aws

開発者ガイド

Amazon Textract

Amazon Textract: 開発者ガイド

Table of Contents

Amazon Textract とは

Amazon Textract では、ドキュメントのテキストの検出と分析をアプリケーションに追加できます。Amazon Textract を使用すると、お客様は以下を実行できます。

- 財務報告書、医療記録、税務フォームなど、さまざまな文書で入力された手書きのテキストを検出します。
- Amazon Textract ドキュメント分析 API を使用して、構造化データを含むドキュメントからテキスト、フォーム、テーブルを抽出します。
- AnalyzeExpense API を使用して請求書と領収書を処理します。
- AnalyzeID API を使用して、米国政府が発行した運転免許証やパスポートなどのID文書を処理します。

Amazon Textract は、Amazon のコンピュータビジョン科学者が日々何十億ものイメージやビデオを分析するために開発したものと同じ、実証済みの非常にスケーラブルなディープラーニングテクノロジに基づいています。使用にあたって機械学習の専門知識は必要ありません。Amazon Textract には、画像ファイルや PDF ファイルを分析できる、シンプルで使いやすい API が含まれています。Amazon Textract は、常に新しいデータを学習させ、Amazon は継続的に新しい機能をサービスに追加しています。

Amazon Textract を使用する一般的なユースケースは以下のとおりです。

- インテリジェント検索インデックスの作成— Amazon Textract を使用すると、イメージファイルおよび PDF ファイルで検出されるテキストのライブラリを作成できます。
- 自然言語処理 (NLP) にインテリジェントテキスト抽出を使用する— Amazon Textract では、NLP アプリケーションの入力としてテキストをグループ化する方法を制御できます。では、テキストを単語と行として抽出できます。また、Amazon Textract ドキュメントテーブル分析が有効になっている場合は、テーブルのセルごとにテキストをグループ化します。
- 異なるソースからのデータのキャプチャと正規化を加速— Amazon Textract では、財務文書、調査レポート、医療メモなど、さまざまなドキュメントからテキストおよび表形式のデータを抽出できます。Amazon Textract ドキュメント分析 API を使用すると、ドキュメントから非構造化データと構造化データを簡単かつ迅速に抽出できます。
- フォームからのデータキャプチャの自動化— Amazon Textract を使用すると、フォームから構造化データを抽出できます。Amazon Textract Analysis API を使用すると、既存のビジネスワークフ

ローに抽出機能を組み込んで、フォームを介して送信されたユーザーデータを使用可能な形式に抽出できます。

Amazon Textract を使用する利点のいくつかを以下に示します。

- ドキュメントテキスト検出をアプリに統合する— Amazon Textract では、シンプルな API を使用して強力かつ精度の高い分析を実行できるようにすることで、テキスト検出機能をアプリケーションに簡単に組み込むことができます。Amazon Textract を使用してドキュメントのテキストを検出するには、コンピュータビジョンや深層学習の専門知識は必要ありません。Amazon Textract Text API を使用すると、ウェブ、モバイル、コネクテッドデバイスのアプリケーションにテキスト検出を簡単に構築できます。
- スケーラブルなドキュメント分析— Amazon Textract を使用すると、何百万ものドキュメントからデータを迅速に分析および抽出できるため、意思決定を迅速化できます。
- 低コスト— Amazon Textract では、分析したドキュメント分のみお支払いいただくだけで利用可能です。最低料金や前払いの義務はありません。無料でお試しください。利用の増大に伴って、当社の階層型の料金モデルでさらに節約できます。

同期処理では、Amazon Textract は、レイテンシーが重要なアプリケーションについて、単一ページのドキュメントを分析できます。Amazon Textract は、複数ページのドキュメントのサポートを拡張するための非同期オペレーションも提供します。

Amazon Textract 初めて使用する方

Amazon Textract を初めて使用する場合は、次のセクションを順に読むことをお勧めします。

1. Amazon Textract 仕組み— このセクションでは、Amazon Textract コンポーネントと、エンドツーエンドエクスペリエンスを実現するための Amazon Textract コンポーネントとその連携方法を紹介します。
2. Amazon Textract の開始方法— このセクションでは、アカウントをセットアップして Amazon Textract API をテストします。

Amazon Textract をAWSSDK

AWS ソフトウェア開発キット (SDK) は、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コードの例
AWS SDK for C++	AWS SDK for C++コードの例
AWS SDK for Go	AWS SDK for Goコードの例
AWS SDK for Java	AWS SDK for Javaコードの例
AWS SDK for JavaScript	AWS SDK for JavaScriptコードの例
AWS SDK for .NET	AWS SDK for .NETコードの例
AWS SDK for PHP	AWS SDK for PHPコードの例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3)コードの例
AWS SDK for Ruby	AWS SDK for Rubyコードの例

可用性の例

必要なものが見つからなかった場合。このページの下部にある [Provide feedback] (フィードバックを送信) リンクを使用して、コードの例をリクエストしてください。

Amazon Textract 仕組み

Amazon Textract では、単一ページまたは複数ページの入力ドキュメント内のテキストを検出して分析できます (入力ドキュメント).

Amazon Textract は、以下のアクションに対するオペレーションを提供します。

- テキストのみを検出します。詳細については、「」を参照してください。テキストの検出。
- テキスト間の関係を検出して分析する。詳細については、「」を参照してください。ドキュメントを分析する。
- 請求書と領収書のテキストの検出と分析。詳細については、「」を参照してください。請求書と領収書の分析。
- 政府の身分証明書内のテキストの検出と分析。詳細については、「」を参照してください。アイデンティティドキュメントの分析。

Amazon Textract は、小さな単一ページのドキュメントとほぼリアルタイムの応答を処理するための同期オペレーションを提供します。詳細については、「同期操作によるドキュメントの処理」を参照してください。Amazon Textract は、大規模な複数ページのドキュメントを処理するために使用できる非同期オペレーションも提供します。非同期応答はリアルタイムではありません。詳細については、「非同期操作によるドキュメントの処理」を参照してください。

Amazon Textract オペレーションがドキュメントを処理すると、結果は次の配列で返されます。the section called “Block”オブジェクトまたは配列the section called “ExpenseDocument”オブジェクト。両方のオブジェクトには、ドキュメント上の位置やドキュメント上の他のアイテムとの関係など、アイテムについて検出された情報が含まれています。詳細については、「Amazon Textract レスポンスオブジェクト」を参照してください。使用方法を示す例です。Blockオブジェクト、「」を参照してください。チュートリアル。

トピック

- テキストの検出
- ドキュメントを分析する
- 請求書と領収書の分析
- アイデンティティドキュメントの分析
- 入力ドキュメント
- Amazon Textract レスポンスオブジェクト

- ドキュメントページ上のアイテムの場所

テキストの検出

Amazon Textract は、ドキュメント内で検出されたテキストのみを返す同期および非同期オペレーションを提供します。両方の操作セットで、次の情報が複数返されます。the section called “Block”オブジェクト。

- 検出されたテキストの行と単語
- 検出されたテキストの行と単語の関係
- 検出されたテキストが表示されるページ
- ドキュメントページ上のテキストの行と単語の位置

詳細については、「the section called “テキストの行と単語”」を参照してください。

テキストを同期的に検出するには、DetectDocumentTextAPI 操作。ドキュメントファイルを入力として渡します。結果セット全体が操作によって返されます。詳細と例については、「同期操作によるドキュメントの処理」を参照してください。

> **Note**
>
> Amazon Rekognition API オペレーションDetectTextとは異なります。DetectDocumentText。あなたは使うDetectTextポスターや道路標識などのライブシーン内のテキストを検出します。

テキストを非同期で検出するには、StartDocumentTextDetectionをクリックして、入力ドキュメントファイルの処理を開始します。結果を取得するには、GetDocumentTextDetection。結果は 1 つ以上の応答で返されます。GetDocumentTextDetection。詳細と例については、「非同期操作によるドキュメントの処理」を参照してください。

ドキュメントを分析する

Amazon Textract は、検出されたテキスト間の関係についてドキュメントとフォームを分析します。Amazon Textract 分析オペレーションでは、テキスト、フォーム、表の 3 つのカテゴリのドキュメント抽出が返されます。請求書と領収書の分析は、別のプロセスで処理されます。詳細については、を参照してください。請求書と領収書の分析。

テキスト抽出

ドキュメントから抽出された生のテキスト。詳細については、「」を参照してください。テキストの行と単語。

フォームの抽出

フォームデータは、ドキュメントから抽出されたテキストアイテムにリンクされます。Amazon Textract は、キーと値のペアでフォームデータを表します。次の例では、Amazon Textract によって検出されたテキスト行の 1 つが次のようになります。名前: Jane Doe。Amazon Textract はキーも識別します (名前:) と値 (Jane Doe). 詳細については、「」を参照してください。フォームデータ (キーバリューペア)。

名前: Jane Doe

住所: 123 Any Street, エニータウン, アメリカ合衆国

生年月日: 12-26-1980

キーと値のペアは、フォームから抽出されたチェックボックスまたはオプションボタン (ラジオボタン) を表すためにも使用されます。

male:☑

詳細については、「」を参照してください。選択エレメント。

表抽出

Amazon Textract は、テーブル、テーブルセル、およびテーブルセル内のアイテムを抽出でき、JSON、.csv、または.txt ファイルに結果を返すようにプログラムできます。

[Name] (名前)	Address
アナ・カロライナ	123 Any Town

詳細については、テーブルを参照してください。選択要素は、テーブルから抽出することもできます。詳細については、「」を参照してください。選択エレメント。

分析された商品の場合、Amazon Textract は以下を複数で返します。the section called “Block”オブジェクト:

- 検出されたテキストの行と単語
- 検出されたアイテムの内容
- 検出されたアイテム間の関係
- アイテムが検出されたページ
- ドキュメントページ上のアイテムの場所

同期または非同期操作を使用して、文書内のテキストを分析できます。テキストを同期的に分析するには、AnalyzeDocument操作を行い、ドキュメントを入力として渡します。AnalyzeDocument結果セット全体を返します。詳細については、「Amazon Textract を使用したドキュメントテキストの分析」を参照してください。

テキストを非同期で検出するには、StartDocumentAnalysisをクリックして処理を開始します。結果を取得するには、GetDocumentAnalysis。結果は 1 つ以上の応答で返されます。GetDocumentAnalysis。詳細と例については、「複数ページドキュメント内のテキストの検出または分析」を参照してください。

実行する解析のタイプを指定するには、FeatureTypes入力パラメータをリストします。表セル、セルテキスト、セル内の選択要素など、入力ドキュメントで検出された表に関する情報を返すには、TABLES を一覧に追加します。FORMS を追加して、キーと値のペアや選択要素などの単語の関係を返します。両方のタイプの分析を実行するには、TABLES と FORMS の両方をFeatureTypes。

ドキュメント内で検出されたすべての行と単語が応答に含まれます (次の値に関係しないテキストを含む) 。FeatureTypes).

請求書と領収書の分析

Amazon Textract は、テンプレートや設定を必要とせずに、ほぼすべての請求書や領収書から、連絡先情報、購入した商品、ベンダー名などの関連データを抽出します。請求書や領収書では多くの場合、さまざまなレイアウトが使用されるため、大規模なデータを手動で抽出するのは困難で時間がかかります。Amazon Textract は ML を使用して請求書と領収書のコンテキストを理解し、お客様のビジネスニーズに合わせて請求書または受領日、請求書または領収書番号、商品価格、合計金額、支払い条件などのデータを自動的に抽出します。

Amazon Textract は、ワークフローにとって重要であるが明示的にラベル付けされていないベンダー名も識別します。たとえば、Amazon Textract は、明示的なキーと値のペアの組み合わせなしで、ページ上部の口ゴ内にのみ表示されていても、レシートでベンダー名を見つけることができま

す。Amazon Textract を使用すると、同じコンセプトに異なる単語を使用するさまざまな領収書や請求書からの入力を簡単に統合できます。たとえば、Amazon Textract は、顧客番号、顧客番号、アカウント ID などの異なるドキュメント内のフィールド名間の関係をマッピングし、標準分類を次のように出力します。INVOICE_RECEIPT_ID。この場合、Amazon Textract は、異なるドキュメントタイプにわたって一貫してデータを表します。標準タクソノミと一致しないフィールドは、次のように分類されます。OTHER。

以下に、AnalyzeExpense で現在サポートされている標準フィールドを示します。

- ベンダー名:VENDOR_NAME
- 合計:TOTAL
- 受信者のアドレス:RECEIVER_ADDRESS
- 請求書/受領日:INVOICE_RECEIPT_DATE
- 請求書/領収書 ID:INVOICE_RECEIPT_ID
- 支払条件:PAYMENT_TERMS
- 小計:SUBTOTAL
- 期日:DUE_DATE
- 税:TAX
- 請求書納税者 ID (SSN/ITIN または EIN):TAX_PAYER_ID
- 項目名:ITEM_NAME
- 商品価格:PRICE
- 商品数量:QUANTITY

AnalyzeExpense API は、特定のドキュメントページに対して次の要素を返します。

- ページ内の領収書または請求書の件数は、次のように表されます。ExpenseIndex
- として表される個々のフィールドの標準化された名前Type
- ドキュメントに表示される実際のフィールド名。次のように表示されます。LabelDetection
- 次のように表される対応するフィールドの値ValueDetection
- 送信されたドキュメント内のページ数が次のように表されます。Pages
- フィールド、値、またはラインアイテムが検出されたページ番号。PageNumber
- ページ上の個々のフィールド、値、またはラインアイテムの境界ボックスと座標位置を含むジオメトリ。Geometry

- ドキュメントで検出された各データに関連付けられた信頼スコア。次のように表されます。Confidence
- 購入された個々の明細項目の行全体。EXPENSE_ROW

以下は、AnalyzeExpense によって処理された領収書の API 出力の一部で、標準フィールドとして抽出されたドキュメントの「合計:$55.64」を示しています。TOTAL、ドキュメント上の実際のテキストを「合計」、信頼スコア「97.1」、ページ番号「1」、合計値を「$55.64」、境界ボックスとポリゴン座標：


```
{
    "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
}
```


同期操作を使用して、請求書または受入を分析できます。これらの文書を分析するには、AnalyzeExpense オペレーションを使用して、領収書または請求書をそれに渡します。AnalyzeExpense結果セット全体を返します。詳細については、「Amazon Textract を使用した請求書と領収書の分析」を参照してください。

請求書と領収書を非同期に分析するには、StartExpenseAnalysisをクリックして、入力ドキュメントファイルの処理を開始します。結果を取得するには、GetExpenseAnalysis。に対する特定の呼び

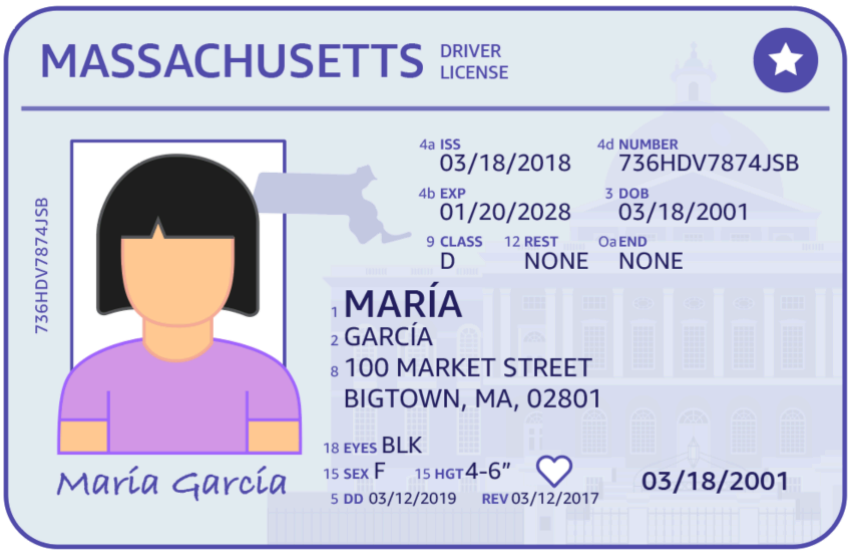
出しの結果StartExpenseAnalysisによって返されるGetExpenseAnalysis。詳細と例については、「非同期操作によるドキュメントの処理」を参照してください。

アイデンティティドキュメントの分析

Amazon Textract は、AnalyzeID API を使用して、米国政府が発行したパスポート、運転免許証、およびその他の身分証明書から関連情報を抽出できます。Analyze ID を使用すると、企業は米国の運転免許証、州ID、および異なるテンプレートや形式を持つパスポートなど、IDから迅速かつ正確に情報を抽出できます。AnalyzeID API は、次の 2 つのカテゴリのデータタイプを返します。

- 生年月日、発行日、ID 番号、クラス、制限など ID で使用できるキーと値のペア。
- 名前、住所、発行者など、明示的なキーが関連付けられていないドキュメント上の暗黙のフィールド。

キー名はレスポンス内で標準化されています。たとえば、運転免許証に「LIC# (ライセンス番号)」と表示され、パスポートに「パスポート番号」と表示されている場合、Analyze ID 応答は標準化されたキーを raw キー (例:LIC#) とともに「ドキュメント ID」として返します。この標準化により、顧客は、同じ概念に対して異なる用語を使用する多くの ID 間で情報を簡単に組み合わせることができます。



Analyze ID は、と呼ばれる構造体の情報を返します。IdentityDocumentFields。これらはJSON正規化された Type と Type に関連付けられた値の 2 つの情報を含む構造体。これら両方とも信頼スコアを持っています。詳細については、「ID ドキュメントの応答オブジェクト」を参照してください。

同期操作を使用して、運転免許証またはパスポートを分析できます。これらのドキュメントを分析するには、AnalyzeID オペレーションを使用して、ID ドキュメントを渡します。`AnalyzeID`結果セット全体を返します。詳細については、「Amazon Textract を使用したアイデンティティドキュメントの分析」を参照してください。

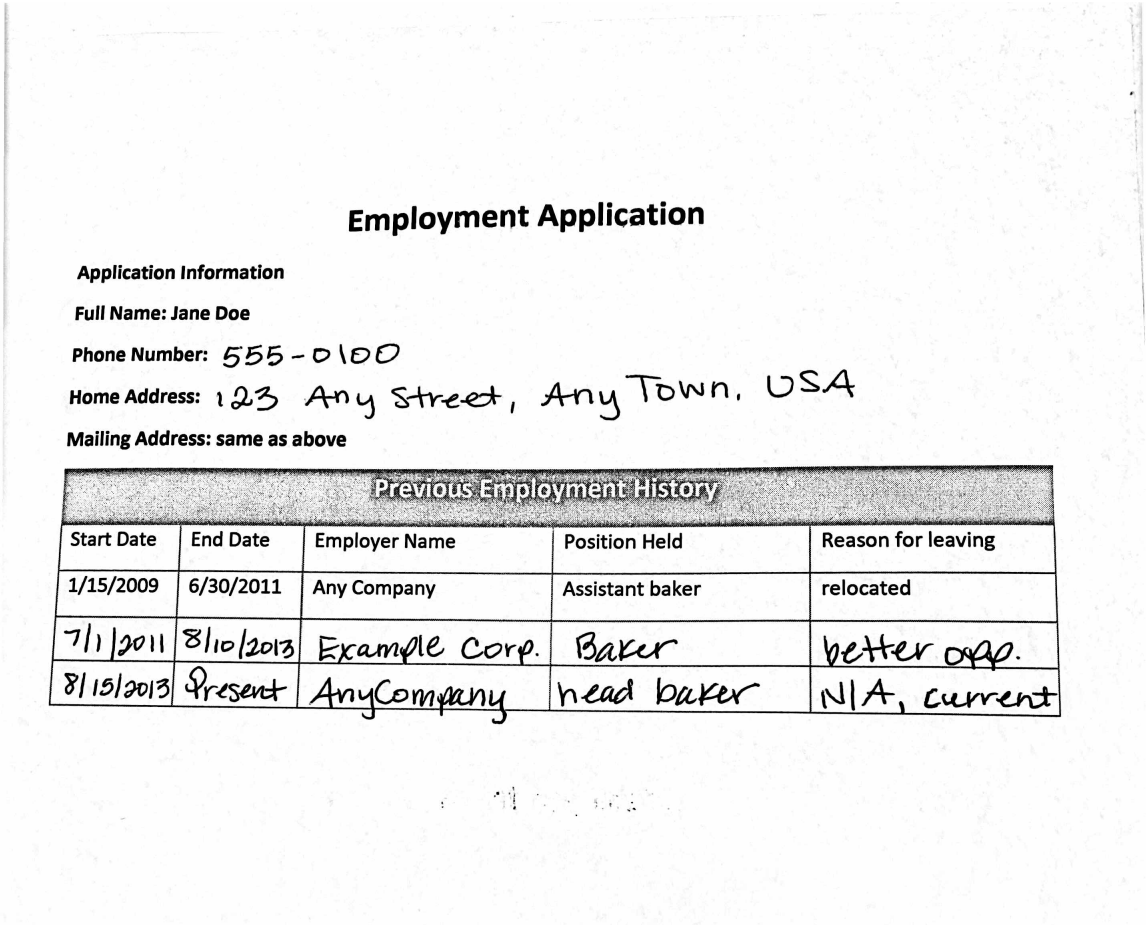
> **Note**
>
> 運転免許証などの一部の身分証明書には、2つの側面があります。同じ Analyze ID API リクエスト内で、運転免許証の表と背面画像を別々の画像として渡すことができます。

入力ドキュメント

Amazon Textract オペレーションに適した入力は、単一ページまたは複数ページのドキュメントです。いくつかの例は、法的文書、フォーム、ID、または手紙です。フォームは、ユーザーが回答を提供するための質問またはプロンプトを含むドキュメントです。例としては、患者登録フォーム、納税フォーム、または保険請求フォームなどがあります。

ドキュメントは、JPEG、PNG、PDF、または TIFF 形式にすることができます。PDF および TIFF 形式のファイルを使用すると、複数ページのドキュメントを処理できます。Amazon Textract がドキュメントを次のように表現する方法の詳細については`Block`オブジェクト、「」を参照してください。テキスト検出および文書分析応答オブジェクト。

以下に、入力ドキュメントの例を示します。



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

ドキュメントの制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

Amazon Textract 同期オペレーションでは、Amazon S3 バケットに格納されている入力ドキュメントを使用するか、base64 でエンコードされたイメージバイトを渡すことができます。詳細については、「Amazon Textract 同期オペレーションを呼び出す」を参照してください。非同期オペレーションの場合は、Amazon S3 バケットで入力ドキュメントを指定する必要があります。詳細については、「Amazon Textract 非同期オペレーションを呼び出す」を参照してください。

Amazon Textract レスポンスオブジェクト

Amazon Textract オペレーションは、実行するオペレーションに応じて異なるタイプのオブジェクトを返します。テキストを検出し、一般文書を分析する場合、この操作は Block オブジェクトを返します。請求書または領収書を分析するために、オペレーションは expenseDocuments オブジェクトを返します。アイデンティティドキュメントを分析するために、この操作は IdentityDocumentFields オブジェクトを返します。これらの応答オブジェクトの詳細については、以下のセクションを参照してください。

トピック

- テキスト検出および文書分析応答オブジェクト
- 請求書および領収書応答オブジェクト
- ID ドキュメントの応答オブジェクト

テキスト検出および文書分析応答オブジェクト

Amazon Textract がドキュメントを処理すると、次のリストが作成されます。Block検出または解析されたテキストのオブジェクト。各ブロックには、検出されたアイテム、それがどこにあるか、Amazon Textract が処理の精度に持っている信頼度に関する情報が含まれています。

