obtener más información, consulteCómo inician sesión los usuarios en la cuentaen laIAM User Guide.

> **Note**
>
> Un usuario de IAM con permisos de administrador tiene acceso ilimitado alAWSservicios de tu cuenta. En los ejemplos de código de esta guía se presupone que tiene un usuario con la`AmazonTextractFullAccess`permisos.`AmazonS3ReadOnlyAccess`se requiere para ver ejemplos que tienen acceso a los documentos que se almacenan en un bucket Amazon S3. En función de sus requisitos de seguridad, es posible que desee utilizar un grupo de IAM que se limite a estos permisos. Para obtener más información, consulteCreación de grupos de IAM.

Para obtener más información sobre IAM, consulte lo siguiente:

- AWS Identity and Access Management (IAM)
- Introducción
- Guía del usuario de IAM

Paso siguiente

Paso 2: Configurar laAWS CLIyAWSSDK de

Paso 2: Configurar laAWS CLIyAWSSDK de

En los pasos siguientes se muestra cómo instalar la AWS Command Line Interface (AWS CLI) y los AWS SDK que utilizan los ejemplos de esta documentación.

Hay una serie de diferentes maneras para autenticar las llamadas al AWS SDK. En los ejemplos que aparecen en esta guía se presupone que utiliza un perfil de credenciales predeterminado para llamar a los comandos de la AWS CLI y las operaciones de API de AWS SDK. Sus credenciales predeterminadas funcionarán en todos los servicios, por lo que si ya ha configurado sus credenciales, no tendrá que volver a hacerlo. Sin embargo, si desea crear otro conjunto de credenciales para este servicio, puede crear un perfil de nombres. Para obtener más información acerca de la creación de perfiles,consulte Perfiles con nombres.

Para ver una lista de disponiblesAWSRegiones, consulteRegiones y puntos de enlace deen laReferencia general de Amazon Web Services.

Para configurar la AWS CLI y los AWS SDK

1. Descargue e instale la AWS CLI y los SDK de AWS que desea utilizar. En esta guía se ofrecen ejemplos para laAWS CLI, Java y Python. Para obtener más información sobre otros SDK de AWS, consulte Herramientas para Amazon Web Services.

   - AWS CLI
   - AWS SDK for Java
   - AWS SDK for Python (Boto3)

2. Cree una clave de acceso para el usuario que ha creado enCreación de un usuario de IAM.

   a. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.

   b. En el panel de navegación, seleccione Users.

   c. Elija el nombre del usuario que ha creado enCreación de un usuario de IAM.

   d. Seleccione la pestaña de credenciales de seguridad.

   e. Elija Create access key (Crear clave de acceso). A continuación, elija Download .csv file (Descargar archivo .csv) para guardar el ID de clave de acceso y la clave de acceso secreta en un archivo CSV de su equipo. Guarde el archivo en un lugar seguro. No podrá obtener acceso de nuevo a la clave de acceso secreta cuando este cuadro de diálogo se cierre. Cuando descargue el archivo CSV, elijaCerrar.

3. Configure las credenciales en el archivo de perfil de credenciales de AWS del sistema local, situado en:

   - `~/.aws/credentials`en Linux, macOS o Unix.
   - `C:\Users\USERNAME\.aws\credentials`en Windows.

   La`.aws`no existe antes de la primera configuración inicial de la instancia de AWS. La primera vez que configure sus credenciales con la CLI, se creará esta carpeta. Para obtener más información acerca de las credenciales de AWS, consulteAjustes de los archivos de configuración y credenciales.

   Este archivo debe contener líneas con el siguiente formato:

```
[default]
aws_access_key_id = your_access_key_id
```

```
aws_secret_access_key = your_secret_access_key
```

Reemplace la ID de clave de acceso y la clave de acceso secretayour_access_key_idyyour_secret_access_key.

4. Defina la región de AWS predeterminada en AWS`config`en el sistema local, situado en:

   - `~/.aws/config`en Linux, macOS o Unix.
   - `C:\Users\USERNAME\.aws\config`en Windows.

   La`.aws`no existe antes de la primera configuración inicial de la instancia de AWS. La primera vez que configure sus credenciales con la CLI, se creará esta carpeta. Para obtener más información acerca de las credenciales de AWS, consulteAjustes de los archivos de configuración y credenciales.

   Este archivo debe contener las líneas siguientes:

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

   Reemplace la región de AWS que desee (por ejemplo «us-west-2") portu_aws_region.

   > **Note**
   >
   > Si no elige una región, se usa us-east-1 de forma predeterminada.

Paso siguiente

Paso 3: Introducción al uso de laAWS CLIyAWSSDK API

Paso 3: Introducción al uso de laAWS CLIyAWSSDK API

Después de configurar elAWS CLIyAWSLos SDK que desea utilizar, puede compilar aplicaciones que usen Amazon Textract. En los siguientes temas, se muestra cómo comenzar a utilizar Amazon Textract.

- Análisis del texto del documento con Amazon Textract

Formato de los ejemplos de AWS CLI

Los ejemplos de AWS CLI incluidos en esta guía tienen formato para el sistema operativo Linux. Para utilizar los ejemplos con Microsoft Windows, deberá cambiar el formato JSON del parámetro `--document`, así como los saltos de línea de barras diagonales inversas (\) por signos de intercalación (^). Para obtener más información sobre el formato JSON, consulte Especificación de valores de parámetros para la AWS Command Line Interface.

Procesamiento de documentos con operaciones síncronas

Amazon Textract Texact puede detectar y analizar texto en documentos de una sola página que se proporcionan como imágenes en formato JPEG, PNG, PDF y TIFF. Las operaciones son sincrónicas y devuelven los resultados casi en tiempo real. Para obtener más información sobre documentos, consulte Objetos de respuesta de detección de texto y análisis de documentos.

En esta sección se explica cómo puede utilizar Amazon Textract para detectar y analizar el texto de un documento de una sola página de forma sincrónica. Para detectar y analizar texto en documentos de varias páginas o para detectar documentos JPEG y PNG de forma asíncrona, consulteProcesamiento de documentos con operaciones asíncronas.

Puede utilizar operaciones síncronas de Amazon Textract para lo siguiente:

- Detección de texto: puede detectar líneas y palabras en una imagen de documento de una sola página mediante elDetectDocumentText. Para obtener más información, consulte Detección de texto.
- Análisis de texto: puede identificar las relaciones entre el texto detectado en un documento de una sola página mediante elAnalyzeDocument. Para obtener más información, consulte Análisis de documentos.
- Análisis de facturas y recibos: puede identificar las relaciones financieras entre el texto detectado en una factura o un recibo de una sola página mediante la operación AnalyzeExpense. Para obtener más información, consulteAnálisis de facturas y recibos
- Análisis de documentos de identidad: puede analizar los documentos de identidad emitidos por el gobierno de EE. UU. y extraer información junto con los tipos comunes de información que se encuentran en los documentos de identidad. Para obtener más información, consulteAnálisis de documentos de identidad.

Temas

- Llamar a operaciones sincrónicas de Amazon Textract
- Detección de texto de documento con Amazon Textract
- Análisis del texto del documento con Amazon Textract
- Análisis de facturas y recibos con Amazon Textract
- Análisis de la documentación de identidad con Amazon Textract

Llamar a operaciones sincrónicas de Amazon Textract

Las operaciones de Amazon Textract procesan imágenes de documentos que se almacenan en un sistema de archivos local o imágenes de documentos almacenadas en un depósito de Amazon S3. Especifique dónde se encuentra el documento de entrada mediante elDocumentparámetro de entrada. La imagen del documento puede estar en formato PNG, JPEG, PDF o TIFF. Los resultados de las operaciones sincrónicas se devuelven inmediatamente y no se almacenan para recuperarlos.

Para obtener un ejemplo completo, consulteDetección de texto de documento con Amazon Textract.

Solicitud

A continuación se describe cómo funcionan las solicitudes en Amazon Textract.

Documentos pasados como bytes de imagen

Puede pasar una imagen de documento a una operación de Amazon Textract pasando la imagen como matriz de bytes codificada en base64. Un ejemplo es una imagen de documento cargada desde un sistema de archivos local. Es posible que el código no necesite codificar bytes de archivos de documentos si está utilizando unAWSSDK para llamar a las operaciones de la API Amazon Textract Texact.

Los bytes de imagen se especifican en el`Bytes`del`Document`parámetro de entrada. En el siguiente ejemplo se muestra el JSON de entrada para una operación de Amazon Textract Texact que pasa los bytes de imagen en el`Bytes`parámetro de entrada.

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

> **Note**
>
> Si utilizaAWS CLI, no puedes pasar bytes de imagen a las operaciones de Amazon Textract. En su lugar, debe hacer referencia a una imagen almacenada en un bucket de Amazon S3.

El siguiente código de Java muestra cómo cargar una imagen desde un sistema de archivos local y llamar a una operación de Amazon Textract.

```
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);
```

Documentos almacenados en un bucket de Amazon S3

Amazon Textract puede analizar imágenes de documentos almacenadas en un bucket de Amazon S3. Puede especificar el bucket y el nombre del archivo mediante laS3ObjectdelDocumentparámetro de entrada. El siguiente ejemplo de muestra el JSON de entrada de una operación de Amazon Textract que procesa un documento almacenado en un bucket Amazon S3.

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

El siguiente ejemplo de muestra cómo llamar a una operación de Amazon Textract utilizando una imagen almacenada en un bucket Amazon S3.

```
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);
```

Respuesta

El ejemplo siguiente es la respuesta JSON de una llamada a`DetectDocumentText`. Para obtener más información, consulte Detección de texto.

```
{
{
  "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
  }
}
}
```


Detección de texto de documento con Amazon Textract

Para detectar texto de un documento, utilice elDetectDocumentTexty pasa un archivo de documento como entrada.`DetectDocumentText`devuelve una estructura JSON que contiene líneas y palabras del texto detectado, la ubicación del texto en el documento y las relaciones entre el texto detectado. Para obtener más información, consulte Detección de texto.

Puede proporcionar un documento de entrada como matriz de bytes de imagen (bytes de imagen con codificación en base64) o como objeto Amazon S3. En este procedimiento cargará un archivo de imagen en su bucket de S3; y especificará el nombre de archivo.

Para detectar texto en un documento (API de)

1. Si aún no lo ha hecho:

   a. Crear o actualizar un usuario de IAM con`AmazonTextractFullAccess`y`AmazonS3ReadOnlyAccess`permisos. Para obtener más información, consulte Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM.

   b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte Paso 2: Configurar laAWS CLIyAWSSDK de.

2. Cargue un documento en el bucket de S3.

   Para obtener instrucciones, consulteCarga de objetos en Amazon S3en laAmazon Simple Storage Service Manual del usuario.

3. Utilice los siguientes ejemplos para llamar a la operación `DetectDocumentText`.

   Java

   El código de ejemplo siguiente muestra el documento y los cuadros alrededor de las líneas del texto detectado.

   En la función`main`, sustituya los valores de`bucket`y`document`con los nombres del bucket de Amazon S3 y del documento que utilizó en el paso 2.

```
//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

Este comando de la AWS CLI muestra la salida de JSON para la operación `detect-document-text` de la CLI.

Sustituir los valores de`Bucket`y`Name`con los nombres del bucket de Amazon S3 y del documento que utilizó en el paso 2.

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

Python

El siguiente código de ejemplo muestra el documento y los cuadros alrededor de líneas de texto detectadas.

En la funciónmain, sustituya los valores debucketydocumentcon los nombres del bucket de Amazon S3 y del documento que utilizó en el paso 2.

```
#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

El siguiente código de ejemplo de Node.js muestra el documento y los cuadros alrededor de las líneas de texto detectadas, lo que muestra una imagen de los resultados en el directorio desde el que ejecuta el código. Hace uso de`image-size`y`images`paquetes.

En la función`main`, sustituya los valores de`bucket`y`document`con los nombres del bucket de Amazon S3 y del documento que utilizó en el paso 2. Sustituir el valor de`regionConfig`con el nombre de la región en la que se encuentra tu cuenta.

```
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. Ejecute el ejemplo. Los ejemplos de Python y Java muestran la imagen del documento. Un cuadro negro rodea cada línea de texto detectado. Una línea vertical verde es el principio de una palabra detectada. Una línea vertical roja es el final de una palabra detectada. LaAWS CLImuestra solo la salida de JSON para la`DetectDocumentText`.

Análisis del texto del documento con Amazon Textract

Para analizar el texto de un documento, utilice elAnalyzeDocumenty pasa un archivo de documento como entrada.`AnalyzeDocument`devuelve una estructura JSON que contiene el texto analizado. Para obtener más información, consulte Análisis de documentos.

Puede proporcionar un documento de entrada como matriz de bytes de imagen (bytes de imagen con codificación en base64) o como objeto Amazon S3. En este procedimiento cargará un archivo de imagen en su bucket de S3; y especificará el nombre de archivo.

Para analizar el texto de un documento (API)

1. Si aún no lo ha hecho:

   a. Crear o actualizar un usuario de IAM con`AmazonTextractFullAccess`y`AmazonS3ReadOnlyAccess`permisos. Para obtener más información, consulte Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM.

   b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte Paso 2: Configurar laAWS CLIyAWSSDK de.

2. Cargue una imagen que contenga un documento en su bucket de S3.

   Para obtener instrucciones, consulteCarga de objetos en Amazon S3en laAmazon Simple Storage Service Manual del usuario.

3. Utilice los siguientes ejemplos para llamar a la operación `AnalyzeDocument`.

   Java

   El siguiente código de ejemplo muestra el documento y los cuadros alrededor de los elementos detectados.

   En la función`main`, sustituya los valores de`bucket`y`document`con los nombres del bucket de Amazon S3 e imagen de documento que utilizó en el paso 2.

```
//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

Este comando de la AWS CLI muestra la salida de JSON para la operación `detect-document-text` de la CLI.

Sustituir los valores de`BucketyName`con los nombres del bucket de Amazon S3 y del documento que utilizó en el paso 2.

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

Python

El siguiente código de ejemplo muestra el documento y los cuadros alrededor de los elementos detectados.

En la funciónmain, sustituya los valores debucketydocumentcon los nombres del bucket de Amazon S3 y del documento que utilizó en el paso 2.

```
#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

El siguiente código de ejemplo muestra el documento y los cuadros alrededor de los elementos detectados.

En el siguiente código, sustituya los valores de`bucket`y`photo`con los nombres del bucket de Amazon S3 y del documento que utilizó en el paso 2. Sustituir el valor de`region`con la región asociada a la cuenta de.

```
// 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. Ejecute el ejemplo. Los ejemplos de Python y Java muestran la imagen del documento con los siguientes cuadros delimitadores de colores:

   - Rojo — Objetos KEY Block
   - Verde — VALUE Bloquear objetos
   - Azul — TABLA Bloquear objetos
   - Amarillo — objetos CELL Block

   Los elementos de selección seleccionados se rellenan de azul.

   LaAWS CLImuestra solo la salida de JSON para la`AnalyzeDocument`.

Análisis de facturas y recibos con Amazon Textract

Para analizar documentos de factura y recibo, utiliza la API AnalyzeExpense y pasa un archivo de documento como entrada.`AnalyzeExpense`es una operación sincrónica que devuelve una estructura JSON que contiene el texto analizado. Para obtener más información, consulte Análisis de facturas y recibos.

Para analizar facturas y recibos de forma asíncrona, utiliceStartExpenseAnalysispara empezar a procesar un archivo de documento de entrada y utilizarGetExpenseAnalysispara obtener los resultados.

Puede proporcionar un documento de entrada como matriz de bytes de imagen (bytes de imagen con codificación en base64) o como objeto Amazon S3. En este procedimiento cargará un archivo de imagen en su bucket de S3; y especificará el nombre de archivo.

Para analizar una factura o un recibo (API)

1. Si aún no lo ha hecho:

   a. Crear o actualizar un usuario de IAM conAmazonTextractFullAccessyAmazonS3ReadOnlyAccesspermisos. Para obtener más información, consulte Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM.

   b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte Paso 2: Configurar laAWS CLIyAWSSDK de.

2. Cargue una imagen que contenga un documento en su bucket de S3.

   Para obtener instrucciones, consulteCarga de objetos en Amazon S3en laAmazon Simple Storage Service Manual del usuario.

3. Utilice los siguientes ejemplos para llamar a la operación AnalyzeExpense.

   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. Esto le proporcionará la salida de JSON para la`AnalyzeExpense`.

Análisis de la documentación de identidad con Amazon Textract

Para analizar documentos de identidad, utiliza la API AnalyzeID y pasa un archivo de documento como entrada.`AnalyzeID`devuelve una estructura JSON que contiene el texto analizado. Para obtener más información, consulte Análisis de documentos de identidad.

Puede proporcionar un documento de entrada como matriz de bytes de imagen (bytes de imagen con codificación en base64) o como objeto Amazon S3. En este procedimiento cargará un archivo de imagen en su bucket de S3; y especificará el nombre de archivo.

Para analizar un documento de identidad (API)

1. Si aún no lo ha hecho:

   a. Crear o actualizar un usuario de IAM con`AmazonTextractFullAccessy`AmazonS3ReadOnlyAccesspermisos. Para obtener más información, consulte Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM.

   b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte Paso 2: Configurar laAWS CLIyAWSSDK de.

2. Cargue una imagen que contenga un documento en su bucket de S3.

   Para obtener instrucciones, consulteCarga de objetos en Amazon S3en laAmazon Simple Storage Service Manual del usuario.

3. Utilice los siguientes ejemplos para llamar a la operación `AnalyzeID`.

   CLI

   En el siguiente ejemplo, se toma un archivo de entrada de un bucket de S3; y ejecuta la`AnalyzeID`Operación en él. En el siguiente código, sustituya el valor de*`balde`*con el nombre de su bucket de S3, el valor de*`archivo`*con el nombre del archivo del bucket y el valor de*`región`*con el nombre del`region`asociada a la cuenta de.

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

   También puedes llamar a la API con la parte delantera y trasera de una licencia de conducir añadiendo otro objeto S3 a la entrada.

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

Si está accediendo a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y escapa de las comillas dobles internas mediante barra invertida (es decir, \) para corregir cualquier error de analizador que pueda surgir. Para un ejemplo, consulte a continuación:

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

Python

En el siguiente ejemplo, se toma un archivo de entrada de un bucket de S3; y ejecuta la`AnalyzeID`en él, devolviendo los pares clave-valor detectados. En el siguiente código, sustituya el valor de*bucket_name*con el nombre de su bucket de S3, el valor de*file_name*con el nombre del archivo del bucket y el valor de*región*con el nombre del`region`asociada a la cuenta de.

```
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

En el siguiente ejemplo, se toma un archivo de entrada de un bucket de S3; y ejecuta la`AnalyzeID`Operación en él, devolviendo los datos detectados. En la función principal, reemplace los valores des3bucketysourceDoccon los nombres del bucket de Amazon S3 e imagen de documento que utilizó en el paso 2.

```
/*
   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. Esto le proporcionará la salida de JSON para la`AnalyzeID`.

Procesamiento de documentos con operaciones asíncronas

Amazon Textract Texact puede detectar y analizar texto en documentos de varias páginas en formato PDF o TIFF. Esto incluye facturas y recibos. El procesamiento de documentos de varias páginas es asíncrona. El procesamiento asíncrono de documentos resulta útil para procesar documentos grandes de varias páginas. Por ejemplo, un archivo PDF con más de 1.000 páginas tarda un tiempo en procesarse. El procesamiento del archivo PDF de forma asíncrona permite que la aplicación complete otras tareas mientras espera a que se complete el proceso.

En esta sección se explica cómo puede utilizar Amazon Textract para detectar y analizar texto de forma asíncrona en un documento de varias páginas o de una sola página. Los documentos de varias páginas deben estar en formato PDF o TIFF. Los documentos de una sola página procesados con operaciones asíncronas pueden estar en formato JPEG, PNG, TIFF o PDF.

Puede utilizar operaciones asíncronas de Amazon Textract Texact para lo siguiente:

- Detección de texto: puede detectar líneas y palabras en un documento de varias páginas. Las operaciones asíncronas sonStartDocumentTextDetectionyGetDocumentTextDetection. Para obtener más información, consulte Detección de texto.
- Análisis de texto: puede identificar las relaciones entre el texto detectado en un documento de varias páginas. Las operaciones asíncronas sonStartDocumentAnalysisyGetDocumentAnalysis. Para obtener más información, consulte Análisis de documentos.
- Análisis de gastos: puede identificar las relaciones de datos en facturas y recibos de varias páginas. Amazon Textract Texact trata cada factura o página de recepción de un documento de varias páginas como un recibo individual o una factura. No conserva el contexto de una página a otra de un documento de varias páginas. Las operaciones asíncronas sonStartExpenseAnalysisyGetExpenseAnalysis. Para obtener más información, consulte Análisis de facturas y recibos.

Temas

Llamar a operaciones asíncronas de Amazon Textract

Amazon Textract Texact proporciona una API asíncrona que puede utilizar para procesar documentos de varias páginas en formato PDF o TIFF. También puede utilizar operaciones asíncronas para procesar documentos de una sola página en formato JPEG, PNG, TIFF o PDF.

La información de este tema utiliza operaciones de detección de texto para mostrar cómo utilizar las operaciones asíncronas de Amazon Textract Texact. El mismo enfoque funciona con las operaciones de análisis de textos dethe section called “StartDocumentAnalysis”ythe section called “GetDocumentAnalysis”. También funciona igual conthe section called “StartExpenseAnalysis”ythe section called “GetExpenseAnalysis”.

Para ver un ejemplo, consulte Detección o análisis de texto en un documento de varias páginas.

Amazon Textract procesa de forma asíncrona un documento almacenado en un bucket de Amazon S3. Empiezas a procesar llamando a un`Start`operación, tales comoStartDocumentTextDetection. El estado de realización de la solicitud se publica en un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener el estado de realización del tema de Amazon SNS, puede utilizar una cola de Amazon Simple Queue Service (Amazon SQS) o unaAWS Lambdafunción. Una vez que disponga del estado de realización, llame a una operación `Get`, como GetDocumentTextDetection, para obtener los resultados de la solicitud.

Los resultados de las llamadas asíncronas se cifran y almacenan durante 7 días en un bucket propiedad de Amazon Textract Texact de forma predeterminada, a menos que especifique un bucket de Amazon S3 mediante una operación`OutputConfig`argumento.

En la tabla siguiente se muestran las operaciones Start y Get correspondientes para los distintos tipos de procesamiento asíncrono admitidos por Amazon Textract:

Iniciar/obtener operaciones de API para operaciones asíncronas de Amazon Textract

Tipo de procesamiento	API de inicio	Obtenga la API
Detección de texto	StartDocumentTextDetection	GetDocumentTextDetection
Análisis de texto	StartDocumentAnalysis	GetDocumentAnalysis
Análisis de gastos	Iniciar análisis de gastos	Obtener análisis de gastos

Para un ejemplo que utilizaAWS Lambdafunciones, consulteProcesamiento de documentos a gran escala con Amazon Textract.

En el siguiente diagrama se muestra el proceso para detectar el texto de un documento en una imagen de documento almacenada en un bucket de Amazon S3. En el diagrama, una cola de Amazon SQS obtiene el estado de realización a partir del tema de Amazon SNS.

El proceso que muestra el diagrama anterior es el mismo para analizar texto y facturas/recibos. Empiezas a analizar texto llamandothe section called “StartDocumentAnalysis”y comienza a analizar facturas/recibos llamandothe section called “StartExpenseAnalysis”Obtienes los resultados llamandothe section called “GetDocumentAnalysis”othe section called “GetExpenseAnalysis”respectivamente.

Iniciar la detección de texto

Para iniciar una solicitud de detección de texto de Amazon Textract, llame aStartDocumentTextDetection. El siguiente es un ejemplo de una solicitud JSON que ha transferido `StartDocumentTextDetection`.

```
{
    "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"
}
```

El parámetro de entrada`DocumentLocation`proporciona el nombre de archivo de documento y el bucket de Amazon S3 desde el que recuperarlo.`NotificationChannel`contiene el nombre de recurso de Amazon (ARN) del tema de Amazon SNS que Amazon Textract notifica cuando finaliza la solicitud de detección de texto. El tema de Amazon SNS debe estar en la misma región de AWS que el punto de enlace Amazon Textract Texact al que está llamando.`NotificationChannel`también

contiene el ARN de un rol que permite a Amazon Textract Texact publicar en el tema de Amazon SNS. Puede conceder permisos de Amazon Textract Texact a sus temas de Amazon SNS creando un rol de servicio de IAM. Para obtener más información, consulte Configuración de Amazon Textract Texact para operaciones asíncronas.

También puede especificar un parámetro de entrada opcional,`JobTag`, que le permite identificar el trabajo o los grupos de trabajos en el estado de realización que se ha publicado en el tema de Amazon SNS. Por ejemplo, puede utilizar`JobTag`para identificar el tipo de documento que se está procesando, como un formulario fiscal o un recibo.