ドキュメントは、次のタイプから構成されます。Blockオブジェクト。

- ページ
- テキストの行と単語
- フォームデータ (キーバリューペア)
- テーブルおよびセル
- 選択エレメント

ブロックの内容は、呼び出す操作によって異なります。テキスト検出操作のいずれかを呼び出すと、検出されたテキストのページ、行、および単語が返されます。詳細については、「テキストの検出」を参照してください。文書分析操作のいずれかを呼び出すと、検出されたページ、キーと値のペア、テーブル、選択要素、およびテキストに関する情報が返されます。詳細については、「ドキュメントを分析する」を参照してください。

ある程度Blockオブジェクトフィールドは、両方のタイプの処理に共通しています。たとえば、各ブロックに一意の識別子があります。

使用方法を示す例です。Blockオブジェクト、「」を参照してください。チュートリアル。

ドキュメントレイアウト

Amazon Textract は、ドキュメントの表現をさまざまなタイプのリストとして返します。Block親子関係またはキーと値のペアでリンクされているオブジェクト。ドキュメント内のページ数を示すメタデータも返されます。以下に、一般的な JSON を示します。Block型のオブジェクト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
    }
}
```

ドキュメントは、1 つ以上のもので作成されます。PAGEブロック。各ページには、テキスト行や表など、ページで検出されたプライマリアイテムの子ブロックのリストが含まれています。詳細については、「ページ」を参照してください。

のタイプを判断できます。Blockオブジェクトを検査してBlockTypeフィールド。

あるBlockオブジェクトには関連のリストが含まれています。Blockのオブジェクトの Relationshipsフィールド。これは、の配列です。Relationshipオブジェクト。あるRelationships配列は CHILD 型または VALUE 型のいずれかです。CHILD 型の配列は、現在のブロックの子である項目を一覧表示するために使用されます。たとえば、現在のブロックのタイプが LINE の場合、Relationshipsテキスト行を構成する WORD ブロックの ID のリストが含まれます。VALUE 型の配列は、キーと値のペアを格納するために使用されます。関係のタイプを調べて、関係の種類を判断できます。TypeのフィールドRelationshipオブジェクト。

子ブロックには、親ブロックオブジェクトに関する情報がありません。

示す例についてはBlock詳細については、を参照してください。同期操作によるドキュメントの処理。

Confidence

Amazon Textract オペレーションは、検出された商品の精度に対する Amazon Textract の信頼度のパーセンテージを返します。信頼を得るには、ConfidenceのフィールドBlockオブジェクト。値が高いほど、信頼度が高いことを示します。シナリオによっては、信頼性の低い検出では、人間による視覚的な確認が必要になる場合があります。

ジオメトリ

Amazon Textract オペレーションは、ID 分析を除き、ドキュメントページ上で検出されたアイテムの場所に関する位置情報を返します。場所を取得するには、GeometryのフィールドBlockオブジェクト。詳細については、「」を参照してください。ドキュメントページ上のアイテムの場所

ページ

ドキュメントは 1 ページ以上で構成されます。あるthe section called “Block”型のオブジェクトPAGEドキュメントの各ページに存在します。あるPAGEブロックオブジェクトには、ドキュメントページで検出されたテキスト行、キーと値のペア、およびテーブルの子 ID のリストが含まれます。

の JSONPAGEblock は次の例のようになります。

```
{

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

PDF 形式の複数ページのドキュメントで非同期操作を使用している場合は、Pageのフィールドオブジェクト。スキャンした画像 (JPEG、PNG、PDF、または TIFF 形式の画像) は、画像に複数のドキュメントページがあっても、単一ページのドキュメントと見なされます。非同期操作は常にPageスキャンした画像の値 1。Block

合計ページ数は、PagesのフィールドDocumentMetadata。DocumentMetadataの各リストとともに返されます。BlockAmazon Textract オペレーションによって返されるオブジェクト。

テキストの行と単語

Amazon Textract オペレーションによって返される検出されたテキストは、the section called “Block”オブジェクト。これらのオブジェクトは、文書ページで検出されたテキスト行またはテキスト単語を表します。次のテキストは、複数の単語から作成された 2 行のテキストを示しています。

これはテキストです。

2 つの別々の行で。

検出されたテキストはTextのフィールドBlockオブジェクト。-BlockTypeフィールドでは、テキストがテキスト行 (LINE) または単語 (WORD) のどちらであるかを指定します。ある単語スペースで区切られていない、1 個以上の ISO 基本ラテンアルファベットです。あるラインは、タブ区切りの連続した単語の文字列です。

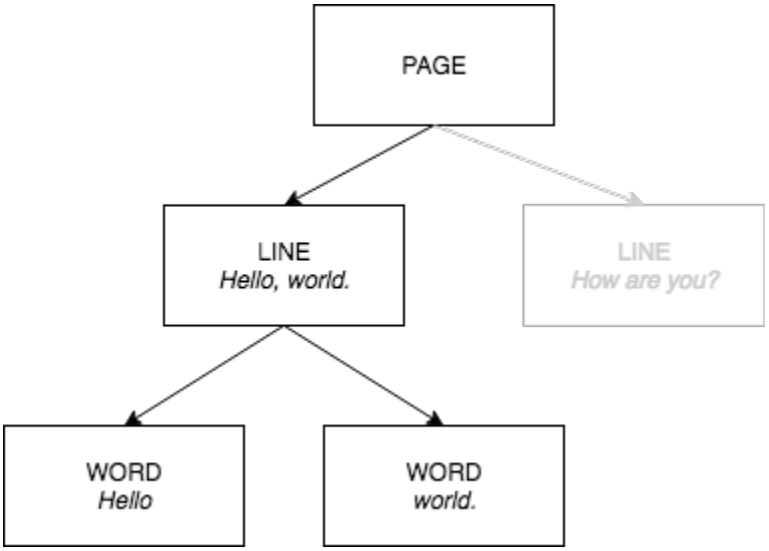
さらに、Amazon Textract は、テキストが手書きまたは印刷されたかを判断します。TextTypesフィールド。これらはそれぞれ、手書き文字として返され、印刷されます。

他のBlockプロパティは、ID、信頼度、ジオメトリ情報など、すべてのブロックタイプに共通です。詳細については、「the section called “テキスト検出および文書分析応答オブジェクト”」を参照してください。

行と単語のみを検出するには、DetectDocumentTextまたはStartDocumentTextDetection。詳細については、「テキストの検出」を参照してください。検出されたテキスト（行と単語）と、それが文書の他の部分（表など）とどのように関連しているかについての情報を取得するには、AnalyzeDocumentまたはStartDocumentAnalysis。詳細については、「ドキュメントを分析する」を参照してください。

PAGE,LINE, およびWORDブロックは、親と子の関係で互いに関連しています。あるPAGEブロックはすべての人の親ですLINEドキュメントページ上のオブジェクトをブロックします。LINEは1つ以上の単語を持つことができるため、RelationshipsLINE ブロックの配列は、テキスト行を構成する子の WORD ブロックの ID を格納します。

次の図は、この線の仕組みを示しています。Hello worldのテキストHello world お元気ですか？で表される。Blockオブジェクト。



以下に、からの JSON 出力を示します。DetectDocumentText文がいつあるかHello world お元気ですか？が検出されました。最初の例は、ドキュメントページの JSON です。子 ID によってドキュメント内を移動できる点に注意してください。

```
{
    "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"
},
```

以下は、「Hello, World」という行を構成するLINEブロックの JSON です。

```
{
    "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"
},
```

以下は、単語の WORD ブロックの JSON です。Hello,:

```
{
```

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

最後の JSON は、その単語の WORD ブロックです。世界。:

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

フォームデータ (キーと値のペア)

Amazon Textract は、キーと値のペアで、ドキュメントからフォームデータを抽出できます。たとえば、次のテキストでは、Amazon Textract はキーを識別できます (名前:) と値 (アナ・カロライナ).

名前: アナ・カロライナ

検出されたキーと値のペアは、次のように返されます。Blockからの応答内のオブジェクトAnalyzeDocumentそしてGetDocumentAnalysis。♪FeatureTypesキーと値のペア、テーブル、またはその両方に関する情報を取得するための入力パラメータ。キーと値のペアの場合のみ、値を使用します。FORMS。例については、フォームドキュメントからのキーと値のペアの抽出 を参照してください。でドキュメントがどのように表されるかに関する一般情報については、Blockオブジェクト、「」を参照してください。テキスト検出および文書分析応答オブジェクト。

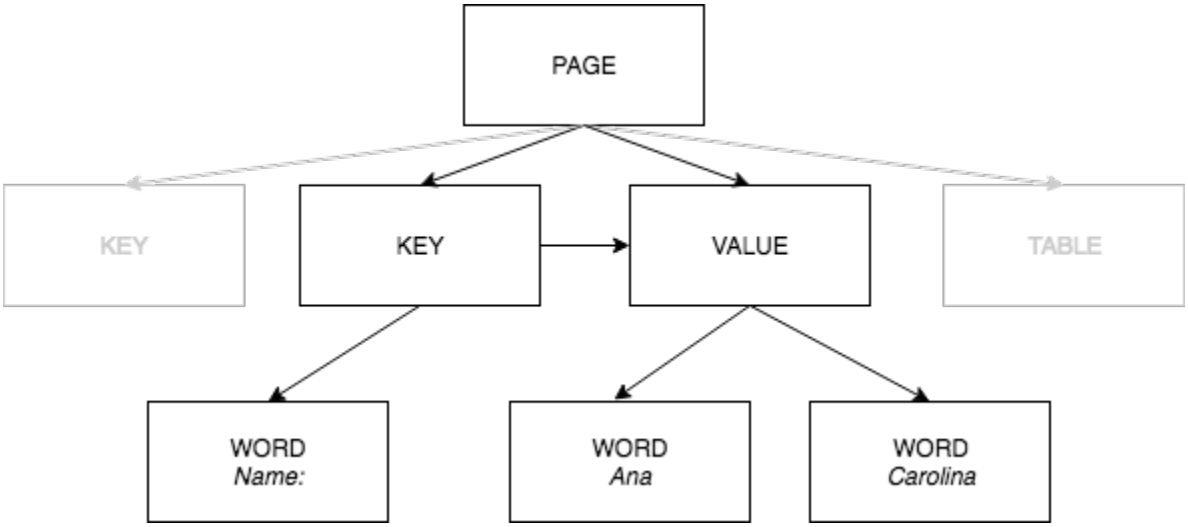
KEY_VALUE_SET 型のブロックオブジェクトは、文書内で検出されたリンクされたテキストアイテムに関する情報を格納する KEY または VALUE ブロックオブジェクトのコンテナです。♪EntityType属性を使用して、ブロックがKEYかVALUEかを決定します。

- あるキーオブジェクトには、リンクテキストのキーに関する情報が含まれています。たとえば、名前:。KEY ブロックには 2 つのリレーションシップリストがあります。VALUE 型の関係は、キーに関連付けられた VALUE ブロックの ID を含むリストです。CHILD 型の関係は、キーのテキストを構成する WORD ブロックの ID のリストです。

- ある値オブジェクトには、キーに関連付けられているテキストに関する情報が含まれています。前の例では、以下のようになっています。アナ· カロライナキーの値はです。名前:。VALUE ブロックは、WORD ブロックを識別する CHILD ブロックのリストとの関係を持ちます。各 WORD ブロックには、値のテキストを構成する単語の 1 つが含まれています。あるVALUEオブジェクトには、選択した要素に関する情報を含めることもできます。詳細については、「選択エレメント」を参照してください。

KEY_VALUE_SET の各インスタンスBlockオブジェクトはページの子ですBlock現在のページに対応するオブジェクト。

次の図は、キーと値のペアの仕組みを示しています。名前: アナ· カロライナで表される。Blockオブジェクト。



次の例は、キーと値のペアの仕組みを示しています。名前: アナ· カロライナは JSON で表されます。

PAGE ブロックにはタイプの CHILD ブロックがありますKEY_VALUE_SETドキュメント内で検出された KEY および VALUE ブロックごとに。

```
{
    "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
},
```

以下のJSONは、KEYブロック (52be177-53f7-42f6-a7cf-6d09bdc15a30) がVALUEブロック (7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c) との関係を示している。また、キーのテキストを含むWORDブロック (c734fca6-c4c4-415c-B6c1-30f7510b72ee) のCHILDブロックもある。名前:).

```
{
    "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
},
```

次の JSON は、VALUE ブロック 7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c に、値のテキストを構成する単語ブロックの ID の子リストがあることを示しています (アナそしてカロライナ).

```
{
    "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
}
```

以下の JSON は、Block単語のためのオブジェクト名前:,アナ, およびカロライナ。

```
{
    "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"
},
```

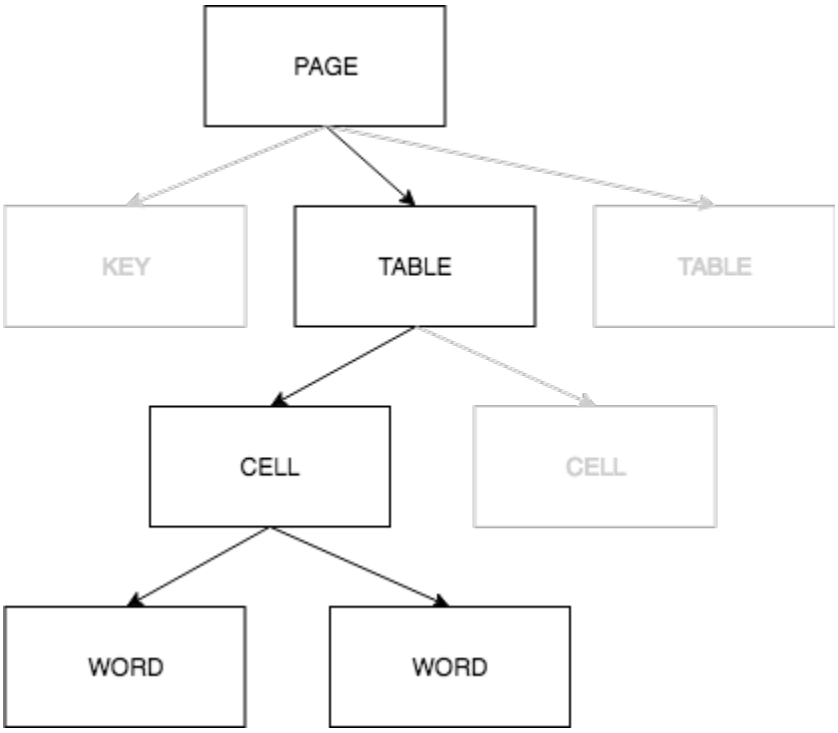
テーブル

Amazon Textract は、テーブルおよびテーブル内のセルを抽出できます。たとえば、フォームで次のテーブルが検出されると、Amazon Textract は 4 つのセルを含むテーブルを検出します。

[Name] (名前)	Address
アナ・カロライナ	123 Any Town

検出されたテーブルは次のように返されます。Blockからの応答内のオブジェクトAnalyzeDocumentそしてGetDocumentAnalysis。♪FeatureTypesキーと値のペア、テーブル、またはその両方に関する情報を取得するための入力パラメータ。テーブルの場合のみ、値を使用します。TABLES。例については、CSV ファイルへのテーブルのエクスポート を参照してください。でドキュメントがどのように表されるかに関する一般情報については、Blockオブジェクト、「」を参照してください。テキスト検出および文書分析応答オブジェクト。

次の図は、表内の 1 つのセルがどのように表されるかを示しています。Blockオブジェクト。



セルの内容WORD検出された単語のブロック、およびSELECTION_ELEMENTチェックボックスなどの選択要素のブロック。

次に、4 つのセルがある前のテーブルの部分 JSON を示します。

PAGE ブロックオブジェクトには、TABLE ブロックの子ブロック ID と検出されたテキストの各行のリストがあります。

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

TABLE ブロックには、テーブル内のセルの子 ID のリストが含まれます。TABLE ブロックには、ドキュメント内のテーブル位置のジオメトリ情報も含まれます。次の JSON は、テーブルに 4 つのセルがあることを示しています。Ids配列。

```
{
    "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"
},
```

表のセルの [ブロックタイプ] は [セル] です。-Block各セルのオブジェクトには、表内の他のセルと比較したセルの位置に関する情報が含まれます。また、ドキュメント上のセルの位置に関するジオメトリ情報も含まれます。前の例では、以下のようになっています。505e9581-0d1c-42fb-a214-6ff736822e8c単語を含むセルの子 ID はありますか名前。次の例は、セルの情報です。

```
{
    "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"
},
```

各セルは表内の位置を持ち、最初のセルは 1,1 です。前の例では、値が設定されたセル名前は行 1、列 1 にあります。値を持つセル123 Any Townは行 2、列 2 にあります。セルブロックオブジェクトには、この情報がRowIndexそしてColumnIndexフィールド。子リストには、セル内のテキストを含む WORD Block オブジェクトの ID が含まれます。リスト内の単語は、セルの左上からセルの右下まで、検出された順に表示されます。前の例では、セルに値 e9108c8e-0167-4482-989e-8b6cd3c3653e を持つ子 ID があります。次の出力は、E9108c8e-0167-4482-989e-8b6cd3c3653e のワードブロックに対するものです。

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

選択エレメント

Amazon Textract では、ドキュメントページのオプションボタン (ラジオボタン) やチェックボックスなどの選択要素を検出できます。選択エレメントはで検出できますフォームデータそしてでテーブル。たとえば、フォームで次の表が検出されると、Amazon Textract はテーブルのセル内のチェックボックスを検出します。

	同意	普通	反対する
優れたサービス	☑	☐	☐
使いやすさ	☐	☑	☐
公正価格	☑	☐	☐

検出された選択要素は、次のように返されます。Blockからの応答内のオブジェクトAnalyzeDocumentそしてGetDocumentAnalysis。

> **Note**
>
> ♪FeatureTypesキーと値のペア、テーブル、またはその両方に関する情報を取得するための入力パラメータ。たとえば、テーブルでフィルタリングすると、テーブルで検出された選択エレメントが応答に含まれます。キーと値のペアで検出された選択要素は、応答に含まれません。

選択要素に関する情報は、Block型のオブジェクトSELECTION_ELEMENT。選択可能な要素のステータスを確認するには、SelectionStatusのフィールドSELECTION_ELEMENTブロック。このステータスは、選択されたまたはNOT_SELECTED。たとえば、の値SelectionStatus前のイメージは選択された。

あるSELECTION_ELEMENT Blockオブジェクトは、キーと値のペアまたはテーブルセルのいずれかに関連付けられています。あるSELECTION_ELEMENT Blockオブジェクトには、選択エレメントのバウンディングボックス情報がGeometryフィールド。あるSELECTION_ELEMENT Blockオブジェクトは a の子ではありませんPAGE Blockオブジェクト。

フォームデータ (キーと値のペア)

キーと値のペアは、フォーム上で検出された選択要素を表すために使用されます。-KEYblock には、選択要素のテキストが含まれます。-VALUEブロックには SELECTION_ELEMENT ブロックが含まれます。次の図は、選択要素がどのように表示されるかを示しています。the section called “Block”オブジェクト。

キーと値のペアの詳細については、「」を参照してください。フォームデータ (キーと値のペア)。

次の JSON スニペットは、選択要素 () を含むキーと値のペアのキーを示しています。male ☑). 子 ID (ID bd14cfd5-9005-498b-a7f3-45ceb171f0ff) は、選択要素のテキストを含む単語ブロックのIDです (男性). 値ID (ID 24aaac7f-fcce-49c7-a4f0-3688b05586d4) は、VALUEを含むブロックSELECTION_ELEMENTブロックオブジェクト。

```
{
    "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"
}
```

次の JSON スニペットは、その単語の WORD ブロックです。男。WORD ブロックには親の LINE ブロックもあります。

```
{
    "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"
},
```

VALUE ブロックには、SELECTION_ELEMENT ブロックである子 (ID f2f5e8cd-e73a-4e99-a095-053acd3b6bfb) があります。

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

次の JSON は SELECTION_ELEMENT ブロックです。の価値SelectionStatusチェックボックスがオンになっていることを示します。

```
{
    "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"
}
```

表のセル

Amazon Textract は、テーブルセル内の選択要素を検出できます。たとえば、次の表のセルにはチェックボックスがあります。

	同意	普通	反対する
優れたサービス	☑	☐	☐
使いやすさ	☐	☑	☐
公正価格	☑	☐	☐

あるCELLブロックには子を含めることができます。SELECTION_ELEMENT選択工レメントのオブジェクト、および子工レメントのオブジェクトWORD検出されたテキストのブロック。

テーブルの詳細については、「」を参照してください。テーブル。

ザ· テーブルBlock前のテーブルのオブジェクトはこのようになります。

```
{
    "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"
}
```

セルのあるBLOCKチェックボックスが含まれるセルのオブジェクト (ID c63ad40d-5a14-4646-a8df-2d4304213dbc) を指定します。優れたサービス次のようになります。子供も含まれますBlock (Id = 26d122fd-c5f4-4b53-92c4-0ae92730ee1e) とはSELECTION_ELEMENT Blockチェックボックスのオブジェクト。

```
{
    "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_ELEMENTBlockチェックボックスのオブジェクトは次のとおりです。の価値SelectionStatusチェックボックスがオンになっていることを示します。

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

請求書および領収書応答オブジェクト

AnalyzeExpense API に請求書または領収書を送信すると、一連の expenseDocuments オブジェクトが返されます。各 expenseDocument は、LineItemGroupsそしてSummaryFields。ほとんどの請求書および領収書には、仕入先名、受入番号、受入日、合計金額などの情報が含まれています。AnalyzeExpense はこの情報をSummaryFields。領収書と請求書には、購入した商品に関する詳細も含まれています。AnalyzeExpense API はこの情報をLineItemGroups。-ExpenseIndexフィールドは、経費を一意に識別し、適切なものを関連付けますSummaryFieldsそしてLineItemGroupsその費用で検出されました。

AnalyzeExpense レスポンスの最も詳細なデータレベルは、Type,ValueDetection, およびLabelDetection(オプション). 個々のエンティティは次のとおりです。

- [Type] (タイプ): ハイレベルで検出される情報の種類を示します。
- ラベル検出 : ドキュメントのテキスト内の関連する値のラベルを参照します。LabelDetectionはオプションで、ラベルが書き込まれた場合にのみ返されます。
- 値検出: 返されるラベルまたは型の値を参照します。

AnalyzeExpense API も検出します。ITEM,QUANTITY, およびPRICE正規化されたフィールドとしてラインアイテム内。SKU や詳細な説明など、領収書画像上の行項目に他のテキストがある場合は、次のようにJSONに含まれます。EXPENSE_ROW以下の例に示すように、

```
            {
                                "Type": {
```

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

上の例は、AnalyzeExpense API が 2.5 ドルで販売された 5 つのバナナに関するラインアイテム情報を含む領収書の行全体を返す方法を示しています。

[Type] (タイプ)

以下に、キーと値のペアの標準型または正規化されたタイプの例を示します。

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

領収書には「ベンダー名」が明示的にリストされていませんでした。ただし、経費の分析 API は文書を領収書として認識し、値「AMAZON」をタイプとして分類しました。VENDOR_NAME。

ラベル検出

以下は、顧客ドキュメントページに表示されるテキストの例です。

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

例文書には「CASHIER Mina」が含まれていました。経費の分析 API は、現状の値を抽出して、LabelDetection。レシートに「キー」が明示的に表示されていない「ベンダー名」などの暗黙的な値については、LabelDetectionAnalyzeExpense 要素には含まれません。このような場合、AnalyzeExpense API は返されません。LabelDetection。

値検出

以下に、キーと値のペアの「値」の例を示します。

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

```
                    "Confidence": 87.89806365966797
                }
            }
```

この例では、文書には「CASHIER Mina」が含まれていました。AnalyzeExpense API はキャッシャーの値を Mina として検出し、`ValueDetection`。

ID ドキュメントの応答オブジェクト

AnalyzeID API にアイデンティティドキュメントを送信すると、一連のドキュメントが返されます。`IdentityDocumentField`オブジェクト。これらの各オブジェクトには次のものが含まれます。`Type`, および`Value`。TypeAmazon Textract が検出した正規化フィールドを記録します。`Value`正規化フィールドに関連付けられたテキストを記録します。

以下に、`IdentityDocumentField`、簡潔にするために短縮されます。

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

IdentityDocumentFields は、長い応答から切り取られた 2 つの例です。検出された型とその型の値の間には区切りがあります。ここでは、それぞれ姓名です。この構造は、含まれているすべての情報で繰り返されます。型が正規化フィールドとして認識されない場合は、「other」としてリストされます。

以下は、運転免許証の正規化されたフィールドのリストです。

- 名
- 姓
- ミドルネーム
- サフィックス
- 住所の市区町村
- 住所の郵便番号
- 住所の状態
- 郡
- ドキュメント番号
- 有効期限日
- 生年月日
- ステート名
- 発行日
- class
- 制限
- 保証
- ID タイプ ID
- ベテラン
- アドレス

米国パスポートの正規化フィールドのリストを以下に示します。

- 名
- 姓
- ミドルネーム
- ドキュメント番号

- 有効期限日
- 生年月日
- 生年月日
- 発行日
- ID タイプ ID

ドキュメントページ上のアイテムの場所

Amazon Textract オペレーションは、ドキュメントページで見つかったアイテムの場所とジオメトリを返します。DetectDocumentTextそしてGetDocumentTextDetectionラインと単語の位置とジオメトリを返します。AnalyzeDocumentそしてGetDocumentAnalysisキーと値のペア、表、セル、選択要素の位置とジオメトリを返します。

ドキュメントページのアイテムがどこにあるかを確認するには、境界ボックス (Geometry) Amazon Textract オペレーションによって返される情報Blockオブジェクト。-Geometryオブジェクトには、検出されたアイテムに関する 2 種類の位置情報とジオメトリ情報が含まれています。

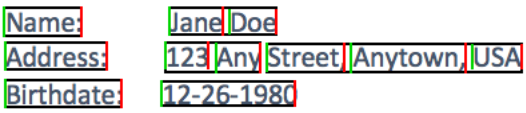
- 軸に位置合わせされたものBoundingBoxアイテムの左上の座標、および幅と高さを含むオブジェクト。
- アイテムのアウトラインを記述するポリゴンオブジェクト。の配列として指定します。Pointを含むオブジェクトX(横軸) とY(垂直軸) 各ポイントのドキュメントページ座標。

の JSONBlockオブジェクトは次のようになります。注意してくださいBoundingBoxそしてPolygonフィールド。

```
{
    "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"
},
```


ジオメトリ情報を使用して、検出されたアイテムの周囲に境界ボックスを描画できます。の例を挙げてBoundingBoxそしてPolygon各単語の先頭と末尾の線と垂直線の周りにボックスを描くための情報。Amazon Textract でドキュメントテキストの検出。出力例は、次の例のようになります。



境界ボックス

境界ボックス (BoundingBox) には以下のプロパティがあります。

- Height — ドキュメントページ全体の高さの比率としての境界ボックスの高さ。
- Left — ドキュメントページ全体の幅の比率としての境界ボックスの左上ポイントの X 座標。
- Top-ドキュメントページ全体の高さの比率としての境界ボックスの左上のポイントの Y 座標。
- Width — ドキュメントページ全体の幅の比率としての境界ボックスの幅。

各 BoundingBox プロパティには、0 から 1 の値があります。値は、画像全体の幅の比率です (LeftそしてWidth) または高さ (に適用)HeightそしてTop). たとえば、入力イメージが 700 x 200

ピクセルの場合、境界ボックスの左上の座標は 350,50 ピクセルで、API はLeft0.5 (350/700) の値と aTop0.25 (50/200) の値です。

次の図は、各 BoundingBox プロパティがカバーするドキュメントページの範囲を示しています。

適切な場所に適切なサイズで境界ボックスを表示するには、ピクセル値を取得するには、BoundingBox 値にドキュメントのページの幅または高さ (必要な値に応じて) を掛ける必要があります。境界ボックスを表示するには、ピクセル値を使用します。たとえば、幅608 ピクセル x 高さ588 ピクセルのドキュメントページと、分析されたテキストに次の境界ボックス値を使用します。

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

テキスト境界ボックスの位置 (ピクセル) は次のように計算されます。

```
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
```

これらの値を使用して、解析されたテキストの周囲に境界ボックスを表示します。次の Java および Python の例で、境界ボックスを表示する方法を示しています。

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

この Python の例では、responseによって返されるDetectDocumentTextAPI オペレーション。

```
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

によって返されるポリゴンAnalyzeDocumentの配列です。Pointオブジェクト。EachyPointは、ドキュメントページ上の特定の場所の X 座標と Y 座標を持ちます。BoundingBox 座標と同様に、ポリゴン座標はドキュメントの幅と高さに正規化され、0 ~ 1 の間になります。

ポリゴン配列内のポイントを使用して、細粒度の境界ボックスを囲むように表示することができます。Blockオブジェクト。ドキュメントページ上の各ポリゴンポイントの位置を計算するには、BoundingBoxes。X 座標にドキュメントページの幅を掛け、Y 座標にドキュメントページの高さを掛けます。

次の例は、ポリゴンの垂直線を表示する方法を示しています。

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

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

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

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

   }
```

Amazon Textract の開始方法

このセクションでは、Amazon Textract の使用を開始する方法について説明します。Amazon Textract を初めて使用する場合は、最初に「」の概念と用語に目をおすすめします。Amazon Textract 仕組み。

Amazon Textract コンソールでデモを使用して API を試すことができます。詳細については、「」を参照してください。https://console.aws.amazon.com/textract/。

トピック

- ステップ 1: AWS アカウントを設定して IAM ユーザーの作成
- ステップ 2: のセットアップAWS CLIそしてAWSSDK
- ステップ 3: の使用開始AWS CLIそしてAWSSDK API

ステップ 1: AWS アカウントを設定して IAM ユーザーの作成

Amazon Textract を初めて使用する場合は、事前に以下のタスクをすべて実行してください。

1. AWS にサインアップする.
2. IAM ユーザーを作成する.

AWS にサインアップする

Amazon Web Services (AWS) にサインアップすると、AWS アカウントが AWS でリリースされているすべてのサービスに自動的にサインアップされます。サービスを実際に使用した分の料金のみが請求されます。

Amazon Textract は、使用したリソース分のみお支払いいただくだけで利用可能です。Amazon Textract の使用料の詳細については、「」を参照してください。Amazon Textract の料金。AWS の新規のお客様の場合、Amazon Textract は無料で使い始めることができます。詳細については、「AWS 無料利用枠」を参照してください。

すでに AWS アカウントをお持ちの場合は次のタスクに進んでください。AWS アカウントをお持ちでない場合は、次に説明する手順を実行してアカウントを作成してください。

AWS アカウントを作成するには

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを用いて確認コードを入力するように求められます。

AWS アカウント ID は次のタスクでも必要となるので、メモしておいてください。

IAM ユーザーを作成する

Amazon Textract などの AWS のサービスは、サービスにアクセスする際に認証情報を提供する必要があります。これにより、サービスのリソースにアクセスする権限の有無が判定されます。コンソールを使用するにはパスワードが必要です。AWS アカウントによる AWS CLI または API へのアクセス用にアクセスキーを作成できます。ただし、AWS アカウントの認証情報を使用して AWS にアクセスすることはお勧めしません。代わりに、以下をお勧めします。

- を使用するAWS Identity and Access Management(IAM) を使用して IAM ユーザーを作成します。
- 管理権限を持つ IAM グループにユーザーを追加します。

その後、特別な URL とその IAM ユーザーの認証情報を使用して AWS にアクセスできます。

AWS にサインアップしても、ご自分の IAM ユーザーをまだ作成していない場合は、IAM コンソールを使用して作成できます。アカウントに IAM ユーザーを作成するには、次の手順に従います。

IAM ユーザーを作成してコンソールにサインインするには

1. AWS アカウントに管理者権限を持つ IAM ユーザーを作成します。手順については、IAM ユーザーガイドの「最初の IAM 管理者のユーザーおよびユーザーグループの作成」を参照してください。
2. IAM ユーザーとして、特別な URL を使用して AWS マネジメントコンソール にサインインします。詳細については、「」を参照してください。ユーザーがアカウントにサインインする方法のIAM ユーザーガイド。

> **Note**
>
> 管理者権限を持つ IAM ユーザーには、AWSアカウント内のサービス。
> このガイドのコード例では、AmazonTextractFullAccessアクセス許
> 可。AmazonS3ReadOnlyAccessAmazon S3 バケットに格納されているドキュメントにア
> クセスする場合は、が必要です。セキュリティ要件によっては、これらのアクセス権限に制
> 限した IAM グループを使用することもできます。詳細については、「」を参照してくださ
> い。IAM グループの作成。

IAM の詳細については、以下を参照してください。

- AWS Identity and Access Management (IAM)
- 開始方法
- IAM ユーザーガイド

次のステップ

ステップ 2: のセットアップAWS CLIそしてAWSSDK

ステップ 2: のセットアップAWS CLIそしてAWSSDK

以下の手順では、以下の例で使用する AWS Command Line Interface (AWS CLI) と AWS SDK をインストールする方法を示します。

AWS SDK の呼び出しは、さまざまな方法で認証できます。以下の例では、デフォルトの認証情報プロファイルを使用して AWS CLI コマンドと AWS SDK API オペレーションを呼び出します。デフォルトの認証情報はサービス間で機能するため、認証情報をすでに設定している場合は、再度行う必要はありません。ただし、このサービスの別のクレデンシャルセットを作成する場合は、名前プロファイルを作成できます。プロファイルの作成の詳細については、名前を指定されたプロファイルを参照してください。

使用可能なのリストについてはAWSリージョン、「」を参照してくださいのリージョンとエンドポイントのAmazon Web Services 全般リファレンス。

AWS CLI と AWS SDK をセットアップするには

1. 使用する AWS CLI と AWS SDK をダウンロードしてインストールします。このガイドでは、の例を示しています。AWS CLI、Java、Python、など。AWS SDK の詳細については、Amazon Web Services のツールを参照してください。

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

2. 「」で作成したユーザーのアクセスキーを作成します。IAM ユーザーを作成する。

   a. AWS マネジメントコンソール にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

   b. ナビゲーションペインで [Users] (ユーザー) を選択します。

   c. 「」で作成したユーザーの名前を選択します。IAM ユーザーを作成する。

   d. [Security credentials] タブを選択します。

   e. [アクセスキーの作成] を選択します。次に、[.csv ファイルのダウンロード] を選択して、ご使用のコンピュータにアクセスキー ID とシークレットアクセスキーをファイルに保存します。このファイルは安全な場所に保存してください。このダイアログボックスを閉じた後で、シークレットアクセスキーに再度アクセスすることはできません。CSV ファイルをダウンロードしたら、Close。

3. 次の場所にあるローカルシステム上の AWS 認証プロファイルファイルで認証情報を設定する

   - `~/.aws/credentials`Linux、macOS、Unix の場合は。
   - `C:\Users\USERNAME\.aws\credentials`Windows の場合は。

   -.awsフォルダは、AWS インスタンスの最初の初期設定の前に存在しません。CLI で認証情報を初めて設定すると、このフォルダが作成されます。AWS 認証情報の詳細については、「」を参照してください。設定ファイルと認証情報ファイルの設定。

   このファイルには以下の形式の行が含まれている必要があります。

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

アクセスキー ID とシークレットアクセスキーをyour_access_key_idそしてsecret_access_key。

4. AWS でデフォルトの AWS リージョンを設定するconfigローカルシステム上のファイル。次の場所にあります。

   - ~/.aws/configLinux、macOS、Unix の場合は。
   - C:\Users\USERNAME\.aws\configWindows の場合は。

   -.awsフォルダは、AWS インスタンスの最初の初期設定の前に存在しません。CLI で認証情報を初めて設定すると、このフォルダが作成されます。AWS 認証情報の詳細については、「」を参照してください。設定ファイルと認証情報ファイルの設定。

   このファイルには次の行が含まれています。

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

   使用する AWS リージョン (「us-west-2" など) に置き換えます。your_aws_region。

   > **Note**
   >
   > リージョンを選択しないと、デフォルトで us-east-1 が使用されます。

次のステップ

ステップ 3: の使用開始AWS CLIそしてAWSSDK API

ステップ 3: の使用開始AWS CLIそしてAWSSDK API

設定後AWS CLIそしてAWS使用する SDK は、Amazon Textract を使用するアプリケーションを構築できます。以下のトピックでは、Amazon Textract の開始方法を示しています。

- Amazon Textract を使用したドキュメントテキストの分析

AWS CLI のサンプルの書式設定

このガイドの AWS CLI の例は、Linux オペレーティングシステム用に書式設定されています。Microsoft Windows で例を使用するには、`--document` パラメータの JSON 形式を変更し、改行をバックスラッシュ (\) からキャレット (^) に変える必要があります。JSON 形式の詳細については、「AWS Command Line Interface のパラメータ値の指定」を参照してください。

同期操作によるドキュメントの処理

Amazon Textract は、JPEG、PNG、PDF、および TIFF 形式の画像として提供される単一ページのドキュメント内のテキストを検出して分析できます。オペレーションは同期しており、ほぼリアルタイムに結果を返します。ドキュメントの詳細については、「テキスト検出および文書分析応答オブジェクト」を参照してください。

このセクションでは、Amazon Textract を使用して、単一ページのドキュメント内のテキストを同期的に検出して分析する方法について説明します。複数ページのドキュメント内のテキストの検出と分析、または JPEG および PNG ドキュメントを非同期的に検出するには、を参照してください。非同期操作によるドキュメントの処理。

Amazon Textract 同期オペレーションは以下の目的に使用できます。

- テキスト検出-単一ページのドキュメントイメージの行と単語を検出するには、DetectDocumentTextオペレーション. 詳細については、「テキストの検出」を参照してください。
- テキスト分析-単一ページのドキュメントで検出されたテキスト間の関係を特定するには、AnalyzeDocumentオペレーション. 詳細については、「ドキュメントを分析する」を参照してください。
- 請求書と領収書の分析 — AnalyzeExpense操作を使用して、単一ページの請求書または領収書で検出されたテキスト間の財務関係を特定できます。詳細については、「」を参照してください。請求書と領収書の分析
- アイデンティティドキュメント分析 — 米国政府によって発行された身分証明書を分析し、アイデンティティドキュメントで見つかった一般的な種類の情報とともに情報を抽出できます。詳細については、「」を参照してください。アイデンティティドキュメントの分析。

トピック

- Amazon Textract 同期オペレーションを呼び出す
- Amazon Textract でドキュメントテキストの検出
- Amazon Textract を使用したドキュメントテキストの分析
- Amazon Textract を使用した請求書と領収書の分析
- Amazon Textract を使用したアイデンティティドキュメントの分析

Amazon Textract 同期オペレーションを呼び出す

Amazon Textract オペレーションは、ローカルファイルシステムに保存されているドキュメントイメージ、または Amazon S3 バケットに格納されているドキュメントイメージを処理します。入力ドキュメントの場所を指定するには、Document入力パラメータ。ドキュメントイメージは PNG、JPEG、PDF、または TIFF のいずれかの形式にすることができます。同期操作の結果はすぐに返され、取得用に格納されません。

詳しい例については、「」を参照してください。Amazon Textract でドキュメントテキストの検出。

リクエスト

以下では、Amazon Textract でのリクエストの仕組みについて説明します。

イメージバイトとして渡されたドキュメント

イメージを base64 でエンコードされたバイト配列として渡すことで、ドキュメントイメージを Amazon Textract オペレーションに渡すことができます。例として、ローカルファイルシステムからロードされたドキュメントイメージがあります。を使用している場合、コードでは、ドキュメントファイルのバイトをエンコードする必要がない場合があります。AWSAmazon Textract API オペレーションを呼び出すためのSDK。

イメージバイトは、Bytesのフィールドの入力パラメータ。次の例は、イメージバイトを渡す Amazon Textract オペレーションの入力 JSON を示しています。Bytes入力パラメータ。

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

> **Note**
>
> 使用しているAWS CLIでは、Amazon Textract オペレーションにイメージバイトを渡すことはできません。代わりに、Amazon S3 バケットに保存されたイメージを参照する必要があります。

次の Java コードは、ローカルファイルシステムからイメージをロードし、Amazon Textract オペレーションを呼び出します。

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

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

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


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

Amazon S3 バケットに保存されたドキュメント

Amazon Textract は、Amazon S3 バケットに保存されているドキュメントイメージを分析できます。を使用してバケットとファイル名を指定します。S3ObjectのフィールドDocument入力パラメータ。次の例では、Amazon S3 バケットに格納されているドキュメントを処理する Amazon Textract オペレーションの入力 JSON を示しています。

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

次の例では、Amazon S3 バケットに格納されているイメージを使用して Amazon Textract オペレーションを呼び出す方法を示しています。

```
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);
```

対処

への呼び出しの JSON レスポンスの例を次に示します。DetectDocumentText。詳細については、「テキストの検出」を参照してください。

```
{
{
  "DocumentMetadata": {
    "Pages": 1
  },
  "Blocks": [
    {
      "BlockType": "PAGE",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.9995205998420715,
          "Height": 1.0,
          "Left": 0.0,
          "Top": 0.0
        },
        "Polygon": [
          {
            "X": 0.0,
            "Y": 0.0
          },
          {
            "X": 0.9995205998420715,
            "Y": 2.297314024515845E-16
          },
          {
            "X": 0.9995205998420715,
            "Y": 1.0
          },
          {
            "X": 0.0,
            "Y": 1.0
          }
        ]
```

```
      },
      "Id": "ca4b9171-7109-4adb-a811-e09bbe4834dd",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "26085884-d005-4144-b4c2-4d83dc50739b",
            "ee9d01bc-d91c-401d-8c0a-eec76f5f7862",
            "404bb3d3-d7ab-4008-a195-5dec87a08664",
            "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
            "47aab5ab-be2c-4c73-97c7-d0a45454e843",
            "dd06bb49-6a56-4ea7-beec-a2aa09835c3c",
            "8837153d-81b8-4031-a49f-83a3d81803c2",
            "5dae3b74-9e95-4b62-99b7-93b88fe70648",
            "4508da80-64d8-42a8-8846-cfafe6eab10c",
            "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
            "f04bb223-d075-41c3-b328-7354611c826b",
            "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
            "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
            "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
            "359f3870-7183-43f5-b638-970f5cefe4d5",
            "b9deea0a-244c-4d54-b774-cf03fbaaa8b1",
            "e2a43881-f620-44f2-b067-500ce7dc8d4d",
            "41756974-64ef-432d-b4b2-34702505975a",
            "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
            "bc907357-63d6-43c0-ab87-80d7e76d377e",
            "2d727ca7-3acb-4bb9-a564-5885c90e9325",
            "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
            "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
            "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
            "ac4b9ee0-c9b2-4239-a741-5753e5282033",
            "ebc18885-48d7-45b8-90e3-d172b4357802",
            "babf6360-789e-49c1-9c78-0784acc14a0c"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.93761444091797,
      "Text": "Employment Application",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3391372561454773,
```

```
          "Height": 0.06906412541866302,
          "Left": 0.29548385739326477,
          "Top": 0.027493247762322426
        },
        "Polygon": [
          {
            "X": 0.29548385739326477,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.0965573713183403
          },
          {
            "X": 0.29548385739326477,
            "Y": 0.0965573713183403
          }
        ]
      },
      "Id": "26085884-d005-4144-b4c2-4d83dc50739b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "ed48dacc-d089-498f-8e93-1cee1e5f39f3",
            "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.91246795654297,
      "Text": "Application Information",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.19878505170345306,
          "Height": 0.03754019737243652,
          "Left": 0.03988289833068848,
          "Top": 0.14050349593162537
```

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

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

```
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.32843729853630066
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.32843729853630066
          }
        ]
      },
      "Id": "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "d782f847-225b-4a1b-b52d-f252f8221b1f",
            "fa69c5cd-c80d-4fac-81df-569edae8d259",
            "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.48902893066406,
      "Text": "Home Address: 123 Any Street, Any Town. USA",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.7431139945983887,
          "Height": 0.09577702730894089,
          "Left": 0.03359385207295418,
          "Top": 0.3258342146873474
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.3258342146873474
          },
```

```
          {
            "X": 0.7767078280448914,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "47aab5ab-be2c-4c73-97c7-d0a45454e843",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "acfbed90-4a00-42c6-8a90-d0a0756eea36",
            "046c8a40-bb0e-4718-9c71-954d3630e1dd",
            "82b838bc-4591-4287-8dea-60c94a4925e4",
            "5cdcde7a-f5a6-4231-a941-b6396e42e7ba",
            "beafd497-185f-487e-b070-db4df5803e94",
            "ef1b77fb-8ba6-41fe-ba53-dce039af22ed",
            "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e",
            "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.89382934570312,
      "Text": "Mailing Address: same as above",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.26575741171836853,
          "Height": 0.039571404457092285,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
```

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

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

```
          },
          {
            "X": 0.1175319030880928,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319030880928,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
          }
        ]
      },
      "Id": "5dae3b74-9e95-4b62-99b7-93b88fe70648",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45",
            "91e582cd-9871-4e9c-93cc-848baa426338"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86717224121094,
      "Text": "End Date",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07581500709056854,
          "Height": 0.03223184868693352,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
```

```
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
          },
          {
            "X": 0.14846202731132507,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "4508da80-64d8-42a8-8846-cfafe6eab10c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "7c97b56b-699f-49b0-93f4-98e6d90b107c",
            "7af04e27-0c15-447e-a569-b30edb99a133"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9539794921875,
      "Text": "Employer Name",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1347292959690094,
          "Height": 0.0392492413520813,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.3994368314743042,
            "Y": 0.6140711903572083
          },
          {
```

```
            "X": 0.3994368314743042,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
            "Y": 0.6533204317092896
          }
        ]
      },
      "Id": "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9bfeb55-75cd-47cd-b953-728e602a3564",
            "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.35584259033203,
      "Text": "Position Held",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11393272876739502,
          "Height": 0.03415105864405632,
          "Left": 0.49973347783088684,
          "Top": 0.614840030670166
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.6489911079406738
          },
```

```
          {
            "X": 0.49973347783088684,
            "Y": 0.6489911079406738
          }
        ]
      },
      "Id": "f04bb223-d075-41c3-b328-7354611c826b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "6d5edf02-845c-40e0-9514-e56d0d652ae0",
            "3297ab59-b237-45fb-ae60-a108f0c95ac2"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9817886352539,
      "Text": "Reason for leaving",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16511960327625275,
          "Height": 0.04062700271606445,
          "Left": 0.7430596351623535,
          "Top": 0.6116235852241516
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6522505879402161
```