Para evitar la duplicación accidental de trabajos de análisis, tiene la opción de proporcionar un token idempotente, `ClientRequestToken`. Si proporciona un valor para`ClientRequestToken`, el`Start`La operación devuelve lo mismo`JobId`para varias llamadas idénticas al`Start`operación, tales como`StartDocumentTextDetection`. Un token `ClientRequestToken` tiene una vida útil de 7 días. Después de 7 días, puede volver a utilizarla. Si reutiliza el token durante el ciclo de vida del token, sucede lo siguiente:

- Si reutiliza el token con la misma operación `Start` y los mismos parámetros de entrada, se devuelve el mismo `JobId`. El trabajo no se vuelve a realizar de nuevo y Amazon Textract no envía un estado de realización al tema de Amazon SNS registrado.
- Si vuelve a utilizar el token con la misma operación `Start` y un cambio de parámetro de entrada menor, obtendrá una excepción `idempotentparametermismatchexception` (código de estado HTTP: 400).
- Si reutiliza el token con otra operación `Start` distinta, la operación se realiza correctamente.

Otro parámetro opcional disponible es`OutputConfig`, que le permite ajustar dónde se colocará la salida. De forma predeterminada, Amazon Textract Texact almacenará los resultados internamente y solo se puede acceder a ellos mediante las operaciones de Get API. con`OutputConfig`habilitado, puede establecer el nombre del depósito al que se enviará la salida y el prefijo de archivo de los resultados, donde puede descargar los resultados. También puede configurar la`KMSKeyID`parámetro a una clave administrada por el cliente para cifrar la salida. Sin este conjunto de parámetros, Amazon Textract Texact cifrará el lado del servidor mediante elClave administrada de AWSpara Amazon S3

> **Note**
>
> Antes de utilizar este parámetro, asegúrese de tener el permiso PutObject para el bucket de salida. Además, asegúrese de tener los permisos Descifrar, ReEncrypt, GenerateDataKey y DescribeKey para elAWS KMSclave si decides usarla.

La respuesta a la operación `StartDocumentTextDetection` es un identificador de trabajo (`JobId`). Usar`JobId`para realizar un seguimiento de las solicitudes y obtener los resultados de análisis después de que Amazon Textract haya publicado el estado de realización en el tema de Amazon SNS. A continuación se muestra un ejemplo:

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

Si empiezas demasiados trabajos simultáneamente, llama a`StartDocumentTextDetection`RAISE UN`LimitExceededException`excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Textract Texact.

Si descubre que las excepciones LimitExceededException se producen con picos de actividad, considere la posibilidad de usar una cola de Amazon SQS para administrar las solicitudes entrantes. ContactoAWSSupport técnico si descubre que el número medio de solicitudes simultáneas no se puede administrar con una cola de Amazon SQS y sigue recibiendo`LimitExceededException`excepciones.

Obtención del estado de realización de una solicitud de Amazon Textract Analysis

Amazon Textract envía una notificación a la realización de análisis al tema de Amazon SNS registrado. La notificación incluye el identificador de trabajo y el estado de realización de la operación en una cadena de JSON. Una solicitud de detección de texto correcta tiene un`SUCCEEDED`estado. Por ejemplo, el siguiente resultado muestra el procesamiento correcto de un trabajo de detección de texto.

```
{
    "JobId": "642492aea78a86a40665555dc375ee97bc963f342b29cd05030f19bd8fd1bc5f",
    "Status": "SUCCEEDED",
```

```
    "API": "StartDocumentTextDetection",
    "JobTag": "Receipt",
    "Timestamp": 1543599965969,
    "DocumentLocation": {
        "S3ObjectName": "document",
        "S3Bucket": "bucket"
    }
}
```

Para obtener más información, consulte Notificación de resultados de Amazon Textract.

Para obtener la información de estado que Amazon Textract ha publicado en el tema de Amazon SNS, utilice una de las siguientes opciones:

- AWS Lambda— Puede suscribirse a unAWS Lambdaque escribe en un tema de Amazon SNS. Se llama a la función cuando Amazon Textract notifica al tema de Amazon SNS que la solicitud se ha completado. Utilice una función Lambda si desea que el código del servidor procese los resultados de una solicitud de detección de texto. Por ejemplo, es posible que desee utilizar el código del servidor para anotar la imagen o crear un informe sobre el texto detectado antes de devolver la información a una aplicación cliente.
- Amazon SQS— Puede suscribir una cola de Amazon SQS a un tema de Amazon SNS. A continuación, sondee la cola de Amazon SQS para recuperar el estado de realización publicado por Amazon Textract cuando se completa una solicitud de detección de texto. Para obtener más información, consulte Detección o análisis de texto en un documento de varias páginas. Utilice una cola de Amazon SQS si desea llamar a operaciones de Amazon Textract Texact solo desde una aplicación cliente.

> **Important**
>
> No le recomendamos obtener el estado de realización de solicitud llamando repetidamente a Amazon Textract`Get`. Esto se debe a que Amazon Textract Texact limita el`Get`si se realizan demasiadas solicitudes. Si está procesando varios documentos al mismo tiempo, es más sencillo y más eficaz supervisar una cola de SQS para la notificación de realización que sondear Amazon Textract detectar el estado de cada trabajo individualmente.

Obtener resultados de detección de texto de Amazon Textract

Para obtener los resultados de una solicitud de detección de texto, en primer lugar, asegúrese de que el estado de realización que se ha recuperado del tema de Amazon SNS es`SUCCEEDED`. A continuación, llame a `GetDocumentTextDetection`, que transfiere el valor `JobId` que se devuelve desde `StartDocumentTextDetection`. El JSON de la solicitud es similar al siguiente ejemplo:

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

`JobId`es el identificador de la operación de detección de texto. Como la detección de texto puede generar grandes cantidades de datos, utilice`MaxResults`para especificar el número máximo de resultados que se devuelven en un solo`Get`. El valor predeterminado de`MaxResults`es 1000. Si especifica un valor superior a 1 000, solo se devuelven 1 000 resultados. Si la operación no devuelve todos los resultados, se devuelve un token de paginación para la página siguiente. Para obtener la siguiente página de resultados, especifique el token en el`NextToken`parámetro.

> **Note**
>
> Amazon Textract Texact conserva los resultados de operaciones asíncronas durante siete días. No puede recuperar los resultados transcurrido este plazo.

La`GetDocumentTextDetection`JSON de respuesta de operación es similar al siguiente. El número total de páginas detectadas se devuelve en`DocumentMetadata`. El texto detectado se devuelve en el`Blocks`matriz. Para obtener información sobre`Block`objetos, consulte[Objetos de respuesta de detección de texto y análisis de documentos](#).

```
{
    "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
        }
    ]
}
```

Configuración de Amazon Textract Texact para operaciones asíncronas

En los siguientes procedimientos se muestra cómo configurar Amazon Textract para que lo utilice con un tema de Amazon Simple Notification Service (Amazon SNS) y una cola de Amazon Simple Queue Service (Amazon SQS).

> **Note**
>
> Si está utilizando estas instrucciones para configurar elDetección o análisis de texto en un documento de varias páginasejemplo, no es necesario realizar los pasos 3 a 6. El ejemplo incluye código que permite crear y configurar el tema de Amazon SNS y la cola de Amazon SQS.

Para configurar Amazon Textract

1. Configurar unAWScuenta para acceder a Amazon Textract. Para obtener más información, consulte Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM.

   Asegúrese de que el usuario tiene al menos los permisos siguientes:

   - AmazonTextractFullAccess
   - AmazonS3ReadOnlyAccess
   - AmazonSNSFullAccess
   - AmazonSQSFullAccess
2. Instale y configure el SDK de AWS necesario. Para obtener más información, consulte Paso 2: Configurar laAWS CLIyAWSSDK de.
3. Cree un tema de Amazon SNS.. Anexa el nombre del tema conExtracto de Amazon. Anote el nombre de recurso de Amazon (ARN) del tema. Asegúrese de que el tema de está en la misma región de que elAWSpunto de enlace que está utilizando con su cuenta de AWS.
4. Cree una cola estándar de Amazon SQSmediante el uso de laConsola de Amazon SQS. Anote el ARN de la cola.
5. Suscriba la cola al tema que creó en el paso 3.
6. Conceder permiso al tema de Amazon SNS para enviar mensajes a la cola de Amazon SQS.
7. Cree una función de servicio de IAM para dar a Amazon Textract Texact acceso a sus temas de Amazon SNS. Anote el nombre de recurso de Amazon (ARN) del rol de servicio. Para obtener más información, consulte Conceder acceso a Amazon Textract a su tema de Amazon SNS.
8. Añada la siguiente política insertadaal usuario de IAM que ha creado en el paso 1.

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

Asigne un nombre a la política insertada.

9. Ahora puede ejecutar los ejemplos deDetección o análisis de texto en un documento de varias páginas.

Conceder acceso a Amazon Textract a su tema de Amazon SNS

Amazon Textract Texact necesita permiso para enviar un mensaje al tema de Amazon SNS cuando se haya completado una operación asíncrona. Utilice un rol de servicio de IAM para dar a Amazon Textract Texact acceso al tema de Amazon SNS.

Al crear el tema de Amazon SNS, debe añadir el nombre del tema con**`AmazonTextract`**—por ejemplo,**`AmazonTextractMyTopicName`**.

1. Inicie sesión en la consola de IAM (https://console.aws.amazon.com/iam).
2. Seleccione Roles en el panel de navegación.
3. Elija Create role.
4. En Select type of trusted entity (Seleccionar tipo de entidad de confianza), elija AWS service (Servicio de AWS).
5. ParaElija el servicio que utilizará este rol, eligeTextract.
6. Seleccione Next (Siguiente): Permisos.
7. Verifique que laAmazonTextractServiceRolese ha incluido en la lista de políticas adjuntas. Para mostrar la política en la lista, escriba parte del nombre de la política en elPolíticas de filtrado.
8. Seleccione Next (Siguiente): Tags (Etiquetas).
9. No es necesario añadir etiquetas, así que elijaSiguiente: Consulte.
10. En la sección Review (Revisar), en Role Name (Nombre de rol), escriba un nombre para el rol (por ejemplo, `TextractRole`). EnDescripción del rol, actualice la descripción de la función y, a continuación, elijaCrear rol.
11. Elija el rol nuevo para abrir la página de detalles del rol.
12. En Summary (Resumen), copie el valor de Role ARN (ARN del rol) y guárdelo.
13. Seleccione Trust Relationships.
14. ElegirModificar relación de confianzay garantizar que la política de fideicomiso tenga el siguiente aspecto.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "textract.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

15. Elija Update Trust Policy (Actualizar política de confianza).

Detección o análisis de texto en un documento de varias páginas

Este procedimiento muestra cómo detectar o analizar texto en un documento de varias páginas mediante operaciones de detección de Amazon Textract Texact, un documento almacenado en un depósito de Amazon S3, un tema de Amazon SNS y una cola de Amazon SQS. El procesamiento de documentos de varias páginas es asíncrona. Para obtener más información, consulte Llamar a operaciones asíncronas de Amazon Textract.

Puede elegir el tipo de procesamiento que desea que realice el código: detección de texto, análisis de texto o análisis de gastos.

Los resultados del procesamiento se devuelven en una matriz dethe section called “Block”objetos que varían según el tipo de procesamiento que utilice.

Para detectar texto o analizar documentos de varias páginas, haga lo siguiente:

1. Cree el tema de Amazon SNS y la cola de Amazon SQS.
2. Suscriba la cola de al tema de.
3. Conceda permiso al tema de para enviar mensajes a la cola de.
4. Comience a procesar el documento. Utilice la operación adecuada para el tipo de análisis elegido:
   - StartDocumentTextDetectionpara tareas de detección de texto.
   - StartDocumentAnalysispara tareas de análisis de texto.
   - StartExpenseAnalysispara tareas de análisis de gastos.
5. Obtenga el estado de realización a partir de la cola de Amazon SQS. El código de ejemplo hace un seguimiento del identificador del trabajo (`JobId`) devuelto por el`Start`. Solo obtiene los

resultados de identificadores de trabajo coincidentes que se leen desde el estado de realización. Esto es importante si otras aplicaciones están utilizando la misma cola de y tema de. Para simplificar, en el ejemplo se eliminan los trabajos que no coinciden. Plantéese añadir los trabajos eliminados a una cola de mensajes erróneos de Amazon SQS para examinarlos posteriormente.

6. Obtenga y muestre los resultados del procesamiento llamando a la operación adecuada para el tipo de análisis elegido:
   - GetDocumentTextDetectionpara tareas de detección de texto.
   - GetDocumentAnalysispara tareas de análisis de texto.
   - GetExpenseAnalysispara tareas de análisis de gastos.
7. Elimine el tema de Amazon SNS y la cola de Amazon SQS.

Realización de operaciones asíncronas

El código de ejemplo de este procedimiento se proporciona en Java, Python yAWS CLI. Antes de comenzar, instale el correspondienteAWSSDK. Para obtener más información, consulte Paso 2: Configurar laAWS CLIyAWSSDK de.

Para detectar o analizar texto en un documento de varias páginas

1. Configure el acceso de usuario a Amazon Textract y configure el acceso de Amazon Textract Texact a Amazon SNS. Para obtener más información, consulte Configuración de Amazon Textract Texact para operaciones asíncronas. Para completar este procedimiento, se necesita un archivo de documento de varias páginas en formato PDF. Omita los pasos 3 a 6, ya que el código de ejemplo crea y configura el tema de Amazon SNS y la cola de Amazon SQS. Si es complejotEn el ejemplo de la CLI, no es necesario configurar una cola de SQS.
2. Cargue un archivo de documento de varias páginas en formato PDF o TIFF a su bucket de Amazon S3. (También se pueden procesar documentos de una sola página en formato JPEG, PNG, TIFF o PDF).

   Para obtener instrucciones, consulteCarga de objetos en Amazon S3en laAmazon Simple Storage Service.
3. Utilice lo siguienteAWS SDK for Java, SDK for Python (Boto3) oAWS CLIcódigo para detectar texto o analizar texto en un documento de varias páginas. En el navegador`main`función:

   - Sustituir el valor de`roleArn`con el ARN del rol de IAM que ha guardado enConceder acceso a Amazon Textract a su tema de Amazon SNS.

- Sustituir los valores de`bucket`y`document`con el nombre del bucket de y el nombre del archivo de documento que especificó en el paso 2.
- Sustituya el valor de la`type`parámetro de entrada del`ProcessDocument`función con el tipo de procesamiento que desea realizar. Usar`ProcessType.DETECTION`para detectar texto. Usar`ProcessType.ANALYSIS`para analizar texto.
- Para el ejemplo de Python, sustituya el valor de`region_name`con la región en la que opera su cliente.

Para el registro`AWS CLI`ejemplo, haga lo siguiente:

- Al llamar`StartDocumentTextDetection`, sustituya el valor de`bucket-name`con el nombre de su bucket de S3 y reemplace`file-name`con el nombre del archivo que especificó en el paso 2. Especifique la región de su depósito reemplazando`region-name`con el nombre de su región. Tenga en cuenta que el ejemplo de la CLI no utiliza SQS.
- Al llamar`GetDocumentTextDetection`replace`job-id-number`con`job-id`devuelto por`StartDocumentTextDetection`. Especifique la región de su depósito reemplazando`region-name`con el nombre de su región.

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

EsteAWS CLIinicia la detección asincrónica del texto de un documento especificado. Este método devuelve un objeto`job-id`que se puede utilizar para recuperar los resultados de la detección.

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

EsteAWS CLIdevuelve los resultados de una operación asíncrona de Amazon Textract cuando se proporciona con un`job-id`.

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

Si está accediendo a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y escapa de las comillas dobles internas mediante barra invertida (es decir, \) para corregir cualquier error de analizador que pueda surgir. Para un ejemplo, consulte lo siguiente

```
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

En este ejemplo, reemplace el valor de`roleArn`con el ARN del rol de IAM que ha guardado enConceder acceso a Amazon Textract a su tema de Amazon SNS. Sustituir los valores de`bucket`y`document`con el nombre del bucket de y el nombre del archivo de documento que especificó en el paso 2 anterior. Sustituir el valor de`processType`con el tipo de procesamiento que quieres utilizar en el documento de entrada. Por último, sustituya el valor de`REGION`con la región en la que opera su cliente.

```
 // 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. Ejecute el código. La operación podría llevar algún tiempo. Una vez terminada, se muestra una lista de bloques para el texto detectado o analizado.

Notificación de resultados de Amazon Textract

Amazon Textract publica los resultados de una solicitud de análisis de Amazon Textract, incluido el estado de realización, en un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener la notificación de un tema de Amazon SNS, utilice una cola de Amazon SQS o unAWS Lambdafunción. Para obtener más información, consulte Llamar a operaciones asíncronas de Amazon Textract. Para ver un ejemplo, consulte Detección o análisis de texto en un documento de varias páginas.

Los resultados tienen el siguiente formato JSON:

```
{
  "JobId": "String",
  "Status": "String",
  "API": "String",
  "JobTag": "String",
  "Timestamp": Number,
  "DocumentLocation": {
    "S3ObjectName": "String",
    "S3Bucket": "String"
  }
}
```

En esta tabla se describen los diferentes parámetros de una respuesta de Amazon Textract Texact.

Parámetro	Descripción
JobId	Identificador exclusivo que Amazon Textract asigna al trabajo. Coincide con un identificador de trabajo que devuelve desde un `Start` operación, tales como StartDocumentTextDetection.
Estado	El estado del trabajo. Los valores válidos son Correctos, Fallados o Error.
API	La operación de Amazon Textract utilizada para analizar el documento de entrada, como por ejemplo StartDocumentTextDetection o StartDocumentAnalysis.
JobTag	El identificador especificado por el usuario del trabajo. Especificas `JobTag` en una llamada al `Start` operación, tales como StartDocumentTextDetection.
Marca temporal	La marca de tiempo de Unix que indica cuándo finalizó el trabajo, se devuelve en milisegundos.
Ubicación del documento	Detalles sobre el documento que se procesó. Incluye el nombre de archivo y el bucket de Amazon S3 en el que está almacenado el archivo.

Gestión de llamadas limitadas y conexiones interrumpidas

Una operación de Amazon Textract Texact puede fallar si supera el número máximo de transacciones por segundo (TPS), lo que hace que el servicio limite la aplicación o cuando se cae la conexión. Por ejemplo, si realiza demasiadas llamadas a operaciones de Amazon Textract Texact en un corto período de tiempo, limita las llamadas y envía un`ProvisionedThroughputExceededException`error en la respuesta de la operación. Para obtener información sobre cuotas de Amazon Textract TPS, consulte[Cuotas de Amazon Textract](textract-limits.md).

Puede administrar la limitación y las conexiones caídas si vuelve a intentar automáticamente la operación. Puede especificar el número de reintentos incluyendo el`Config`cuando se crea el cliente de Amazon Textract. Recomendamos un recuento de reintentos de 5. LaAWSEl SDK reintenta una operación la cantidad especificada de veces antes de que se produzca el error y de que se lanza una excepción. Para obtener más información, consulte [Reintentos de error y retardo exponencial en AWS](https://docs.aws.amazon.com/general/latest/gr/api-retries.html).

> **Note**
>
> Los reintentos automáticos funcionan tanto para operaciones síncronas como asíncronas. Antes de especificar reintentos automáticos, asegúrese de que tiene la versión más reciente del SDK de AWS. Para obtener más información, consulte [Paso 2: Configurar laAWS CLIyAWSSDK de](setup-awscli-sdk.md).

En el siguiente ejemplo se muestra cómo reintentar las operaciones de Amazon Textract automáticamente cuando se procesan varios documentos.

Requisitos previos

- Si aún no lo ha hecho:

  a. Crear o actualizar un usuario de IAM con`AmazonTextractFullAccess`y`AmazonS3ReadOnlyAccess`permisos. Para obtener más información, consulte [Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM](setup-awsaccount-and-user.md).

  b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar laAWS CLIyAWSSDK de](setup-awscli-sdk.md).

Para reintentar automáticamente las operaciones

1. Cargue varias imágenes de documento en el bucket de S3 para que se lure el ejemplo sincrónico. Cargue un documento de varias páginas en su bucket de S3 y ejecute`StartDocumentTextDetection`para ejecutar el ejemplo asíncrono.

   Para obtener instrucciones, consulteCarga de objetos en Amazon S3en laAmazon Simple Storage Service User Guide.

2. En los siguientes ejemplos se muestra cómo utilizar el`Config`para volver a intentar automáticamente una operación. El ejemplo sincrónico llama al`DetectDocumentText`, mientras que el ejemplo asíncrono llama al`GetDocumentTextDetection`.

   Sync Example

   Utilice los siguientes ejemplos para llamar al`DetectDocumentText`en los documentos de su bucket de Amazon S3. En`main`, cambie el valor de`bucket`en su bucket de S3. Cambiar el valor de`documents`a los nombres de las imágenes de documento que ha subido en el paso 2.

```
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

Utilice los siguientes ejemplos para llamar a la operación `GetDocumentTextDetection`. Supone que ya ha llamado`StartDocumentTextDetection`en los documentos de su bucket de Amazon S3 y obtuvo un`JobId`. En`main`, cambie el valor de`bucket`en su bucket de S3 y el valor de`roleArn`al Arn asignado a su rol Textract. También necesitará cambiar el valor de`document`al nombre del documento de varias páginas del bucket de Amazon S3. Por último, sustituya el valor de`region_name`con el nombre de su región y proporcione el`GetResults`función con el nombre de su`jobId`.

```
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()
```

Prácticas recomendadas para Amazon Textract

Amazon Textract utiliza el aprendizaje automático para leer documentos como lo haría una persona. Extrae texto, tablas y formularios de documentos. Utilice las siguientes prácticas recomendadas para obtener los mejores resultados de sus documentos.

Proporcionar un documento de entrada óptimo

A continuación se muestra una lista de algunas formas de optimizar los documentos de entrada para obtener mejores resultados.

- Asegúrese de que el texto del documento esté en un idioma compatible con Amazon Textract Texact. Actualmente, Amazon Textract admite inglés, español, español, español, italiano, francés y portugués.
- Proporcione una imagen de alta calidad, idealmente al menos 150 PPP.
- Si el documento ya está en uno de los formatos de archivo compatibles con Amazon Textract Texact (PDF, TIFF, JPEG y PNG), no convierta ni reduzca la muestra del documento antes de cargarlo en Amazon Textract.

Para obtener los mejores resultados al extraer texto de tablas de documentos, asegúrese de que:

- Las tablas del documento están separadas visualmente de los elementos circundantes de la página. Por ejemplo, la tabla no se superpone sobre una imagen o un patrón complejo.
- El texto de la tabla está en posición vertical. Por ejemplo, el texto no se gira en relación con otro texto de la página.

Al extraer texto de tablas, es posible que veas resultados incoherentes cuando:

- Celdas de tabla combinadas que abarcan varias columnas.
- Tablas con celdas, filas o columnas diferentes de otras partes de la misma tabla.

Recomendamos utilizardetección de textocomo solución provisional.

Usar puntuación de confianza

Debe tener en cuenta las puntuaciones de confianza devueltas por las operaciones de la Amazon Textract Texact y la sensibilidad de su caso de uso. Una puntuación de confianza es un número entre 0 y 100 que indica la probabilidad de que una predicción determinada sea correcta. Le ayuda a tomar decisiones informadas sobre cómo utiliza los resultados.

En aplicaciones sensibles a errores de detección (falsos positivos), aplique un umbral mínimo de puntuación de confianza. La aplicación debe descartar los resultados por debajo de ese umbral o indicar situaciones que requieren un mayor nivel de escrutinio humano.

El umbral óptimo depende de la aplicación. Para fines de archivo, como documentar notas escritas a mano, puede ser de hasta un 50%. Los procesos empresariales que implican decisiones financieras pueden requerir umbrales del 90% o más.

Considere utilizar revisión humana

Considere también incorporar la revisión humana en sus flujos de trabajo. Esto es especialmente importante para aplicaciones sensibles, como los procesos empresariales que implican decisiones financieras.

Tutoriales

the section called “Block”los objetos que se devuelven de las operaciones de Amazon Textract Texact contienen los resultados de las operaciones de detección de texto y análisis de texto, tales comothe section called “AnalyzeDocument”. En los siguientes tutoriales de Python se muestran algunas de las diferentes formas en las que puede utilizar objetos de bloque. Por ejemplo, puede exportar información de tabla a un archivo de valores separados por comas (CSV).

Los tutoriales utilizan operaciones sincrónicas de Amazon Textract que devuelven todos los resultados. Si desea utilizar operaciones asíncronas tales comothe section called “StartDocumentAnalysis”, debe cambiar el código de ejemplo para acomodar varios lotes de devueltos`Block`objetos. Para utilizar el ejemplo de operaciones asíncronas, asegúrese de haber seguido las instrucciones que se dan enConfiguración de Amazon Textract Texact para operaciones asíncronas.

Para ver ejemplos que le muestran otras formas de utilizar Amazon Textract, consulteEjemplos de código adicionales.

Temas

- Requisitos previos
- Extracción de pares clave-valor de un documento de formulario
- Exportación de tablas a un archivo CSV
- Creación de unAWS LambdaFunción
- Ejemplos de código adicionales

Requisitos previos

Antes de ejecutar los ejemplos de esta sección, debe configurar su entorno.

Para configurar el entorno

1. Crear o actualizar un usuario de IAM con`AmazonTextractFullAccess`permisos. Para obtener más información, consulte Paso 1: Configuración de una cuenta de AWS y creación de un usuario de IAM.
2. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte Paso 2: Configurar laAWS CLIyAWSSDK de.

Extracción de pares clave-valor de un documento de formulario

El siguiente ejemplo de Python muestra cómo extraer pares clave-valor en documentos de formulario dethe section called “Block”objetos que se almacenan en un mapa. Los objetos de bloque se devuelven de una llamada athe section called “AnalyzeDocument”. Para obtener más información, consulte Datos de formulario (pares clave-valor).

Utiliza las siguientes funciones:

- `get_kv_map`— LlamadasAnalyzeDocumenty almacena los objetos KEY y VALUE BLOCK en un mapa.
- `get_kv_relationshipyfind_value_block`: construye las relaciones clave-valor desde el mapa.

Para extraer pares clave-valor de un documento de formulario

1. Configure el entorno. Para obtener más información, consulte Requisitos previos.
2. Copie el siguiente código de ejemplo en un archivo llamadotextract_python_kv_parser.py que es.