```
          }
        ]
      },
      "Id": "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "f4b8cf26-d2da-4a76-8345-69562de3cc11",
            "386d4a63-1194-4c0e-a18d-4d074a0b1f93",
            "a8622541-1896-4d54-8d10-7da2c800ec5c"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.77413177490234,
      "Text": "1/15/2009",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08799663186073303,
          "Height": 0.03832906484603882,
          "Left": 0.03175082430243492,
          "Top": 0.691371738910675
        },
        "Polygon": [
          {
            "X": 0.03175082430243492,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.7297008037567139
          },
          {
            "X": 0.03175082430243492,
            "Y": 0.7297008037567139
          }
        ]
```

```
      },
      "Id": "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.72286224365234,
      "Text": "6/30/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843101561069489,
          "Height": 0.03991425037384033,
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
          }
        ]
      },
      "Id": "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
      "Relationships": [
        {
```

```
          "Type": "CHILD",
          "Ids": [
            "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86936950683594,
      "Text": "Any Company",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11800950765609741,
          "Height": 0.03943679481744766,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
          {
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.736709475517273
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.736709475517273
          }
        ]
      },
      "Id": "359f3870-7183-43f5-b638-970f5cefe4d5",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "77749c2b-aa7f-450e-8dd2-62bcaf253ba2",
            "713bad19-158d-4e3e-b01f-f5707ddb04e5"
```

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

```
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994903564453,
          "Height": 0.033302485942840576,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "e2a43881-f620-44f2-b067-500ce7dc8d4d",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98190307617188,
```

```
      "Text": "7/1/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067441940307617,
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.8451982140541077
          },
          {
            "X": 0.028500309213995934,
            "Y": 0.8451982140541077
          }
        ]
      },
      "Id": "41756974-64ef-432d-b4b2-34702505975a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0f711065-1872-442a-ba6d-8fababaa452a"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98418426513672,
      "Text": "8/10/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10664612054824829,
```

```
          "Height": 0.06439518928527832,
          "Left": 0.14159755408763885,
          "Top": 0.7791688442230225
        },
        "Polygon": [
          {
            "X": 0.14159755408763885,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.8435640335083008
          },
          {
            "X": 0.14159755408763885,
            "Y": 0.8435640335083008
          }
        ]
      },
      "Id": "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a92d8eef-db28-45ba-801a-5da0f589d277"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98075866699219,
      "Text": "Example Corp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.2114926278591156,
          "Height": 0.058415766805410385,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
```

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

```
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.847984790802002
          },
          {
            "X": 0.5098910331726074,
            "Y": 0.847984790802002
          }
        ]
      },
      "Id": "2d727ca7-3acb-4bb9-a564-5885c90e9325",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "00adeaef-ed57-44eb-b8a9-503575236d62"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.93852233886719,
      "Text": "better opp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18919607996940613,
          "Height": 0.06994765996932983,
          "Left": 0.7428008317947388,
          "Top": 0.7928366661071777
        },
        "Polygon": [
          {
            "X": 0.7428008317947388,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
```

```
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
            "Y": 0.8627843260765076
          },
          {
            "X": 0.7428008317947388,
            "Y": 0.8627843260765076
          }
        ]
      },
      "Id": "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "c0fc9a58-7a4b-4f69-bafd-2cff32be2665",
            "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.92573547363281,
      "Text": "8/15/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10257463902235031,
          "Height": 0.05412459373474121,
          "Left": 0.027909137308597565,
          "Top": 0.8608770370483398
        },
        "Polygon": [
          {
            "X": 0.027909137308597565,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.8608770370483398
          },
          {
```

```
            "X": 0.13048377633094788,
            "Y": 0.915001630783081
          },
          {
            "X": 0.027909137308597565,
            "Y": 0.915001630783081
          }
        ]
      },
      "Id": "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "5384f860-f857-4a94-9438-9dfa20eed1c6"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.99625396728516,
      "Text": "Present",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888341903686523,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
          {
```

```
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
      "Id": "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611276149749756,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
```

```
      },
      "Id": "ac4b9ee0-c9b2-4239-a741-5753e5282033",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "25343360-d906-440a-88b7-92eb89e95949"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.99549102783203,
      "Text": "head baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1937451809644699,
          "Height": 0.056156039237976074,
          "Left": 0.49359121918678284,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "ebc18885-48d7-45b8-90e3-d172b4357802",
      "Relationships": [
        {
```

```
          "Type": "CHILD",
          "Ids": [
            "0ef3c194-8322-4575-94f1-82819ee57e3a",
            "d296acd9-3e9a-4985-95f8-f863614f2c46"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98360443115234,
      "Text": "N/A, current",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.22544169425964355,
          "Height": 0.06588292121887207,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
        },
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.9381561279296875
          },
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "babf6360-789e-49c1-9c78-0784acc14a0c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "195cfb5b-ae06-4203-8520-4e4b0a73b5ce",
```

```
            "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
          ]
        }
      ]
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94815826416016,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.17462396621704102,
          "Height": 0.06266549974679947,
          "Left": 0.29548385739326477,
          "Top": 0.03389188274741173
        },
        "Polygon": [
          {
            "X": 0.29548385739326477,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.0965573862195015
          },
          {
            "X": 0.29548385739326477,
            "Y": 0.0965573862195015
          }
        ]
      },
      "Id": "ed48dacc-d089-498f-8e93-1cee1e5f39f3"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92706298828125,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.15933875739574432,
          "Height": 0.062391020357608795,
          "Left": 0.47528234124183655,
          "Top": 0.027493247762322426
        },
        "Polygon": [
          {
            "X": 0.47528234124183655,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.08988427370786667
          },
          {
            "X": 0.47528234124183655,
            "Y": 0.08988427370786667
          }
        ]
      },
      "Id": "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9821548461914,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09610454738140106,
          "Height": 0.03656719997525215,
          "Left": 0.03988289833068848,
          "Top": 0.14147649705410004
        },
        "Polygon": [
          {
            "X": 0.03988289833068848,
            "Y": 0.14147649705410004
          },
```

```
          {
            "X": 0.13598744571208954,
            "Y": 0.14147649705410004
          },
          {
            "X": 0.13598744571208954,
            "Y": 0.1780436933040619
          },
          {
            "X": 0.03988289833068848,
            "Y": 0.1780436933040619
          }
        ]
      },
      "Id": "efe3fc6d-becb-4520-80ee-49a329386aee"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.84278106689453,
      "Text": "Information",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10029315203428268,
          "Height": 0.03209415823221207,
          "Left": 0.13837480545043945,
          "Top": 0.14050349593162537
        },
        "Polygon": [
          {
            "X": 0.13837480545043945,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.17259766161441803
          },
          {
            "X": 0.13837480545043945,
            "Y": 0.17259766161441803
```

```
          }
        ]
      },
      "Id": "c2260852-6cfd-4a71-9fc6-62b2f9b02355"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.83993530273438,
      "Text": "Full",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03039788082242012,
          "Height": 0.031106330454349518,
          "Left": 0.03899926319718361,
          "Top": 0.21361036598682404
        },
        "Polygon": [
          {
            "X": 0.03899926319718361,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.24471670389175415
          },
          {
            "X": 0.03899926319718361,
            "Y": 0.24471670389175415
          }
        ]
      },
      "Id": "e94eb587-9545-4215-b0fc-8e8cb1172958"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93611907958984,
      "Text": "Name:",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.05555811896920204,
          "Height": 0.030184319242835045,
          "Left": 0.07123806327581406,
          "Top": 0.2137702852487564
        },
        "Polygon": [
          {
            "X": 0.07123806327581406,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2439546138048172
          },
          {
            "X": 0.07123806327581406,
            "Y": 0.2439546138048172
          }
        ]
      },
      "Id": "090aeba5-8428-4b7a-a54b-7a95a774120e"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.91043853759766,
      "Text": "Jane",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03905024006962776,
          "Height": 0.02941947989165783,
          "Left": 0.12933772802352905,
          "Top": 0.214289128780365
        },
        "Polygon": [
          {
            "X": 0.12933772802352905,
            "Y": 0.214289128780365
          },
```

```
          {
            "X": 0.16838796436786652,
            "Y": 0.214289128780365
          },
          {
            "X": 0.16838796436786652,
            "Y": 0.24370861053466797
          },
          {
            "X": 0.12933772802352905,
            "Y": 0.24370861053466797
          }
        ]
      },
      "Id": "64ff0abb-736b-4a6b-aa8d-ad2c0086ae1d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86123657226562,
      "Text": "Doe",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.035229459404945374,
          "Height": 0.030427640303969383,
          "Left": 0.17110899090766907,
          "Top": 0.21377210319042206
        },
        "Polygon": [
          {
            "X": 0.17110899090766907,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.244199737906456
          },
          {
            "X": 0.17110899090766907,
            "Y": 0.244199737906456
```

```
          }
        ]
      },
      "Id": "565ffc30-89d6-4295-b8c6-d22b4ed76584"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92633056640625,
      "Text": "Phone",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.052783288061618805,
          "Height": 0.03104414977133274,
          "Left": 0.03604753687977791,
          "Top": 0.28701552748680115
        },
        "Polygon": [
          {
            "X": 0.03604753687977791,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.31805968284606934
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.31805968284606934
          }
        ]
      },
      "Id": "d782f847-225b-4a1b-b52d-f252f8221b1f"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86275482177734,
      "Text": "Number:",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.07424934208393097,
          "Height": 0.030300479382276535,
          "Left": 0.0915418416261673,
          "Top": 0.28639692068099976
        },
        "Polygon": [
          {
            "X": 0.0915418416261673,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.3166973888874054
          },
          {
            "X": 0.0915418416261673,
            "Y": 0.3166973888874054
          }
        ]
      },
      "Id": "fa69c5cd-c80d-4fac-81df-569edae8d259"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97282409667969,
      "Text": "555-0100",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.17021971940994263,
          "Height": 0.047169629484415054,
          "Left": 0.17732827365398407,
          "Top": 0.2812676727771759
        },
        "Polygon": [
          {
            "X": 0.17732827365398407,
            "Y": 0.2812676727771759
          },
```

```
          {
            "X": 0.3475480079650879,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.32843729853630066
          },
          {
            "X": 0.17732827365398407,
            "Y": 0.32843729853630066
          }
        ]
      },
      "Id": "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.66238403320312,
      "Text": "Home",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.049357783049345016,
          "Height": 0.03134990110993385,
          "Left": 0.03359385207295418,
          "Top": 0.36172014474868774
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.3930700421333313
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.3930700421333313
```

```
          }
        ]
      },
      "Id": "acfbed90-4a00-42c6-8a90-d0a0756eea36"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.6871109008789,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07411003112792969,
          "Height": 0.0314042791724205,
          "Left": 0.08516156673431396,
          "Top": 0.3600046932697296
        },
        "Polygon": [
          {
            "X": 0.08516156673431396,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3914089798927307
          },
          {
            "X": 0.08516156673431396,
            "Y": 0.3914089798927307
          }
        ]
      },
      "Id": "046c8a40-bb0e-4718-9c71-954d3630e1dd"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93781280517578,
      "Text": "123",
      "TextType": "HANDWRITING",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.05761868134140968,
          "Height": 0.05008566007018089,
          "Left": 0.1750781387090683,
          "Top": 0.35484206676483154
        },
        "Polygon": [
          {
            "X": 0.1750781387090683,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.40492773056030273
          },
          {
            "X": 0.1750781387090683,
            "Y": 0.40492773056030273
          }
        ]
      },
      "Id": "82b838bc-4591-4287-8dea-60c94a4925e4"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.96530151367188,
      "Text": "Any",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06814215332269669,
          "Height": 0.06354366987943649,
          "Left": 0.2550157308578491,
          "Top": 0.35471394658088684
        },
        "Polygon": [
          {
            "X": 0.2550157308578491,
            "Y": 0.35471394658088684
          },
```

```
          {
            "X": 0.3231579065322876,
            "Y": 0.35471394658088684
          },
          {
            "X": 0.3231579065322876,
            "Y": 0.41825762391090393
          },
          {
            "X": 0.2550157308578491,
            "Y": 0.41825762391090393
          }
        ]
      },
      "Id": "5cdcde7a-f5a6-4231-a941-b6396e42e7ba"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.87527465820312,
      "Text": "Street,",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12156613171100616,
          "Height": 0.05449587106704712,
          "Left": 0.3357025980949402,
          "Top": 0.3550415635108948
        },
        "Polygon": [
          {
            "X": 0.3357025980949402,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.4095374345779419
          },
          {
            "X": 0.3357025980949402,
            "Y": 0.4095374345779419
```

```
          }
        ]
      },
      "Id": "beafd497-185f-487e-b070-db4df5803e94"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99514770507812,
      "Text": "Any",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07748188823461533,
          "Height": 0.07339789718389511,
          "Left": 0.47723668813705444,
          "Top": 0.3482133150100708
        },
        "Polygon": [
          {
            "X": 0.47723668813705444,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.47723668813705444,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "ef1b77fb-8ba6-41fe-ba53-dce039af22ed"
    },
    {
      "BlockType": "WORD",
      "Confidence": 96.80656433105469,
      "Text": "Town.",
      "TextType": "HANDWRITING",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.11213835328817368,
          "Height": 0.057233039289712906,
          "Left": 0.5563329458236694,
          "Top": 0.3331930637359619
        },
        "Polygon": [
          {
            "X": 0.5563329458236694,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3904260993003845
          },
          {
            "X": 0.5563329458236694,
            "Y": 0.3904260993003845
          }
        ]
      },
      "Id": "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98260498046875,
      "Text": "USA",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08771833777427673,
          "Height": 0.05706935003399849,
          "Left": 0.6889894604682922,
          "Top": 0.3258342146873474
        },
        "Polygon": [
          {
            "X": 0.6889894604682922,
            "Y": 0.3258342146873474
          },
```

```
          {
            "X": 0.7767078280448914,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.3829035460948944
          },
          {
            "X": 0.6889894604682922,
            "Y": 0.3829035460948944
          }
        ]
      },
      "Id": "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9583969116211,
      "Text": "Mailing",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06291338801383972,
          "Height": 0.03957144916057587,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
            "X": 0.03068041242659092,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.4730895459651947
          },
          {
            "X": 0.03068041242659092,
            "Y": 0.4730895459651947
```

```
          }
        ]
      },
      "Id": "d7261cdc-6ac5-4711-903c-4598fe94952d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.87476348876953,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07364854216575623,
          "Height": 0.03147412836551666,
          "Left": 0.0954652726650238,
          "Top": 0.43450701236724854
        },
        "Polygon": [
          {
            "X": 0.0954652726650238,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.465981125831604
          },
          {
            "X": 0.0954652726650238,
            "Y": 0.465981125831604
          }
        ]
      },
      "Id": "287f80c3-6db2-4dd7-90ec-5f017c80aa31"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94071960449219,
      "Text": "same",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.04640670120716095,
          "Height": 0.026415130123496056,
          "Left": 0.17156922817230225,
          "Top": 0.44010937213897705
        },
        "Polygon": [
          {
            "X": 0.17156922817230225,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.46652451157569885
          },
          {
            "X": 0.17156922817230225,
            "Y": 0.46652451157569885
          }
        ]
      },
      "Id": "ce31c3ad-b51e-4068-be64-5fc9794bc1bc"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.76510620117188,
      "Text": "as",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.02041218988597393,
          "Height": 0.025104399770498276,
          "Left": 0.2207803726196289,
          "Top": 0.44124215841293335
        },
        "Polygon": [
          {
            "X": 0.2207803726196289,
            "Y": 0.44124215841293335
          },
```

```
          {
            "X": 0.24119256436824799,
            "Y": 0.44124215841293335
          },
          {
            "X": 0.24119256436824799,
            "Y": 0.4663465619087219
          },
          {
            "X": 0.2207803726196289,
            "Y": 0.4663465619087219
          }
        ]
      },
      "Id": "e96eb92c-6774-4d6f-8f4a-68a7618d4c66"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9301528930664,
      "Text": "above",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05268359184265137,
          "Height": 0.03216424956917763,
          "Left": 0.24375422298908234,
          "Top": 0.4354657828807831
        },
        "Polygon": [
          {
            "X": 0.24375422298908234,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4676300287246704
          },
          {
            "X": 0.24375422298908234,
            "Y": 0.4676300287246704
```

```
          }
        ]
      },
      "Id": "88b85c05-427a-4d4f-8cc4-3667234e8364"
    },
    {
      "BlockType": "WORD",
      "Confidence": 85.3905029296875,
      "Text": "Previous",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09860499948263168,
          "Height": 0.04000622034072876,
          "Left": 0.3194798231124878,
          "Top": 0.5194430351257324
        },
        "Polygon": [
          {
            "X": 0.3194798231124878,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5594492554664612
          },
          {
            "X": 0.3194798231124878,
            "Y": 0.5594492554664612
          }
        ]
      },
      "Id": "8b324501-bf38-4ce9-9777-6514b7ade760"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.14524841308594,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.14039960503578186,
          "Height": 0.04645847901701927,
          "Left": 0.4214291274547577,
          "Top": 0.5219109654426575
        },
        "Polygon": [
          {
            "X": 0.4214291274547577,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.568369448184967
          },
          {
            "X": 0.4214291274547577,
            "Y": 0.568369448184967
          }
        ]
      },
      "Id": "b0cea99a-5045-464d-ac8a-a63ab0470995"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.48454284667969,
      "Text": "History",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08361124992370605,
          "Height": 0.05192042887210846,
          "Left": 0.5668527483940125,
          "Top": 0.5172380208969116
        },
        "Polygon": [
          {
            "X": 0.5668527483940125,
            "Y": 0.5172380208969116
          },
```

```
          {
            "X": 0.6504639983177185,
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
            "Y": 0.5691584348678589
          },
          {
            "X": 0.5668527483940125,
            "Y": 0.5691584348678589
          }
        ]
      },
      "Id": "b92a6ee5-ca59-44dc-9c47-534c133b11e7"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.78699493408203,
      "Text": "Start",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.041341401636600494,
          "Height": 0.030926469713449478,
          "Left": 0.034429505467414856,
          "Top": 0.6124123334884644
        },
        "Polygon": [
          {
            "X": 0.034429505467414856,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
```

```
          }
        ]
      },
      "Id": "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.55198669433594,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03923053666949272,
          "Height": 0.03072454035282135,
          "Left": 0.07830137014389038,
          "Top": 0.6123942136764526
        },
        "Polygon": [
          {
            "X": 0.07830137014389038,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6431187391281128
          },
          {
            "X": 0.07830137014389038,
            "Y": 0.6431187391281128
          }
        ]
      },
      "Id": "91e582cd-9871-4e9c-93cc-848baa426338"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8897705078125,
      "Text": "End",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.03212086856365204,
          "Height": 0.03193363919854164,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6439804434776306
          },
          {
            "X": 0.14846202731132507,
            "Y": 0.6439804434776306
          }
        ]
      },
      "Id": "7c97b56b-699f-49b0-93f4-98e6d90b107c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8445816040039,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03987143933773041,
          "Height": 0.03142518177628517,
          "Left": 0.1844055950641632,
          "Top": 0.612853467464447
        },
        "Polygon": [
          {
            "X": 0.1844055950641632,
            "Y": 0.612853467464447
          },
```

```
          {
            "X": 0.22427703440189362,
            "Y": 0.612853467464447
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
          },
          {
            "X": 0.1844055950641632,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "7af04e27-0c15-447e-a569-b30edb99a133"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9652328491211,
      "Text": "Employer",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08150768280029297,
          "Height": 0.0392492301762104,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
            "Y": 0.6533204317092896
```

```
          }
        ]
      },
      "Id": "a9bfeb55-75cd-47cd-b953-728e602a3564"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94273376464844,
      "Text": "Name",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05018233880400658,
          "Height": 0.03248906135559082,
          "Left": 0.34925445914268494,
          "Top": 0.6162016987800598
        },
        "Polygon": [
          {
            "X": 0.34925445914268494,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6486907601356506
          },
          {
            "X": 0.34925445914268494,
            "Y": 0.6486907601356506
          }
        ]
      },
      "Id": "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
    },
    {
      "BlockType": "WORD",
      "Confidence": 98.85071563720703,
      "Text": "Position",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.07007700204849243,
          "Height": 0.03255689889192581,
          "Left": 0.49973347783088684,
          "Top": 0.6164342164993286
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6489911079406738
          },
          {
            "X": 0.49973347783088684,
            "Y": 0.6489911079406738
          }
        ]
      },
      "Id": "6d5edf02-845c-40e0-9514-e56d0d652ae0"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86096954345703,
      "Text": "Held",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.04017873853445053,
          "Height": 0.03292537108063698,
          "Left": 0.5734874606132507,
          "Top": 0.614840030670166
        },
        "Polygon": [
          {
            "X": 0.5734874606132507,
            "Y": 0.614840030670166
          },
```

```
          {
            "X": 0.6136662364006042,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136662364006042,
            "Y": 0.6477653980255127
          },
          {
            "X": 0.5734874606132507,
            "Y": 0.6477653980255127
          }
        ]
      },
      "Id": "3297ab59-b237-45fb-ae60-a108f0c95ac2"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97740936279297,
      "Text": "Reason",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06497219949960709,
          "Height": 0.03248770162463188,
          "Left": 0.7430596351623535,
          "Top": 0.6136704087257385
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6461580991744995
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6461580991744995
```

```
          }
        ]
      },
      "Id": "f4b8cf26-d2da-4a76-8345-69562de3cc11"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98371887207031,
      "Text": "for",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.029645200818777084,
          "Height": 0.03462234139442444,
          "Left": 0.8108851909637451,
          "Top": 0.6117717623710632
        },
        "Polygon": [
          {
            "X": 0.8108851909637451,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6463940739631653
          },
          {
            "X": 0.8108851909637451,
            "Y": 0.6463940739631653
          }
        ]
      },
      "Id": "386d4a63-1194-4c0e-a18d-4d074a0b1f93"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98424530029297,
      "Text": "leaving",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.06517849862575531,
          "Height": 0.040626998990774155,
          "Left": 0.8430007100105286,
          "Top": 0.6116235852241516
        },
        "Polygon": [
          {
            "X": 0.8430007100105286,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
          {
            "X": 0.8430007100105286,
            "Y": 0.6522505879402161
          }
        ]
      },
      "Id": "a8622541-1896-4d54-8d10-7da2c800ec5c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.77413177490234,
      "Text": "1/15/2009",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08799663186073303,
          "Height": 0.03832906112074852,
          "Left": 0.03175082430243492,
          "Top": 0.691371738910675
        },
        "Polygon": [
          {
            "X": 0.03175082430243492,
            "Y": 0.691371738910675
          },
```

```
          {
            "X": 0.11974745243787766,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.7297008037567139
          },
          {
            "X": 0.03175082430243492,
            "Y": 0.7297008037567139
          }
        ]
      },
      "Id": "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.72286224365234,
      "Text": "6/30/2011",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843102306127548,
          "Height": 0.03991425037384033,
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
```