```
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. En el símbolo del sistema, escriba el siguiente comando. Reemplazar`file`con el archivo de imagen del documento que desea analizar.

```
textract_python_kv_parser.py file
```

4. Cuando se te solicite, introduce una clave que se encuentra en el documento de entrada. Si el código detecta la clave, muestra el valor de la clave.

Exportación de tablas a un archivo CSV

En estos ejemplos de Python se muestran cómo exportar tablas de una imagen de un documento a un archivo de valores separados por comas (CSV).

El ejemplo de análisis de documentos síncrono recopila información de tabla de una llamada athe section called “AnalyzeDocument”. El ejemplo de análisis de documentos asíncrono realiza una llamada athe section called “StartDocumentAnalysis”y, a continuación, recupera los resultados dethe section called “GetDocumentAnalysis”como`Block`objetos.

La información de la tabla se devuelve comothe section called “Block”objetos de una llamada athe section called “AnalyzeDocument”. Para obtener más información, consulte Tablas. La`Block`los objetos se almacenan en una estructura de mapa que se utiliza para exportar los datos de la tabla a un archivo CSV.

Synchronous

En este ejemplo, utilizará las funciones:

- `get_table_csv_results`— LlamadasAnalyzeDocumenty crea un mapa de tablas que se detectan en el documento. Crea una representación CSV de todas las tablas detectadas.
- `generate_table_csv`: genera el archivo CSV de una tabla individual.
- `get_rows_columns_map`— Obtiene las filas y columnas del mapa.
- `get_text`— Obtiene el texto de una celda.

Para exportar tablas a un archivo CSV

1. Configure el entorno. Para obtener más información, consulte Requisitos previos.
2. Copie el siguiente código de ejemplo en un archivo llamadotextract_python_table_parser.py que es.

```
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. En el símbolo del sistema, escriba el siguiente comando. Reemplazar`file`con el nombre del archivo de imagen de documento que desee analizar.

```
python textract_python_table_parser.py file
```

Al ejecutar el ejemplo, la salida CSV se guarda en un archivo denominado`output.csv`.

Asynchronous

En este ejemplo, utilizará hacer uso de dos scripts diferentes. El primer guión inicia el proceso de análisis asincrónico de documentos con`StartDocumentAnalysis`y obtiene el`Block`información devuelta por`GetDocumentAnalysis`. El segundo guión toma el devuelto`Block`información de cada página, da formato a los datos como tabla y guarda las tablas en un archivo CSV.

Para exportar tablas a un archivo CSV

1. Configure el entorno. Para obtener más información, consulte Requisitos previos.
2. Asegúrese de haber seguido las instrucciones que se dan en verConfiguración de Amazon Textract Texact para operaciones asíncronas. El proceso documentado en esa página le permite enviar y recibir mensajes sobre el estado de finalización de los trabajos asíncronos.

3. En el siguiente ejemplo de código, sustituya el valor de`roleArn`con el Arn asignado al rol que creó en el paso 2. Sustituir el valor de`bucket`con el nombre del bucket de S3 que contenga el documento. Sustituir el valor de`document`con el nombre del documento de su bucket de S3. Sustituir el valor de`region_name`con el nombre de la región de su bucket.

   Copie el siguiente código de ejemplo en un archivo llamado`start_doc_analysis_for_table_extraction.py`que es..

```
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. Ejecute el código. El código imprimirá un JobId. Copia este JobId hacia abajo.
5. Espere a que el trabajo finalice el procesamiento y, una vez finalizado, copie el siguiente código en un archivo denominadoget_doc_analysis_for_table_extraction.py que es. Sustituir el valor de`jobId`con el ID de Job que copiaste anteriormente. Sustituir el valor de`region_name`con el nombre de la región asociada a su rol Textract. Sustituir el valor de`file_name`con el nombre que desea asignar al archivo CSV de salida.

```
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. Ejecute el código.

   Una vez que haya obtenido los resultados, asegúrese de eliminar los recursos SNS y SQS asociados o, de lo contrario, puede acumular cargos por ellos.

Creación de unAWS LambdaFunción

Puede llamar a las operaciones de la Amazon Textract Texact desde unAWS Lambdafunción. Las siguientes instrucciones muestran cómo crear una función de Lambda en Python que llamethe section called “DetectDocumentText”. devuelve una lista dethe section called “Block”objetos. Para ejecutar este ejemplo, necesita un depósito de Amazon S3 que contenga un documento en formato PNG o JPEG. Para crear la función, utilice la consola.

Para ver un ejemplo que utiliza funciones Lambda para procesar documentos a gran escala, consulteProcesamiento de documentos a gran escala con Amazon Textract.

Para llamar a la operación DetectDocumentText desde una función Lambda:

Paso 1: Creación de un paquete de implementación de Lambda

1. Abra una ventana del sistema.
2. Especifique los siguientes comandos para crear un paquete de implementación con la versión más reciente deAWSSDK.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

Paso 2: Creación de una función de Lambda

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Create function (Crear función).
3. Especifique lo siguiente.
   - Elija Author from scratch (Crear desde cero).
   - En Function name (Nombre de función), escriba un nombre.
   - ParaRuntime (Tiempo de ejecución):, eligePython 3.7oPython 3.6.
   - ParaElija o cree un rol de ejecución, eligeCrear un nuevo rol con permisos básicos de Lambda.
4. ElegirCrear funciónpara crear la función Lambda.
5. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
6. En el panel de navegación, elijaRoles de.
7. En la lista de recursos, elija el rol de IAM que Lambda creó para usted. El nombre del rol comienza con el nombre de su función de Lambda.
8. Elija el iconoPermisospestaña y, a continuación, eligeAsociar políticas.
9. Selecciona las políticas de acceso de solo lectura de Amazon Textract y Amazon S3.
10. SelectAsociar política.

Para obtener más información, consulteCreación de una función Lambda con la consola

Paso 3: Crear y agregar una capa

1. Abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. En el panel de navegación, elija Layers (Capas).
3. Elija Crear capa.
4. ParaNombre, escriba un nombre.
5. En Description (Descripción), escriba una descripción.
6. ParaCode entry type, eligeCargar un archivo .zipy seleccioneCargar.
7. En el cuadro de diálogo, seleccione el archivo zip (boto3-layer.zip), el zip que creó enPaso 1: Creación de un paquete de implementación de Lambda.

8. ParaTiempos de ejecución compatibles, elige la versión del motor de ejecución que eligió enPaso 2: Creación de una función de Lambda.
9. ElegirCrearpara crear la capa.
10. Seleccione el icono del menú del panel de navegación.
11. Seleccione Functions (Funciones) en el panel de navegación.
12. En la lista de recursos, seleccione la función que ha creado enPaso 2: Creación de una función de Lambda.
13. ElegirConfiguracióny en elDiseñador, elijaCapas de(bajo el nombre de la función Lambda).
14. En el navegadorCapas de, elijaAñadir una capa.
15. ElegirSeleccionar de una lista de capas compatibles en tiempo de ejecución.
16. EnCapas compatibles, seleccione elNombreyVersionde la capa que creó en el paso 3.
17. Elija Add (Agregar).

Paso 4: Agregar código de python a la función

1. EnDiseñador, elija su función.
2. En el editor de código de función, agregue lo siguiente al archivolambda_function.py. Cambie los valores de`bucket`y`document`a su depósito y documento.

```
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. ElegirGuardarpara guardar la función de Lambda.

Paso 5: Pruebe su Lambda

1. SelectPruebas.
2. Introduzca un valor paraNombre del evento:.
3. Elija Create (Crear).
4. La salida, una lista dethe section called “Block”objetos, aparece en el panel Resultados de ejecución.

Si el archivo deAWS Lambdadevuelve un error de tiempo de espera, una llamada a la operación de la API de Amazon Texact podría ser la causa. Para obtener información sobre cómo ampliar el período de espera de unAWS Lambdafunción, consulteConfiguración de funciones de AWS Lambda.

Para obtener información sobre cómo invocar una función de Lambda desde el código, consulteInvocaciónAWS LambdaFunciones.

Ejemplos de código adicionales

La tabla siguiente contiene enlaces a más ejemplos de código de Amazon Textract.

Ejemplo	Descripción
Ejemplos de código de Amazon Textract	Muestra varias formas en las que puede utilizar Amazon Textract.
Procesamiento de documentos a gran escala con Amazon Textract	Muestra una arquitectura de referencia sin servidor que procesa documentos a gran escala.
Amazon Textract Parser	Muestra cómo analizar elthe section called “Block”objetos devueltos por las operaciones de Amazon Textract Texact.

Ejemplo	Descripción
Ejemplos de códigos de documentación de Amazon Textract	Ejemplos de código utilizados en esta guía.
Extractor	Muestra cómo convertir la salida de Amazon Textract Texact a varios formatos.
Generar documentos PDF con capacidad de búsqueda con Amazon Textract	Muestra cómo crear un documento PDF con capacidad de búsqueda a partir de diferente s tipos de documentos de entrada, como imágenes en formato JPG/PNG y documentos PDF escaneados.

Ejemplos de código para Amazon Textract

En los siguientes ejemplos de código, se muestra cómo utilizar Amazon Textract con unAWSkit de desarrollo de software (SDK).

Los ejemplos se dividen en las categorías siguientes:

Acciones

Fragmentos de código que muestran cómo llamar a funciones individuales de servicio.

Ejemplos de servicios combinados

Aplicaciones de ejemplo que funcionan en variosAWSServicios de .

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Ejemplos de código

- Acciones de Amazon Textract
  - Analizar un documento mediante Amazon Textract y unAWSSDK
  - Detectar texto en un documento mediante Amazon Textract y unAWSSDK
  - Obtener datos sobre un trabajo de análisis de documentos de Amazon Textract mediante unAWSSDK
  - Iniciar un análisis asíncrono de un documento utilizando Amazon Textract y unAWSSDK
  - Iniciar la detección de texto asíncrona con Amazon Textract Texact y unAWSSDK
- Ejemplos de servicios combinados para Amazon Textract
  - Creación de una aplicación de exploración de Amazon Textract
  - Detectar entidades en el texto extraído de una imagen mediante unAWSSDK

Acciones de Amazon Textract

En los siguientes ejemplos de código, se muestra cómo realizar acciones individuales de Amazon Textract conAWSSDK de. Estos extractos llaman a la API de Amazon Textract Texact y no están

diseñados para ejecutarse de forma aislada. En cada ejemplo se incluye un enlace a GitHub, con instrucciones de configuración y ejecución del código en contexto.

Los ejemplos siguientes incluyen solo las acciones que se utilizan con mayor frecuencia. Para obtener una lista completa, consulteReferencia de la API de Amazon Textract.

Ejemplos

- Analizar un documento mediante Amazon Textract y unAWSSDK
- Detectar texto en un documento mediante Amazon Textract y unAWSSDK
- Obtener datos sobre un trabajo de análisis de documentos de Amazon Textract mediante unAWSSDK
- Iniciar un análisis asíncrono de un documento utilizando Amazon Textract y unAWSSDK
- Iniciar la detección de texto asíncrona con Amazon Textract Texact y unAWSSDK

Analizar un documento mediante Amazon Textract y unAWSSDK

En los siguientes ejemplos de código, se muestra cómo analizar un documento mediante Amazon Textract.

Java

SDK para 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);
        }
    }
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteAnalyzeDocumentenAWS SDK for Java 2.xReferencia de la API.

Python

SDK para 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
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteAnalyzeDocumentenAWSReferencia de API de SDK for Python (Boto3).

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Detectar texto en un documento mediante Amazon Textract y unAWSSDK

En los siguientes ejemplos de código, se muestra cómo detectar texto en un documento mediante Amazon Textract.

Java

SDK para Java 2.x

Detecta texto de un documento de entrada.

```
   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);
        }
    }
```

Detectar texto de un documento ubicado en un bucket de Amazon S3.

```
    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);
        }
    }
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteDetectDocumentTextenAWS SDK for Java 2.xReferencia de la API.

Python

SDK para 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
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteDetectDocumentTextenAWSReferencia de API de SDK for Python (Boto3).

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Obtener datos sobre un trabajo de análisis de documentos de Amazon Textract mediante unAWSSDK

El siguiente ejemplo de código muestra cómo obtener datos sobre un trabajo de análisis de documentos de Amazon Textract.

Python

SDK para 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
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteGetDocumentAnalysisenAWSReferencia de API de SDK for Python (Boto3).

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Iniciar un análisis asíncrono de un documento utilizando Amazon Textract y unAWSSDK

En los siguientes ejemplos de código, se muestra cómo iniciar un análisis asíncrono de un documento mediante Amazon Textract.

Java

SDK para 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 "";
    }
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteStartDocumentAnalysisenAWS SDK for Java 2.xReferencia de la API.

Python

SDK para Python (Boto3)

Inicie un trabajo asíncrono para analizar un documento.

```
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
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteStartDocumentAnalysisenAWSReferencia de API de SDK for Python (Boto3).

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Iniciar la detección de texto asíncrona con Amazon Textract Texact y unAWSSDK

El siguiente ejemplo de código muestra cómo iniciar la detección de texto asíncrono en un documento mediante Amazon Textract.

Python

SDK para Python (Boto3)

Iniciar un trabajo asíncrono para detectar texto en un documento.

```
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
```

- Encuentre instrucciones y más código en GitHub.
- Para obtener más información sobre la API, consulteStartDocumentTextDetectionenAWSReferencia de API de SDK for Python (Boto3).

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Ejemplos de servicios combinados para Amazon Textract

Las siguientes aplicaciones de ejemplo utilizanAWSSDK para combinar Amazon Textract con otrosAWSServicios de . En cada ejemplo, se incluye un enlace a GitHub, donde puedes encontrar instrucciones sobre cómo configurar y ejecutar la aplicación.

Ejemplos

- Creación de una aplicación de exploración de Amazon Textract
- Detectar entidades en el texto extraído de una imagen mediante unAWSSDK

Creación de una aplicación de exploración de Amazon Textract

Los siguientes ejemplos de código indican cómo explorar la salida de Amazon Textract mediante una aplicación interactiva.

JavaScript

SDK para JavaScript V3

Indica cómo utilizar el AWS SDK for JavaScript para crear una aplicación React que utilice Amazon Textract para extraer datos de la imagen de un documento y presentarlos en una página web interactiva. Este ejemplo se ejecuta en un navegador web y requiere una identidad autenticada de Amazon Cognito para las credenciales. Para el almacenamiento utiliza Amazon Simple Storage Service (Amazon S3) y para las notificaciones consulta una cola de Amazon Simple Queue Service (Amazon SQS) que está suscrita a un tema de Amazon Simple Notification Service (Amazon SNS).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Indica cómo utilizar AWS SDK for Python (Boto3) con Amazon Textract para detectar elementos de texto, formularios y tablas en la imagen de un documento. La imagen de entrada y la salida de Amazon Textract aparecen en una aplicación Tkinter que permite explorar los elementos detectados.

- Envía la imagen de un documento a Amazon Textract y explora el resultado de los elementos detectados.
- Envía imágenes directamente a Amazon Textract o mediante un bucket de Amazon Simple Storage Service (Amazon S3).
- Utilice las API asíncronas para iniciar un trabajo que publique una notificación en un tema de Amazon Simple Notification Service (Amazon SNS) cuando el trabajo se finalice.

- Consulta una cola de Amazon Simple Queue Service (Amazon SQS) en busca de un mensaje de finalización de trabajo y muestra los resultados.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Detectar entidades en el texto extraído de una imagen mediante unAWSSDK

En el siguiente ejemplo de código se muestra cómo utilizar Amazon Comprehend para detectar entidades en el texto extraído por Amazon Textract Texact de una imagen almacenada en Amazon S3.

Python

SDK para Python (Boto3)

Muestra cómo utilizar elAWS SDK for Python (Boto3)en un bloc de notas de Jupyter para detectar entidades del texto que se extrae de una imagen. En este ejemplo, se utiliza Amazon Textract para extraer texto de una imagen guardada en Amazon Simple Storage Service (Amazon S3) y Amazon Comprehend para detectar entidades en el texto extraído.

Este ejemplo es un bloc de notas Jupyter y debe ejecutarse en un entorno que pueda alojar blocs de notas. Para obtener instrucciones sobre cómo ejecutar el ejemplo mediante Amazon SageMaker, consulte las instrucciones deText Extraer y comprender notebook.ipyNB.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Para obtener una lista completa deAWSGuías para desarrolladores de SDK y ejemplos de código, consulteUso de Amazon Textract con unAWSSDK. En este tema también se incluye información sobre cómo empezar y detalles sobre versiones anteriores del SDK.

Uso de Augmented AI de Amazon para agregar revisión humana a la salida de Amazon Textract

La Augmented AI Amazon (Amazon A2I) es un servicio de Machine Learning (ML) que facilita la creación de flujos de trabajo para la revisión humana de los análisis de ML.

Amazon Textract está integrado con Amazon A2I. Puede utilizarlo para enrutar los resultados del análisis de documentos que tienen una baja puntuación de confianza a los revisores humanos.

Puedes usar Amazon Textract`AnalyzeDocument`API para extraer datos de formularios y de la consola Amazon A2I. Puede especificar las condiciones en las que Amazon A2I envía las predicciones a los revisores. Las condiciones se establecen en función del umbral de confianza de las claves de formulario importantes. Por ejemplo, puede enviar un documento a un revisor humano si la claveNombreo su valor asociadoJane Doese detectó con poca confianza.

Temas

- Conceptos básicos de Amazon A2I
- Comience a utilizar Amazon A2I

Conceptos básicos de Amazon A2I

Revise los siguientes términos para familiarizarse con los conceptos principales de Amazon A2I.

Condiciones de activación de revisión humana

Puede utilizar Amazon A2Icondiciones de activaciónpara especificar cuándo se envía un documento a humanos para su revisión y el contenido del formulario que se pide a los trabajadores que revisen.

Por ejemplo, puede establecer una condición de activación para que Amazon Textract envíe formularios a Amazon A2I cuando se detecta una clave importante con poca confianza, comoPhone Number. En este ejemplo, se pide a los revisores humanos que revisen elPhone Numbercampo y valor asociados detectados por Amazon Textract.

Puede utilizar las siguientes condiciones de activación para especificar cuándo se envían los formularios a los humanos para su revisión:

- Desencadenar una revisión humana para claves de formulario específicas basadas en la puntuación de confianza de clave de formulario. Se pide a los revisores humanos que revisen estas claves de formulario y valores asociados.
- Desencadenar una revisión humana cuando falten claves de formulario específicas. Se pide a los revisores humanos que identifiquen estas claves de formulario y los valores asociados.
- Desencadenar la revisión humana para todas las claves de formulario identificadas por Amazon Textract Texact con puntuaciones de confianza en un rango especificado.
- Enviar aleatoriamente una muestra de formularios a los humanos para su revisión. Se pide a los revisores humanos que revisen todas las claves de formulario y valores detectados por Amazon Textract.

Cuando su condición de activación depende de las puntuaciones de confianza clave del formulario, puede utilizar dos tipos de umbrales de confianza de predicción para activar la revisión humana:

- Fianza de identificación— La puntuación de confianza de los pares clave-valor detectados dentro de un formulario.
- Confianza de cualificación— La puntuación de confianza del texto contenido en un par clave-valor de un formulario.

Si especifica un umbral de confianza, Amazon A2I dirige únicamente las predicciones que se encuentran dentro del umbral a revisores humanos. Puede ajustar estos umbrales en cualquier momento para lograr el equilibrio adecuado entre precisión y rentabilidad. Esto puede ayudarle a implementar auditorías para supervisar con regularidad la precisión de la predicción.

> **Note**
>
> Puede personalizar aún más las condiciones en las que se envían los documentos a los humanos para su revisión mediante el tipo de tarea personalizada de Amazon A2I. Con este tipo de tarea, se especifican las condiciones para una revisión humana directamente en su aplicación. Para obtener información, consulteUtilizar la Augmented AI de Amazon con tipos de tareas personalizadosen la guía para desarrolladores de Amazon SageMaker.

Flujo de trabajo de revisión humana (definición de flujo)

Usas unflujo de trabajo de revisión humana, también conocido comoDefinición del flujo, para especificar los recursos utilizados para crear el flujo de trabajo de revisión humana y especificar las condiciones de activación.

Los recursos que especifique son:

- Un rol de IAM con permiso para llamar a las operaciones de la API de Amazon A2I
- Un bucket de Amazon S3 en el que desea almacenar la salida de la revisión humana
- Tu humanoequipo de trabajo
- UNAplantilla de tarea de trabajoque incluye instrucciones y ejemplos para ayudar a los trabajadores a completar la tarea de revisión

También se utiliza el flujo de trabajo de revisión humana para especificar las condiciones de activación. Para obtener más información, consulte Condiciones de activación de revisión humana.

Puede utilizar un único flujo de trabajo de revisión humana para crear variosBucles humanos.

Puede crear un flujo de trabajo de revisión humana en la consola de SageMaker o con la API de SageMaker. Para obtener información, consulte Crear un flujo de trabajo de revisión humana.

Plantilla de tareas de trabajador

Usas unplantilla de tarea de trabajopara crear una interfaz de usuario de trabajador que se utilice para sus tareas de revisión humana.

La interfaz de usuario del trabajador muestra los documentos y las instrucciones del trabajador. También proporciona herramientas que los trabajadores utilizan para completar las tareas.

Puede utilizar la consola de SageMaker para configurar la plantilla de tarea de trabajo al crear un flujo de trabajo de revisión humana. Para obtener información, consulte Crear un flujo de trabajo de revisión humana.

Equipo de trabajo

UNAequipo de trabajoes un grupo de trabajadores humanos al que envías tus tareas de revisión humana.

Al crear un flujo de trabajo de revisión humana, especifica un único equipo de trabajo.

Con Amazon A2I, puede utilizar un grupo de revisores dentro de su propia organización. También puede acceder a la plantilla compuesta por más de 500.000 contratistas independientes que ya están realizando tareas de aprendizaje automático a través de Amazon Mechanical Turk. Otra opción es utilizar proveedores de mano de obra preseleccionados por AWS en función de su calidad y su cumplimiento de los procedimientos de seguridad.

Amazon A2I también proporciona a los revisores una interfaz web que consta de todas las instrucciones y herramientas que necesitan para completar sus tareas de revisión.

Para cada tipo de fuerza de trabajo (privada, proveedor y Mechanical Turk), puede crear varios equipos de trabajo. Puede utilizar cada equipo de trabajo en varios flujos de trabajo de revisión humana. Para aprender a crear una fuerza de trabajo y equipos de trabajo, consulteCreación y administración de personalen la guía para desarrolladores de Amazon SageMaker.

> **Important**
>
> Clicaquípara ver los programas de cumplimiento que cubren la Augmented AI de Amazon en este momento. Si utiliza Amazon Augmented AI junto con otros servicios de AWS (como Amazon Rekognition y Amazon Textract), tenga en cuenta que Amazon Augmented AI puede no estar dentro del alcance de los mismos programas de cumplimiento que esos otros servicios. Es responsable de cómo utiliza Amazon Augmented AI, incluida la comprensión de cómo el servicio procesará o almacenará datos de los clientes y cualquier impacto en la conformidad del entorno de datos. Debe analizar los objetivos de su carga de trabajo con el equipo de cuentas de AWS, que puede evaluar si el servicio es adecuado para su caso de uso y arquitectura propuestos.
> En la actualidad, la Augmented AI de Amazon cumple con PCI, excepto en los casos de personal público y de proveedores.
> Para obtener información sobre el cumplimiento HIPAA de Augmented AI de Amazon, haga clic enaquí.

Bucles humanos

Usa un bucle humano para crear una tarea de revisión humana.

Asigna una tarea de revisión humana a un trabajador de su equipo de trabajadores humanos. Se solicita al trabajador que revise los pares clave-valor detectados por Amazon Textract Texact en el documento de entrada que especificó en las condiciones de activación.

Por ejemplo, supongamos que una imagen cae entre el cincuenta y el sesenta por ciento de confianza de que contiene una manzana. Esto se encuentra dentro de su umbral de confianza para la revisión humana y se envía a un trabajador. El trabajador comprueba si hay una manzana en el documento y la marca como que contiene o no una manzana. A continuación, Amazon A2I vuelve a enviar el documento al flujo de trabajo.

Cuando llamas a Amazon Textract`AnalyzeDocument`y especifique un flujo de trabajo de revisión humana (definición de flujo) y un nombre de bucle humano, se crea una tarea de revisión humana cuando se cumplen las condiciones de activación especificadas en el flujo de trabajo de revisión humana. Estas tareas se crean utilizando los recursos especificados en el flujo de trabajo de revisión humana.

Comience a utilizar Amazon A2I

Los siguientes pasos le ayudan a integrar Amazon A2I en una tarea de análisis de documentos de una sola página de Amazon Texact. Siga estas instrucciones:

1. Cree un flujo de trabajo de revisión humana mediante la consola Amazon A2I (recomendada para nuevos usuarios) o la API de Amazon A2I.
2. Para analizar un formulario e incluir la revisión humana cuando sea necesario, utilice el`AnalyzeDocument`y especifique el nombre de recurso de Amazon (ARN) del flujo de trabajo de revisión humana. La respuesta indica si se requiere revisión humana.
3. Supervise su bucle humano con la consola y la API de Amazon A2I.
4. Revise los resultados de la revisión humana en un depósito de Amazon S3 al que se envían los resultados.

Para configurar una instancia de bloc de notas de SageMaker y utilizar un bloc de notas de ejemplo, consulteDemostración integral mediante Amazon Textract e Augmented AIen laGuía para desarrolladores de Amazon SageMaker

> **Note**
>
> En esta sección se explica cómo crear un flujo de trabajo de revisión humana para el tipo de tarea Amazon A2I, Amazon Textract Texact. Para personalizar aún más la integración de Amazon A2I y Amazon Textract, puede utilizar el tipo de tarea personalizada de Amazon A2I. Con esta opción, proporciona una plantilla de tareas de trabajador personalizada y especifica las condiciones en las que se envía un documento para revisión humana directamente en la

aplicación. Para obtener información, consulteUtilizar la Augmented AI de Amazon con tipos de tareas personalizadosen laGuía para desarrolladores de Amazon SageMaker.

Temas

- Crear un flujo de trabajo de revisión humana
- Análisis del documento
- Monitor Human Loop
- Ver datos de salida y métricas de trabajadores

Crear un flujo de trabajo de revisión humana

Puede crear un flujo de trabajo de revisión humana mediante la consola Amazon A2I (recomendada para nuevos usuarios) o Amazon A2I`CreateFlowDefinition`.

Temas

- Crear un flujo de trabajo de revisión humana (consola)
- Crear un flujo de trabajo de revisión humana (API)

Crear un flujo de trabajo de revisión humana (consola)

Puede completar este ejemplo utilizando su propio documento en Amazon S3 o bien descargarloeste documento de ejemploy colóquelo en su bucket de Amazon S3.

Asegúrese de que el bucket de S3 esté en el mismoAWSRegión en la que está utilizando Amazon Textract. Para crear un bucket, consulteCrear un bucketen laGuía del usuario de Amazon Simple Storage Service Console.

> **Note**
>
> La consola Amazon A2I está integrada en la consola de SageMaker. Para utilizar la consola, necesita permisos para acceder a la consola de SageMaker y crear un equipo de trabajo. Para empezar, puede utilizar laAmazonSageMakerFullAccessLa política administrada de IAM que incluye todos los permisos necesarios para realizar la mayoría de las acciones en SageMaker. Para obtener más información, consulteIdentity and Access Management para Amazon SageMakeren laGuía para desarrolladores de Amazon SageMaker.

Temas

Paso 1: Crear un equipo de trabajo (consola)

En primer lugar, cree un equipo de trabajo en la consola de Amazon A2I y añádase a sí mismo como trabajador para que pueda obtener una vista previa de la tarea de revisión humana en el portal de trabajadores, los miembros del equipo de trabajo pueden examinar diferentes tareas y documentos asignados a ellos.

Para crear un personal privado con correos electrónicos de trabajadores (consola)

1. Abra la consola de SageMaker enhttps://console.aws.amazon.com/sagemaker/.
2. En el panel de navegación, enGround Truth, eligeLabeling de personal.
3. Elija Private (Privado), a continuación elija Create private team (Crear equipo privado).
4. Elija Invite new workers by email (Invitar a nuevos trabajadores por correo electrónico).
5. En este ejemplo, introduzca su dirección de correo electrónico y la dirección de correo electrónico de cualquier otra persona que desee obtener una vista previa del portal de trabajadores. Puede pegar o escribir una lista de hasta 50 direcciones de correo electrónico separadas por comas en laDirecciones de correo electrónico.
6. Introduzca un nombre de organización y un correo electrónico de contacto.
7. Elija Create private team (Crear equipo privado).

Si te agregas a un equipo de trabajo privado, recibirás un correo electrónico deno-`reply@verificationemail.com`con información de inicio de sesión. Utilice el enlace de este correo electrónico para restablecer la contraseña e iniciar sesión en el portal de trabajadores. Aquí es donde aparecerán tus tareas de revisión humana después de llamar`AnalyzeDocument`.

Paso 2: Crear un flujo de trabajo de revisión humana (consola)

En este paso, creará un flujo de trabajo de revisión humana Amazon Textract Texact.

Para crear un flujo de trabajo de revisión humana (consola)

1. Abra la consola de Amazon A2I enhttps://console.aws.amazon.com/a2ipara obtener acceso aFlujos de trabajo de revisión(Se ha creado el certificado).

2. ElegirCrear flujo de trabajo de revisión humana.
3. ParaNombre, introduzca un nombre de flujo de trabajo.
4. ParaS3 bucket, elija el bucket de donde desea que Amazon A2I almacene los resultados de las tareas de revisión humana. Si no elige un bucket, cámbielo para introducir el nombre del bucket.
5. UNDERRol de IAM, seleccioneCreate a new role (Crear un nuevo rol). Aparece una ventana con el títuloCreación de un rol de IAM. Utilice esta ventana para especificar los depósitos de Amazon S3 a los que desea que tenga acceso a este rol. Si no seleccionaCualquier bucket de S3, especifique el depósito de salida que especificó en el paso 4 y el bucket que contiene el documento de entrada.
6. ParaTipo de tarea, eligeTextract: extracción de par clave-valor.
7. EnExtracción de formularios de Amazon Textract - Condiciones para invocar la revisión humana, especifique las condiciones de activación. Le recomendamos que establezca un umbral de puntuación de confianza elevado para al menos una clave del documento para activar una revisión humana, de modo que pueda obtener una vista previa de una tarea de trabajador en el portal de trabajadores.

   Si ha utilizado el documento de ejemplo proporcionado en este tutorial, especifique las condiciones de activación de la siguiente manera:

   a. ElegirDesencadenar una revisión humana para claves de formulario específicas basadas en la puntuación de confianza de clave de formulario o cuando faltan claves de formulario específicas.
   b. ParaNombre de la clave, introduzca**`Mail Address`**.
   c. Establecimiento de la propiedad deIdentificación de confianzaumbral entre0y99.
   d. Establecimiento de la propiedad deConfianza de cualificaciónumbral entre0y99.
   e. ElegirDesencadenar una revisión humana para todas las claves de formulario identificadas por Amazon Textract Texact con puntuaciones de confianza en un rango específico.
   f. Para**`identification confidence`**, elija cualquier número entre 0 y 90.
   g. Para**`qualification confidence`**, elija cualquier número entre 0 y 90.

   Esto desencadena una revisión humana si Amazon Textract devuelve un puntaje de confianza inferior al 99 paraDirección de correoy su valor, o si devuelve una puntuación de confianza inferior a 90 para cualquier par clave-valor detectado en el documento.

8. UNDERCreación de plantillas de tareas de trabajo, seleccioneCrear a partir de una plantilla predeterminada.
9. ParaNombre de la plantilla, introduzca un nombre descriptivo..
10. ParaTask description, agregue algo similar a lo siguiente:

    ```
    Read the instructions and review the document.
    ```

11. ParaTrabajadoreselegirPrivate.
12. En el menú elige el equipo privado que has creado.
13. Seleccione Create (Crear).

Una vez creado el flujo de trabajo de revisión humana, aparece en la tabla delFlujos de trabajo de revisión(Se ha creado el certificado). CuandoEstadoesActivo, copia y guarda el ARN del flujo de trabajo.

Crear un flujo de trabajo de revisión humana (API)

Puede crear un flujo de trabajo de revisión humana o unDefinición del flujo, utilizando Amazon A2I,`CreateFlowDefinition`.

En este ejemplo, puede utilizar su propio documento en Amazon S3 o descargareste documento de ejemploy guárdelo en su bucket S3.

Asegúrese de que el bucket de Amazon S3 esté en el mismoAWSRegión a la que planeas usar para llamar`AnalyzeDocument`. Para crear un bucket, siga las instrucciones en Crear un bucket en la Guía del usuario de la consola de Amazon Simple Storage Service.

Requisitos previos

Para utilizar la API de Amazon A2I para crear un flujo de trabajo de revisión humana, debe completar los siguientes requisitos previos:

- Configure un rol de IAM con permiso para llamar a las operaciones de la API de Amazon A2I y Amazon Textract. Para empezar, puede adjuntar las políticas de AWS, AmazonAugmentedAIFullAccess y AmazonTextractFullAccess a un rol de IAM. Registre el nombre de recursos de Amazon (ARN) de IAM, ya que lo necesitará más adelante.

  Para obtener permisos más detallados al utilizar Amazon Textract, consulteEjemplos de políticas basadas en identidades de Amazon Textract. Para Amazon A2I, consultePermisos y seguridad en la Augmented AI de Amazonen laGuía para desarrolladores de Amazon SageMaker.

- Cree un equipo de trabajo privado y registre el ARN del equipo de trabajo. Si es un usuario nuevo de Amazon A2I, siga las instrucciones dePaso 1: Crear un equipo de trabajo (consola).
- Cree una plantilla de tarea del trabajador. Siga las instrucciones enCrear una plantilla de tarea de trabajopara crear una plantilla mediante la consola Amazon A2I. Cuando estés creando la plantilla, eligeExtracción de forma de textoparaTipo de plantilla. En la plantilla, sustituya`s3_arn`con el ARN de Amazon S3 de su documento. Añadir instrucciones adicionales para el trabajador en`<full-instructions header="Instructions"></full-instructions>`.

  Si desea obtener una vista previa de la plantilla, asegúrese de que su rol de IAM tenga los permisos descritos enHabilitar vistas previas de plantillas de tareas del trabajador.

  Después de crear la plantilla, registre el ARN de la plantilla de tarea del trabajador.

Usa los recursos que ha creado enRequisitos previospara configurar`CreateFlowDefinition`request. En esta solicitud, también especificará las condiciones de activación en formato JSON. Para obtener información sobre cómo configurar las condiciones de activación, consulteUtilizar el esquema JSON de condiciones de activación del bucle humano con Amazon Textract.

Creación de un flujo de trabajo de revisión humana (SDK de AWS para Python (Boto3))

Para utilizar este ejemplo, reemplace la*red*texto con sus especificaciones y recursos.

En primer lugar, codifique las condiciones de activación en un objeto JSON utilizando el siguiente código. Esto desencadena una revisión humana si Amazon Textract devuelve un puntaje de confianza inferior al 99 paraDirección de correoy su valor, o si devuelve una puntuación de confianza inferior a 90 para cualquier par clave-valor detectado en el documento. Si utiliza el documento de ejemplo que se proporciona en este ejemplo, estas condiciones de activación crean una tarea de revisión humana.

```
import json

humanLoopActivationConditions = json.dumps("{
                "Conditions": [
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "Mail Address",
                            "KeyValueBlockConfidenceLessThan": 99,
                            "WordBlockConfidenceLessThan": 99
```

```
                    }
                },
                {
                    "ConditionType": "ImportantFormKeyConfidenceCheck",
                    "ConditionParameters": {
                        "ImportantFormKey": "*",
                        "KeyValueBlockConfidenceLessThan": 90,
                        "WordBlockConfidenceLessThan": 90
                    }
                }
            ]
        }"
)
```

Usarhuman`LoopActivationConditions`para configurar el`create_flow_definition`request. El siguiente ejemplo utiliza el SDK for Python (Boto3) para llamar`create_flow_definition`en us-west-2 de la región de AWS. Especifica el uso de un equipo de trabajo privado.

```
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"
)
```

Análisis del documento

Para incorporar Amazon A2I en un flujo de trabajo de análisis de Amazon Textract Texact, configureHumanLoopConfigen la`AnalyzeDocument`.

En`HumanLoopConfig`, especifica el ARN del flujo de trabajo de revisión humana (definición de flujo) en`FlowDefinitionArn`, y dale un nombre a tu bucle humano en`HumanLoopName`.

Analyze the Document (AWS SDK for Python (Boto3))

El siguiente ejemplo utiliza el SDK for Python (Boto3) para llamar`analyze_document`en us-west-2. Reemplace el*rojo, cursiva*texto con sus recursos. Para obtener más información, consulteanalyze_documenten laAWSSDK para la referencia de API de Python (Boto).

```
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)

El siguiente ejemplo utiliza laAWSCLI para llamar`analyze_document`. Estos ejemplos son compatibles conAWSCLI versión 2. El primero es la sintaxis abreviada, el segundo en sintaxis JSON. Para obtener más información, consulteDocumento-análisisen laAWS CLIReferencia de los comandos.

```
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**
>
> Evite espacios blancos en el parámetro —human-loop-config, ya que esto puede causar problemas de procesamiento para el código.

La respuesta a esta solicitud contieneSalida de activación de bucle humano, que indica si se creó un bucle humano y, de ser así, por qué. Si se creó un bucle humano, este objeto también contiene el`HumanLoopArn`.

Para obtener más información acerca de y ejemplos de uso de la`AnalyzeDocument`operación, consulteAnálisis del texto del documento con Amazon Textract.

Monitor Human Loop

Puede ver detalles sobre su bucle humano y detener un bucle humano activo en caso de error mediante la consola y la API de Amazon A2I.

Ver detalles de Human Loop

Puede ver el estado del bucle humano en la consola de Amazon A2I y mediante elAPI de tiempo de ejecución de Amazon A2I.

Para obtener detalles sobre su bucle humano (consola)

1. Abra la consola de Amazon A2I enhttps://console.aws.amazon.com/a2ipara obtener acceso aFlujos de trabajo de revisión(Se ha creado el certificado).

2. Elija el flujo de trabajo de revisión humana que utilizó también para configurar`HumanLoopConfig`en`AnalyzeDocument`.
3. En el navegadorBucles humanos, elija el bucle humano cuyos detalles desea ver.

Para obtener detalles sobre su bucle humano (API):

Usar Amazon A2I`DescribeHumanLoop`. Especifique el nombre del bucle humano que utilizó para llamar`AnalyzeDocument`.

El siguiente SDK for Python (Boto3) llama de ejemplo`describe_human_loop`.

```
response = client.describe_human_loop(HumanLoopName="human-loop-name")
```

Detener un bucle humano

Una vez iniciado un bucle humano, puede detenerlo utilizando la consola y la API de Amazon A2I.

Para detener el bucle humano (consola)

1. Abra la consola de Amazon A2I enhttps://console.aws.amazon.com/a2ipara obtener acceso aFlujos de trabajo de revisión(Se ha creado el certificado).
2. Elija el flujo de trabajo de revisión humana que utilizó para configurar`HumanLoopConfig`en la`AnalyzeDocument`.
3. En el navegadorBucles humanos, elija el bucle humano que desea detener.
4. Elija Detener.

Para detener tu bucle humano (API)

Usar Amazon A2I`StopHumanLoop`. Especifique el nombre del bucle humano al que utilizó para llamar`AnalyzeDocument`.

El siguiente SDK for Python (Boto3) llama de ejemplo`stop_human_loop`.

```
response = client.stop_human_loop(HumanLoopName="human-loop-name")
```

Ver datos de salida y métricas de trabajadores

Cuando un trabajador completa una tarea de revisión humana, Amazon A2I almacena los datos de salida en el bucket de Amazon S3 que especificó en el flujo de trabajo de revisión humana.

Si utiliza un personal privado, los datos de salida contienen metadatos de trabajador que puede utilizar para realizar un seguimiento de la actividad de los trabajadores individuales.

Buscar datos de salida en Amazon S3

Amazon A2I utiliza el nombre del flujo de trabajo de revisión humana como prefijo del nombre del archivo que almacena los datos de salida de los bucles humanos creados mediante ese flujo de trabajo de revisión humana.

La ruta a una salida de bucle humano utiliza el siguiente patrón en el que*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*representa la fecha de creación del bucle humano con año (YYYY), mes (MM) y día (DD) y el tiempo de creación con hora (hh), minuto (mm) y segundo (ss).

```
s3://output-bucket-specified-in-flow-definition/flow-definition-
name/YYYY/MM/DD/hh/mm/ss/human-loop-name/output.json
```

Para ver el resultado de un bucle humano, utilice la consola Amazon A2I.

Para ver la salida de bucle humano

1. Abra la consola de Amazon A2I enhttps://console.aws.amazon.com/a2ipara obtener acceso aFlujos de trabajo de revisión(Se ha creado el certificado).
2. Elija el flujo de trabajo de revisión humana que utilice para configurarHumanLoopConfigenAnalyzeDocument.
3. En el navegadorBucles humanos, elija el bucle humano cuya salida desea revisar.
4. UNDERUbicación de salida, elija el enlace a los datos de salida.

Seguimiento de la actividad de trabajadores

Cuando utiliza un personal privado para tareas de revisión humana, los datos de salida incluyen la siguiente información sobre el trabajador que completó la revisión:

- El valor workerId.

- En `workerMetadata`:
  - `identityProviderType`— El servicio utilizado para administrar la fuerza de trabajo privada.
  - `issuer`— El grupo de usuarios de Amazon Cognito o el emisor OIDC Identity Provider (IdP) asociado al equipo de trabajo asignado a esta tarea de revisión humana.
  - `sub`: identificador único que hace referencia al trabajador. Si creó un personal con Amazon Cognito, puede recuperar detalles sobre este empleado (como el nombre o el nombre de usuario) con este ID de mediante Amazon Cognito. Para obtener más información, consulteGestión y búsqueda de cuentas de usuarioenGuía para desarrolladores de Amazon Cognito.

A continuación se muestra un ejemplo de la salida que podría ver si utilizó Amazon Cognito para crear un personal privado.

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Cognito",
                    "issuer": "https://cognito-idp.aws-region.amazonaws.com/aws-
region_123456789",
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

A continuación se muestra un ejemplo de la salida que podría ver si utilizó su propio proveedor de identidad OIDC para crear una plantilla privada:

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Oidc",
                    "issuer": "https://example-oidc-ipd.com/adfs",
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

Para obtener más información acerca del uso de personal privado, consulteCrear personal privadoen laGuía para desarrolladores de Amazon SageMaker.

Seguridad en Amazon Textract

La seguridad en la nube de AWS es la mayor prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y un centro de datos que están diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

Utilice los siguientes temas para aprender a proteger los recursos de Amazon Textract.

Temas

Protección de datos en Amazon Textract

Amazon Textract Texact se ajusta alAWS Modelo de responsabilidad compartida, que incluye reglamentos y directrices para la protección de datos.AWSes responsable de proteger la infraestructura global en la que se ejecutan todos losAWSServicios de .AWSmantiene el control sobre los datos alojados en esta infraestructura, incluidos los controles de la configuración de seguridad para la gestión del contenido y los datos personales de los clientes.AWSclientes yAPNlos socios de negocios, que actúan como controladores de datos o procesadores de datos, son responsables de los datos personales que ponen en laAWSCloud.

Para fines de protección de datos, recomendamos proteger las credenciales de la cuenta de AWS y configurar cuentas de usuario individuales con AWS Identity and Access Management (IAM). Esto garantiza que a cada usuario solo se le otorguen los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de las siguientes formas:

- Utilice Multi-Factor Authentication (MFA) con cada cuenta.

- Utilice SSL/TLS para comunicarse con los recursos de AWS.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de AWS.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon S3.

Le recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, números de cuenta de sus clientes, en los campos de formato libre, como el campo Name (Nombre). Esto incluye cuando trabaje con Amazon Textract u otroAWSservicios que utilizan la consola, API,AWS CLI, o bienAWSSDK. Es posible que cualquier dato que ingrese Amazon Textract o en otros servicios se incluya en los registros de diagnóstico. Cuando proporcione una URL a un servidor externo, no incluya información de credenciales en la URL para validar la solicitud para ese servidor.

Para obtener más información sobre la protección de datos, consulte la entrada de blog relativa al modelo de responsabilidad compartida de AWS y GDPR en el blog de seguridad de AWS.

Cifrado en Amazon Textract

El cifrado de datos se refiere a proteger los datos en tránsito y en reposo. Puede proteger sus datos mediante las claves administradas de Amazon S3 oAWS KMS keyen reposo, junto con la seguridad de la capa de transporte estándar mientras está en tránsito.

Cifrado en reposo

El método principal de cifrado de datos en Amazon Textract es el cifrado del lado del servidor. Amazon S3 cifra los documentos de entrada transferidos desde los depósitos de Amazon S3 y se descifran cuando accede a ellos. Siempre que autentique su solicitud y tenga permiso de acceso, no existe diferencia alguna en la forma de obtener acceso a objetos cifrados o sin cifrar. Por ejemplo, si comparte objetos con una URL prefirmada, esa URL funcionará igual para objetos cifrados y sin cifrar. Además, al enumerar objetos en su bucket, el`ListLa` API devuelve una lista de todos los objetos, independientemente de si están cifrados.

Amazon Textract utiliza dos métodos de cifrado del lado del servidor de excluyentes.

Cifrado del lado del servidor con claves administradas por Amazon S3 (SSE-S3)

Cuando utiliza el cifrado del lado del servidor con claves de cifrado administradas por Amazon S3 (SSE-S3), cada objeto se cifra con una clave exclusiva. Como medida de seguridad adicional, este método cifra la propia clave con una clave maestra que rota regularmente. El cifrado del lado del servidor de Amazon S3 utiliza uno de los cifrados de bloques más seguros disponibles, Advanced Encryption Standard de 256 bits (AES-256), para cifrar sus datos. Para obtener más información, consulte Protección de datos con el cifrado del lado del servidor con claves de cifrado administradas por Amazon S3 (SSE-S3).

Cifrado del lado del servidor con claves de KMS almacenadas en AWS Key Management Service (SSE-KMS)

El cifrado del lado del servidor con claves de KMS almacenadas en AWS Key Management Service (SSE-KMS) es similar a SSE-S3, pero con algunos cargos y beneficios adicionales por el uso de este servicio. Hay permisos separados para usar una clave de KMS que proporcionan protección adicional contra el acceso no autorizado de sus objetos en Amazon S3. SSE-KMS le proporciona también un seguimiento de auditoría que muestra cuándo se usó la clave de KMS y quién la usó. Además, puede crear y administrar claves KMS o utilizarClaves administradas por AWSque sean únicas para usted, su servicio y su región. Para obtener más información, consulteProtección de datos con el cifrado del lado del servidorcon claves de KMS almacenadas en AWS Key Management Service (SSE-KMS).

Cifrado en tránsito

Para los datos en tránsito, Amazon Textract Texact utiliza Transport Layer Security (TLS) para cifrar los datos enviados entre el servicio y el agente. Además, Amazon Textract utiliza endpoints de VPC para enviar datos entre los distintos microservicios utilizados cuando Amazon Textract procesa un documento.

Privacidad del tráfico entre redes

Amazon Textract se comunica exclusivamente a través de puntos finales HTTPS, que son compatibles con todas las regiones compatibles con Amazon Textract

Identity and Access Management para Amazon Textract

AWS Identity and Access Management (IAM) es un servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién puede serautenticado(iniciado sesión) yautorizado(tienen permisos) para utilizar los recursos de Amazon Textract. IAM es un servicio de AWS que puede utilizar sin cargo adicional.

Temas

Público

Cómo se utilizaAWS Identity and Access Management(IAM) varía, en función del trabajo que realice en Amazon Textract.

Usuario de servicio: si utiliza el servicio Amazon Textract para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Amazon Textract para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos a su administrador. Si no puede acceder a una característica de Amazon Textract, consulteSolución de problemas de identidad y acceso de Amazon Textract.

Administrador de servicios— Si está a cargo de los recursos de Amazon Textract Texact de su empresa, probablemente tenga acceso completo a Amazon Textract. Su trabajo consiste en determinar qué a características y recursos de Amazon Textract pueden acceder los empleados. A continuación, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con Amazon Textract, consulteCómo funciona Amazon Textract con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer información sobre cómo escribir políticas para administrar el acceso a Amazon Textract. Para consultar ejemplos de políticas basadas en la identidad de Amazon Textract que puede utilizar en IAM, consulteEjemplos de políticas basadas en identidades de Amazon Textract.

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Para obtener más información acerca de cómo iniciar sesión con la AWS Management Console, consulte

Inicio de sesión en la AWS Management Console como usuario de IAM o usuario raíz en la Guía del usuario de IAM.

Debe estar autenticado (haber iniciado sesión en AWS) como el usuario raíz de la Cuenta de AWS, como un usuario de IAM o asumiendo un rol de IAM. También puede utilizar la autenticación de inicio de sesión único de la empresa o incluso iniciar sesión con Google o Facebook. En estos casos, su administrador habrá configurado previamente la federación de identidad mediante roles de IAM. Cuando obtiene acceso a AWS mediante credenciales de otra empresa, asume un rol indirectamente.