```
          }
        ]
      },
      "Id": "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92295837402344,
      "Text": "Any",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.034067559987306595,
          "Height": 0.037968240678310394,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
          {
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.7352409362792969
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.7352409362792969
          }
        ]
      },
      "Id": "77749c2b-aa7f-450e-8dd2-62bcaf253ba2"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.81578063964844,
      "Text": "Company",
      "TextType": "PRINTED",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.08160992711782455,
          "Height": 0.03890080004930496,
          "Left": 0.29906952381134033,
          "Top": 0.6978086829185486
        },
        "Polygon": [
          {
            "X": 0.29906952381134033,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.736709475517273
          },
          {
            "X": 0.29906952381134033,
            "Y": 0.736709475517273
          }
        ]
      },
      "Id": "713bad19-158d-4e3e-b01f-f5707ddb04e5"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.37964630126953,
      "Text": "Assistant",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.0789310410618782,
          "Height": 0.03139699995517731,
          "Left": 0.49814170598983765,
          "Top": 0.7005078196525574
        },
        "Polygon": [
          {
            "X": 0.49814170598983765,
            "Y": 0.7005078196525574
          },
```

```
          {
            "X": 0.5770727396011353,
            "Y": 0.7005078196525574
          },
          {
            "X": 0.5770727396011353,
            "Y": 0.7319048047065735
          },
          {
            "X": 0.49814170598983765,
            "Y": 0.7319048047065735
          }
        ]
      },
      "Id": "989944f9-f684-4714-87d8-9ad9a321d65c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.784912109375,
      "Text": "baker",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.050264399498701096,
          "Height": 0.03237773850560188,
          "Left": 0.5806865096092224,
          "Top": 0.699238657951355
        },
        "Polygon": [
          {
            "X": 0.5806865096092224,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.7316163778305054
          },
          {
            "X": 0.5806865096092224,
            "Y": 0.7316163778305054
```

```
          }
        ]
      },
      "Id": "ae82e2aa-1601-4e0c-8340-1db7ad0c9a31"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994158506393,
          "Height": 0.03330250084400177,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98190307617188,
      "Text": "7/1/2011",
      "TextType": "HANDWRITING",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067439705133438,
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.8451982140541077
          },
          {
            "X": 0.028500309213995934,
            "Y": 0.8451982140541077
          }
        ]
      },
      "Id": "0f711065-1872-442a-ba6d-8fababaa452a"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98418426513672,
      "Text": "8/10/2013",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10664612054824829,
          "Height": 0.06439515948295593,
          "Left": 0.14159755408763885,
          "Top": 0.7791688442230225
        },
        "Polygon": [
          {
            "X": 0.14159755408763885,
            "Y": 0.7791688442230225
          },
```

```
          {
            "X": 0.24824367463588715,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.843563973903656
          },
          {
            "X": 0.14159755408763885,
            "Y": 0.843563973903656
          }
        ]
      },
      "Id": "a92d8eef-db28-45ba-801a-5da0f589d277"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97722625732422,
      "Text": "Example",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12127546221017838,
          "Height": 0.05682983994483948,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
        "Polygon": [
          {
            "X": 0.26764172315597534,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.8512446284294128
          },
          {
            "X": 0.26764172315597534,
            "Y": 0.8512446284294128
```

```
          }
        ]
      },
      "Id": "d6962efb-34ab-4ffb-9f2f-5f263e813558"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98429870605469,
      "Text": "Corp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07650306820869446,
          "Height": 0.05481306090950966,
          "Left": 0.4026312530040741,
          "Top": 0.7980174422264099
        },
        "Polygon": [
          {
            "X": 0.4026312530040741,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.8528305292129517
          },
          {
            "X": 0.4026312530040741,
            "Y": 0.8528305292129517
          }
        ]
      },
      "Id": "1876c8ea-d3e8-4c39-870e-47512b3b5080"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.91166687011719,
      "Text": "Baker",
      "TextType": "HANDWRITING",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.09931197017431259,
          "Height": 0.06008723005652428,
          "Left": 0.5098910331726074,
          "Top": 0.787897527217865
        },
        "Polygon": [
          {
            "X": 0.5098910331726074,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.8479847311973572
          },
          {
            "X": 0.5098910331726074,
            "Y": 0.8479847311973572
          }
        ]
      },
      "Id": "00adeaef-ed57-44eb-b8a9-503575236d62"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98870849609375,
      "Text": "better",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10782185196876526,
          "Height": 0.06207133084535599,
          "Left": 0.7428008317947388,
          "Top": 0.7928366661071777
        },
        "Polygon": [
          {
            "X": 0.7428008317947388,
            "Y": 0.7928366661071777
          },
```

```
          {
            "X": 0.8506226539611816,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.8506226539611816,
            "Y": 0.8549079895019531
          },
          {
            "X": 0.7428008317947388,
            "Y": 0.8549079895019531
          }
        ]
      },
      "Id": "c0fc9a58-7a4b-4f69-bafd-2cff32be2665"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8883285522461,
      "Text": "opp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07421936094760895,
          "Height": 0.058906231075525284,
          "Left": 0.8577775359153748,
          "Top": 0.8038780689239502
        },
        "Polygon": [
          {
            "X": 0.8577775359153748,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8627843260765076
          },
          {
            "X": 0.8577775359153748,
            "Y": 0.8627843260765076
```

```
        }
      ]
    },
    "Id": "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.92573547363281,
    "Text": "8/15/2013",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.10257463902235031,
        "Height": 0.05412459000945091,
        "Left": 0.027909137308597565,
        "Top": 0.8608770370483398
      },
      "Polygon": [
        {
          "X": 0.027909137308597565,
          "Y": 0.8608770370483398
        },
        {
          "X": 0.13048377633094788,
          "Y": 0.8608770370483398
        },
        {
          "X": 0.13048377633094788,
          "Y": 0.915001630783081
        },
        {
          "X": 0.027909137308597565,
          "Y": 0.915001630783081
        }
      ]
    },
    "Id": "5384f860-f857-4a94-9438-9dfa20eed1c6"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.99625396728516,
    "Text": "Present",
    "TextType": "HANDWRITING",
    "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888339668512344,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
          {
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
      "Id": "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611273169517517,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
```

```
          {
            "X": 0.4476994276046753,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994276046753,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
      },
      "Id": "25343360-d906-440a-88b7-92eb89e95949"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99523162841797,
      "Text": "head",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07429949939250946,
          "Height": 0.05485520139336586,
          "Left": 0.49359121918678284,
          "Top": 0.8714361190795898
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.926291286945343
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.926291286945343
```

```
          }
        ]
      },
      "Id": "0ef3c194-8322-4575-94f1-82819ee57e3a"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99574279785156,
      "Text": "baker",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1019822508096695,
          "Height": 0.05615599825978279,
          "Left": 0.585354208946228,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
            "X": 0.585354208946228,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.585354208946228,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "d296acd9-3e9a-4985-95f8-f863614f2c46"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9880599975586,
      "Text": "N/A,",
      "TextType": "HANDWRITING",
      "Geometry": {
```

```
        "BoundingBox": {
          "Width": 0.08230073750019073,
          "Height": 0.06588289886713028,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
        },
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.9381561279296875
          },
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "195cfb5b-ae06-4203-8520-4e4b0a73b5ce"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97914123535156,
      "Text": "current",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12791454792022705,
          "Height": 0.04768490046262741,
          "Left": 0.8387037515640259,
          "Top": 0.8843405842781067
        },
        "Polygon": [
          {
            "X": 0.8387037515640259,
            "Y": 0.8843405842781067
          },
```

```
          {
            "X": 0.9666182994842529,
            "Y": 0.8843405842781067
          },
          {
            "X": 0.9666182994842529,
            "Y": 0.9320254921913147
          },
          {
            "X": 0.8387037515640259,
            "Y": 0.9320254921913147
          }
        ]
      },
      "Id": "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
    }
  ],
  "DetectDocumentTextModelVersion": "1.0",
  "ResponseMetadata": {
    "RequestId": "337129e6-3af7-4014-842b-f6484e82cbf6",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "337129e6-3af7-4014-842b-f6484e82cbf6",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "45675",
      "date": "Mon, 09 Nov 2020 23:54:38 GMT"
    },
    "RetryAttempts": 0
  }
}
}
```


Amazon Textract でドキュメントテキストの検出

文書内のテキストを検出するには、DetectDocumentText操作を行い、ドキュメントファイルを入力として渡します。DetectDocumentText検出されたテキストの行と単語、ドキュメント内のテキストの位置、および検出されたテキスト間の関係を含む JSON 構造を返します。詳細については、「テキストの検出」を参照してください。

入力ドキュメントとして、イメージのバイト配列 (base64 エンコードされたイメージのバイト) を指定するか、Amazon S3 オブジェクトを指定できます。以下の手順では、イメージファイルを S3 バケットにアップロードし、そのファイル名を指定します。

ドキュメント内のテキストを検出するには (API)

1. まだ実行していない場合:

   a. を使用して IAM ユーザーを作成または更新するAmazonTextractFullAccessそしてAmazonS3ReadOnlyAccessアクセス許可。詳細については、「ステップ 1: AWS アカウントを設定して IAM ユーザーの作成」を参照してください。

   b. AWS CLI と AWS SDK をインストールして設定します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

2. ドキュメントを S3 バケットにアップロードします。

   手順については、以下を参照してください。Amazon S3 へのオブジェクトのアップロードのAmazon Simple Storage Service ユーザーガイド。

3. 以下の例を使用して、DetectDocumentText オペレーションを呼び出します。

   Java

   次のサンプルコードは、検出されたテキストの行を囲むドキュメントとボックスを表示します。

   関数内でmainで、の値を置き換えます。bucketそしてdocumentは、ステップ 2 で使用した Amazon S3 バケット名とドキュメント名を使用します。

```
//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

この AWS CLI コマンドでは、detect-document-text CLI オペレーションの JSON 出力を表示します。

の値を置換するBucketそしてNameステップ 2 で使用した Amazon S3 バケット名とドキュメント名を指定します。

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

Python

次のコード例では、テキスト行周辺のドキュメントとボックスを表示します。

関数内でmainで、の値を置き換えます。bucketそしてdocumentステップ 2 で使用した Amazon S3 バケット名とドキュメント名を指定します。

```
#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

次の Node.js サンプルコードは、検出されたテキスト行の周囲のドキュメントとボックスを表示し、コードの実行元のディレクトリに結果のイメージを出力します。これは、`image-size`そして`images`パッケージ。

関数内で`main`で、の値を置き換えます。`bucket`そして`document`ステップ 2 で使用した Amazon S3 バケット名とドキュメント名を指定します。の値を置換する`regionConfig`アカウントが属しているリージョンの名前を指定します。

```
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. 例を実行します。Python および Java の例には、ドキュメントイメージが表示されます。検出されたテキストの各行をブラックボックスで囲みます。緑色の縦線は、検出された単語の先頭です。赤い縦線は、検出された単語の終わりです。-AWS CLI例では、の JSON 出力のみを表示します。DetectDocumentTextオペレーション.

Amazon Textract を使用したドキュメントテキストの分析

文書内のテキストを分析するには、AnalyzeDocument操作を行い、ドキュメントファイルを入力として渡します。AnalyzeDocumentは、分析されたテキストを含む JSON 構造を返します。詳細については、「ドキュメントを分析する」を参照してください。

入力ドキュメントとして、イメージのバイト配列 (base64 エンコードされたイメージのバイト) を指定するか、Amazon S3 オブジェクトを指定できます。以下の手順では、イメージファイルを S3 バケットにアップロードし、そのファイル名を指定します。

ドキュメント内のテキストを分析するには (API)

1. まだ実行していない場合:

   a. を使用して IAM ユーザーを作成または更新するAmazonTextractFullAccessそしてAmazonS3ReadOnlyAccessアクセス許可。詳細については、「ステップ 1: AWS アカウントを設定して IAM ユーザーの作成」を参照してください。

   b. AWS CLI と AWS SDK をインストールして設定します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

2. ドキュメントが含まれているイメージを S3 バケットにアップロードします。

   手順については、以下を参照してください。Amazon S3 へのオブジェクトのアップロードのAmazon Simple Storage Service ユーザーガイド。

3. 以下の例を使用して、AnalyzeDocument オペレーションを呼び出します。

   Java

   次のコード例では、検出されたアイテムの周りのドキュメントとボックスを表示します。

   関数内でmainで、の値を置き換えます。bucketそしてdocumentステップ 2 で使用した Amazon S3 バケット名とドキュメントイメージ名を指定します。

```
//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

この AWS CLI コマンドでは、`detect-document-text` CLI オペレーションの JSON 出力を表示します。

の値を置換するBucketそしてNameステップ 2 で使用した Amazon S3 バケット名とドキュメント名を指定します。

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

Python

次のコード例では、検出されたアイテムの周りのドキュメントとボックスを表示します。

関数内でmainで、の値を置き換えます。bucketそしてdocumentステップ 2 で使用した Amazon S3 バケット名とドキュメント名を指定します。

```
#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

次のコード例では、検出されたアイテムの周りのドキュメントとボックスを表示します。

以下のコードで、の値を置き換えます。bucketそしてphotoステップ 2 で使用した Amazon S3 バケット名とドキュメント名を指定します。の値を置換するregionアカウントに関連付けられているリージョンをアカウントに関連付けます。

```
// 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. 例を実行します。Python および Java の例では、次のカラーのバウンディングボックスでドキュメントイメージを表示します。

   - 赤 — キーブロックオブジェクト
   - 緑 — VALUE ブロックオブジェクト
   - 青 — TABLE ブロックオブジェクト
   - 黄色 — CELL ブロックオブジェクト

   選択された選択工レメントは青で塗りつぶされます。

   -AWS CLI例では、の JSON 出力のみを表示します。`AnalyzeDocument`オペレーション.

Amazon Textract を使用した請求書と領収書の分析

請求書および領収書文書を分析するには、AnalyzeExpense API を使用し、ドキュメントファイルを入力として渡します。`AnalyzeExpense`は、分析されたテキストを含む JSON 構造体を返す同期操作です。詳細については、「請求書と領収書の分析」を参照してください。

請求書と領収書を非同期に分析するには、`StartExpenseAnalysis`入力ドキュメントファイルの処理を開始し、`GetExpenseAnalysis`結果を取得します。

入力ドキュメントとして、イメージのバイト配列 (base64 エンコードされたイメージのバイト) を指定するか、Amazon S3 オブジェクトを指定できます。以下の手順では、イメージファイルを S3 バケットにアップロードし、そのファイル名を指定します。

請求書または領収書を分析するには (API)

1. まだ実行していない場合:

   a. を使用して IAM ユーザーを作成または更新するAmazonTextractFullAccessそしてAmazonS3ReadOnlyAccessアクセス許可。詳細については、「ステップ 1: AWS アカウントを設定して IAM ユーザーの作成」を参照してください。

   b. AWS CLI と AWS SDK をインストールして設定します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

2. ドキュメントが含まれているイメージを S3 バケットにアップロードします。

   手順については、以下を参照してください。Amazon S3 へのオブジェクトのアップロードのAmazon Simple Storage Service ユーザーガイド。

3. 以下の例を使用して、AnalyzeExpense オペレーションを呼び出します。

   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. これにより、の JSON 出力が提供されます。AnalyzeExpenseオペレーション.

Amazon Textract を使用したアイデンティティドキュメントの分析

アイデンティティドキュメントを分析するには、AnalyzeID API を使用し、ドキュメントファイルを入力として渡します。AnalyzeIDは、分析されたテキストを含む JSON 構造を返します。詳細については、「アイデンティティドキュメントの分析」を参照してください。

入力ドキュメントとして、イメージのバイト配列 (base64 エンコードされたイメージのバイト) を指定するか、Amazon S3 オブジェクトを指定できます。以下の手順では、イメージファイルを S3 バケットにアップロードし、そのファイル名を指定します。

アイデンティティドキュメント (API) を分析するには

1. まだ実行していない場合:

   a. を使用して IAM ユーザーを作成または更新するAmazonTextractFullAccessそしてAmazonS3ReadOnlyAccessアクセス許可。詳細については、「ステップ 1: AWS アカウントを設定して IAM ユーザーの作成」を参照してください。

   b. AWS CLI と AWS SDK をインストールして設定します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

2. ドキュメントが含まれているイメージを S3 バケットにアップロードします。

   手順については、以下を参照してください。Amazon S3 へのオブジェクトのアップロードのAmazon Simple Storage Service ユーザーガイド。

3. 以下の例を使用して、AnalyzeID オペレーションを呼び出します。

   CLI

   次の例では、S3 バケットから入力ファイルを取り込み、AnalyzeIDそれに対する操作。以下のコードで、の値を置き換えます。####の S3 バケットの名前で、####は、バケット内のファイル名と##の名前でregionアカウントに関連付けられます。

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

   入力に別の S3 オブジェクトを追加することで、運転免許証の前面と背面で API を呼び出すこともできます。

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

Windows デバイスで CLI にアクセスする場合は、一重引用符ではなく二重引用符を使用し、内部の二重引用符をバックスラッシュ (つまり\ ) でエスケープして、発生する可能性のあるパーサーエラーに対処します。例については、以下を参照してください。

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

Python

次の例では、S3 バケットから入力ファイルを取り込み、AnalyzeIDそれを操作し、検出されたキーと値のペアを返します。以下のコードで、の値を置き換えます。*bucket_name*のS3 バケットの名前で、*file_name*は、バケット内のファイル名と##の名前でregionアカウントに関連付けられます。

```
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

次の例では、S3 バケットから入力ファイルを取り込み、AnalyzeIDそれを操作し、検出されたデータを返します。関数 main で、次の値を置き換えます。s3bucketそしてsourceDocステップ 2 で使用した Amazon S3 バケット名とドキュメントイメージ名を指定します。

```
/*
   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. これにより、の JSON 出力が提供されます。AnalyzeIDオペレーション.

非同期操作によるドキュメントの処理

Amazon Textract は、PDF または TIFF 形式の複数ページのドキュメント内のテキストを検出して分析できます。これには、請求書と領収書が含まれます。複数ページのドキュメント処理は、非同期オペレーションです。ドキュメントの非同期処理は、大規模な複数ページのドキュメントを処理する場合に便利です。たとえば、1,000 ページを超える PDF ファイルの処理には時間がかかります。PDF ファイルを非同期的に処理すると、アプリケーションはプロセスの完了を待っている間に他のタスクを完了できます。

このセクションでは、Amazon Textract を使用して、複数ページまたは単一ページのドキュメントのテキストを非同期的に検出して分析する方法について説明します。複数ページのドキュメントは PDF 形式または TIFF 形式である必要があります。非同期操作で処理される単一ページのドキュメントは、JPEG、PNG、TIFF、または PDF 形式にすることができます。

Amazon Textract 非同期オペレーションは、次の目的に使用できます。

- テキスト検出-複数ページの文書の行と単語を検出できます。非同期操作は次のとおりです。StartDocumentTextDetectionそしてGetDocumentTextDetection。詳細については、「テキストの検出」を参照してください。
- テキスト分析-複数ページのドキュメントで検出されたテキスト間の関係を特定できます。非同期操作は次のとおりです。StartDocumentAnalysisそしてGetDocumentAnalysis。詳細については、「ドキュメントを分析する」を参照してください。
- 経費分析 — 複数ページの請求書と領収書のデータ関係を特定できます。Amazon Textract は、複数ページのドキュメントの各請求書または領収書ページを個別の領収書または請求書として扱います。複数ページドキュメントの 1 つのページから別のページへのコンテキストは保持されません。非同期操作は次のとおりです。StartExpenseAnalysisそしてGetExpenseAnalysis。詳細については、「請求書と領収書の分析」を参照してください。

トピック

- Amazon Textract 非同期オペレーションを呼び出す
- 非同期オペレーション用の Amazon Textract の設定
- 複数ページドキュメント内のテキストの検出または分析
- Amazon Textract 結果通知

Amazon Textract 非同期オペレーションを呼び出す

Amazon Textract は、PDF または TIFF 形式で複数ページのドキュメントを処理するために使用できる非同期 API を提供します。非同期操作を使用して、JPEG、PNG、TIFF、または PDF 形式の単一ページのドキュメントを処理することもできます。

このトピックの情報は、テキスト検出操作を使用して Amazon Textract の非同期オペレーションの使用方法を示しています。同じアプローチが、のテキスト解析操作でも機能します。the section called “StartDocumentAnalysis”そしてthe section called “GetDocumentAnalysis”。また、で動作します。the section called “StartExpenseAnalysis”そしてthe section called “GetExpenseAnalysis”。

例については、複数ページドキュメント内のテキストの検出または分析 を参照してください。

Amazon Textract は、Amazon S3 バケットに保存されたドキュメントを非同期的に処理します。処理を開始するには、`Start`などの操作StartDocumentTextDetection。リクエストの完了ステータスが、Amazon Simple Notification Service (Amazon SNS) トピックに発行されます。Amazon SNS トピックから完了ステータスを取得するには、Amazon Simple Queue Service (Amazon SQS) キューまたはAWS Lambdafunction. 完了ステータスを取得したら、GetDocumentTextDetection などの `Get` オペレーションを呼び出してリクエストの結果を取得します。

非同期呼び出しの結果は、オペレーションを使用して Amazon S3 バケットを指定しない限り、デフォルトで暗号化され、Amazon Textract が所有するバケットに 7 日間保存されます。`OutputConfig`引数。

次の表に、Amazon Textract でサポートされているさまざまなタイプの非同期処理に対応する開始オペレーションと Get オペレーションを示します。

Amazon Textract 非同期オペレーションの API オペレーションの開始/取得

処理タイプ	API を起動する	API の取得
テキストの検出	StartDocumentTextDetection	GetDocumentTextDetection
テキスト分析	StartDocumentAnalysis	GetDocumentAnalysis
経費分析	経費分析の開始	経費分析を取得

という例を挙げてAWS Lambda関数、「」を参照してください。Amazon Textract による大規模なドキュメント処理。

以下の図は、Amazon S3 バケットに保存されているドキュメントイメージ内のドキュメントテキストを検出するプロセスを示しています。この図では、Amazon SQS キューによって Amazon SNS トピックから完了ステータスが表示されます。

前の図に表示されるプロセスは、テキストおよび請求書/領収書の分析と同じです。テキスト分析を開始するには、を呼び出します。the section called “StartDocumentAnalysis”そして、電話をかけて請求書/領収書の分析を開始するthe section called “StartExpenseAnalysis”あなたは呼び出すことによって結果を得ますthe section called “GetDocumentAnalysis”またはthe section called “GetExpenseAnalysis”それぞれ。

テキストの検出の開始

Amazon Textract テキストリクエストを開始するには、を呼び出します。StartDocumentTextDetection。以下は、StartDocumentTextDetection により渡される JSON リクエストの例を示しています。