Para iniciar sesión directamente en la AWS Management Console, utilice la contraseña con su dirección de email de usuario raíz o con su nombre de usuario de IAM. Puede acceder a AWS mediante programación utilizando sus claves de acceso de usuario raíz o usuario de IAM. AWS proporciona SDK y herramientas de línea de comandos para firmar criptográficamente su solicitud con sus credenciales. Si no utiliza las herramientas de AWS, debe firmar usted mismo la solicitud. Para ello, utilice Signature Version 4, un protocolo para autenticar solicitudes de API de entrada. Para obtener más información acerca de cómo autenticar solicitudes, consulte Proceso de firma de Signature Version 4 en la Referencia general de AWS.

Independientemente del método de autenticación que utilice, es posible que también deba proporcionar información de seguridad adicional. Por ejemplo, AWS le recomienda el uso de la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Uso de la autenticación multifactor (MFA) en AWS en la Guía del usuario de IAM.

Cuenta de AWS usuario raíz

Cuando se crea por primera vez una Cuenta de AWS, se comienza con una única identidad de inicio de sesión que tiene acceso completo a todos los servicios y recursos de AWS de la cuenta. Esta identidad recibe el nombre de usuario raíz de la Cuenta de AWS y se accede a ella iniciando sesión con el email y la contraseña que utilizó para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz en sus tareas cotidianas, ni siquiera en las tareas administrativas. En lugar de ello, es mejor ceñirse a la práctica recomendada de utilizar el usuario final exclusivamente para crear al primer usuario de IAM. A continuación, guarde las credenciales del usuario raíz en un lugar seguro y utilícelas tan solo para algunas tareas de administración de cuentas y servicios.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad de la Cuenta de AWS que dispone de permisos específicos para una sola persona o aplicación. Un usuario de IAM puede tener credenciales a largo plazo, como un nombre de usuario y una contraseña o un conjunto de claves de acceso. Para obtener información sobre cómo generar claves de acceso, consulte Administración de claves de acceso de los usuarios de IAM en la Guía del usuario de IAM. Al generar claves de acceso para un usuario de IAM, asegúrese de ver y guardar de forma segura el par de claves. No puede recuperar la clave de acceso secreta en el futuro. En su lugar, debe generar un nuevo par de claves de acceso.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para obtener más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad de la Cuenta de AWS que dispone de permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente un rol de IAM en la AWS Management Consolecambiando de roles. Puede asumir un rol llamando a una operación de la AWS CLI o de la API de AWS, o utilizando una URL personalizada. Para obtener más información acerca de los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso de usuarios federados: en lugar de crear un usuario de IAM, puede utilizar identidades existentes de AWS Directory Service, del directorio de usuarios de su empresa o de un proveedor de identidades web. A estas identidades se les llama usuarios federados. AWS asigna una función a un usuario federado cuando se solicita acceso a través de un proveedor de identidad. Para

obtener más información acerca de los usuarios federados, consulte Usuarios federados y roles en la Guía del usuario de IAM.

- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunos servicios de AWS, puede asociar una política directamente a un recurso (en lugar de utilizar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.
- Acceso entre servicios: algunos servicios de AWS utilizan características de otros servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado a servicios.
  - Permisos principales: cuando utiliza un usuario o un rol de IAM para llevar a cabo acciones en AWS, se lo considera una entidad principal. Las políticas conceden permisos a una entidad principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. En este caso, debe tener permisos para realizar ambas acciones. Para ver si una acción requiere acciones dependientes adicionales en una política, consulteClaves de condición, recursos y acciones de Amazon Textracten laReferencia de autorizaciones de servicio.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un servicio de AWS en la Guía del usuario de IAM.
  - Rol vinculado a un servicio: un rol vinculado a un servicio es un tipo de función del servicio que está vinculado a un servicio de AWS. El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados a servicios aparecen en la cuenta de IAM y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.
- Aplicaciones que se ejecutan en Amazon EC2: puede utilizar un rol de IAM que le permita administrar credenciales temporales para las aplicaciones que se ejecutan en una instancia de EC2 y realizan solicitudes a la AWS CLI o a la API de AWS. Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar un rol de AWS a una instancia de EC2 y ponerla a disposición de todas las aplicaciones, cree un perfil de instancia asociado a

la misma. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para obtener más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades de IAM o recursos de AWS. Una política es un objeto de AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. Puede iniciar sesión como usuario raíz o usuario de IAM o puede asumir un rol de IAM. Cuando realiza una solicitud, AWS evalúa las políticas relacionadas basadas en identidades o en recursos. Los permisos en las políticas determinan si la solicitud se permite o se deniega. Las mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

Cada entidad de IAM (usuario o rol) comienza sin permisos. En otras palabras, de forma predeterminada, los usuarios no pueden hacer nada, ni siquiera cambiar sus propias contraseñas. Para conceder permiso a un usuario para hacer algo, el administrador debe asociarle una política de permisos. O bien el administrador puede agregar al usuario a un grupo que tenga los permisos necesarios. Cuando el administrador concede permisos a un grupo, todos los usuarios de ese grupo obtienen los permisos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con dicha política puede obtener información del usuario de la AWS Management Console, la AWS CLI o la API de AWS.

Políticas basadas en la identidad

Las políticas basadas en identidades son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas

controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y bajo qué condiciones. Para obtener más información sobre cómo crear una política basada en identidades, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede asociar a varios usuarios, grupos y roles de su Cuenta de AWS. Las políticas administradas incluyen las políticas administradas por AWS y las políticas administradas por el cliente. Para obtener más información acerca de cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política basada en recursos. Las entidades principales pueden incluir cuentas, usuarios, roles, usuarios federados o servicios de AWS.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas por AWS en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de política JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite otros tipos de políticas adicionales menos frecuentes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le otorgan.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidades puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una identidad. Los permisos resultantes son la intersección de las políticas basadas en identidades de la entidad y los límites de sus permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicio (SCP): las SCP son políticas de JSON que especifican los permisos máximos de una organización o una unidad organizativa en AWS Organizations. AWS Organizations es un servicio que le permite agrupar y administrar de manera centralizada varias Cuentas de AWS que posea su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o todas sus cuentas. Las SCP limitan los permisos de las entidades de las cuentas miembro, incluido cada usuario raíz de la Cuenta de AWS. Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política basada en recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información acerca de cómo AWS decide si permitir o no una solicitud cuando hay varios tipos de políticas implicados, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon Textract con IAM

Antes de utilizar IAM para administrar el acceso a Amazon Textract Texact, debe conocer qué características de IAM se encuentran disponibles con Amazon Textract. Para obtener una perspectiva general de cómo Amazon Textract y otrosAWSLos servicios funcionan con IAM, consulteAWSServicios que funcionan con IAMen laIAM User Guide.

Temas

Políticas basadas en identidades de Amazon Textract

Con las políticas basadas en identidades de IAM, puede especificar las acciones permitidas o denegadas, así como los recursos y las condiciones en las que se permiten o deniegan las acciones. Amazon Textract admite acciones, claves de condiciones y recursos específicos. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte Referencia de los elementos de las políticas JSON de IAM en la Guía del usuario de IAM.

Acciones

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede llevar a cabo acciones en qué recursos y bajo qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para permitir o denegar el acceso en una política. Las acciones de la política generalmente tienen el mismo nombre que la operación de API de AWS asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones asíncronas de Amazon Textract requieren dos permisos de acción que se otorguen, uno para acciones de inicio y otro para acciones Get. Además, si utilizas un depósito de Amazon S3 para pasar documentos, tendrás que conceder acceso de lectura a tu cuenta.

En Amazon Textract, todas las acciones de política empiezan por: `textract:`. Por ejemplo, para conceder a alguien permiso para ejecutar una operación de Amazon Textract Texact con Amazon Textract`AnalyzeDocument`, incluye el`textract:AnalyzeDocument`acción en su política. Las instrucciones de la política deben incluir un elemento `Action` o un elemento `NotAction`. Amazon

Textract Texact define su propio conjunto de acciones que describen las tareas que se pueden realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo.

```
"Action": [
      "textract:action1",
      "textract:action2"
```

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción.

```
"Action": "textract:Describe*"
```

Para obtener una lista de acciones de Amazon Textract, consulteAcciones definidas por Amazon Textracten laIAM User Guide.

Recursos

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Amazon Textract no admite especificar los ARN de los recursos en una política.

Claves de condición

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición con una operación lógica `OR`. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para obtener más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

Amazon Textract Texact no proporciona ninguna clave de condición específica del servicio, pero sí admite el uso de algunas claves de condición globales. Para ver una lista de todas lasAWSClaves de condición globales, consulteAWSClaves de contexto de condición globales deen laIAM User Guide.

Ejemplos

Para ver ejemplos de políticas basadas en identidades de Amazon Textract, consulteEjemplos de políticas basadas en identidades de Amazon Textract.

Políticas basadas en recursos de Amazon Textract

Amazon Textract no admite las políticas basadas en recursos.

Autorización basada en etiquetas de Amazon Textract

Amazon Textract no admite el etiquetado de recursos ni el control de acceso basado en etiquetas.

Roles de IAM de Amazon Textract

Un rol de IAM es una entidad de la cuenta de AWS que dispone de permisos específicos.

Uso de credenciales temporales con Amazon Textract

Puede utilizar credenciales temporales para iniciar sesión con federación, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Las credenciales de seguridad temporales se obtienen mediante una llamada a operaciones de la API de AWS STS, como AssumeRole o GetFederationToken.

Amazon Textract admite el uso de credenciales temporales.

Roles vinculados a servicios

Los roles vinculados a servicios permiten a los servicios de AWS obtener acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles vinculados a servicios aparecen en la cuenta de IAM y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Amazon Textract no admite roles vinculados a servicios.

> **Note**
>
> Como Amazon Textract Texact no admite roles vinculados a servicios, no admite los principios de servicio de AWS. Para obtener más información acerca de los directores de servicio, consulteEntidades de servicios de AWSen laIAM User Guide

Roles de servicio

Esta característica permite que un servicio asuma un rol de servicio en su nombre. Este rol permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en su cuenta de IAM y son propiedad de la cuenta. Esto significa que un administrador de IAM puede cambiar los permisos de este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

Amazon Textract admite roles de servicio.

Ejemplos de políticas basadas en identidades de Amazon Textract

De forma predeterminada, los usuarios y roles de IAM no tienen permiso para crear ni modificar recursos de Amazon Textract. Tampoco pueden realizar tareas mediante la AWS Management

Console, la AWS CLI, o la API de AWS. Un administrador de IAM debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe adjuntar esas políticas a los usuarios o grupos de IAM que necesiten esos permisos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM con estos documentos de políticas JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Temas

- Prácticas recomendadas relativas a políticas
- Permitir a los usuarios ver sus propios permisos
- Acceso a operaciones sincrónicas en Amazon Textract
- Acceso a operaciones asíncronas en Amazon Textract

Prácticas recomendadas relativas a políticas

Las políticas basadas en identidades son muy eficaces. Determinan si alguien puede crear, acceder o eliminar recursos de Amazon Textract de la cuenta. Estas acciones pueden generar costes adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Aprenda a usarAWSpolíticas administradas— Para empezar a utilizar Amazon Textract Texact rápidamente, utiliceAWSpolíticas administradas para proporcionar a los empleados los permisos necesarios. Estas políticas ya están disponibles en su cuenta, y las mantiene y actualiza AWS. Para obtener más información, consulte Introducción sobre el uso de permisos con políticas administradas por AWS en la Guía del usuario de IAM.
- Conceder privilegios mínimos: al crear políticas personalizadas, conceda solo los permisos necesarios para llevar a cabo una tarea. Comience con un conjunto mínimo de permisos y conceda permisos adicionales según sea necesario. Por lo general, es más seguro que comenzar con permisos que son demasiado tolerantes e intentar hacerlos más estrictos más adelante. Para obtener más información, consulte Conceder privilegios mínimos en la Guía del usuario de IAM.
- Habilitar la MFA para operaciones confidenciales: para mayor seguridad, obligue a los usuarios de IAM a utilizar la autenticación multifactor (MFA) para acceder a recursos u operaciones de API confidenciales. Para obtener más información, consulte Uso de la autenticación multifactor (MFA) en AWS en la Guía del usuario de IAM.

- Utilizar condiciones de política para mayor seguridad: en la medida en que sea práctico, defina las condiciones en las que las políticas basadas en identidades permitan el acceso a un recurso. Por ejemplo, puede escribir condiciones para especificar un rango de direcciones IP permitidas desde el que debe proceder una solicitud. También puede escribir condiciones para permitir solicitudes solo en un intervalo de hora o fecha especificado o para solicitar el uso de SSL o MFA. Para obtener más información, consulteElemento de la política de JSON de IAM: Condiciónen laIAM User Guide.

Permitir a los usuarios ver sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de AWS.

```
{
    "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": "*"
        }
    ]
}
```

Acceso a operaciones sincrónicas en Amazon Textract

En esta política de ejemplo se concede acceso a las acciones sincrónicas de Amazon Textract a un usuario de IAM en suAWSaccount.

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument"
            ],
            "Resource": "*"
        }
    ]
```

Acceso a operaciones asíncronas en Amazon Textract

En la siguiente política de ejemplo se proporciona a un usuario de IAM en suAWSacceso a la cuenta a todas las operaciones asíncronas utilizadas en Amazon Textract.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:StartDocumentTextDetection",
                "textract:StartDocumentAnalysis",
                "textract:GetDocumentTextDetection",
                "textract:GetDocumentAnalysis"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

Solución de problemas de identidad y acceso de Amazon Textract

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que es posible que surjan cuando trabaje con Amazon Textract e IAM.

Temas

- No tengo autorización para realizar una acción en Amazon Textract
- No tengo autorización para realizar la operación iam:PassRole
- Quiero ver mis claves de acceso
- Soy administrador y deseo permitir que otros obtengan acceso a Amazon Textract
- Quiero permitir a personas externas a miAWSCuenta para acceder a los recursos de My Amazon Textract

No tengo autorización para realizar una acción en Amazon Textract

Si la AWS Management Console le indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su administrador para recibir ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

El siguiente error de ejemplo se produce cuando el`mateojackson`El usuario de IAM intenta ejecutar`DetectDocumentText`en una imagen de prueba pero no tiene`textract:DetectDocumentText`permisos.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 textract:DetectDocumentText on resource: textimage.png
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso `textimage.png` mediante la acción `textract:DetectDocumentText`.

No tengo autorización para realizar la operación iam:PassRole

Si recibe un error que indica que no está autorizado para llevar a cabo la acción `iam:PassRole`, debe ponerse en contacto con su administrador para recibir ayuda. Su administrador es la persona

que le facilitó su nombre de usuario y contraseña. Pida a la persona que actualice las políticas de forma que pueda transferir un rol a Amazon Textract.

Algunos servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado`marymajor`intenta utilizar la consola para realizar una acción en Amazon Textract. Sin embargo, la acción requiere que el servicio cuente con permisos otorgados por un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, Mary pide a su administrador que actualice sus políticas para que pueda realizar la acción `iam:PassRole`.

Quiero ver mis claves de acceso

Después de crear sus claves de acceso de usuario de IAM, puede ver su ID de clave de acceso en cualquier momento. Sin embargo, no puede volver a ver su clave de acceso secreta. Si pierde la clave de acceso secreta, debe crear un nuevo par de claves de acceso.

Las claves de acceso se componen de dos partes: un ID de clave de acceso (por ejemplo, `AKIAIOSFODNN7EXAMPLE`) y una clave de acceso secreta (por ejemplo, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). El ID de clave de acceso y la clave de acceso secreta se utilizan juntos, como un nombre de usuario y contraseña, para autenticar sus solicitudes. Administre sus claves de acceso con el mismo nivel de seguridad que para el nombre de usuario y la contraseña.

> **Important**
>
> No proporcione las claves de acceso a terceros, ni siquiera para que le ayuden a buscar el ID de usuario canónico. Si lo hace, podría conceder a otra persona acceso permanente a su cuenta.

Cuando cree un par de claves de acceso, se le pide que guarde el ID de clave de acceso y la clave de acceso secreta en un lugar seguro. La clave de acceso secreta solo está disponible en

el momento de su creación. Si pierde la clave de acceso secreta, debe agregar nuevas claves de acceso a su usuario de IAM. Puede tener un máximo de dos claves de acceso. Si ya cuenta con dos, debe eliminar un par de claves antes de crear uno nuevo. Para consultar las instrucciones, consulte Administración de claves de acceso en la Guía del usuario de IAM.

Soy administrador y deseo permitir que otros obtengan acceso a Amazon Textract

Para permitir que otros accedan a Amazon Textract Texact, debe crear una entidad de IAM (usuario o rol) para la persona o aplicación que necesita acceso. Esta persona utilizará las credenciales de la entidad para acceder a AWS. A continuación, debe asociar una política a la entidad que les conceda los permisos correctos en Amazon Textract.

Para comenzar de inmediato, consulte Creación del primer grupo y usuario delegado de IAM en la Guía del usuario de IAM.

Quiero permitir a personas externas a miAWSCuenta para acceder a los recursos de My Amazon Textract

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon Textract admite estas características, consulteCómo funciona Amazon Textract con IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuentas de AWS de su propiedad, consulte Proporcionar acceso a un usuario de IAM a otra Cuenta de AWS de la que es propietario en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos a Cuentas de AWS de terceros, consulte Proporcionar acceso a Cuentas de AWS que son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una identidad federada, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.

- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Registro y monitorización

Para supervisar Amazon Textract, utilice Amazon CloudWatch. Esta sección proporciona información sobre cómo configurar la monitorización de Amazon Textract. También proporciona contenido de referencia para las métricas de Amazon Textract Texact.

Temas

Monitorización de Amazon Textract

Con CloudWatch, obtendrá métricas de las operaciones de Amazon Textract individuales o las métricas de Amazon Textract Texact globales de su cuenta. Puede utilizar las métricas para hacer un seguimiento del estado de la solución basada en Amazon Textract y configurar alarmas que le avisen cuando una o varias métricas superen un umbral definido. Por ejemplo, puede ver métricas del número de errores de servidor que se han producido. También puede consultar métricas del número de veces que se ha realizado correctamente una operación de Amazon Textract específica. Para ver las métricas, puede utilizarAmazon CloudWatch, elAWS CLI, o elAPI de CloudWatch.

Uso de las métricas de CloudWatch para Amazon Textract

Para utilizar métricas, debe especificar la siguiente información:

- La dimensión de la métrica o ninguna dimensión. Una dimensión es un par de nombre-valor que le ayuda a identificar una métrica de forma inequívoca. Amazon Textract tiene una dimensión denominadaOperación. Proporciona métricas para una operación específica. Si no especifica ninguna dimensión, el ámbito de la métrica se establece en todas las operaciones de Amazon Textract Texact dentro de su cuenta.
- El nombre de la métrica, como `UserErrorCount`.

Puede obtener datos de monitorización de la Amazon Textract mediante laAWS Management Console, elAWS CLIo API de CloudWatch. También puede utilizar la API de CloudWatch API a través de uno de los kits de desarrollo de software (SDK) de Amazon AWS o las herramientas de la API de CloudWatch. La consola muestra una serie de gráficos basados en los datos sin procesar de la API de CloudWatch. En función de sus necesidades, es posible que prefiera utilizar los gráficos que se muestran en la consola o que se recuperan de la API.

En la siguiente lista se indican algunos usos frecuentes de las métricas. Se trata de sugerencias que puede usar como punto de partida y no de una lista completa.

¿Cómo?	Métricas relevantes
¿Cómo puedo saber si mi aplicación ha alcanzado el número máximo de solicitudes por segundo?	Monitorice la estadística `Sum` de la métrica `ThrottledCount` .
¿Cómo puedo monitorizar los errores de solicitud?	Utilice la estadística `Sum` de la métrica `UserErrorCount` .
¿Cómo puedo encontrar el número total de solicitudes?	Utilice la estadística `SampleCount` de la métrica `ResponseTime` . Esto incluye cualquier solicitud que genere un error. Si desea ver únicamente las llamadas a operacion es que se han realizado con éxito, use la métrica `SuccessfulRequestCount` .
¿Cómo puedo monitorizar la latencia de las llamadas a operaciones de Amazon Textract?	Utilice la métrica `ResponseTime` .

Debe disponer de los permisos de CloudWatch adecuados para monitorear Amazon Textract con CloudWatch. Para obtener más información, consulte Autenticación y control de acceso de Amazon CloudWatch.

Acceso a las métricas de Amazon Textract

En los siguientes ejemplos se muestra cómo tener acceso a métricas de Amazon Textract mediante la consola de CloudWatch, laAWS CLIy la API de CloudWatch.

Para ver las métricas (consola)

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. ElegirMétricas, elige elTodas las métricaspestaña y, a continuación, elijaAmazon Textract.
3. ElegirPor operacióny, a continuación, elija una métrica.

   Por ejemplo, elijaStartDocumentAnalysismétrica para medir cuántas veces se ha iniciado el análisis de documentos asíncrono.
4. Elija un valor para el intervalo de fechas. El número de métricas se muestra en el gráfico.

Para ver las métricas para tener éxito**`StartDocumentAnalysis`**llamadas a operaciones que se han realizado durante un periodo de tiempo (CLI)

- Abra la AWS CLI y escriba el siguiente comando:

  ```
  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
  ```

  Este ejemplo muestra las llamadas a la operación `StartDocumentAnalysis` que se han realizado correctamente durante un periodo de tiempo. Para obtener más información, consulte get-metric-statistics.

Para tener acceso a las métricas (API de CloudWatch)

- Llame a `GetMetricStatistics`. Para obtener más información, consulte laReferencia de API de Amazon CloudWatch.

Crear una alarma

Puede crear una alarma de CloudWatch que envíe un mensaje de Amazon Simple Notification Service (Amazon SNS) cuando la alarma cambia de estado. Una alarma vigila una métrica determinada durante el periodo especificado. Realiza una o varias acciones según el valor de

la métrica con respecto a un umbral dado durante varios períodos de tiempo. La acción es una notificación que se envía a un tema de Amazon SNS o a una política de Auto Scaling.

Las alarmas invocan acciones únicamente para los cambios de estado prolongados. Las alarmas de CloudWatch no invocan acciones simplemente porque estén en un estado particular. El estado debe haber cambiado y debe haberse mantenido durante el número de periodos de tiempo especificado.

Para configurar una alarma (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, elijaAlarmas, y elijaCrear alarma. Esto abreAsistente de creación de alarmas de.
3. Elija Select Metric (Seleccionar métrica).
4. En el navegadorTodas las métricaspestaña, elijaTextract.
5. ElegirPor operacióny, a continuación, elija una métrica.

   Por ejemplo, elijaStartDocumentAnalysispara definir una alarma para un número máximo de operaciones de análisis de documentos asíncronas.
6. Elija la pestaña Graphed metrics.
7. En Statistic (Estadística), elija Sum (Suma).
8. Elija Select Metric (Seleccionar métrica).
9. Rellene Name y Description. Para Whenever, elija >= e introduzca un valor máximo de su elección.
10. Si desea que CloudWatch le envíe un correo electrónico cuando se alcance el estado de la alarma, paraSiempre que esta alarma:, eligeEl estado es ALARM. Para enviar alarmas a un tema de Amazon SNS existente, enEnviar notificación a:, elija un tema de SNS existente. Para definir el nombre y las direcciones de correo electrónico para una nueva lista de suscripción de correo electrónico, elijaNueva lista. CloudWatch guarda la lista y la muestra en el campo para que pueda utilizarla para definir nuevas alarmas.

> **Note**
>
> Si usaNueva listaPara crear un nuevo tema de Amazon SNS, debe verificar las direcciones de correo electrónico para que los destinatarios previstos puedan recibir las notificaciones. Amazon SNS envía solo mensajes de correo electrónico cuando

la alarma entra en un estado de alarma. Si este cambio en el estado de la alarma se produce antes de que se verifiquen las direcciones de correo electrónico, los destinatarios no reciben ninguna notificación.

11. Elija Create Alarm.

Para configurar una alarma (AWS CLI)

- Abra la AWS CLI y escriba el siguiente comando. Cambiar el valor de`alarm-actions`para hacer referencia al tema de Amazon SNS que ha creado anteriormente.

```
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
```

  Este ejemplo muestra cómo crear una alarma cuando se producen más de 10 errores de usuario en 5 minutos para las llamadas a`StartDocumentAnalysis`. Para obtener más información, consulte put-metric-alarm.

Para configurar una alarma (API de CloudWatch)

- Llame a `PutMetricAlarm`. Para obtener más información, consulteReferencia de API de Amazon CloudWatch.

Métricas de CloudWatch para Amazon Textract

Esta sección contiene información acerca de las métricas de Amazon CloudWatch y laOperaciónque están disponibles para Amazon Textract.

También puede ver una vista completa de métricas de Amazon Textract desde la consola de Amazon Textract.

Métricas de CloudWatch para Amazon Textract

En la siguiente tabla se resumen las métricas de Amazon Textract.

Métrica	Descripción
SuccessfulRequestCount	El número de solicitudes realizadas correctamente. El intervalo de códigos de respuesta para una solicitud realizada correctamente comprende de 200 a 299. Unidad: Recuento Estadísticas válidas: `Sum,Average`
ThrottledCount	El número de solicitudes restringidas. Amazon Textract restringe una solicitud cuando recibe más solicitudes que el límite de transacciones por segundo de su cuenta. Si el límite establecido para su cuenta se supera con frecuencia, puede solicitar un aumento del límite. Para solicitar un aumento, consulte Límites de los servicios de AWS. Unidad: Recuento Estadísticas válidas: `Sum,Average`
ResponseTime	El tiempo en milisegundos que tarda Amazon Textract en calcular la respuesta. Unidades: 1. Recuento para la estadística `Data Samples` 2. Milisegundos para la estadística `Average` Estadísticas válidas: `Data Samples,Average`

Métrica	Descripción
	Note La `ResponseTime` métrica no se incluye en el panel de métricas de Amazon Textract.
ServerErrorCount	El número de errores de servidor. El intervalo de códigos de respuesta de un error de servidor comprende de 500 a 599. Unidad: Recuento Estadísticas válidas: `Sum,Average`
UserErrorCount	El número de errores de usuario (parámetros no válidos, imagen no válida, sin permiso, etc.). El intervalo de códigos de respuesta de un error de usuario comprende de 400 a 499. Unidad: Recuento Estadísticas válidas: `Sum,Average`

Dimension de CloudWatch para Amazon Textract

Para recuperar métricas específicas de la operación, utilice el espacio de nombres `AWS/Textract` y proporcione una dimensión de operación. Para obtener más información acerca de las dimensiones, consulteDimensionesen laGuía del usuario de Amazon CloudWatch.

Registro de llamadas a la API de Amazon Textract conAWS CloudTrail

Amazon Textract se integra conAWS CloudTrail, un servicio que proporciona un registro de las acciones que realiza un usuario, un rol o un rolAWSservicio en Amazon Textract. CloudTrail captura todas las llamadas a la API para Amazon Textract como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de Amazon Textract llamadas desde el código a las operaciones de la API de Amazon Textract Texact.

Si crea un registro de seguimiento, puede habilitar la entrega continua de eventos de CloudTrail a un bucket de Amazon S3, incluidos los eventos de Amazon Textract. Si no configura un registro de seguimiento, puede ver los eventos más recientes de la consola de CloudTrail en el Event history (Historial de eventos). Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon Textract Texact, la dirección IP desde la que se realizó la solicitud, quién realizó la solicitud, cuándo se realizó y detalles adicionales.

Para obtener más información acerca de CloudTrail, consulte la AWS CloudTrailGuía del usuario de .

Información de Amazon Textract en CloudTrail

CloudTrail se habilita en su cuenta de AWS cuando la crea. Cuando se produce una actividad en Amazon Textract Texact, esta se registra en un evento de CloudTrail junto con otrosAWSeventos de servicio enHistorial de eventos. Puede ver, buscar y descargar los últimos eventos de la cuenta de AWS. Para obtener más información, consulte Ver eventos con el historial de eventos de CloudTrail.

Para mantener un registro continuo de los eventos deAWS, incluidos los eventos de Amazon Textract, cree un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. El registro de seguimiento registra los eventos de todas las regiones de la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. Además, puede configurar otrosAWSservicios para analizar y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte los siguientes:

- Introducción a la creación de registros de seguimiento
- Servicios e integraciones compatibles con CloudTrail
- Configuración de notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de registro de CloudTrail de varias regiones y Recibir archivos de registro de CloudTrail de varias cuentas

CloudTrail registra todas las operaciones de Amazon Textract y se documentan en laReferencia de la API. Por ejemplo, las llamadas a las acciones `DetectDocumentText`, `AnalyzeDocument` y `GetDocumentText` generan entradas en los archivos de log de CloudTrail.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario AWS Identity and Access Management (IAM) o credenciales de usuario raíz.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail.

Parámetros de solicitud y campos de respuesta que no están registrados

Por motivos de privacidad, algunos parámetros de solicitud y campos de respuesta no se registran, por ejemplo, bytes de imagen de solicitud o información del cuadro delimitador de respuestas. Los nombres de bucket y los nombres de archivo de Amazon S3 suministrados en los parámetros de solicitud se proporcionan en las entradas de registro de CloudTrail No se proporciona información sobre los bytes de imagen transmitidos en una solicitud en un registro de CloudTrail. En la tabla siguiente se muestran los parámetros de entrada y los parámetros de respuesta que no se registran para cada operación de Amazon Textract Texact.

Operación	Parámetros de solicitud	Campos de respuesta
AnalyzeDocument	`Bytes`	Todos
DetectDocumentText	`Bytes`	Todos
StartDocumentAnalysis	Ninguno	Ninguno
GetDocumentAnalysis	Ninguno	Todos
StartDocumentTextDetection	Ninguno	Ninguno
GetDocumentTextDetection	Ninguno	Todos

Descripción de las entradas de archivos de registro de Amazon Textract

Un registro de seguimiento es una configuración que permite la entrega de eventos como archivos de registros en un bucket de Amazon S3 que especifique. Los archivos log de CloudTrail pueden contener una o varias entradas de log. Un evento representa una única solicitud de cualquier origen e incluye información sobre la operación solicitada, la fecha y la hora de la operación, los parámetros

de la solicitud, etcétera. Los archivos de registro de CloudTrail no rastrean el orden en la pila de las llamadas públicas a la API, por lo que estas no aparecen en ningún orden específico.

En el ejemplo que sigue se muestra una entrada de registro de CloudTrail que ilustra la operación `AnalyzeDocument`. Los bytes de imagen de la entrada`document`y los resultados del análisis (`responseElements`) no están registrados.

```
{
    "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"
}
```

En el siguiente ejemplo, se muestra una entrada de registro de CloudTrail para la`StartDocumentAnalysis`. La entrada de registro incluye el nombre del bucket de Amazon S3 y el nombre del archivo de imagen en`documentLocation`. El registro también incluye la respuesta de la operación.

```
{
```

```
    "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"
        }

    ]
}
```

Validación de la conformidad para Amazon Textract

Auditores externos evalúan la seguridad y la conformidad de Amazon Textract como parte de variosAWSprogramas de conformidad. Estos incluyen HIPAA, SOC, ISO y PCI.

> **Note**
>
> Si está procesando datos a través del servicio Textract que está sujeto al cumplimiento PCI DSS, debe cancelar su cuenta poniéndose en contacto con AWS Support y siguiendo el proceso que se le ha proporcionado.

Para obtener una lista de servicios de AWS en el ámbito de programas de conformidad específicos, consulte Servicios de AWS en el ámbito del programa de conformidad. Para obtener información general, consulte Programas de conformidad de AWS.

Puede descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Amazon Texact se determina en función de la confidencialidad de los datos, los objetivos de conformidad de su empresa, así como de la legislación y los reglamentos aplicables.AWSproporciona los siguientes recursos para ayudar con la conformidad:

- Security and Compliance Quick Start Guides (Guías de inicio rápido de seguridad y conformidad) (Guías de inicio rápido de seguridad y conformidad): Estas guías de implementación analizan consideraciones sobre arquitectura y proporcionan los pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- Documento técnico sobre arquitectura para seguridad y conformidad de HIPAA : en este documento técnico, se describe cómo las empresas pueden utilizar AWS para crear aplicaciones conformes con HIPAA.
- Recursos de conformidad de AWS: este conjunto de manuales y guías podría aplicarse a su sector y ubicación.
- Evaluación de recursos con reglas en la Guía para desarrolladores de AWS Config: el servicio AWS Config evalúa en qué medida las configuraciones de sus recursos cumplen las prácticas internas, las directrices del sector y las normativas.

- AWS Security Hub— EstoAWSproporciona una perspectiva completa de su estado de seguridad dentro deAWS. El centro de seguridad le ayuda a verificar el grado de conformidad con las prácticas recomendadas y los estándares sectoriales.

Resiliencia en Amazon Textract

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las regiones y zonas de disponibilidad de AWS, consulte Infraestructura global de AWS.

> **Note**
>
> No se permite la transferencia de datos entre regiones debido al Reglamento General de Protección de Datos (RGPD).

Seguridad de la infraestructura en Amazon Textract

Al tratarse de un servicio administrado, Amazon Textract está protegido por elAWSprocedimientos de seguridad de red globales que se describen en elAmazon Web Services: Información general de los procesos de seguridaddocumento técnico.

UsasAWSpublicadas en la API de para obtener acceso a Amazon Textract a través de la red. Los clientes deben ser compatibles con Transport Layer Security (TLS) 1.0 o una versión posterior. Recomendamos TLS 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad de seguridad de IAM. También puede utilizar AWS

Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Configuración y análisis de vulnerabilidades en Amazon Textract

La configuración y los controles de TI son una responsabilidad compartida entre AWS y usted, nuestro cliente. Para obtener más información, consulte el modelo de responsabilidad compartida de AWS.

Puntos de enlace de la VPC de interfaz y Textract (AWS PrivateLink)

Puede establecer una conexión privada entre su VPC y Amazon Textract mediante la creación de unpunto de enlace de la VPC de la interfaz. Los puntos de enlace de tipo interfaz cuentan conAWS PrivateLink, una tecnología que le permite acceder de forma privada a las API de Amazon Textract sin una gateway a Internet, un dispositivo NAT, una conexión VPN o una AWS Direct Connect conexión. Las instancias de la VPC no necesitan direcciones IP públicas para comunicarse con las API de Amazon Textract. El tráfico entre la VPC y Amazon Textract no sale de la red de AWS.

Cada punto de enlace de la interfaz está representado por una o más interfaces de red elásticas en las subredes.

Para obtener más información, consulte Puntos de enlace de la VPC de tipo interfaz (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Consideraciones para los puntos de enlace de la VPC de Amazon Textract

Antes de configurar un punto de enlace de la VPC de interfaz para Amazon Textract, asegúrese de revisarPropiedades y limitaciones de los puntos de enlace de interfazen laAmazon VPC User Guide.

Amazon Textract admite realizar llamadas a todas sus acciones de API desde su VPC.

Creación de un punto de enlace de la VPC de interfaz para Amazon Textract

Puede crear un punto de enlace de la VPC para el servicio Amazon Textract mediante la consola de Amazon VPC o laAWS Command Line Interface(AWS CLI). Para obtener más información, consulte Creación de un punto de enlace de interfaz en la Guía del usuario de Amazon VPC.

Cree un punto de enlace de la VPC para Amazon Textract utilizando el siguiente nombre de servicio:

- com.amazonaws.*región*.textract: para crear un endpoint para la mayoría de las operaciones de Amazon Textract Texact.
- com.amazonaws.*región*.textract-fips: para crear un punto de enlace para Amazon Textract Texact que cumpla con la norma del gobierno de EE. UU. Federal Information Processing Standard (FIPS) Publication 140-2.

Si habilita un DNS privado para el punto de enlace, puede realizar solicitudes de API a Amazon Textract Texact utilizando el nombre de DNS predeterminado de la región, por ejemplo,`textract.us-east-1.amazonaws.com`.

Para obtener más información, consulte Acceso a un servicio a través de un punto de enlace de interfaz en la Guía del usuario de Amazon VPC.

Creación de una política de punto de enlace de la VPC para Amazon Textract

Puede asociar una política de puntos de enlace con su punto de enlace de la VPC que controla el acceso a Amazon Textract. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para obtener más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la guía del usuario de Amazon VPC.

Ejemplo: Política de puntos de enlace de la VPC para acciones de Amazon Textract

A continuación, se muestra un ejemplo de una política de punto de enlace para Amazon Textract. Cuando se asocia a un punto de enlace, esta política concede acceso a las acciones de Amazon Textract enumeradas para todos las entidades principales de todos los recursos.

Este ejemplo de política permite obtener acceso solo a las operaciones`DetectDocumentText`y`AnalyzeDocument`. Los usuarios aún pueden llamar a operaciones de Amazon Textract desde fuera del punto de enlace de la VPC.

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "textract:DetectDocumentText",
            "textract:AnalyzeDocument",
         ],
         "Resource":"*"
      }
   ]
}
```

Referencia de la API

Esta sección proporciona documentación sobre las operaciones de la API de Amazon Textract.

Temas

Acciones

Se admiten las siguientes acciones:

AnalyzeDocument

Analiza un documento de entrada en busca de relaciones entre elementos detectados.

Los tipos de información devuelta son los siguientes:

- Datos de formulario (pares de clave-valor). La información relacionada se devuelve en dosBlockobjetos, cada uno de tipoKEY_VALUE_SET: una LLAVEBlockobjeto y un VALORBlockobjeto. Por ejemplo,Name: Ana Silva Carolinacontiene una clave y un valor. Name:es la clave. Ana Silva Carolinaes el valor.
- Datos de celdas de tabla y tabla. UNA TABLABlockcontiene información sobre una tabla detectada. UNA CELDABlockse devuelve para cada celda de una tabla.
- Líneas y palabras de texto. UNA LÍNEABlockcontiene uno o varios WORDBlockobjetos. Se devuelven todas las líneas y palabras detectadas en el documento (incluido el texto que no tiene relación con el valor de`FeatureTypes`).

Los elementos de selección, tales como casillas de verificación y botones de opción (botones de opción) se pueden detectar en los datos del formulario y en las tablas. UN ELEMENTO SELECTION_ELEMENTBlockcontiene información sobre un elemento de selección, incluido el estado de la selección.

Puede elegir qué tipo de análisis desea realizar especificando la`FeatureTypes`lista.

La salida se devuelve en una lista de`Block`objetos.

`AnalyzeDocument`es una operación síncrona. Para analizar documentos de forma asíncrona, utiliceStartDocumentAnalysis.

Para obtener más información, consulteAnálisis de texto de documentos.

Sintaxis de la solicitud

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
```

```
    },
    "FeatureTypes": [ "string" ],
    "HumanLoopConfig": {
       "DataAttributes": {
          "ContentClassifiers": [ "string" ]
       },
       "FlowDefinitionArn": "string",
       "HumanLoopName": "string"
    }
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

Document

El documento de entrada como bytes codificados en base64 o un objeto Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Textract Texact, no puede pasar bytes de imagen. El documento debe ser una imagen en formato JPEG, PNG, PDF o TIFF.

Si utiliza un SDK de AWS para llamar a Amazon Textract, es posible que no tenga que codificar en base 64 bytes de imagen que se pasan mediante el`Bytes`.

Tipo: objeto Document

Obligatorio: Sí

FeatureTypes

Lista de los tipos de análisis que se van a realizar. Agregue TABLES a la lista para devolver información sobre las tablas detectadas en el documento de entrada. Añada FORMULARIOS para devolver los datos del formulario detectados. Para realizar ambos tipos de análisis, agregue TABLES y FORMS a`FeatureTypes`. Todas las líneas y palabras detectadas en el documento se incluyen en la respuesta (incluido el texto que no está relacionado con el valor de`FeatureTypes`).

Type: Matriz de cadenas

Valores válidos: `TABLES | FORMS`

Obligatorio: Sí

HumanLoopConfig

Establece la configuración del flujo de trabajo humano en bucle para analizar documentos.

Tipo: objeto HumanLoopConfig

Obligatorio: No

Sintaxis de la respuesta

```
{
   "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"
      }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

AnalyzeDocumentModelVersion

La versión del modelo que se utiliza para analizar el documento.

Type: Cadena

Blocks

Los elementos detectados y analizados por`AnalyzeDocument`.

Type: Matriz deBlockobjects

DocumentMetadata

Metadatos sobre el documento analizado. Un ejemplo es el número de páginas.

Tipo: objeto DocumentMetadata

HumanLoopActivationOutput

Muestra los resultados de la evaluación humana en bucle.

Tipo: objeto HumanLoopActivationOutput

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

BadDocumentException

Amazon Textract Texact no puede leer el documento. Para obtener más información sobre los límites de documentos en Amazon Textract, consulteLímites máximos de Amazon Textract.

Código de estado HTTP: 400

DocumentTooLargeException

El documento no se puede procesar porque es demasiado grande. Tamaño máximo de documento para operaciones síncronas de 10 MB. El tamaño máximo de documento para las operaciones asíncronas es de 500 MB para los archivos PDF.

Código de estado HTTP: 400

HumanLoopQuotaExceededException

Indica que ha superado la cantidad máxima de personas activas en los flujos de trabajo de bucle disponibles

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact no puede obtener acceso al objeto de S3 que se especifica en la solicitud. Para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

UnsupportedDocumentException

El formato del documento de entrada no se admite. Los documentos para operaciones pueden estar en formato PNG, JPEG, PDF o TIFF.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python

- SDK de AWS para Ruby V3

AnalyzeExpense

`AnalyzeExpense`analiza sincrónicamente un documento de entrada para ver las relaciones financieras entre texto.

La información se devuelve como`ExpenseDocuments`y separados de la siguiente manera.

- `LineItemGroups`- Conjunto de datos que contiene`LineItems`que almacenan información sobre las líneas de texto, como un artículo comprado y su precio en un recibo.
- `SummaryFields`- Contiene toda la demás información de un recibo, como la información del encabezado o el nombre del proveedor.

Sintaxis de la solicitud

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

Document

El documento de entrada, ya sea en bytes o como objeto S3.

Puede transferir bytes de imágenes a una operación API Amazon Textract Texact utilizando la`Bytes`propiedad. Por ejemplo, usaría el`Bytes`para pasar un documento cargado desde un sistema de archivos local. Bytes de imagen pasados mediante el`Bytes`debe tener codificación base64. Es posible que el código no necesite codificar bytes de archivos de documentos si utiliza un SDK de AWS para llamar a las operaciones de la API de Amazon Textract Texact.

Puede transferir imágenes almacenadas en un bucket de S3 a una operación API Amazon Textract Texact utilizando el`S3Object`propiedad. Los documentos almacenados en un bucket de S3 no tienen por qué estar codificados en base64.

La región de AWS para el bucket de S3 que contiene el objeto S3 debe coincidir con la región de AWS que utiliza para las operaciones de Amazon Textract Texact.

Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Textract Texact, no es posible transferir bytes de imágenes utilizando la propiedad Bytes. Debe cargar primero el documento en un bucket de Amazon S3 y, a continuación, llamar a la operación utilizando la propiedad S3Object.

Para que Amazon Textract Texact procese un objeto de S3, el usuario debe tener permiso para acceder al objeto de S3.

Tipo: objeto Document

Obligatorio: Sí

Sintaxis de la respuesta

```
{
   "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"
                 }
              }
           ]
        }
     ]
```

```
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

DocumentMetadata

Información sobre el documento de entrada.

Tipo: objeto DocumentMetadata

ExpenseDocuments

Los gastos detectados por Amazon Textract.

Type: Matriz deExpenseDocumentobjects

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

BadDocumentException

Amazon Textract Texact no puede leer el documento. Para obtener más información sobre los límites de documentos en Amazon Textract, consulteLímites máximos de Amazon Textract.

Código de estado HTTP: 400

DocumentTooLargeException

El documento no se puede procesar porque es demasiado grande. Tamaño máximo de documento para operaciones síncronas de 10 MB. El tamaño máximo de documento para operaciones asíncronas es de 500 MB para los archivos PDF.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract no puede obtener acceso al objeto de S3 que se especifica en la solicituda. para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

UnsupportedDocumentException

El formato del documento de entrada no es posible. Los documentos para operaciones pueden estar en formato PNG, JPEG, PDF o TIFF.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

AnalyzeID

Analiza los documentos de identidad para obtener información relevante. Esta información se extrae y se devuelve como`IdentityDocumentFields`, que registra tanto el campo normalizado como el valor del texto extraído. A diferencia de otras operaciones de Amazon Textract Texact,`AnalyzeID`no devuelve ningún dato de geometría.

Sintaxis de la solicitud

```
{
   "DocumentPages": [
      {
         "Bytes": blob,
         "S3Object": {
            "Bucket": "string",
            "Name": "string",
            "Version": "string"
         }
      }
   ]
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

DocumentPages

El documento que se pasa a AnalyzeID.

Type: Matriz deDocumentobjects

Miembros de la matriz: Número mínimo de 1 elemento. Número máximo de 2 elementos.

Obligatorio: Sí

Sintaxis de la respuesta

```
{
   "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"
               }
            }
         ]
      }
   ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[AnalyzeIDModelVersion](#)

Versión de la API AnalyzeIdentity que se utiliza para procesar documentos.

Type: Cadena

[DocumentMetadata](#)

Información sobre el documento de entrada.

Tipo: objeto DocumentMetadata

IdentityDocuments

Lista de documentos procesados por AnalyzeID. Incluye un número que indica su lugar en la lista y la estructura de respuestas del documento.

Type: Matriz deIdentityDocumentobjects

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

BadDocumentException

Amazon Textract Texact no puede leer el documento. Para obtener más información sobre los límites de documentos en Amazon Textract, consulteLímites máximos de Amazon Textract.

Código de estado HTTP: 400

DocumentTooLargeException

El documento no se puede procesar porque es demasiado grande. Tamaño máximo de documento para operaciones síncronas de 10 MB. El tamaño máximo de documento para las operaciones asíncronas es de 500 MB para los archivos PDF.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact no puede obtener acceso al objeto de S3 que se especificó en la solicitud. Para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

UnsupportedDocumentException

El formato del documento de entrada no es compatible. Los documentos para operaciones pueden estar en formato PNG, JPEG, PDF o TIFF.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript

- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

DetectDocumentText

Detecta el texto del documento de entrada. Amazon Textract Texact puede detectar líneas de texto y las palabras que componen una línea de texto. El documento de entrada debe ser una imagen en formato JPEG, PNG, PDF o TIFF.`DetectDocumentText`devuelve el texto detectado en una matriz de`Block`objetos.

Cada página de documento tiene como asociado`Block`de tipo PAGE. Cada PÁGINA`Block`object es el principal de LINE`Block`objetos que representan las líneas del texto detectado en una página. UNA LÍNEA`Block`objeto es un padre para cada palabra que forma la línea. Las palabras están representadas por`Block`objetos de tipo WORD.

`DetectDocumentText`es una operación síncrona. Para analizar documentos de forma asíncrona, utilice`StartDocumentTextDetection`.

Para obtener más información, consulte`Detección de texto de documentos`.

Sintaxis de la solicitud

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

Document

El documento de entrada como bytes codificados en base64 o un objeto Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Textract Texact, no puede pasar bytes de imagen. El documento debe ser una imagen en formato JPEG o PNG.

Si utiliza un SDK de AWS para llamar a Amazon Textract, es posible que no tenga que codificar en base 64 bytes de imagen que se pasan mediante el`Bytes`.

Tipo: objeto Document

Obligatorio: Sí

Sintaxis de la respuesta

```
{
   "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
   }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Blocks

Una matriz de`Block`objetos que contienen el texto detectado en el documento.

Type: Matriz deBlockobjects

DetectDocumentTextModelVersion

Type: Cadena

DocumentMetadata

Metadatos sobre el documento. Contiene el número de páginas detectadas en el documento.

Tipo: objeto DocumentMetadata

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

BadDocumentException

Amazon Textract Texact no puede leer el documento. Para obtener más información sobre los límites de documentos en Amazon Textract, consulteLímites máximos de Amazon Textract.

Código de estado HTTP: 400

DocumentTooLargeException

El documento no se puede procesar porque es demasiado grande. Tamaño máximo de documento para operaciones síncronas de 10 MB. El tamaño máximo de documento para las operaciones asíncronas es de 500 MB para los archivos PDF.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract no puede obtener acceso al objeto de S3 que se especificó en la solicituda. para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

UnsupportedDocumentException

El formato del documento de entrada no se admite. Los documentos para operaciones pueden estar en formato PNG, JPEG, PDF o TIFF.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

GetDocumentAnalysis

Obtiene los resultados de una operación asíncrona de Amazon Textract Texact que analiza el texto de un documento.

Para iniciar un análisis de texto asíncrono, llama aStartDocumentAnalysis, que devuelve un identificador de trabajo (`JobId`). Cuando finaliza la operación de análisis de texto, Amazon Textract Texact publica un estado de finalización en el tema de Amazon Simple Notification Service (Amazon SNS) registrado en la llamada inicial a`StartDocumentAnalysis`. Para obtener los resultados de la operación de detección de texto, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es`SUCCEEDED`. Si es así, llame`GetDocumentAnalysis`y pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta`StartDocumentAnalysis`.

`GetDocumentAnalysis`devuelve una matriz deBlockobjetos. Se devuelven los siguientes tipos de información:

- Datos del formulario (pares de clave-valor). La información relacionada se devuelve en dos.Blockobjetos, cada uno de tipo`KEY_VALUE_SET`: una LLAVE`Block`objeto y un VALOR`Block`objeto. Por ejemplo,Name: Ana Silva Carolinacontiene una clave y un valor. Name:es la clave. Ana Silva Carolinaes el valor.
- Datos de celdas de tabla y tabla. UNA MESA`Block`contiene información sobre una tabla detectada. UNA CELDA`Block`se devuelve para cada celda de una tabla.
- Líneas y palabras de texto. UNA LÍNEA`Block`un objeto contiene uno o varios WORD`Block`objetos. Se devuelven todas las líneas y palabras detectadas en el documento (incluido el texto que no tiene relación con el valor del`StartDocumentAnalysis` `FeatureTypes`parámetro de entrada).

Los elementos de selección, tales como casillas de verificación y botones de opción (botones de opción) se pueden detectar en los datos del formulario y en las tablas. A SELECTION_ELEMENT`Block`contiene información sobre un elemento de selección, incluido el estado de selección.

Usar`MaxResults`para limitar el número de bloques devueltos. Si hay más resultados de los especificados en`MaxResults`, el valor de`NextToken`en la respuesta de la operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame a`GetDocumentAnalysis`, y rellénela`NextToken`parámetro request con el valor de token que se devuelve de la llamada anterior a`GetDocumentAnalysis`.

Para obtener más información, consulteAnálisis de texto de documentos.

Sintaxis de la solicitud

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

JobId

Un identificador único del trabajo de detección de texto. La`JobId`se devuelve desde`StartDocumentAnalysis`. UNA`JobId`El valor solo es válido durante 7 días.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Obligatorio: Sí

MaxResults

El número máximo de resultados que devolver por llamada paginada. El valor mayor que puede especificar es 1,000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1,000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

NextToken

Si la respuesta anterior estaba incompleta (porque hay más bloques que recuperar), Amazon Texact devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de bloques.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `.*\S.*`

Obligatorio: No

Sintaxis de la respuesta

```
{
   "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 ]
         }
      ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

AnalyzeDocumentModelVersion

Type: Cadena

Blocks

Los resultados de la operación de análisis de texto.