```
{
    "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"
}
```

入力パラメータDocumentLocationは、ドキュメントファイル名と、取得先の Amazon S3 バケットを提供します。NotificationChannelには、Amazon SNS トピックの Amazon リソースネーム (ARN) と、テキスト検出リクエストの完了時に Amazon Textract が通知します。Amazon SNS トピックは、呼び出し先の Amazon Textract エンドポイントと同じ AWS リージョン内に存在する必要があります。NotificationChannelには、Amazon Textract を Amazon SNS トピックに発行できるようにするためのロールの ARN も含まれています。Amazon SNS トピックへの Amazon Textract 発行アクセス許可を付与するには、IAM サービスロールを作成します。詳細については、「非同期オペレーション用の Amazon Textract の設定」を参照してください。

オプションの入力パラメータを指定することもできます。JobTagを選択すると、Amazon SNS トピックに発行された完了ステータスのジョブまたはジョブのグループを特定できます。たとえば、を使用すると、JobTag税金フォームや領収書など、処理中の文書のタイプを識別します。

ClientRequestToken というべき等トークンをオプションで設定すると、分析ジョブが誤って重複するのを防ぐことができます。の値を指定した場合ClientRequestTokenとすると、Startオペレーションが同じを返すJobIdへの複数の同一のコールに対してStartなどの操作StartDocumentTextDetection。ClientRequestToken トークンの有効期間は 7 日です。7 日後に再利用することができます。トークンの有効期間中にトークンを再利用すると、以下のことが発生します。

- 同じ Start オペレーション、同じ入力パラメータでトークンを再利用すると、同じ JobId が返されます。ジョブは再度実行されず、Amazon Textract は、登録されている Amazon SNS トピックに完了ステータスを送信しません。
- 同じ Start オペレーションで、入力パラメータを少し変更してトークンを再利用すると、idempotentparametermismatchexception (HTTP ステータスコード: 400) 例外を受け取ります。
- 別の Start オペレーションでトークンを再利用すると、オペレーションは正常に実行されます。

使用可能なもう 1 つのオプションパラメータは次のとおりです。OutputConfigで、出力の配置場所を調整できます。デフォルトでは、Amazon Textract は結果を内部的に保存し、Get API オペレーションによってのみアクセスできます。とOutputConfig有効にすると、出力が送信されるバケットの名前と、結果をダウンロードできる結果のファイルプレフィックスを設定できます。さらに、KMSKeyIDパラメータを使用して、出力を暗号化するためにカスタマー管理キーを指定します。このパラメータを設定しないと、Amazon Textract は、AWS マネージドキーAmazon S3 におけるもの

> **Note**
>
> このパラメータを使用する前に、出力バケットの PutObject 権限があることを確認してください。さらに、の [復号化]、[ReEncrypt]、[generateDataKey]、および [describeKey] 権限があることを確認します。AWS KMS使用することに決めた場合はキーを指定します。

StartDocumentTextDetection オペレーションに対する応答は、ジョブ識別子 (JobId) です。を使用するJobIdAmazon Textract が完了ステータスを Amazon SNS トピックに発行したら、リクエストを追跡し、分析結果を取得します。次に例を示します。

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

同時に開始するジョブが多すぎると、StartDocumentTextDetectionを育てるLimitExceededException同時に実行されるジョブの数が Amazon Textract のサービスの制限を下回るまで、の例外 (HTTP ステータスコード:400)。

アクティビティのバーストにより LimitExceededException 例外が発生した場合は、Amazon SQS キューを使用して受信リクエストを管理することを検討してください。連絡先AWS同時リクエストの平均数を Amazon SQS キューで管理できないと、引き続き受信されている場合のSupportLimitExceededException例外。

Amazon Textract 分析リクエストの完了ステータスの取得

Amazon Textract は、登録されている Amazon SNS トピックに分析完了の通知を送信します。通知には、ジョブ識別子およびオペレーション完了ステータスが JSON 文字列で含まれています。成功したテキスト検出リクエストには、SUCCEEDEDstatus。たとえば、次の結果は、テキスト検出ジョブが正常に処理されたことを示しています。

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

詳細については、「Amazon Textract 結果通知」を参照してください。

Amazon Textract によって Amazon SNS トピックに発行されたステータス情報を取得するには、次のいずれかのオプションを使用します。

- AWS Lambda— をサブスクライブできますAWS LambdaAmazon SNS トピックに書き込む関数。この関数は、Amazon Textract が Amazon SNS トピックにリクエスト完了を通知したときに呼び出されます。Lambda 関数を使用すると、サーバー側のコードでテキスト検出リクエストの結果を処理できます。たとえば、クライアントアプリケーションに情報を返す前に、イメージに注釈を付けたり、検出されたテキストに関するレポートを作成したりする場合があります。

- Amazon SQS— Amazon SNS トピックに Amazon SQS キューをサブスクライブできます。Amazon SQS キューをポーリングすることで、リクエストの完了時に Amazon Textract が発行する完了ステータスを取得できます。詳細については、「複数ページドキュメント内のテキストの検出または分析」を参照してください。Amazon Textract オペレーションをクライアントアプリケーションからのみ呼び出す場合は、Amazon SQS キューを使用します。

> **Important**
>
> Amazon Textract を繰り返し呼び出すことでリクエスト完了ステータスを取得することは推奨されません。Getオペレーション. これは、Amazon Textract がGet実行されるリクエストが多すぎると、操作が実行されます。複数のドキュメントを同時に処理する場合は、Amazon Textract をポーリングして各ジョブのステータスを個別に取得するよりも、1 つの SQS キューの完了通知を監視するほうがシンプルかつ効率的です。

Amazon Textract テキスト検出結果の取得

テキスト検出リクエストの結果を取得するには、Amazon SNS トピックから取得された完了ステータスがであることを最初に確認します。SUCCEEDED。次に GetDocumentTextDetection を呼び出し、StartDocumentTextDetection から返された JobId の値を渡します。リクエストの JSON は次の例のようになります。

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

JobIdテキスト検出操作の識別子です。テキスト検出によって大量のデータが生成されるため、MaxResultsは、1 件で返す結果の最大数を指定します。Getオペレーション. のデフォルト値MaxResultsは1,000です。1,000 より大きい値を指定した場合、1,000 件の結果だけが返されます。オペレーションによってすべての結果が返されない場合、次のページのページ分割トークンが返されます。次の結果ページを取得するには、NextTokenパラメータ。

> **Note**
>
> Amazon Textract は非同期オペレーションの結果を 7 日間保持します。この時間を過ぎると結果を取得することはできません。

-GetDocumentTextDetectionオペレーションの応答の JSON は次のようになります。検出されたページの総数がDocumentMetadata。検出されたテキストはBlocks配列。の詳細については、Blockオブジェクト、「」を参照してください。テキスト検出および文書分析応答オブジェクト。

```
{
    "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
        }
    ]
}
```

非同期オペレーション用の Amazon Textract の設定

以下の手順は、Amazon Simple Notification Service (Amazon SNS) トピックと Amazon Simple Queue Service (Amazon SQS) キューと一緒に使用する Amazon Textract を設定する方法を示しています。

> **Note**
>
> これらの手順を使用してをセットアップする場合複数ページドキュメント内のテキストの検出または分析たとえば、ステップ 3 ~ 6 を実行する必要はありません。この例には、Amazon SNS トピックと Amazon SQS キューを作成して設定するコードが含まれています。

Amazon Textract を設定するには

1. をセットアップするAWSAmazon Textract にアクセスするためのアカウント。詳細については、「ステップ 1: AWS アカウントを設定して IAM ユーザーの作成」を参照してください。

   ユーザーに少なくとも以下のアクセス許可があることを確認します。

   - AmazonTextractFullAccess
   - AmazonS3ReadOnlyAccess
   - AmazonSNSFullAccess
   - AmazonSQSFullAccess
2. 必要な AWS SDK をインストールして設定します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。
3. Amazon SNS トピックを作成する。トピック名の先頭に「」を追加します。AmazonExtract。その Amazon リソースネーム (ARN) をメモします。トピックが、と同じリージョンにあることを確認します。AWSAWS アカウントで使用しているエンドポイント。
4. Amazon SQS 標準キューを作成するを使用してAmazon SQS コンソール。キューの ARN をメモします。
5. ステップ 3 で作成したトピックにキューをサブスクライブします。
6. Amazon SQS キューにメッセージを送信するアクセス許可を Amazon SNS トピックに付与する。

7. IAM サービスロールを作成して、Amazon Textract に Amazon SNS トピックへのアクセス権を付与します。このサービスロールの Amazon リソースネーム (ARN) をメモします。詳細については、「Amazon SNS トピックへの Amazon Textract アクセスを付与する」を参照してください。
8. 以下のインラインポリシーを追加します。をステップ 1 で作成した IAM ユーザーに指定します。

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

   インラインポリシーに名前を付けます。
9. これで、の例を実行できます。複数ページドキュメント内のテキストの検出または分析。

Amazon SNS トピックへの Amazon Textract アクセスを付与する

Amazon Textract では、非同期操作が完了したときに Amazon SNS トピックにメッセージを送信するためのアクセス許可が必要です。IAM サービスロールを使用して、Amazon Textract に Amazon SNS トピックへのアクセス権を付与します。

Amazon SNS トピックを作成するときは、トピック名の先頭に「」を付ける必要があります。**`AmazonTextract`**— たとえば、**`AmazonTextractMyTopicName`**。

1. IAM コンソール (https://console.aws.amazon.com/iam) にサインインします。
2. ナビゲーションペインで [Roles (ロール)] を選択します。
3. [ロールの作成] を選択します。
4. [Select type of trusted entity (信頼されたエンティティのタイプの選択)] で、[AWS サービス] を選択します。
5. を使用する場合このロールを使用するサービスを選択で、Textract。

6. [Next: (次へ:)] を選択します アクセス許可.
7. になっていることを確認します。AmazonTextractServiceRoleポリシーがアタッチされたポリシーのリストに含まれている。リストにポリシーを表示するには、にポリシー名の一部を入力します。フィルタポリシー。
8. [Next: (次へ:)] を選択します タグ
9. タグの作成は必要ないため、次へ: 確認.
10. [確認] セクションの [ロール名] に、ロールの名前を入力します (例: `TextractRole`)。Eclipseロールの説明で、ロールの説明を更新して、ロールの作成。
11. 新しいロールを選択して、ロールの詳細ページを開きます。
12. [概要] で、[ロール ARN] の値をコピーして保存します。
13. [Trust relationships (信頼関係)] を選択します。
14. 選択信頼関係を編集するを選択し、信頼ポリシーが次のようになっていることを確認します。

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

15. [Update Trust Policy] (信頼ポリシーの更新) をクリックします。

複数ページドキュメント内のテキストの検出または分析

この手順では、Amazon Textract 検出オペレーション、Amazon S3 バケットに格納されたドキュメント、Amazon SNS トピック、および Amazon SQS キューを使用して、複数ページのドキュメント内のテキストを検出または分析する方法について説明します。複数ページのドキュメント処理は、非同期オペレーションです。詳細については、「Amazon Textract 非同期オペレーションを呼び出す」を参照してください。

コードで実行する処理のタイプ (テキスト検出、テキスト分析、または経費分析) を選択できます。

処理結果は、次の配列で返されます。the section called “Block”オブジェクト。使用する処理の種類によって異なります。

複数ページのドキュメント内のテキストを検出したり、分析したりするには、次の操作を行います。

1. Amazon SNS トピックと Amazon SQS キューを作成します。
2. トピックをキューにサブスクライブします。
3. キューにメッセージを送信するアクセス許可をトピックに付与します。
4. ドキュメントの処理を開始します。選択した解析タイプに適切な操作を使用します。
   - StartDocumentTextDetectionテキスト検出タスク用。
   - StartDocumentAnalysisテキスト分析タスク用。
   - StartExpenseAnalysis経費分析タスクの場合。
5. 完了ステータスを Amazon SQS キューから取得します。サンプルコードは、ジョブ識別子 () を追跡します。`JobId`) によって返される`Start`オペレーション. 完了ステータスから読み取られた、一致するジョブ識別子の結果だけを取得します。これは、他のアプリケーションが同じキューとトピックを使用している場合、重要です。わかりやすいように、この例では一致しないジョブが削除されます。削除したジョブを Amazon SQS デッドレターキューに追加して詳しく調査することを検討してください。
6. 選択した解析タイプの適切なオペレーションを呼び出して、処理結果を取得して表示します。
   - GetDocumentTextDetectionテキスト検出タスク用。
   - GetDocumentAnalysisテキスト分析タスク用。
   - GetExpenseAnalysis経費分析タスクの場合。
7. Amazon SNS トピックと Amazon SQS キューを削除します。

非同期操作の実行

この手順のコード例は Java、Python、AWS CLI。開始する前に、適切なをインストールします。AWSSDK。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

複数ページのドキュメント内のテキストを検出または分析するには

1. Amazon Textract へのユーザーアクセスを設定し、Amazon SNS への Amazon Textract アクセスを設定します。詳細については、「非同期オペレーション用の Amazon Textract の設定」を参照してください。この手順を完了するには、PDF 形式の複数ページのドキュメントファイル

が必要です。コード例によって Amazon SNS トピックと Amazon SQS キューが作成されて設定されるため、ステップ 3 ~ 6 をスキップします。もし複雑ならtCLI の例では、SQS キューを設定する必要はありません。

2. PDF または TIFF 形式の複数ページのドキュメントファイルを Amazon S3 バケットにアップロードします。(JPEG、PNG、TIFF、または PDF 形式の単一ページのドキュメントも処理できます)。

   手順については、以下を参照してください。Amazon S3 へのオブジェクトのアップロードのAmazon Simple Storage Service ユーザーガイド。

3. 以下を使用します。AWS SDK for Java、SDK for Python (Boto3)、またはAWS CLI複数ページの文書でテキストを検出したり、テキストを分析したりするためのコードです。左`main`関数:

   - の値を置換する`roleArn`で保存した IAM ロール ARN を持つAmazon SNS トピックへの Amazon Textract アクセスを付与する。
   - の値を置換する`bucket`そして`document`に、ステップ 2 で指定したバケット名とドキュメントファイル名を使用します。
   - の値を置換します。`type`の入力パラメータ`ProcessDocument`関数を使用して、実行する処理の種類を指定します。を使用する`ProcessType.DETECTION`テキストを検出します。を使用する`ProcessType.ANALYSIS`テキストを分析します。
   - Python の例では、の値を置き換えます。`region_name`クライアントが操作している地域で。

   向けのAWS CLIたとえば、以下を実行します。

   - 呼び出し時StartDocumentTextDetectionで、の値を置き換えます。`bucket-name`を S3 バケットの名前に置き換えます。`file-name`に、ステップ 2 で指定したファイル名を使用します。置き換えてバケットのリージョンを指定します。`region-name`お住まいの地域の名前を付けて。CLI の例では SQS を使用しないことに注意してください。
   - 呼び出し時GetDocumentTextDetection置換`job-id-number`と`job-id`によって返されるStartDocumentTextDetection。置き換えてバケットのリージョンを指定します。`region-name`お住まいの地域の名前を付けて。

   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

このAWS CLIコマンドは、指定したドキュメント内のテキストの非同期検出を開始します。このメソッドからは、job-idこれにより、検出結果を取得できます。

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

このAWS CLIコマンドを使用すると、Amazon Textract 非同期オペレーションの結果が返されます。job-id。

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

Windows デバイスで CLI にアクセスする場合は、一重引用符ではなく二重引用符を使用し、内部の二重引用符をバックスラッシュ (つまり\ ) で エスケープして、発生する可能性のあるパーサーエラーに対処します。例については、以下を参照してください

```
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

この例では、の値を置き換えます。roleArnで保存した IAM ロール ARN を持つAmazon SNS トピックへの Amazon Textract アクセスを付与する。の値を置換するbucketそしてdocumentは、上記のステップ 2 で指定したバケット名とドキュメントファイル名を使用します。の値を置換するprocessType入力ドキュメントで使用する処理の種類を指定します。最後に、の値を置き換えます。REGIONクライアントが操作している地域で。

```
 // 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. コードを実行します。オペレーションの完了までに時間がかかる場合があります。完了すると、検出または分析されたテキストのブロックのリストが表示されます。

Amazon Textract 結果通知

Amazon Textract は、完了ステータスを含む Amazon Textract 分析リクエストの結果を、Amazon Simple Notification Service (Amazon SNS) トピックに発行します。Amazon SNS トピックから通知を取得するには、Amazon SQS キューまたはAWS Lambdafunction. 詳細については、「Amazon Textract 非同期オペレーションを呼び出す」を参照してください。例については、複数ページドキュメント内のテキストの検出または分析 を参照してください。

結果の JSON 形式は次のとおりです。

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

```
    "S3ObjectName": "String",
    "S3Bucket": "String"
  }
}
```

次の表に、Amazon Textract レスポンス内のさまざまなパラメータを示します。

パラメータ	説明
JobId	Amazon Textract がジョブに割り当てる一意の識別子。これは、から返されたジョブ識別子と一致します。Startなどの操作StartDocumentTextDetection。
[Status] (ステータス)	ジョブのステータス。有効な値は、[成功]、[失敗]、または [エラー] です。
API	入力ドキュメントの分析に使用された Amazon Textract オペレーション。StartDocumentTextDetectionまたはStartDocumentAnalysis。
JobTag	ジョブのユーザー指定の識別子。指定するJobTagへの呼び出しでStartなどの操作StartDocumentTextDetection。
タイムスタンプ	ジョブが終了したことを示す Unix タイムスタンプ。ミリ秒単位で返されます。
ドキュメントロケーション	処理されたドキュメントに関する詳細。ファイル名やファイルが保存されている Amazon S3 バケットなどが含まれます。

スロットルコールとドロップされた接続の処理

Amazon Textract オペレーションは、1 秒あたりの最大トランザクション数 (TPS) を超え、サービスがアプリケーションをスロットルしたり、接続が切断されたりすると、失敗することがあります。たとえば、短期間で Amazon Textract オペレーションへの呼び出しが多すぎると、コールがスロットルされ、`ProvisionedThroughputExceededException`オペレーション応答でエラーが発生しました。Amazon Textract TPS クォータの詳細については、「」を参照してください。Amazon Textract クォータ。

操作を自動的に再試行することで、スロットリングとドロップされた接続を管理できます。再試行回数を指定するには、`Config`Amazon Textract クライアントを作成するときのパラメータです。再試行回数は 5 にすることをお勧めします。-AWSSDK では、指定した回数だけ操作が再試行された後に失敗となり、例外がスローされます。詳細については、AWS でのエラーの再試行とエクスポネンシャルバックオフを参照してください。

> **Note**
>
> 自動再試行は、同期操作と非同期操作の両方で機能します。自動再試行を指定する前に、AWS SDK の最新バージョンがインストールされていることを確認します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

次の例では、複数のドキュメントを処理するときに Amazon Textract オペレーションを自動的に再試行する方法を示しています。

前提条件

- まだ実行していない場合:
  a. を使用して IAM ユーザーを作成または更新する`AmazonTextractFullAccess`そして`AmazonS3ReadOnlyAccess`アクセス許可。詳細については、「ステップ 1: AWS アカウントを設定して IAM ユーザーの作成」を参照してください。
  b. AWS CLI と AWS SDK をインストールして設定します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

オペレーションを自動的に再試行するには

1. S3 バケットに複数のドキュメントイメージをアップロードして、Synchronous の例を実行します。複数ページのドキュメントを S3 バケットにアップロードして実行します。StartDocumentTextDetectionその上で、非同期の例を実行します。

   手順については、以下を参照してください。Amazon S3 へのオブジェクトのアップロードのAmazon Simple Storage Service ユーザーガイド。
2. 次の例では、の使い方をデモンストレーションします。Configパラメータを使用して、操作を自動的に再試行します。同期の例では、DetectDocumentTextオペレーション。非同期の例ではGetDocumentTextDetectionオペレーション.

   Sync Example

   以下の例を使用して、DetectDocumentTextAmazon S3 バケット内のドキュメントに対する操作。Eclipsemainで、の値を変更するにはbucketS3 バケットにアップロードします。の値を変更するにはdocumentsステップ 2 でアップロードしたドキュメントイメージの名前を入力します。

```
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

以下の例を使用して、GetDocumentTextDetection オペレーション を呼び出します。これは、既に呼び出されていることを前提としています。StartDocumentTextDetectionAmazon S3 バケットのドキュメントで、JobId。Eclipsemainで、の値を変更するにはbucketS3 バケットの値roleArnTextract ロールに割り当てられた Arn に指定します。また、の値を変更する必要もあります。documentを Amazon S3 バケット内の複数ページドキュメントの名前に変更します。最後に、の値を置き換えます。region_nameお住まいの地域の名前を入力し、GetResultsの名前を持つ関数を使用します。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()
```

Amazon Textract に関するベストプラクティス

Amazon Textract は、人のように機械学習を使用してドキュメントを読み取ります。ドキュメントからテキスト、表、およびフォームを抽出します。以下のベストプラクティスを使用して、ドキュメントから最良の結果を取得してください。

最適な入力ドキュメントを提供する

次に、より良い結果を得るために入力ドキュメントを最適化するいくつかの方法のリストを示します。

- ドキュメントテキストが Amazon Textract がサポートする言語であることを確認します。現在、Amazon Textract は、英語、スペイン語、ドイツ語、イタリア語、フランス語、ポルトガル語をサポートしています。
- 高品質の画像を提供します。理想的には少なくとも 150 DPI です。
- Amazon Textract がサポートするファイル形式 (PDF、TIFF、JPEG、PNG) にドキュメントがすでに含まれている場合は、Amazon Textract にアップロードする前にドキュメントを変換またはダウンサンプルしないでください。

ドキュメント内の表からテキストを抽出するときに最適な結果を得るには、次のことを確認してください。

- ドキュメント内のテーブルは、ページ上の周囲の要素から視覚的に分離されます。たとえば、表はイメージや複雑なパターンにオーバーレイされません。
- テーブル内のテキストは直立しています。たとえば、テキストは、ページ上の他のテキストに対して相対的に回転されません。

表からテキストを抽出するときに、次のような場合に一貫性のない結果が表示されることがあります。

- 複数の列にまたがる結合された表のセル。
- 同じテーブルの他の部分とは異なるセル、行、または列を含むテーブル。

の使用をお勧めしますテキストの検出回避策として。

信頼スコアの使用

Amazon Textract API オペレーションによって返される信頼スコアと、そのユースケースの感度を考慮する必要があります。信頼スコアは 0～100 の間の数値で、与えられた予測が正しい確率を示します。これにより、結果の使用方法について、情報に基づいた決定を下すことができます。

検出エラー（誤検出）の影響を受けやすいアプリケーションでは、最小信頼スコアのしきい値を適用します。アプリケーションは、そのしきい値を下回る結果を破棄するか、より高いレベルの人間の精査を必要とする状況にフラグを付ける必要があります。

最適なしきい値は、アプリケーションによって異なります。手書きメモを文書化するなどのアーカイブ目的では、50% も低い場合があります。財務上の決定を伴うビジネスプロセスでは、90% 以上のしきい値が必要になる場合があります。

ヒューマンレビューの使用を検討する

また、ワークフローにヒューマンレビューを組み込むことも検討してください。これは、財務上の決定を伴うビジネスプロセスなど、機密性の高いアプリケーションでは特に重要です。

チュートリアル

the section called “Block”Amazon Textract オペレーションから返されるオブジェクトには、次のようなテキスト検出およびテキスト分析操作の結果が含まれます。the section called “AnalyzeDocument”。次の Python チュートリアルでは、Block オブジェクトを使用するさまざまな方法をいくつか示します。たとえば、テーブル情報をカンマ区切り値 (CSV) ファイルにエクスポートすることができます。

チュートリアルでは、すべての結果を返す同期 Amazon Textract オペレーションを使用します。次のような非同期操作を使用する場合the section called “StartDocumentAnalysis”では、返された複数のバッチに対応するようにサンプルコードを変更する必要があります。Blockオブジェクト。非同期操作の例を利用するには、に記載されている指示に従っていることを確認してください。非同期オペレーション用の Amazon Textract の設定。

Amazon Textract の他の使用方法を示す例については、「」を参照してください。追加のコードサンプル。

トピック

- 前提条件
- フォームドキュメントからのキーと値のペアの抽出
- CSV ファイルへのテーブルのエクスポート
- の作成AWS Lambda関数
- 追加のコードサンプル

前提条件

このセクションの例を実行する前に、環境を設定する必要があります。

環境を設定するには

1. を使用して IAM ユーザーを作成または更新するAmazonTextractFullAccessアクセス許可。詳細については、「ステップ 1: AWS アカウントを設定して IAM ユーザーの作成」を参照してください。
2. AWS CLI と AWS SDK をインストールして設定します。詳細については、「ステップ 2: のセットアップAWS CLIそしてAWSSDK」を参照してください。

フォームドキュメントからのキーと値のペアの抽出

次の Python の例は、フォームドキュメント内のキーと値のペアをから抽出する方法を示しています。the section called “Block”マップに格納されているオブジェクト。ブロックオブジェクトは、の呼び出しから返されます。the section called “AnalyzeDocument”。詳細については、「フォームデータ (キーと値のペア)」を参照してください。

次の関数を使用します。

- get_kv_map— 呼び出しAnalyzeDocumentをクリックし、KEY および VALUE ブロックオブジェクトをマップに格納します。
- get_kv_relationshipそしてfind_value_block— マップからキーと値の関係を構築します。

フォームドキュメントからキーと値のペアを抽出するには

1. 環境を設定します。詳細については、「前提条件」を参照してください。
2. 次のコード例を、という名前のファイルに保存します。textract_python_kv_parser.py。