Type: Matriz deBlockobjects

DocumentMetadata

Información sobre un documento que Amazon Textract procesó.`DocumentMetadata`se devuelve en cada página de respuestas paginadas de una operación de vídeo de Amazon Textract Texact.

Tipo: objeto DocumentMetadata

JobStatus

El estado actual del trabajo de detección de texto.

Type: Cadena

Valores válidos: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

Si la respuesta se trunca, Amazon Textract devuelve este token. Puede utilizar este token en la solicitud subsiguiente para recuperar el siguiente conjunto de resultados de detección de texto.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `.*\S.*`

StatusMessage

Devuelve si el trabajo de detección no se ha podido completar. Contiene explicación de qué error se ha producido.

Type: Cadena

Warnings

Lista de advertencias que se produjeron durante la operación de análisis de documentos.

Type: Matriz deWarningobjects

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidJobIdException

Se ha pasado un identificador de trabajo no válido aGetDocumentAnalysiso paraGetDocumentAnalysis.

Código de estado HTTP: 400

InvalidKMSKeyException

Indica que no tiene permisos de descifrado con la clave KMS introducida o que la clave KMS se ha introducido de forma incorrecta.

Código de estado HTTP: 400

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact no puede obtener acceso al objeto de S3 especificado en la solicituda. para obtener más información,Configurar el acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

GetDocumentTextDetection

Obtiene los resultados de una operación asíncrona de Amazon Texact que detecta texto de un documento. Amazon Textract Texact puede detectar líneas de texto y las palabras que componen una línea de texto.

Se inicia la detección de texto asíncrona llamandoStartDocumentTextDetection, que devuelve un identificador de trabajo (`JobId`). Cuando finaliza la operación de detección de texto, Amazon Textract Texact publica un estado de finalización en el tema de Amazon Simple Notification Service (Amazon SNS) registrado en la llamada inicial a`StartDocumentTextDetection`. Para obtener los resultados de la operación de detección de texto, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es`SUCCEEDED`. Si es así, llame`GetDocumentTextDetection`y pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta`StartDocumentTextDetection`.

`GetDocumentTextDetection`devuelve una matriz deBlockobjetos.

Cada página de documento tiene como asociado`Block`de tipo PAGE. Cada PÁGINA`Block`el objeto es el principal de LINE`Block`objetos que representan las líneas del texto detectado en una página. UNA LÍNEA`Block`objeto es un padre para cada palabra que forma la línea. Las palabras están representadas por`Block`objetos de tipo WORD.

Utilice el parámetro MaxResults para limitar el número de bloques devueltos. Si hay más resultados de los especificados en`MaxResults`, el valor de`NextToken`en la respuesta de la operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame`GetDocumentTextDetection`y rellénela`NextToken`parámetro request con el valor de token que se devuelve de la llamada anterior a`GetDocumentTextDetection`.

Para obtener más información, consulteDetección de texto de documentos.

Sintaxis de la solicitud

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

JobId

Un identificador único para el trabajo de detección de texto. La`JobId`se devuelve desde`StartDocumentTextDetection`. UNA`JobId`El valor solo es válido durante 7 días.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Obligatorio: Sí

MaxResults

El número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1,000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1,000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

NextToken

Si la respuesta anterior estaba incompleta (porque hay más bloques que recuperar), Amazon Texact devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de bloques.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `.*\S.*`

Obligatorio: No

Sintaxis de la respuesta

```
{
   "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 ]
      }
   ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Blocks

Los resultados de la operación de detección de texto.

Type: Matrices deBlockobjects

DetectDocumentTextModelVersion

Type: Cadena

DocumentMetadata

Información sobre un documento que Amazon Textract procesó.`DocumentMetadata`se devuelve en cada página de respuestas paginadas de una operación de vídeo de Amazon Textract Texact.

Tipo: objeto DocumentMetadata

JobStatus

El estado actual del trabajo de detección de texto.

Type: Cadena

Valores válidos: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

Si la respuesta se trunca, Amazon Textract devuelve este token. Puede utilizar este token en la solicitud subsiguiente para recuperar el siguiente conjunto de resultados de detección de texto.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `.*\S.*`

StatusMessage

Devuelve si el trabajo de detección no se ha podido completar. Contiene explicación de qué error se ha producido.

Type: Cadena

Warnings

Lista de advertencias que se produjeron durante la operación de detección de texto del documento.

Type: Matrices deWarningobjects

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidJobIdException

Se ha pasado un identificador de trabajo no válido aGetDocumentAnalysiso paraGetDocumentAnalysis.

Código de estado HTTP: 400

InvalidKMSKeyException

Indica que no tiene permisos de descifrado con la clave KMS introducida o que la clave KMS se ha introducido de forma incorrecta.

Código de estado HTTP: 400

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact no puede obtener acceso al objeto de S3 especificado en la solicitud. Para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go

- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

GetExpenseAnalysis

Obtiene los resultados de una operación asíncrona de Amazon Textract que analiza facturas y recibos. Amazon Textract encuentra la información de contacto, los artículos comprados y el nombre del proveedor, a partir de facturas y recibos de entrada.

Para iniciar un análisis asíncrono de factura/recibo llamandoStartExpenseAnalysis, que devuelve un identificador de trabajo (`JobId`). Una vez completado el análisis de factura/recepción, Amazon Textract publica el estado de realización en el tema de Amazon Simple Notification Service (Amazon SNS). Este tema debe registrarse en la llamada inicial a`StartExpenseAnalysis`. Para obtener los resultados de la operación de análisis de factura/recepción, en primer lugar, asegúrese de que el valor de estado publicado en el tema de Amazon SNS es`SUCCEEDED`. Si es así, llame`GetExpenseAnalysis`y pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta`StartExpenseAnalysis`.

Utilice el parámetro MaxResults para limitar el número de bloques devueltos. Si hay más resultados de los especificados en`MaxResults`, el valor de`NextToken`en la respuesta de operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame`GetExpenseAnalysis`, y rellénela`NextToken`parámetro request con el valor de token que se devuelve de la llamada anterior a`GetExpenseAnalysis`.

Para obtener más información, consulteAnálisis de facturas y recibos.

Sintaxis de la solicitud

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

JobId

Un identificador único para el trabajo de detección de texto. La`JobId`se devuelve desde`StartExpenseAnalysis`. UNA`JobId`El valor solo es válido durante 7 días.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Obligatorio: Sí

MaxResults

El número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 20. Si especifica un valor superior a 20, se devolverá un máximo de 20 resultados. El valor predeterminado es 20.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

NextToken

Si la respuesta anterior estaba incompleta (porque hay más bloques que recuperar), Amazon Texact devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de bloques.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `.*\S.*`

Obligatorio: No

Sintaxis de la respuesta

```
{
   "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,
```

```
                        "Y": number
                     }
                  ]
               },
               "Text": "string"
            }
         }
      ]
   }
],
"JobStatus": "string",
"NextToken": "string",
"StatusMessage": "string",
"Warnings": [
   {
      "ErrorCode": "string",
      "Pages": [ number ]
   }
]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

AnalyzeExpenseModelVersion

La versión del modelo actual de AnalyzeExpense.

Type: Cadena

DocumentMetadata

Información sobre un documento que Amazon Textract procesó.`DocumentMetadata`se devuelve en cada página de respuestas paginadas de una operación de Amazon Textract Texact.

Tipo: objeto DocumentMetadata

ExpenseDocuments

Los gastos detectados por Amazon Textract.

Type: Matriz deExpenseDocumentobjects

JobStatus

El estado actual del trabajo de detección de texto.

Type: Cadena

Valores válidos: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

Si la respuesta se trunca, Amazon Textract devuelve este token. Puede utilizar este token en la solicitud subsiguiente para recuperar el siguiente conjunto de resultados de detección de texto.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `.*\S.*`

StatusMessage

Devuelve si el trabajo de detección no se ha podido completar. Contiene explicación de qué error se ha producido.

Type: Cadena

Warnings

Lista de advertencias que se produjeron durante la operación de detección de texto del documento.

Type: Matriz deWarningobjects

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidJobIdException

Se ha pasado un identificador de trabajo no válido aGetDocumentAnalysiso paraGetDocumentAnalysis.

Código de estado HTTP: 400

InvalidKMSKeyException

Indica que no tiene permisos de descifrado con la clave KMS introducida o que la clave KMS se ha introducido de forma incorrecta.

Código de estado HTTP: 400

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact no puede obtener acceso al objeto de S3 especificado en la solicitud. Para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

StartDocumentAnalysis

Inicia el análisis asíncrono de un documento de entrada para detectar las relaciones entre elementos detectados como pares clave-valor, tablas y elementos de selección.

`StartDocumentAnalysis`puede analizar el texto de documentos en formato JPEG, PNG, TIFF y PDF. Los documentos se almacenan en un bucket de Amazon S3. UsarDocumentLocationpara especificar el nombre del bucket de y el nombre de archivo del documento.

`StartDocumentAnalysis`devuelve un identificador de trabajo (`JobId`) que utiliza para obtener los resultados de la operación. Cuando el análisis de texto se finalice, Amazon Textract Texact publica un estado de finalización en el tema de Amazon Simple Notification Service (Amazon SNS) que especifique en`NotificationChannel`. Para obtener los resultados de la operación de análisis de texto, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es`SUCCEEDED`. Si es así, llameGetDocumentAnalysisy pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta`StartDocumentAnalysis`.

Para obtener más información, consulteAnálisis de texto en documentos.

Sintaxis de la solicitud

```
{
   "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"
   }
```

```
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

[ClientRequestToken](#)

El token idempotente que utiliza para identificar la solicitud de inicio. Si utilizas el mismo token con variosStartDocumentAnalysissolicitudes, lo mismoJobIdse devuelve. UsarClientRequestTokenpara evitar que el mismo trabajo se inicie accidentalmente más de una vez. Para obtener más información, consulteLlamar a operaciones asíncronas de Amazon Textract.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Obligatorio: No

[DocumentLocation](#)

Ubicación del documento que se va a procesar.

Tipo: objeto DocumentLocation

Obligatorio: Sí

[FeatureTypes](#)

Lista de los tipos de análisis que se van a realizar. Agregue TABLES a la lista para devolver información sobre las tablas detectadas en el documento de entrada. Añada FORMULARIOS para devolver los datos del formulario detectados. Para realizar ambos tipos de análisis, agregue TABLES y FORMS a`FeatureTypes`. Todas las líneas y palabras detectadas en el documento se incluyen en la respuesta (incluido el texto que no está relacionado con el valor de`FeatureTypes`).

Type: Matriz de cadenas

Valores válidos: `TABLES | FORMS`

Obligatorio: Sí

JobTag

Identificador que especifique incluido en la notificación de finalización publicada en el tema de Amazon SNS. Por ejemplo, puede utilizar`JobTag`para identificar el tipo de documento al que corresponde la notificación de finalización (como un formulario fiscal o un recibo).

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `[a-zA-Z0-9_.\-:]+`

Obligatorio: No

KMSKeyId

La clave KMS utilizada para cifrar los resultados de inferencia. Puede estar en formato ID de clave o alias de clave. Cuando se proporciona una clave KMS, la clave KMS se utilizará para el cifrado del lado del servidor de los objetos del depósito de clientes. Cuando este parámetro no está habilitado, el resultado se cifrará en el lado del servidor mediante SSE-S3.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

Obligatorio: No

NotificationChannel

Arn del tema de Amazon SNS en el que desea que Amazon Textract publique el estado de finalización de la operación.

Tipo: objeto NotificationChannel

Obligatorio: No

OutputConfig

Establece si la salida irá a un depósito definido por el cliente. De forma predeterminada, Amazon Textract Texact guardará los resultados internamente para acceder a ellos mediante la operación GetDocumentAnalysis.

Tipo: objeto OutputConfig

Obligatorio: No

Sintaxis de la respuesta

```
{
   "JobId": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

JobId

El identificador del trabajo de detección de texto en documentos. Usar`JobId`para identificar el trabajo en una llamada posterior a`GetDocumentAnalysis`. UNA`JobId`El valor es válido solo durante 7 días.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

BadDocumentException

Amazon Textract Texact no puede leer el documento. Para obtener más información sobre los límites de documentos en Amazon Textract, consulteLímites máximos de Amazon Textract.

Código de estado HTTP: 400

DocumentTooLargeException

El documento no se puede procesar porque es demasiado grande. Tamaño máximo de documento para operaciones síncronas de 10 MB. El tamaño máximo de documento para operaciones asíncronas es de 500 MB para los archivos PDF.

Código de estado HTTP: 400

IdempotentParameterMismatchException

UNA`ClientRequestToken`Se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidKMSKeyException

Indica que no tiene permisos de descifrado con la clave KMS introducida o que la clave KMS se ha introducido de forma incorrecta.

Código de estado HTTP: 400

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract no puede obtener acceso al objeto de S3 que se especifica en la solicituda. para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

LimitExceededException

Se ha superado un límite de servicio Amazon Textract. Por ejemplo, si inicia demasiados trabajos asíncronos simultáneamente, llama para iniciar operaciones (`StartDocumentTextDetection`, por ejemplo) produce una excepción LimitExceededException (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio Amazon Textract Texacte.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

UnsupportedDocumentException

No se admite el formato del documento de entrada. Los documentos para operaciones pueden estar en formato PNG, JPEG, PDF o TIFF.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2

- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

StartDocumentTextDetection

Comienza la detección asincrónica del texto de un documento. Amazon Textract Texact puede detectar líneas de texto y las palabras que componen una línea de texto.

`StartDocumentTextDetection`puede analizar el texto de documentos en formato JPEG, PNG, TIFF y PDF. Los documentos se almacenan en un bucket de Amazon S3. UsarDocumentLocationpara especificar el nombre del bucket de y el nombre de archivo del documento.

`StartTextDetection`devuelve un identificador de trabajo (`JobId`) que utiliza para obtener los resultados de la operación. Una vez finalizada la detección de texto, Amazon Textract publica un estado de finalización en el tema Amazon Simple Notification Service (Amazon SNS) que especifique en`NotificationChannel`. Para obtener los resultados de la operación de detección de texto, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es`SUCCEEDED`. Si es así, llameGetDocumentTextDetectiony pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta`StartDocumentTextDetection`.

Para obtener más información, consulteDetección de texto de documentos.

Sintaxis de la solicitud

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
```

```
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

ClientRequestToken

El token idempotente que se utiliza para identificar la solicitud de inicio. Si utilizas el mismo token con variosStartDocumentTextDetectionsolicitudes, lo mismoJobIdse devuelve. UsarClientRequestTokenpara evitar que el mismo trabajo se inicie accidentalmente más de una vez. Para obtener más información, consulteLlamar a operaciones asíncronas de Amazon Textract.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Obligatorio: No

DocumentLocation

Ubicación del documento que se va a procesar.

Tipo: objeto DocumentLocation

Obligatorio: Sí

JobTag

Identificador que especifique que se incluye en la notificación de finalización publicada en el tema de Amazon SNS. Por ejemplo, puede utilizarJobTagpara identificar el tipo de documento al que corresponde la notificación de finalización (como un formulario fiscal o un recibo).

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `[a-zA-Z0-9_.\-:]+`

Obligatorio: No

KMSKeyId

La clave KMS utilizada para cifrar los resultados de inferencia. Puede estar en formato ID de clave o alias de clave. Cuando se proporciona una clave KMS, la clave KMS se utilizará para el cifrado del lado del servidor de los objetos del depósito de clientes. Cuando este parámetro no está habilitado, el resultado se cifrará en el lado del servidor mediante SSE-S3.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

Obligatorio: No

NotificationChannel

Arn del tema de Amazon SNS en el que desea que Amazon Textract publique el estado de finalización de la operación.

Tipo: objeto NotificationChannel

Obligatorio: No

OutputConfig

Establece si la salida va a ir a un depósito definido por el cliente. De forma predeterminada, Amazon Textract Texact guardará los resultados internamente para acceder a ellos con la operación GetDocumentTextDetection.

Tipo: objeto OutputConfig

Obligatorio: No

Sintaxis de la respuesta

```
{
   "JobId": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

JobId

Identificador del trabajo de detección de texto del documento. Usar`JobId`para identificar el trabajo en una llamada posterior a`GetDocumentTextDetection`. UNA`JobId`El valor solo es válido durante 7 días.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

BadDocumentException

Amazon Textract Texact no puede leer el documento. Para obtener más información sobre los límites de documentos en Amazon Textract, consulteLímites máximos de Amazon Textract.

Código de estado HTTP: 400

DocumentTooLargeException

El documento no se puede procesar porque es demasiado grande. Tamaño máximo de documento para operaciones síncronas de 10 MB. El tamaño máximo de documento para las operaciones asíncronas es de 500 MB para los archivos PDF.

Código de estado HTTP: 400

IdempotentParameterMismatchException

UNA`ClientRequestToken`se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidKMSKeyException

Indica que no tiene permisos de descifrado con la clave KMS introducida o que la clave KMS se ha introducido de forma incorrecta.

Código de estado HTTP: 400

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact no puede obtener acceso al objeto de S3 que se especifica en la solicitud. Para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

LimitExceededException

Se ha superado un límite de servicio Amazon Textract. Por ejemplo, si inicia demasiados trabajos asíncronos simultáneamente, llama para iniciar operaciones (`StartDocumentTextDetection`, por ejemplo) produce una excepción de LimitExceededException (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio Amazon Textract.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

UnsupportedDocumentException

No se admite el formato del documento de entrada. Los documentos para operaciones pueden estar en formato PNG, JPEG, PDF o TIFF.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

StartExpenseAnalysis

Inicia el análisis asíncrono de facturas o recibos de datos como información de contacto, artículos comprados y nombres de proveedores.

`StartExpenseAnalysis`puede analizar el texto de documentos en formato JPEG, PNG y PDF. Los documentos deben almacenarse en un bucket de Amazon S3. UsarDocumentLocationpara especificar el nombre del bucket de S3 y el nombre del documento de ese bucket.

`StartExpenseAnalysis`devuelve un identificador de trabajo (`JobId`) que proporcionarás a`GetExpenseAnalysis`para recuperar los resultados de la operación. Cuando finaliza el análisis de las facturas/recibos de entrada, Amazon Textract publica un estado de finalización en el tema Amazon Simple Notification Service (Amazon SNS) que proporciona al`NotificationChannel`. Para obtener los resultados de la operación de análisis de facturas y recibos, asegúrese de que el valor de estado publicado en el tema de Amazon SNS sea`SUCCEEDED`. Si es así, llameGetExpenseAnalysisy pasa el identificador de trabajo (`JobId`) que se devolvió mediante tu llamada a`StartExpenseAnalysis`.

Para obtener más información, consulteAnálisis de facturas y recibos.

Sintaxis de la solicitud

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
```

```
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

ClientRequestToken

El token idempotente que se utiliza para identificar la solicitud de inicio. Si utilizas el mismo token con variosStartDocumentTextDetectionsolicitudes, lo mismoJobIdse devuelve. UsarClientRequestTokenpara evitar que el mismo trabajo se inicie accidentalmente más de una vez. Para obtener más información, consulteLlamar a operaciones asíncronas de Amazon Textract

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

: obligatorio No

DocumentLocation

Ubicación del documento que se va a procesar.

Tipo: objeto DocumentLocation

: obligatorio Sí

JobTag

Identificador que especifique que se incluye en la notificación de finalización publicada en el tema de Amazon SNS. Por ejemplo, puede utilizarJobTagpara identificar el tipo de documento al que corresponde la notificación de finalización (como un formulario fiscal o un recibo).

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `[a-zA-Z0-9_.\-:]+`

: obligatorio No

KMSKeyId

La clave KMS utilizada para cifrar los resultados de la inferencia. Puede estar en formato ID de clave o alias de clave. Cuando se proporciona una clave KMS, la clave KMS se utilizará para el cifrado del lado del servidor de los objetos del depósito de clientes. Cuando este parámetro no está habilitado, el resultado se cifrará en el lado del servidor mediante SSE-S3.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

: obligatorio No

NotificationChannel

Arn del tema de Amazon SNS en el que desea que Amazon Textract publique el estado de finalización de la operación.

Tipo: objeto NotificationChannel

: obligatorio No

OutputConfig

Establece si la salida va a ir a un depósito definido por el cliente. De forma predeterminada, Amazon Textract Texact guardará los resultados internamente para que pueda acceder a ellos`GetExpenseAnalysis`.

Tipo: objeto OutputConfig

: obligatorio No

Sintaxis de la respuesta

```
{
   "JobId": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

JobId

Un identificador único del trabajo de detección de texto. La`JobId`se devuelve desde`StartExpenseAnalysis`. UNA`JobId`El valor solo es válido durante 7 días.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: `^[a-zA-Z0-9-_]+$`

Errores

AccessDeniedException

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

Código de estado HTTP: 400

BadDocumentException

Amazon Textract Texact no puede leer el documento. Para obtener más información sobre los límites de documentos en Amazon Textract, consulteLímites máximos de Amazon Textract.

Código de estado HTTP: 400

DocumentTooLargeException

El documento no se puede procesar porque es demasiado grande. Tamaño máximo de documento para operaciones síncronas de 10 MB. El tamaño máximo de documento para las operaciones asíncronas es de 500 MB para los archivos PDF.

Código de estado HTTP: 400

IdempotentParameterMismatchException

UNA`ClientRequestToken`Se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

InternalServerError

Amazon Textract ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidKMSKeyException

Indica que no tiene permisos de descifrado con la clave KMS introducida o que la clave KMS se ha introducido de forma incorrecta.

Código de estado HTTP: 400

InvalidParameterException

Un parámetro de entrada infringió una restricción. Por ejemplo, en operaciones sincrónicas, un`InvalidParameterException`se produce cuando ninguno de los`S3Object`o`Bytes`los valores se proporcionan en el`Document`parámetro de solicitud. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact no puede obtener acceso al objeto de S3 especificado en la solicitud. Para obtener más información,Configuración del acceso a Amazon S3Para obtener información sobre la resolución de problemas, consulteSolución de problemas de Amazon S3

Código de estado HTTP: 400

LimitExceededException

Se ha superado un límite de servicio de Amazon Texact. Por ejemplo, si inicia demasiados trabajos asíncronos simultáneamente, llama para iniciar operaciones (`StartDocumentTextDetection`, por ejemplo) produce una excepción LimitExceededException (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Textract.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Textract.

Código de estado HTTP: 400

ThrottlingException

Amazon Textract Texact no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

UnsupportedDocumentException

No se admite el formato del documento de entrada. Los documentos para operaciones pueden estar en formato PNG, JPEG, PDF o TIFF.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

Tipos de datos

Los tipos de datos siguientes son compatibles:

- 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

Se utiliza para contener la información detectada por una operación AnalyzeID.

Contenido

Confidence

La puntuación de confianza del texto detectado.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

NormalizedValue

Solo se devuelve para fechas, devuelve el tipo de valor detectado y la fecha escrita de forma más legible por máquina.

Tipo: objeto NormalizedValue

Obligatorio: No

Text

Texto del campo normalizado o del valor asociado a él.

Type: Cadena

Obligatorio: Sí

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

Block

UNA`Block`representa elementos que se reconocen en un documento dentro de un grupo de píxeles cerca uno del otro. La información devuelta en un`Block`depende del tipo de operación. Detección de texto para documentos (por ejemploDetectDocumentText), obtendrá información sobre las palabras y líneas de texto detectadas. En el análisis de texto (por ejemploAnalyzeDocument), también puede obtener información sobre los campos, tablas y elementos de selección detectados en el documento.

Una matriz de`Block`los objetos se devuelven mediante operaciones síncronas y asíncronas. En operaciones sincrónicas, tales comoDetectDocumentText, la matriz de`Block`objetos es todo el conjunto de resultados. En operaciones asíncronas, tales comoGetDocumentAnalysis, la matriz se devuelve a lo largo de una o más respuestas.

Para obtener más información, consulteFuncionamiento de Amazon Textract.

Contenido

BlockType

El tipo de elemento de texto que se reconoce. En las operaciones de detección de texto, se devuelven los siguientes tipos:

- PÁGINA- Contiene una lista de la línea`Block`objetos detectados en una página de documento.
- PALABRA- Palabra detectada en una página de documento. Una palabra consta de uno o varios caracteres en alfabeto latino básico ISO que no están separados por espacios.
- LÍNEA- Una cadena de palabras contiguas delimitadas por tabuladores que se detectan en una página de documento.

En las operaciones de análisis de texto, se devuelven los siguientes tipos:

- PÁGINA- Contiene una lista de niños`Block`objetos detectados en una página de documento.
- KEY_VALUE_SET- Almacena la CLAVE y el VALOR`Block`objetos para texto vinculado que se detecta en una página de documento. Usar`EntityType`para determinar si un objeto KEY_VALUE_SET es KEY`Block`objeto o VALUE`Block`objeto.
- PALABRA- Palabra que se detecta en una página de documento. Una palabra consta de uno o varios caracteres en alfabeto latino básico ISO que no están separados por espacios.
- LÍNEA- Una cadena de palabras contiguas delimitadas por tabuladores que se detectan en una página de documento.

- TABLA- Tabla que se detecta en una página de documento. Una tabla es información basada en cuadrícula con dos o más filas o columnas, con un rango de celdas de una fila y una columna cada una.
- CELDA- Una celda dentro de una tabla detectada. La celda es el padre del bloque que contiene el texto de la celda.
- SELECTION_ELEMENT- Elemento de selección, como un botón de opción (botón de opción) o una casilla de verificación detectada en una página de documento. Utilice el valor de`SelectionStatus`para determinar el estado del elemento de selección.

Type: Cadena

Valores válidos: `KEY_VALUE_SET | PAGE | LINE | WORD | TABLE | CELL | SELECTION_ELEMENT`

Obligatorio: No

ColumnIndex

Columna en la que aparece una celda de tabla. La primera posición de columna es 1.`ColumnIndex`no es devuelto por`DetectDocumentText`y`GetDocumentTextDetection`.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

ColumnSpan

El número de columnas que abarca una celda de tabla. Actualmente, este valor es siempre 1, incluso si el número de columnas extendidas es mayor que 1.`ColumnSpan`no es devuelto por`DetectDocumentText`y`GetDocumentTextDetection`.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

Confidence

La puntuación de confianza que tiene Amazon Textract Texact en la exactitud del texto reconocido y la precisión de los puntos de geometría alrededor del texto reconocido.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

EntityTypes

El tipo de entidad. Se puede devolver lo siguiente:

- CLAVE- Identificador de un campo del documento.
- VALUE- El texto del campo.

`EntityTypes`no es devuelto por`DetectDocumentText`y`GetDocumentTextDetection`.

Type: Matriz de cadenas

Valores válidos: `KEY | VALUE`

Obligatorio: No

Geometry

La ubicación del texto reconocido en la imagen. Incluye un cuadro delimitador grueso alineado con ejes que rodea el texto y un polígono de grano fino para obtener información espacial más precisa.

Tipo: objeto Geometry

Obligatorio: No

Id

Identificador del texto reconocido. El identificador solo es exclusivo para una sola operación.

Type: Cadena

Patrón: `.*\S.*`

Obligatorio: No

Page

Página en la que se ha detectado un bloque.`Page`se devuelve mediante operaciones asíncronas. Los valores de página superiores a 1 solo se devuelven para documentos de varias páginas en formato PDF o TIFF. Una imagen escaneada (JPEG/PNG), aunque contenga varias páginas de

documentos, se considera un documento de una sola página. El valor de`Page`es siempre 1. Las operaciones sincrónicas no regresan`Page`porque cada documento de entrada se considera un documento de una sola página.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

Relationships

Lista de bloques secundarios del bloque actual. Por ejemplo, un objeto LINE tiene bloques secundarios para cada bloque WORD que forma parte de la línea de texto. No hay objetos Relationship en la lista para relaciones que no existen, como cuando el bloque actual no tiene bloques secundarios. El tamaño de la lista puede ser el siguiente:

- 0 - El bloque no tiene bloques secundarios.
- 1 - El bloque tiene bloques secundarios.

Type: Matriz de[Relationship](#)objects

Obligatorio: No

RowIndex

Fila en la que se encuentra una celda de tabla. La primera posición de la fila es 1.`RowIndex`no es devuelto por`DetectDocumentText`y`GetDocumentTextDetection`.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

RowSpan

El número de filas que abarca una celda de tabla. Actualmente, este valor es siempre 1, incluso si el número de filas extendidas es mayor que 1.`RowSpan`no es devuelto por`DetectDocumentText`y`GetDocumentTextDetection`.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

SelectionStatus

El estado de selección de un elemento de selección, como un botón de opción o una casilla de verificación.

Type: Cadena

Valores válidos: `SELECTED | NOT_SELECTED`

Obligatorio: No

Text

La palabra o línea de texto reconocida por Amazon Textract.

Type: Cadena

Obligatorio: No

TextType

El tipo de texto que Amazon Textract ha detectado. Puede comprobar si hay texto escrito a mano y texto impreso.

Type: Cadena

Valores válidos: `HANDWRITING | PRINTED`

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

BoundingBox

Cuadro delimitador alrededor de la página, texto, par clave-valor, tabla, celda de tabla o elemento de selección detectados en una página de documento. La`left`(coordenada `x`) `y`top(coordenada y) son coordenadas que representan los lados superior e izquierdo del cuadro delimitador. Observe que la esquina superior izquierda de la imagen es el origen (0,0).

La`topyleft`los valores devueltos son proporciones del tamaño total de página del documento. Por ejemplo, si la imagen de entrada es de 700 x 200 píxeles y la coordenada superior izquierda del cuadro delimitador es de 350 x 50 píxeles, la API devuelve un valor de `left` de 0,5 (350/700) y un valor de `top` de 0,25 (50/200).

La`widthyheight`Los valores representan las dimensiones del cuadro delimitador expresada proporcional respecto a la dimensión total de la página del documento. Por ejemplo, si el tamaño de página del documento es 700 x 200 píxeles y el ancho del cuadro delimitador es de 70 píxeles, el ancho devuelto es 0,1.

Contenido

Height

La altura del cuadro delimitador expresada proporcional respecto a la altura total de la página del documento.

Type: Float

Obligatorio: No

Left

La coordenada izquierda del cuadro delimitador expresada proporcional respecto a la anchura total de la página del documento.

Type: Float

Obligatorio: No

Top

La coordenada superior del cuadro delimitador expresada proporcional respecto a la altura total de la página del documento.

Type: Float

Obligatorio: No

Width

La anchura del cuadro delimitador expresada proporcional respecto a la anchura total de la página del documento.

Type: Float

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

Document

El documento de entrada, ya sea en bytes o como objeto S3.

Puede transferir bytes de imágenes a una operación API Amazon Textract Texact utilizando la`Bytes`propiedad. Por ejemplo, debería utilizar el`Bytes`para pasar un documento cargado desde un sistema de archivos local. Bytes de imagen pasados mediante el`Bytes`La propiedad debe tener codificación base64. Es posible que el código no necesite codificar bytes de archivos de documentos si utiliza un SDK de AWS para llamar a las operaciones de la API de Amazon Textract Texact.

Puede transferir imágenes almacenadas en un bucket de S3 a una operación de API Amazon Textract Texact utilizando la`S3Object`propiedad. Los documentos almacenados en un bucket de S3 no tienen por qué estar codificados en base64.

La región de AWS del bucket de S3 que contiene el objeto S3 debe coincidir con la región de AWS que utiliza para las operaciones de Amazon Textract Texact.

Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Textract Texact, no es posible transferir bytes de imágenes utilizando la propiedad Bytes. Debe cargar primero el documento en un bucket de Amazon S3 y, a continuación, llamar a la operación utilizando la propiedad S3Object.

Para que Amazon Textract Texact procese un objeto S3, el usuario debe tener permiso para acceder al objeto S3.

Contenido

Bytes

Blob de bytes de documento codificados en base64. El tamaño máximo de un documento que se proporciona en un blob de bytes es de 5 MB. Los bytes de documento deben estar en formato PNG o JPEG.

Si utiliza un SDK de AWS para llamar a Amazon Textract, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante el`Bytes`.

Type: Objeto de datos binarios codificados en Base64

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 10485760 caracteres.

Obligatorio: No

S3Object

Identifica un objeto S3 como origen del documento. El tamaño máximo de un documento almacenado en un depósito de S3 es de 5 MB.

Tipo: objeto S3Object

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

DocumentLocation

El bucket de Amazon S3 que contiene el documento que se van a procesar. Se utiliza en operaciones asíncronas comoStartDocumentTextDetection.

El documento de entrada puede ser un archivo de imagen en formato JPEG o PNG. También puede ser un archivo en formato PDF.

Contenido

S3Object

El bucket de Amazon S3 que contiene el documento de entrada.

Tipo: objeto S3Object

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

DocumentMetadata

Información sobre el documento de entrada.

Contenido

Pages

Número de páginas detectadas en el documento.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

ExpenseDetection

Objeto utilizado para almacenar información sobre el valor o la etiqueta detectados por Amazon Textract.

Contenido

Confidence

La confianza en la detección, en porcentaje

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

Geometry

Información sobre dónde se encuentran los siguientes elementos en una página de documento: página detectada, texto, pares clave-valor, tablas, celdas de tabla y elementos de selección.

Tipo: objeto Geometry

Obligatorio: No

Text

La palabra o línea de texto reconocida por Amazon Textract

Type: Cadena

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2

- SDK de AWS para Ruby V3

ExpenseDocument

Estructura que contiene toda la información devuelta por AnalyzeExpense

Contenido

ExpenseIndex

Indica de qué factura o recibo del documento procede la información. El primer documento será 1, el segundo 2, el segundo 2, y así sucesivamente.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

LineItemGroups

Información detectada en cada tabla de un documento, separada en`LineItems`.

Type: Matriz deLineItemGroupobjects

Obligatorio: No

SummaryFields

Cualquier información encontrada fuera de una tabla por Amazon Textract.

Type: Matriz deExpenseFieldobjects

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

ExpenseField

Desglose de la información detectada, separada en las categorías Type, LabelDetection y ValueDetection

Contenido

LabelDetection

Etiqueta indicada explícitamente de un elemento detectado.

Tipo: objeto ExpenseDetection

Obligatorio: No

PageNumber

Número de página en el que se ha detectado el valor.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

Type

Etiqueta implícita de un elemento detectado. Presente junto con LabelDetection para elementos explícitos.

Tipo: objeto ExpenseType

Obligatorio: No

ValueDetection

Valor de un elemento detectado. Presente en elementos explícitos e implícitos.

Tipo: objeto ExpenseDetection

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

ExpenseType

Objeto utilizado para almacenar información sobre el tipo detectado por Amazon Textract.

Contenido

Confidence

La confianza de la precisión, como porcentaje.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

Text

Palabra o línea de texto detectada por Amazon Textract.

Type: Cadena

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

Geometry

Información sobre dónde se encuentran los siguientes elementos en una página de documento: página detectada, texto, pares clave-valor, tablas, celdas de tabla y elementos de selección.

Contenido

BoundingBox

Representación gruesa alineada con el eje de la ubicación del elemento reconocido en la página del documento.

Tipo: objeto BoundingBox

Obligatorio: No

Polygon

Dentro del cuadro delimitador, un polígono de grano fino alrededor del elemento reconocido.

Type: Matrices dePointobjects

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

HumanLoopActivationOutput

Muestra los resultados de la evaluación humana en bucle. Si no hay HumanLooParn, la entrada no activó la revisión humana.

Contenido

HumanLoopActivationConditionsEvaluationResults

Muestra el resultado de las evaluaciones de afecciones, incluidas las condiciones que activaron una revisión humana.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 10240 caracteres.

Obligatorio: No

HumanLoopActivationReasons

Muestra si se necesitaba una revisión humana y por qué.

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento.

Obligatorio: No

HumanLoopArn

Nombre de recurso de Amazon (ARN) del HumanLoop creado.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 256 caracteres.

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++

- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

HumanLoopConfig

Configura el flujo de trabajo de revisión humana al que se enviará el documento si se cumple una de las condiciones. También puedes establecer ciertos atributos de la imagen antes de revisar.

Contenido

DataAttributes

Establece los atributos de los datos de entrada.

Tipo: objeto HumanLoopDataAttributes

Obligatorio: No

FlowDefinitionArn

El nombre de recurso de Amazon (ARN) de la definición del flujo.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 256 caracteres.

Obligatorio: Sí

HumanLoopName

El nombre del flujo de trabajo humano utilizado para esta imagen. Esto debe ser único dentro de una región.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 63 caracteres.

Patrón: `^[a-z0-9](-*[a-z0-9])*`

Obligatorio: Sí

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++

- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

HumanLoopDataAttributes

Permite definir los atributos de la imagen. En la actualidad, puedes declarar una imagen libre de información de identificación personal y contenido para adultos.

Contenido

ContentClassifiers

Establece si la imagen de entrada está libre de información de identificación personal o contenido para adultos.

Type: Matriz de cadenas

Miembros de matrices: Número máximo de 256 elementos.

Valores válidos: `FreeOfPersonallyIdentifiableInformation | FreeOfAdultContent`

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

IdentityDocument

Estructura que enumera cada documento procesado en una operación AnalyzeID.

Contenido

DocumentIndex

Indica la ubicación de un documento en la lista IdentityDocument. El primer documento está marcado como 1, el segundo 2, etc.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

IdentityDocumentFields

Estructura utilizada para registrar la información extraída de los documentos de identidad. Contiene el campo normalizado y el valor del texto extraído.

Type: Matriz deIdentityDocumentFieldobjects

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

IdentityDocumentField

Estructura que contiene el tipo normalizado de la información extraída y el texto asociado a ella. Se extraen como Tipo y Valor respectivamente.

Contenido

Type

Se utiliza para contener la información detectada por una operación AnalyzeID.

Tipo: objeto AnalyzeIDDetections

Obligatorio: No

ValueDetection

Se utiliza para contener la información detectada por una operación AnalyzeID.

Tipo: objeto AnalyzeIDDetections

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

LineItemFields

Estructura que contiene información sobre las diferentes líneas que se encuentran en las tablas de un documento.

Contenido

LineItemExpenseFields

ExpenseFields utilizados para mostrar información de líneas detectadas en una tabla.

Type: Matrices deExpenseFieldobjects

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

LineItemGroup

Agrupación de tablas que contienen elementos de línea, con cada tabla identificada por la tabla`LineItemGroupIndex`.

Contenido

LineItemGroupIndex

Número utilizado para identificar una tabla específica de un documento. La primera tabla encontrada tendrá un LineItemGroupIndex de 1, el segundo 2, etc.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

LineItems

Desglose de la información de una línea concreta de una tabla.

Type: Matriz deLineItemFieldsobjects

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

NormalizedValue

Contiene información relativa a las fechas de un documento, incluidos el tipo de valor y el valor.

Contenido

Value

El valor de la fecha, escrito como año-mes-día:minuto:segundo.

Type: Cadena

Obligatorio: No

ValueType

Tipo normalizado del valor detectado. En este caso, FECHA.

Type: Cadena

Valores válidos: `DATE`

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

NotificationChannel

El tema de Amazon Simple Notification Service (Amazon SNS) en el que Amazon Textract Texact publica el estado de finalización de una operación de documento asíncrona, comoStartDocumentTextDetection.

Contenido

RoleArn

Es el nombre de recurso de Amazon (ARN) de un rol de IAM que otorga permisos de publicación a Amazon Texact al tema de Amazon SNS.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Obligatorio: Sí

SNSTopicArn

Tema de Amazon SNS en el que Amazon Textract publica el estado de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 1024 caracteres.

Patrón: `(^arn:([a-z\d-]+):sns:[a-zA-Z\d-]{1,20}:\w{12}:.+$)`

Obligatorio: Sí

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2

- SDK de AWS para Ruby V3

OutputConfig

Establece si la salida irá o no a un bucket creado por el usuario. Se utiliza para establecer el nombre del depósito y el prefijo del archivo de salida.

`OutputConfig`es un parámetro opcional que le permite ajustar dónde se colocará la salida. De forma predeterminada, Amazon Textract almacenará los resultados internamente y solo se puede acceder a ellos mediante las operaciones de Get API. Con OutputConfig habilitado, puedes establecer el nombre del bucket al que se enviará la salida y el prefijo de archivo de los resultados, donde puedes descargar los resultados. Además, puede configurar el`KMSKeyID`a una clave maestra de cliente (CMK) para cifrar la salida. Sin este conjunto de parámetros, Amazon Textract Texact cifrará el lado del servidor mediante la CMK administrada de AWS para Amazon S3.

El descifrado del contenido del cliente es necesario para procesar los documentos por parte de Amazon Textract. Si su cuenta se excluye según una política de exclusión de servicios de IA, todo el Contenido del cliente no cifrado se elimina de forma inmediata y permanente después de que el servicio haya procesado el Contenido del cliente. Amazon Texact no conserva ninguna copia de la salida. Para obtener información acerca de cómo excluirlas, consulte[Administración de políticas de exclusión de servicios de IA.]

Para obtener más información acerca de la privacidad de datos, consulte[Preguntas frecuentes sobre privacidad de datos].

Contenido

S3Bucket

El nombre del depósito al que irá a la salida.

Type: Cadena

Restricciones de longitud: Longitud mínima de 3. La longitud máxima es de 255 caracteres.

Patrón: `[0-9A-Za-z\.\-_]*`

Obligatorio: Sí

S3Prefix

El prefijo de la clave de objeto en el que se va a guardar la salida. Si no está habilitado, el prefijo será «textract_output».

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Patrón: `.*\S.*`

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

Point

Las coordenadas X e Y de un punto de una página de documento. Los valores X e Y que se devuelven son proporciones del tamaño total de página del documento. Por ejemplo, si el documento de entrada es 700 x 200 y la operación devuelve X=0,5 e Y=0,25, el punto se encuentra en la coordenada de píxeles (350,50) de la página del documento.

Una matriz de`Point`objects,`Polygon`se devuelve como parte de laGeometryobjeto que se devuelve en unBlockobject. UNA`Polygon`representa un polígono de grano fino alrededor del texto detectado, un par clave-valor, una tabla, una celda de tabla o un elemento de selección.

Contenido

X

El valor de la coordenada X de un punto de un`Polygon`.

Type: Float

Obligatorio: No

Y

El valor de la coordenada Y de un punto de un`Polygon`.

Type: Float

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

Relationship

Información sobre cómo se relacionan los bloques entre sí. UNA`Block`objeto contiene 0 o más`Relation`objetos de una lista,`Relationships`. Para obtener más información, consulte Block.

La`Type`proporciona el tipo de relación de todos los bloques del`IDs`matriz.

Contenido

Ids

Matriz de ID para bloques relacionados. Puede obtener el tipo de relación del`Type`elemento.

Type: Matriz de cadenas

Patrón: `.*\S.*`

Obligatorio: No

Type

Tipo de relación que tienen los bloques de la matriz IDs con el bloque actual. La relación puede ser`VALUE`o`CHILD`. Una relación de tipo VALUE es una lista que contiene el ID del bloque VALUE asociado a la CLAVE de un par clave-valor. Una relación de tipo CHILD es una lista de ID que identifican bloques WORD en el caso de líneas Bloques de celdas en el caso de Tablas y bloques WORD en el caso de Elementos de selección.

Type: Cadena

Valores válidos: `VALUE | CHILD | COMPLEX_FEATURES`

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2

- SDK de AWS para Ruby V3

S3Object

El nombre del bucket de S3 y el nombre de archivo que identifican el documento.

La región de AWS para el bucket de S3 que contiene el documento debe coincidir con la región que utiliza para las operaciones de Amazon Textract Texact.

Para que Amazon Textract Texact procese un archivo en un bucket de S3, el usuario debe tener permiso para acceder al bucket y al archivo de S3.

Contenido

Bucket

Nombre del bucket de S3. Tenga en cuenta que el carácter # no es válido en el nombre del archivo.

Type: Cadena

Restricciones de longitud: Longitud mínima de 3. La longitud máxima es de 255 caracteres.

Patrón: `[0-9A-Za-z\.\-_]*`

Obligatorio: No

Name

El nombre de archivo del documento de entrada. Las operaciones sincrónicas pueden utilizar archivos de imagen en formato JPEG o PNG. Las operaciones asíncronas también admiten archivos en formato PDF y TIFF.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Patrón: `.*\S.*`

Obligatorio: No

Version

Si en el bucket se ha habilitado el control de versiones, puede especificar la versión del objeto.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Patrón: `.*\S.*`

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

Warning

Una advertencia sobre un problema que se produjo durante el análisis de texto asíncrono (StartDocumentAnalysis) o detección de texto de documentos asíncrona (StartDocumentTextDetection).

Contenido

ErrorCode

El código de error de la advertencia.

Type: Cadena

Obligatorio: No

Pages

Lista de las páginas a las que se aplica la advertencia.

Type: Matriz de números enteros

Rango válido: Valor mínimo de 0.

Obligatorio: No

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- SDK de AWS para Ruby V3

Límites máximos de Amazon Textract

A continuación se muestra una lista de los límites máximos de Amazon Textract, que no pueden modificarse. Para obtener información sobre las limitaciones de la ubicación y los límites que pueden cambiarse, consulteCuotas y puntos de enlace de Amazon Textract. Para obtener información sobre los límites que pueden cambiarse, consulte Service Limits de AWS. Para cambiar un límite, consulte el tema relacionado con la creación de un caso.

Amazon Textract

Límite	Descripción
Formatos de archivo acep	Las operaciones admiten archivos JPEG, PNG, PDF y TIFF. (Se admiten imágenes con codificación JPEG 2000 en PDF).
Tamaño de archivo y límites de recuento de páginas	Para operaciones sincrónicas, los archivos JPEG, PNG, PDF y TIFF tienen un límite de tamaño de 10 MB. Los archivos PDF y TIFF también tienen un límite de 1 página. Para operacion es asíncronas, los archivos JPEG y PNG tienen un límite de tamaño de 10 MB. Los archivos PDF y TIFF tienen un límite de 500 MB. Los archivos PDF y TIFF tienen un límite de 3.000 páginas.
Límites específicos de PDF	La altura y el ancho máximos son 40 pulgadas y 2880 puntos. Los archivos PDF no pueden protegerse con contraseña. Los archivos PDF pueden contener imágenes con formato JPEG 2000.
Rotación de documentos y tamaño de imagen	Amazon Textract admite todas las rotaciones de documentos en el plano, por ejemplo, rotación en plano de 45 grados. Amazon Textract Texact admite imágenes con una resolución inferior o igual a 10000 píxeles en todos los lados.
Alineación de texto	El texto se puede alinear horizontalmente dentro del documento. Amazon Textract Texact no admite la alineación vertical de texto dentro del documento.

Límite	Descripción
Lenguajes	Amazon Textract admite la detección de texto en inglés, francés, alemán, español, inglés y portugués. Amazon Textract no devolverá el idioma detectado en su salida.
Tamaño de personaje	La altura mínima del texto que se va a detectar es de 15 píxeles. Con 150 PPP, sería igual que la fuente de 8 puntos.
Tipo de caracteres	Amazon Textract admite el reconocimiento de caracteres manuscritos e impresos.
Characters	Amazon Textract Texact detecta los siguientes caracteres: • a-z • A-Z • 0-9 • ä Ä Ö Ö ü Ü ç Ç é É â â Ê Ê î Î ô Ô û À À È È Ù Ë Ë Ï Ü Ü Á Á É É Í Ó Ó Ó Ú Ü Ü Ì Ó Ó Ó Ú Ü Ü Ñ Ì Ì Ò Ò ã • ! «# $% '& () * +, -./:; =? @ [\] ^ _ `{|} ~ > <° € £ ¥ # ß ß ¿¡€ £ ¥ # ø ø ©®™ § ¹ ² ³ '
Límites específicos de AnalizadID	AnalyzeID solo admite pasaportes estadounidenses y licencias de conducir estadounidenses.

Historial de documentos de Amazon Textract

En la siguiente tabla, se describen los cambios importantes que se han realizado en cada versión delGuía para desarrolladores de Amazon Textract. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a una fuente RSS.

- Última actualización de la documentación: 29 de mayo de 2019

update-history-change	update-history-description	update-history-date
Integración de ejemplos de código desdeAWSEjemplos de código del SDK de documentos repositorio de GitHub	La guía de Amazon Textract Texact contiene ahora ejemplos de código adicionales. Se ha cambiado el nombre de la sección de ejemplos anteriores a Tutoriales.	30 de enero de 2022
Se ha añadido análisis de gastos	Amazon Textract admite ahora el análisis de los documentos de facturas y recibos mediante la API AnalyzeExpense. Esta función solo está disponible en nuestras regiones de Asia Pacífico (Mumbai), Asia Pacífico (Seúl), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Canadá (Central), Europa (Frankfurt), Europa (Irlanda), Europa (Londres), EE.UU. Este (Ohio) EE. UU. Oeste (California) y EE.UU. Oeste (Oregón) de EE. UU.	26 de julio de 2021
Support para Augmented AI	Amazon Textract ya es compatible con Amazon	3 de diciembre de 2019

	Augmented AI para implement ar revisión humana.	
Nuevo servicio y guía	Amazon Textract ya está disponible para uso general.	29 de mayo de 2019
Support para elementos de selección	Amazon Textract ahora puede detectar elementos de selección (botones de opción y casillas de verificación).	24 de abril de 2019
Publicación de Amazon Textract	Esta es la primera versión de la documentación de Amazon Textract.	28 de noviembre de 2018

Glosario de AWS

Para ver la terminología más reciente de AWS, consulte el Glosario de AWS en la Referencia general de AWS.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.