```
import boto3
import sys
import re
import json


def get_kv_map(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)

    # process using image bytes
    client = boto3.client('textract')
    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['FORMS'])

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

```
    # get key and value maps
    key_map = {}
    value_map = {}
    block_map = {}
    for block in blocks:
        block_id = block['Id']
        block_map[block_id] = block
        if block['BlockType'] == "KEY_VALUE_SET":
            if 'KEY' in block['EntityTypes']:
                key_map[block_id] = block
            else:
                value_map[block_id] = block

    return key_map, value_map, block_map


def get_kv_relationship(key_map, value_map, block_map):
    kvs = {}
    for block_id, key_block in key_map.items():
        value_block = find_value_block(key_block, value_map)
        key = get_text(key_block, block_map)
        val = get_text(value_block, block_map)
        kvs[key] = val
    return kvs


def find_value_block(key_block, value_map):
    for relationship in key_block['Relationships']:
        if relationship['Type'] == 'VALUE':
            for value_id in relationship['Ids']:
                value_block = value_map[value_id]
    return value_block


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
```

```
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] == 'SELECTED':
                            text += 'X '


    return text


def print_kvs(kvs):
    for key, value in kvs.items():
        print(key, ":", value)


def search_value(kvs, search_key):
    for key, value in kvs.items():
        if re.search(search_key, key, re.IGNORECASE):
            return value

def main(file_name):

    key_map, value_map, block_map = get_kv_map(file_name)

    # Get Key Value relationship
    kvs = get_kv_relationship(key_map, value_map, block_map)
    print("\n\n== FOUND KEY : VALUE pairs ===\n")
    print_kvs(kvs)

    # Start searching a key value
    while input('\n Do you want to search a value for a key? (enter "n" for exit)
 ') != 'n':
        search_key = input('\n Enter a search key:')
        print('The value is:', search_value(kvs, search_key))

if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. コマンドプロンプトで次のコマンドを入力します。置換*file*分析するドキュメントイメージファイルを使用します。

```
textract_python_kv_parser.py file
```

4. プロンプトが表示されたら、入力ドキュメントにあるキーを入力します。コードがキーを検出すると、キーの値が表示されます。

CSV ファイルへのテーブルのエクスポート

次の Python の例は、ドキュメントのイメージからテーブルをカンマ区切り値 (CSV) ファイルにエクスポートする方法を示しています。

同期ドキュメント分析の例では、の呼び出しからテーブル情報を収集します。the section called “AnalyzeDocument”。非同期ドキュメント分析の例では、the section called “StartDocumentAnalysis”次に、結果を取得します。the section called “GetDocumentAnalysis”なのでBlockオブジェクト。

テーブル情報は、次のように返されます。the section called “Block”への呼び出しからのオブジェクトthe section called “AnalyzeDocument”。詳細については、「テーブル」を参照してください。-Blockオブジェクトは、テーブルデータを CSV ファイルにエクスポートするために使用されるマップ構造に格納されます。

Synchronous

この例では、次の関数を使用します。

- get_table_csv_results— 呼び出しAnalyzeDocumentをクリックし、ドキュメント内で検出されたテーブルのマップを作成します。検出されたすべてのテーブルの CSV 表現を作成します。
- generate_table_csv— 個々のテーブルの CSV ファイルを生成します。
- get_rows_columns_map— マップから行と列を取得します。
- get_text-セルからテキストを取得します。

テーブルを CSV ファイルにエクスポートするには

1. 環境を設定します。詳細については、「前提条件」を参照してください。
2. 次のコード例を、という名前のファイルに保存します。textract_python_table_parser.py。

```
import webbrowser, os
import json
import boto3
import io
```

```
from io import BytesIO
import sys
from pprint import pprint


def get_rows_columns_map(table_result, blocks_map):
    rows = {}
    for relationship in table_result['Relationships']:
        if relationship['Type'] == 'CHILD':
            for child_id in relationship['Ids']:
                cell = blocks_map[child_id]
                if cell['BlockType'] == 'CELL':
                    row_index = cell['RowIndex']
                    col_index = cell['ColumnIndex']
                    if row_index not in rows:
                        # create new row
                        rows[row_index] = {}

                    # get the text value
                    rows[row_index][col_index] = get_text(cell, blocks_map)
    return rows


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] =='SELECTED':
                            text +=  'X '
    return text


def get_table_csv_results(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)
```

```
    # process using image bytes
    # get the results
    client = boto3.client('textract')

    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['TABLES'])

    # Get the text blocks
    blocks=response['Blocks']
    pprint(blocks)

    blocks_map = {}
    table_blocks = []
    for block in blocks:
        blocks_map[block['Id']] = block
        if block['BlockType'] == "TABLE":
            table_blocks.append(block)

    if len(table_blocks) <= 0:
        return "<b> NO Table FOUND </b>"

    csv = ''
    for index, table in enumerate(table_blocks):
        csv += generate_table_csv(table, blocks_map, index +1)
        csv += '\n\n'

    return csv

def generate_table_csv(table_result, blocks_map, table_index):
    rows = get_rows_columns_map(table_result, blocks_map)

    table_id = 'Table_' + str(table_index)

    # get cells.
    csv = 'Table: {0}\n\n'.format(table_id)

    for row_index, cols in rows.items():

        for col_index, text in cols.items():
            csv += '{}'.format(text) + ","
        csv += '\n'

    csv += '\n\n\n'
```

```
    return csv

def main(file_name):
    table_csv = get_table_csv_results(file_name)

    output_file = 'output.csv'

    # replace content
    with open(output_file, "wt") as fout:
        fout.write(table_csv)

    # show the results
    print('CSV OUTPUT FILE: ', output_file)


if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. コマンドプロンプトで次のコマンドを入力します。置換fileに、分析するドキュメントイメージファイルの名前を指定します。

```
python textract_python_table_parser.py file
```

この例を実行すると、CSV 出力はという名前のファイルに保存されます。output.csv。

Asynchronous

この例では、2 つの異なるスクリプトを使用して make を使用します。最初のスクリプトは、ドキュメントを非同期的に分析するプロセスを開始します。StartDocumentAnalysisを取得し、Blockによって返された情報GetDocumentAnalysis。2 番目のスクリプトは、返されたBlock各ページの情報、データをテーブルとして書式設定し、テーブルを CSV ファイルに保存します。

テーブルを CSV ファイルにエクスポートするには

1. 環境を設定します。詳細については、「前提条件」を参照してください。
2. 「参照」に記載されている指示に従っていることを確認してください。非同期オペレーション用の Amazon Textract の設定。このページに記載されているプロセスでは、非同期ジョブの完了ステータスに関するメッセージを送受信できます。

3. 次のコード例では、の値を置き換えます。roleArnArn をステップ 2 で作成したロールに割り当てます。の値を置き換えるbucketの部分は、お客様のドキュメントを含む S3 バケットの名前を使用します。の値を置き換えるdocumentS3 バケット内のドキュメントの名前を使用した。の値を置き換えるregion_nameバケットのリージョンの名前を指定します。

   次のコード例を、という名前のファイルに保存します。start_doc_analysis_for_table_extraction.py。。

```
import boto3
import time

class DocumentProcessor:

    jobId = ''
    region_name = ''

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

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

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

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

    def ProcessDocument(self):

        jobFound = False

        response =
 self.textract.start_document_analysis(DocumentLocation={'S3Object': {'Bucket':
 self.bucket, 'Name': self.document}},
```

```
                FeatureTypes=["TABLES", "FORMS"],
 NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})
        print('Processing type: Analysis')

        print('Start Job Id: ' + response['JobId'])

        print('Done!')

    def CreateTopicandQueue(self):

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

        # Create SNS topic
        snsTopicName = "AmazonTextractTopic" + millis

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

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

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

        sqsQueueArn = attribs['QueueArn']

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

        # Authorize SNS to write SQS queue
        policy = """{{
      "Version":"2012-10-17",
      "Statement":[
        {{
          "Sid":"MyPolicy",
          "Effect":"Allow",
          "Principal" : {{"AWS" : "*"}},
          "Action":"SQS:SendMessage",
```

```
          "Resource": "{}",
          "Condition":{{
            "ArnEquals":{{
              "aws:SourceArn": "{}"
            }}
          }}
        }}
      ]
    }}""".format(sqsQueueArn, self.snsTopicArn)

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

def main():
    roleArn = 'role-arn'
    bucket = 'bucket-name'
    document = 'document-name'
    region_name = 'region-name'

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

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

4. コードを実行します。コードによって JobId が出力されます。この jobId をコピーします。
5. ジョブの処理が完了するのを待ち、ジョブが終了したら、次のコードをという名前のファイルにコピーします。get_doc_analysis_for_table_extraction.py。の値を置き換えるjobId先ほどコピーしたJob ID を使用します。の値を置き換えるregion_nameTextract ロールに関連付けられたリージョンの名前を指定します。の値を置き換えるfile_name出力の CSV を指定した名前を使用します。

```
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. コードを実行します。

   結果を取得したら、関連する SNS および SQS リソースを必ず削除してください。そうしないと、それらのリソースに対して料金が発生する可能性があります。

の作成AWS Lambda関数

Amazon Textract API オペレーションは、AWS Lambdafunction. 次の手順では、Python で呼び出す Lambda 関数を作成する方法を示しています。the section called “DetectDocumentText”。これは、リストを返しますthe section called “Block”オブジェクト。この例を実行するには、PNG または JPEG 形式のドキュメントを含む Amazon S3 バケットが必要です。関数を作成するには、コンソールを使用します。

Lambda 関数を使用してドキュメントを大規模に処理する例については、を参照してください。Amazon Textract による大規模なドキュメント処理。

Lambda 関数から detectDocumentText オペレーションを呼び出すには、次の手順を実行します。

ステップ 1: Lambda デプロイパッケージの作成

1. コマンドウィンドウを開きます。
2. 次のコマンドを入力して、最新バージョンのAWSSDK。

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

ステップ 2: Lambda 関数の作成

1. AWS マネジメントコンソール にサインインして AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. [Create function] (関数の作成) を選択します。
3. 次を指定します。

   - Author from scratch を選択します。
   - [関数名] に名前を入力します。
   - を使用する場合ランタイムで、Python 3.7またはPython 3.6。
   - を使用する場合実行ロールの選択または作成で、基本的な Lambda アクセス権限で新しいロールを作成する。
4. 選択関数の作成Lambda 関数を作成します。
5. IAM コンソール(https://console.aws.amazon.com/iam/)を開きます。
6. ナビゲーションペインで [ロール] を選択します。
7. [Resources] リストから、Lambda が作成した IAM ロールを選択します。ロール名は Lambda 関数の名前から始まります。
8. [アクセス許可タブを選択し、ポリシーのアタッチ。
9. AmazonTextractFullAccess sポリシーと Amazons3ReadOnlyAccess ポリシーを選択します。
10. Selectポリシーのアタッチ。

詳細については、「」を参照してください。コンソールで Lambda 関数の作成

ステップ 3: レイヤーの作成と追加

1. https://console.aws.amazon.com/lambda/ で、AWS Lambda コンソールを開きます。
2. ナビゲーションペインで [Layers] を選択します。
3. [Create layer] (レイヤーの作成) を選択します。
4. を使用する場合名前[] に、名前を入力します。
5. [説明] に説明を入力します。
6. を使用する場合コード エントリ タイプで、.zip ファイルのアップロードを選択し、アップロード。
7. ダイアログボックスで、で作成した zip ファイル (boto3-layer.zip) を選択します。ステップ 1: Lambda デプロイパッケージの作成。

8. を使用する場合対応ランタイムで、で選択したランタイムのバージョンを選択します。ステップ 2: Lambda 関数の作成。
9. 選択作成をクリックして、レイヤーを作成します。
10. ナビゲーションペインメニューアイコンを選択します。
11. ナビゲーションペインで、[Functions] (関数) を選択します。
12. [Resource] リストで、[] で作成した関数を選択します。ステップ 2: Lambda 関数の作成。
13. 選択設定そして、デザイナー] セクションで、[] を選択しますレイヤー(Lambda 関数名の下)。
14. 左レイヤー] セクションで、[] を選択しますレイヤーを追加する。
15. 選択ランタイム互換レイヤーのリストから選択。
16. Eclipse対応レイヤーで、[] を選択します。名前そしてVersionステップ 3 で作成したレイヤーのうち、
17. [Add] (追加) をクリックします。

ステップ 4: 関数に Python コードを追加する

1. Eclipseデザイナー[] で、関数を選択します。
2. 関数コードエディタで、次の内容をファイルに追加します。lambda_function.py。の値を変更するbucketそしてdocumentバケットとドキュメントに移動します。

```
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. 選択保存Lambda 関数を保存します。

ステップ 5: Lambda をテストする

1. SelectTest。
2. [] の値を入力します。イベント名。
3. [Create] (作成) を選択します。
4. 出力、リストthe section called “Block”オブジェクトは、[実行結果] ペインに表示されます。

そのファイルにAWS Lambda関数がタイムアウトエラーを返します。Amazon Textract API オペレーション呼び出しが原因である可能性があります。のタイムアウト期間の延長については、AWS Lambda関数、「」を参照してください。AWS Lambda 関数の設定。

コードから Lambda 関数を呼び出す方法については、「」を参照してください。を呼び出しますAWS Lambda関数。

追加のコードサンプル

次の表に Amazon Textract のその他のコード例へのリンクを示します。

例	説明
Amazon Textract コードサンプル	Amazon Textract を使用できるさまざまな方法を示します。
Amazon Textract による大規模なドキュメント処理	ドキュメントを大規模に処理するサーバーレスリファレンスアーキテクチャを示します。
Amazon Textract パーサー	の解析方法を示します。the section called “Block”Amazon Textract オペレーションによって返されるオブジェクト。
Amazon Textract ドキュメントのコード例	このガイドで使用するコード例。

例	説明
テクストラクター	Amazon Textract の出力を複数の形式に変換する方法を説明します。
Amazon Textract で検索可能な PDF ドキュメントを生成する	JPG/PNG 形式のイメージやスキャンした PDF ドキュメントなど、さまざまな種類の入力ドキュメントから検索可能な PDF ドキュメントを作成する方法を説明します。

Amazon Textract のコード例

次のコード例は、Amazon Textract をAWSソフトウェア開発キット (SDK)。

例は、次のカテゴリに分類されます。

アクション

個々のサービス関数の呼び出し方法を示すコードの抜粋。

クロスサービスの例

複数のアプリケーションで動作するサンプルアプリケーションAWSのサービス。

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- Amazon Textract のアクション
  - Amazon Textract とAWSSDK
  - Amazon Textract とAWSSDK
  - を使用して Amazon Textract ドキュメント分析ジョブに関するデータを取得するAWSSDK
  - Amazon Textract とAWSSDK
  - Amazon Textract とAWSSDK
- Amazon Textract のクロスサービスの例
  - Amazon Textract エクスプローラーアプリケーションを作成する
  - 画像から抽出されたテキスト内のエンティティを検出するには、AWSSDK

Amazon Textract のアクション

次のコード例は、で個々の Amazon Textract アクションを実行する方法を示しています。AWSSDK。これらの抜粋は Amazon Textract API を呼び出し、単独で実行することを意図していません。それぞれの例には、GitHub へのリンクがあり、コンテキストでコードを設定および実行する方法についての説明です。

次の例には、最も一般的に使用されるアクションだけが含まれています。詳細なリストについては、「」を参照してください。Amazon Textract API リファレンス。

例

- Amazon Textract とAWSSDK
- Amazon Textract とAWSSDK
- を使用して Amazon Textract ドキュメント分析ジョブに関するデータを取得するAWSSDK
- Amazon Textract とAWSSDK
- Amazon Textract とAWSSDK

Amazon Textract とAWSSDK

次のコード例は、Amazon Textract を使用してドキュメントを分析する方法を示しています。

Java

SDK for Java 2.x

```
   public static void analyzeDoc(TextractClient textractClient, String
sourceDoc) {

       try {
           InputStream sourceStream = new FileInputStream(new File(sourceDoc));
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           // Get the input Document object as bytes
           Document myDoc = Document.builder()
                   .bytes(sourceBytes)
                   .build();

           List<FeatureType> featureTypes = new ArrayList<FeatureType>();
           featureTypes.add(FeatureType.FORMS);
           featureTypes.add(FeatureType.TABLES);

           AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
                   .featureTypes(featureTypes)
                   .document(myDoc)
                   .build();
```

```
            AnalyzeDocumentResponse analyzeDocument =
 textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is "
 +block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 手順とその他のコードについては GitHub でご確認いただけます。
- API の詳細については、「」を参照してください。AnalyzeDocumentにAWS SDK for Java 2.xAPI リファレンス。

Python

SDK for Python (Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def analyze_file(
```

```
            self, feature_types, *, document_file_name=None,
document_bytes=None):
        """
        Detects text and additional elements, such as forms or tables, in a local
image
        file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to
detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={'Bytes': document_bytes}, FeatureTypes=feature_types)
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- 手順とその他のコードについては GitHub でご確認いただけます。
- API の詳細については、「」を参照してください。AnalyzeDocumentにAWSSDK for Python (Boto3) API リファレンス。

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Textract とAWSSDK

次のコード例は、Amazon Textract を使用してドキュメント内のテキストを検出する方法を示しています。

Java

SDK for Java 2.x

入力ドキュメントからテキストを検出します。

```
   public static void detectDocText(TextractClient textractClient,String
sourceDoc) {

       try {

           InputStream sourceStream = new FileInputStream(new File(sourceDoc));
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           // Get the input Document object as bytes
           Document myDoc = Document.builder()
                   .bytes(sourceBytes)
                   .build();

           DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                   .document(myDoc)
                   .build();

           // Invoke the Detect operation
           DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);

           List<Block> docInfo = textResponse.blocks();

           Iterator<Block> blockIterator = docInfo.iterator();

           while(blockIterator.hasNext()) {
               Block block = blockIterator.next();
               System.out.println("The block type is "
+block.blockType().toString());
           }
```

```
            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is "
+documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Amazon S3 バケット内のドキュメントからテキストを検出します。

```
    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);
        }
    }
```

- 手順とその他のコードについては「GitHub」でご確認いただけます。
- API の詳細については、「」を参照してください。DetectDocumentTextにAWS SDK for Java 2.xAPI リファレンス。

Python

SDK for Python (Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.
```

```
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={'Bytes': document_bytes})
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- 手順とその他のコードについては GitHub でご確認いただけます。
- API の詳細については、「」を参照してください。DetectDocumentTextにAWSSDK for Python (Boto3) API リファレンス。

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

を使用して Amazon Textract ドキュメント分析ジョブに関するデータを取得するAWSSDK

次のコード例は、Amazon Textract ドキュメント分析ジョブに関するデータを取得する方法を示しています。

Python

SDK for Python (Boto3)

```
class TextractWrapper:
```

```
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
            response = self.textract_client.get_document_analysis(
                JobId=job_id)
            job_status = response['JobStatus']
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
            return response
```

- 手順とその他のコードについては GitHub でご確認いただけます。
- API の詳細については、「」を参照してください。GetDocumentAnalysisにAWSSDK for Python (Boto3) API リファレンス。

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Textract とAWSSDK

次のコード例は、Amazon Textract を使用してドキュメントの非同期分析を開始する方法を示しています。

Java

SDK for Java 2.x

```
   public static String startDocAnalysisS3 (TextractClient textractClient,
String bucketName, String docName) {

       try {

           List<FeatureType> myList = new ArrayList<FeatureType>();
           myList.add(FeatureType.TABLES);
           myList.add(FeatureType.FORMS);

           S3Object s3Object = S3Object.builder()
                   .bucket(bucketName)
                   .name(docName)
                   .build();

           DocumentLocation location = DocumentLocation.builder()
                   .s3Object(s3Object)
                   .build();

           StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                   .documentLocation(location)
                   .featureTypes(myList)
                   .build();

           StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

           // Get the job ID
           String jobId = response.jobId();
           return jobId;

       } catch (TextractException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "" ;
    }

    private static String getJobResults(TextractClient textractClient, String
jobId) {

        boolean finished = false;
        int index = 0 ;
        String status = "" ;

       try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                    .jobId(jobId)
                    .maxResults(1000)
                    .build();

            GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++ ;
        }
        return status;

       } catch( InterruptedException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
       return "";
    }
```

- 手順とその他のコードについては GitHub でご確認いただけます。

- API の詳細については、「」を参照してください。StartDocumentAnalysisにAWS SDK for Java 2.xAPI リファレンス。

Python

SDK for Python (Boto3)

非同期ジョブを開始し、ドキュメントを分析します。

```
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
```

- 手順とその他のコードについては「GitHub」でご確認いただけます。
- API の詳細については、「」を参照してください。StartDocumentAnalysisにAWSSDK for Python (Boto3) API リファレンス。

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Textract とAWSSDK

次のコード例は、Amazon Textract を使用してドキュメントで非同期テキスト検出を開始する方法を示しています。

Python

SDK for Python (Boto3)

ドキュメント内のテキストを検出する非同期ジョブを開始します。

```
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
```

- 手順とその他のコードについては「GitHub」でご確認いただけます。
- API の詳細については、「」を参照してくださ
  い。StartDocumentTextDetectionにAWSSDK for Python (Boto3) API リファレンス。

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Textract のクロスサービスの例

次のサンプルアプリケーションでは、AWSAmazon Textract を他のものと組み合わせる SDKAWS のサービス。各例には GitHub へのリンクが含まれています。ここでは、アプリケーションのセットアップおよび実行の手順を参照してください。

例

- Amazon Textract エクスプローラーアプリケーションを作成する
- 画像から抽出されたテキスト内のエンティティを検出するには、AWSSDK

Amazon Textract エクスプローラーアプリケーションを作成する

次のコード例は、インタラクティブアプリケーションで Amazon Textract の出力を調べる方法を示しています。

JavaScript

SDK for JavaScript V3

AWS SDK for JavaScript を使用して、Amazon Textract を使用してドキュメントイメージからデータを抽出し、インタラクティブなウェブページに表示する React アプリケーションを構築する方法を示します。この例はウェブブラウザで実行され、認証情報に認証された Amazon Cognito ID が必要です。Amazon Simple Storage Service (Amazon S3) をストレージに使用し、通知のために、Amazon Simple Notification Service (Amazon SNS) トピックにサブスクライブした Amazon Simple Queue Service (Amazon SQS) キューをポーリングします。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Cognito ID
- [Amazon S3]
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

Amazon Textract で AWS SDK for Python (Boto3) を使用して、ドキュメントイメージ内のテキスト、フォーム、および表要素を検出する方法を示します。入力イメージと Amazon Textract 出力は、検出された要素を探索できる Tkinter アプリケーションに表示されます。

- Amazon Textract にドキュメントイメージを送信し、検出された要素の出力を調べます。
- Amazon Textract に直接イメージを送信するか、Amazon Simple Storage Service (Amazon S3) バケットを通じてイメージを送信します。
- 非同期 API を使用して、ジョブの完了時に Amazon Simple Notification Service (Amazon SNS) トピックに通知を発行するジョブを開始します。
- Amazon Simple Queue Service (Amazon SQS) キューにジョブ完了メッセージについてポーリングし、結果を表示します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- [Amazon S3]
- Amazon SNS
- Amazon SQS
- Amazon Textract

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

画像から抽出されたテキスト内のエンティティを検出するには、AWSSDK

次のコード例は、Amazon Comprehend を使用して、Amazon S3 に格納されているイメージから Amazon Textract によって抽出されたテキスト内のエンティティを検出する方法を示しています。

Python

SDK for Python (Boto3)

の使用方法を示します。AWS SDK for Python (Boto3)Jupyter ノートブックで、画像から抽出されたテキスト内のエンティティを検出します。この例では、Amazon Textract を使用して、Amazon Simple Storage Service (Amazon S3) およびAmazon Comprehend に格納されたイメージからテキストを抽出して、抽出されたテキスト内のエンティティを検出します。

この例は Jupyter ノートブックであり、ノートブックをホストできる環境で実行する必要があります。Amazon SageMaker を使用してサンプルを実行する方法については、「」を参照してください。TextTractとComprehendNotebook.ipynb。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Comprehend
- [Amazon S3]

- Amazon Textract

の詳細なリストについては、「」を参照してください。AWSSDK 開発者ガイドとコード例については、「」を参照してください。Amazon Textract をAWSSDK。このトピックには、開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Augmented AI を使用して Amazon Textract 出力にヒューマンレビューを追加する

Amazon Augmented AI (Amazon A2I) は、人によるML分析のレビューのためのワークフローを簡単に構築できる、Machine Learning (ML) サービスです。

Amazon Textract は Amazon A2I と統合されています。これを使用して、信頼スコアが低い文書分析結果を人間のレビュー担当者にルーティングできます。

Amazon Textract を使うことができますAnalyzeDocumentフォームと Amazon A2I コンソールからデータを抽出する API。Amazon A2I がレビュー担当者に予測をルーティングする条件を指定できます。重要なフォームキーの信頼度しきい値に基づいて条件を設定します。たとえば、キーの場合、ドキュメントを人間のレビュー担当者に送信できます。名前またはそれに関連する値ジェーン・ドー低い信頼度で検出された。

トピック

- Amazon A2I のコアコンセプト
- Amazon A2I の使用を開始する

Amazon A2I のコアコンセプト

Amazon A2I のコア概念を理解するには、次の用語を確認してください。

ヒューマンレビューのアクティベーション条件

Amazon A2I を使用できます。アクティベーション条件ドキュメントがレビューのために人間に送信されるタイミングと、作業者がレビューを依頼するフォームコンテンツを指定します。

たとえば、重要なキーが信頼性の低い状態で検出されたときに Amazon Textract フォームを Amazon A2I にルーティングするようにアクティベーション条件を設定できます。電話番号。この例では、人間のレビュー担当者は、電話番号Amazon Textract によって検出された関連付けられたフィールドと値。

次のアクティベーション条件を使用して、フォームがレビューのために人間に送信されるタイミングを指定できます。

- フォームキーの信頼性スコアに基づいて、特定のフォームキーのヒューマンレビューをトリガーします。人間のレビュー担当者は、これらのフォームキーと関連する値を確認するように求められます。
- 特定のフォームキーが見つからない場合、ヒューマンレビューをトリガーします。人間のレビュー担当者は、これらのフォームキーと関連する値を特定するよう求められます。
- Amazon Textract によって識別されるすべてのフォームキーについて、指定された範囲の信頼性スコアでヒューマンレビューをトリガーします。
- ランダムにフォームのサンプルを人間に送信してレビューします。人間のレビュー担当者は、Amazon Textract によって検出されたすべてのフォームのキーと値のレビューを求められます。

アクティベーション条件がフォームキーの信頼性スコアに依存する場合は、次の 2 種類の予測信頼度しきい値を使用してヒューマンレビューをトリガーできます。

- 信頼度識別— フォーム内で検出されたキーと値のペアの信頼スコア。
- 資格信頼度数— フォームのキーと値のペアに含まれるテキストの信頼スコア。

信頼度しきい値を指定すると、Amazon A2I はしきい値内に収まる予測のみを人間のレビュー担当者にルーティングします。これらのしきい値はいつでも調整して、精度と費用対効果の適切なバランスを実現できます。これは、監査を実装して予測精度を定期的に監視するのに役立ちます。

> **Note**
>
> Amazon A2I カスタムタスクタイプを使用して、ドキュメントをレビュー用に人間に送信する条件をさらにカスタマイズできます。このタスクタイプでは、アプリケーションで直接ヒューマンレビューの条件を指定します。詳細については、を参照してください。Amazon Augmented AI をカスタムタスクタイプでを使用する」(Amazon SageMaker 開発者ガイド)を参照してください。

ヒューマンレビューワークフロー (フロー定義)

ヒューマンレビューワークフローの作成に使用されるリソースを指定したり、アクティベーション条件を指定したりするには、ヒューマンレビューワークフローを使用します。これは、フロー定義ともいいます。

指定するリソースは次のとおりです。

- Amazon A2I API オペレーションを呼び出す権限を持つ IAM ロール
- ヒューマンレビューの出力を保存する Amazon S3 バケット。
- あなたの人間作業チーム
- あるワーカータスクテンプレート作業者がレビュータスクを完了するのに役立つ手順と例が含まれています

また、ヒューマンレビューワークフローを使用して、アクティベーション条件を指定します。詳細については、「ヒューマンレビューのアクティベーション条件」を参照してください。

1 つのヒューマンレビューワークフローを使用して、複数のを作成できます。ヒューマンループ。

SageMaker コンソールまたは SageMaker API を使用して、ヒューマンレビューワークフローを作成できます。詳細については、ヒューマンレビューワークフローを作成する を参照してください。

ワーカータスクテンプレート

ヒューマンレビュータスクに使用されるワーカー UI を作成するには、ワーカータスクテンプレートを使用します。

ワーカー UI には、ドキュメントと作業者の指示が表示されます。また、ワーカーがタスクを完了するために使用するツールも用意されています。

SageMaker コンソールを使用して、ヒューマンレビューワークフローを作成するときにワーカータスクテンプレートを設定できます。詳細については、ヒューマンレビューワークフローを作成する を参照してください。

作業チーム

ある作業チームは、ヒューマンレビュータスクを送信する人間の作業者のグループです。

ヒューマンレビューワークフローを作成するときは、1 つのワークチームを指定します。

Amazon A2I では、組織内のレビューアーのプールを使用できます。また、Amazon Mechanical Turk を通じてすでに機械学習タスクを実行している 50,000 人以上の独立した請負業者で構成される従業員にアクセスすることもできます。もう 1 つの選択肢は、品質とセキュリティ手順の遵守について、AWS によって事前スクリーニングされたワークフォースベンダーを使用することです。

Amazon A2I は、レビュー担当者がレビュータスクを完了するために必要なすべての手順とツールで構成されるウェブインターフェイスを提供します。

ワークフォースのタイプ (プライベート、ベンダー、Mechanical Turk) ごとに、複数のワークチームを作成できます。各作業チームは、複数のヒューマンレビューワークフローで使用できます。ワークフォースとワークチームの作成方法については、「」を参照してください。ワークフォースの作成と管理」(Amazon SageMaker 開発者ガイド) を参照してください。

> **Important**
>
> [] をクリックします。ここに現時点でAmazon Augmented AI をカバーするコンプライアンスプログラムをご覧ください。Amazon Augmented AI を他の AWS のサービス (Amazon Rekognition や Amazon Textract など) と組み合わせて使用する場合、Amazon Augmented AI は、他のサービスと同じコンプライアンスプログラムの対象外であることに注意してください。サービスの顧客データの処理や保存方法、データ環境のコンプライアンスへの影響について、Amazon Augmented AI の使用方法について、お客様には責任があります。ワークロードの目的や目標について AWS アカウントチームと話し合う必要があります。サービスが、提案されたユースケースやアーキテクチャに適しているかどうかを評価するのに役立ちます。
>
> 現在、Amazon Augmented AI は、パブリックおよびベンダーのワークフォースケースを除いて PCI に準拠しています。
>
> Amazon Augmented AI HIPAA コンプライアンスに関する情報については、ここに。

ヒューマンループ

ヒューマンループを使用して、ヒューマンレビュータスクを作成します。

ヒューマンレビュータスクをヒューマンワーカーチームの作業者に割り当てます。ワーカーは、アクティベーション条件で指定した入力ドキュメントで Amazon Textract によって検出されたキーと値のペアを確認するように求められます。

たとえば、ある絵がリンゴを含むという信頼度が 50 ~ 60% の間にあるとします。これは、ヒューマンレビューの信頼度しきい値内に収まり、作業者に送信されます。作業者は、ドキュメントにリンゴがあるかどうかをチェックし、リンゴを含むか含まれていないとマークします。その後、Amazon A2I はドキュメントをワークフローに送り返します。

Amazon Textract って呼んだらAnalyzeDocumentヒューマンレビューワークフロー (フロー定義) とヒューマンループ名を指定すると、ヒューマンレビューワークフローで指定されたアクティベーション条件が満たされると、ヒューマンレビュータスクが作成されます。これらのタスクは、ヒューマンレビューワークフローで指定したリソースを使用して作成されます。

Amazon A2I の使用を開始する

次の手順は、Amazon A2I を Amazon Textract 単一ページのドキュメント分析タスクに統合するのに役立ちます。次の作業を行います。

1. Amazon A2I コンソール (新規ユーザーに推奨) または Amazon A2I API を使用して、ヒューマンレビューワークフローを作成します。
2. フォームを分析し、必要に応じてヒューマンレビューを含めるには、`AnalyzeDocument`オペレーションを実行し、ヒューマンレビューワークフローの Amazon リソースネーム (ARN) を指定します。応答は、ヒューマンレビューが必要かどうかを示します。
3. Amazon A2I コンソールと API を使用して、人間のループを監視します。
4. 結果が送信される Amazon S3 バケットでヒューマンレビューの結果を確認します。

SageMaker ノートブックインスタンスをセットアップし、サンプルノートブックを使用するには、を参照してください。Amazon Textract とAugmented AI を使用したエンドツーエンドのデモのAmazon SageMaker 開発者ガイド

> **Note**
>
> このセクションでは、Amazon A2I、Amazon Textract タスクタイプのヒューマンレビューワークフローを作成する方法について説明します。Amazon A2I と Amazon Textract の統合をさらにカスタマイズするには、Amazon A2I カスタムタスクタイプを使用できます。このオプションでは、カスタムワーカータスクテンプレートを提供し、ドキュメントがアプリケーションで直接ヒューマンレビューのために送信される条件を指定します。詳細については、 を参照してください。Amazon Augmented AI をカスタムタスクタイプでを使用するのAmazon SageMaker 開発者ガイド。

トピック

- ヒューマンレビューワークフローを作成する
- ドキュメントを分析する
- ヒューマンループを監視する
- 出力データとワーカーメトリクスの表示

ヒューマンレビューワークフローを作成する

Amazon A2I コンソール (新規ユーザーに推奨) または Amazon A2I を使用して、ヒューマンレビューワークフローを作成できます。`CreateFlowDefinition`オペレーション.

トピック

- ヒューマンレビューワークフローを作成する (コンソール)
- ヒューマンレビューワークフローを作成する (API)

ヒューマンレビューワークフローを作成する (コンソール)

Amazon S3 の独自のドキュメントを使用してこの例を完了するか、ダウンロードすることができます。このサンプルドキュメントAmazon S3 バケットに配置します。

S3 バケットが同じであることを確認してください。AWSAmazon Textract を使用しているリージョン。バケットを作成するには、「」を参照してください。バケットの作成のAmazon Simple Storage Service Console。

> **Note**
>
> Amazon A2I コンソールは SageMaker コンソールに埋め込まれています。コンソールを使用するには、SageMaker コンソールへのアクセス権限と、作業チームを作成する権限が必要です。開始するには、AmazonSageMakerFullAccessIAM 管理ポリシー。SageMaker でほとんどのアクションを実行するために必要なアクセス許可がすべて含まれています。詳細については、「」を参照してください。Amazon SageMaker の Identity and Access Management emのAmazon SageMaker 開発者ガイド。

トピック

- ステップ 1: 作業チームの作成 (コンソール)
- ステップ 2: ヒューマンレビューワークフローを作成する (コンソール)

ステップ 1: 作業チームの作成 (コンソール)

まず、Amazon A2I コンソールで作業チームを作成し、ワーカーポータルでヒューマンレビュータスクをプレビューできるように、自分をワーカーとして追加し、作業チームのメンバーは、自分に割り当てられたさまざまなタスクとドキュメントを表示できます。

ワーカーの E メールを使用してプライベートワークフォースを作成するには (コンソール)

1. で SageMaker コンソールを開きます。https://console.aws.amazon.com/sagemaker/。
2. ナビゲーションペインの []Ground Truthで、ワーフォースのラベル付け。
3. [Private (プライベート)]、[Create private team (プライベートチームの作成)] の順に選択します。
4. [Invite new workers by email (E メールで新しいワーカーを招待する)] を選択します。
5. この例では、作業者ポータルをプレビューできるようにする他のユーザーの電子メールアドレスと、電子メールアドレスを入力します。カンマで区切り、最大 50 のメールアドレスを含めたリストを貼り付けるか、入力します。電子メールアドレスボックスに移動するとそのように表示されます。
6. 1 つの組織名と連絡先の E メールを入力します。
7. [Create private team (プライベートチームを作成)] を選択します。

プライベートワークチームに自分を追加すると、次のメールが届きます。no-reply@verificationemail.comログイン情報付き。このメールのリンクを使用して、パスワードをリセットし、ワーカーポータルにログインします。これは、電話をかけた後にヒューマンレビュータスクが表示される場所です。AnalyzeDocument。

ステップ 2: ヒューマンレビューワークフローを作成する (コンソール)

このステップでは、Amazon Textract ヒューマンレビューワークフローを作成します。

ヒューマンレビューワークフローを作成するには (コンソール)

1. Amazon A2I コンソールをhttps://console.aws.amazon.com/a2iにアクセスするにはヒューマンレビューワークフローページで.
2. 選択ヒューマンレビューワークフローの作成。
3. を使用する場合名前で、ワークフロー名を入力します。
4. を使用する場合S3 バケットで、Amazon A2I でヒューマンレビュータスクの結果を保存するバケットを選択します。バケットを選択しない場合は、バケットの名前を入力するように変更します。
5. []IAM ロールで、新規ロールの作成。タイトルが付いたウィンドウが表示されます。IAM ロールを作成する。このウィンドウを使用して、このロールにアクセスする Amazon S3 バケットを指定します。選択しない場合S3 バケットで、ステップ 4 で指定した出力バケットと、入力ドキュメントを含むバケットを指定します。

6. を使用する場合タスクタイプで、Textract-キーと値のペアの抽出。
7. EclipseAmazon Textract フォームの抽出-ヒューマンレビューを呼び出すための条件で、アクティベーション条件を指定します。ワーカーポータルでワーカータスクをプレビューできるように、ドキュメント内の少なくとも 1 つのキーに対して高い信頼度スコアしきい値を設定して、ヒューマンレビューをトリガーすることをお勧めします。

   このウォークスルーで提供されているサンプルドキュメントを使用した場合は、次のようにアクティベーション条件を指定します。

   a. 選択フォームキーの信頼性スコアに基づいて、または特定のフォームキーが欠落している場合に、特定のフォームキーのヒューマンレビューをトリガーします。
   b. を使用する場合キー名で、と入力します。**`Mail Address`**。
   c. 設定:信頼度識別次のしきい値0そして99。
   d. 設定:資格信頼度数次のしきい値0そして99。
   e. 選択Amazon Textract によって識別されるすべてのフォームキーについて、特定の範囲の信頼性スコアでヒューマンレビューをトリガーします。
   f. を使用する場合**`identification confidence`**[] を選択し、0 から 90 までの数値を選択します。
   g. を使用する場合**`qualification confidence`**[] を選択し、0 から 90 までの数値を選択します。

   これにより、Amazon Textract が 99 未満の信頼スコアが返された場合、人間のレビューがトリガーされます。メールアドレスおよびその値、またはドキュメント内で検出されたキーと値のペアについて 90 未満の信頼スコアを返す場合。
8. []ワーカータスクテンプレートの作成で、既定のテンプレートから作成する。
9. を使用する場合テンプレート名で、わかりやすい名前を入力します。
10. を使用する場合タスクの説明のように、次のような結果を追加します。

    `Read the instructions and review the document.`
11. を使用する場合ワーカー選ぶプライベート。
12. メニューから、作成したプライベートチームを選択します。
13. [Create (作成)] を選択します。

ヒューマンレビューワークフローが作成されると、そのワークフローは、ヒューマンレビューワークフローページで. 次の時点ステータスですアクティブで、ワークフロー ARN をコピーして保存します。

ヒューマンレビューワークフローを作成する (API)

ヒューマンレビューワークフローを作成するには、フロー定義、Amazon A2I を使用して、`CreateFlowDefinition`オペレーション.

この例では、Amazon S3 で独自のドキュメントを使用するか、ダウンロードすることができます。このサンプルドキュメントS3 バケットに保存します。

Amazon S3 バケットが同じであることを確認してください。AWS電話に使用するリージョンAnalyzeDocument。バケットを作成するには、「」の手順に従います。バケットの作成のAmazon Simple Storage Service Console。

前提条件

Amazon A2I API を使用してヒューマンレビューワークフローを作成するには、次の前提条件を満たす必要があります。

- Amazon A2I と Amazon Textract API オペレーションの両方を呼び出す権限を持つ IAM ロールを設定します。開始するには、AWS ポリシー、AmazonAugmentedAIFullAccess、および AmazonTextractFullAccess を IAM ロールにアタッチできます。後で必要になるため、IAM ロールの Amazon リソースネーム (ARN) を記録します。

  Amazon Textract を使用する際の詳細な権限については、「」を参照してください。Amazon Textract のアイデンティティベースのポリシーの例。Amazon A2I については、Amazon Augmented AI におけるアクセス許可とセキュリティのAmazon SageMaker 開発者ガイド。
- プライベートワークチームを作成し、ワークチーム ARN を記録します。Amazon A2I を初めて使用する方は、の手順に従います。ステップ 1: 作業チームの作成 (コンソール)。
- ワーカータスクテンプレートを作成します。「」の指示に従って、ワーカータスクテンプレートを作成するをクリックして、Amazon A2I コンソールを使用してテンプレートを作成します。テンプレートを作成するときは、テキスト形式の抽出にとってテンプレートタイプ。テンプレートで、`s3_arn`ドキュメントの Amazon S3 ARN を使用します。での作業者の指示の追加`<full-instructions header="Instructions"></full-instructions>`。

  テンプレートをプレビューする場合は、「」で説明されている権限が IAM ロールに付与されていることを確認してください。ワーカータスクテンプレートプレビューを有効にする。

テンプレートを作成したら、ワーカータスクテンプレート ARN を記録します。

「」で作成したリソースを使用します。前提条件を設定するにはCreateFlowDefinitionリクエスト. このリクエストでは、JSON 形式でアクティベーション条件も指定します。アクティベーション条件を構成する方法については、「」を参照してください。Amazon Textract でヒューマンループアクティブ化条件 JSON スキーマの使用。

ヒューマンレビューワークフローの作成 (AWS SDK for Python (Boto3)))

この例を使用するには、*red*仕様とリソースを含むテキスト。

まず、次のコードを使用して、アクティベーション条件を JSON オブジェクトにエンコードします。これにより、Amazon Textract が 99 未満の信頼スコアが返された場合、人間のレビューがトリガーされます。メールアドレスおよびその値、またはドキュメント内で検出されたキーと値のペアについて 90 未満の信頼スコアを返す場合。この例で提供されているサンプルドキュメントを使用している場合、これらのアクティベーション条件によってヒューマンレビュータスクが作成されます。

```
import json

humanLoopActivationConditions = json.dumps("{
                "Conditions": [
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "Mail Address",
                            "KeyValueBlockConfidenceLessThan": 99,
                            "WordBlockConfidenceLessThan": 99
                        }
                    },
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "*",
                            "KeyValueBlockConfidenceLessThan": 90,
                            "WordBlockConfidenceLessThan": 90
                        }
                    }
                ]
            }"
)
```

を使用するhumanLoopActivationConditionsを構成するにはcreate_flow_definitionリクエスト. 次の例では、SDK for Python (Boto3) を使用してcreate_flow_definitionus-west-2 AWS リージョン内。プライベートワークチームの使用を指定します。

```
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"
)
```

ドキュメントを分析する

Amazon A2I を Amazon Textract ドキュメント分析ワークフローに組み込むには、HumanLoopConfigのAnalyzeDocumentオペレーション.

EclipseHumanLoopConfigでヒューマンレビューワークフロー (フロー定義) ARN を指定します。FlowDefinitionArnで、あなたの人間のループに名前を付けてHumanLoopName。

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

次の例では、SDK for Python (Boto3) を使用してanalyze_documentus-west-2. を交換してください。####リソースを含むテキスト。詳細については、「」を参照してください。documentのAWSSDK for Python (Boto) API リファレンス。

```
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)

次の例ではを使用しています。AWSコールするCLIanalyze_document。これらの例はAWSCLIバージョン 2. 1つ目は短縮構文で、2番目の構文はJSON構文です。詳細については、「」を参照してください。ドキュメント分析のAWS CLIコマンドリファレンス。

```
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**
>
> —human-loop-config パラメーターには空白文字を使用しないでください。これにより、コードの処理の問題が発生する可能性があります。

このリクエストに対する応答には、次のものが含まれます。HumanLoopActivation出力である。これは、人間のループが作成されたかどうかを示し、もしあれば、その理由を示す。ヒューマンループが作成された場合、このオブジェクトにはHumanLoopArn。

の詳細とを使用した例については、AnalyzeDocumentオペレーション、「」を参照してくださいAmazon Textract を使用したドキュメントテキストの分析。

ヒューマンループを監視する

Amazon A2I コンソールと API を使用して、ヒューマンループに関する詳細を表示し、エラーが発生した場合にアクティブなヒューマンループを停止できます。

ヒューマンループ詳細の表示

ヒューマンループのステータスは、Amazon A2I コンソールで、Amazon A2I ランタイム API。

ヒューマンループの詳細を確認するには (コンソール)

1. Amazon A2I コンソールをhttps://console.aws.amazon.com/a2iにアクセスするにはヒューマンレビューワークフローページで.
2. 設定にも使用したヒューマンレビューワークフローを選択します。HumanLoopConfigにAnalyzeDocument。
3. 左ヒューマンループセクションで、詳細を表示するヒューマンループを選択します。

ヒューマンループ (API) の詳細を調べるには:

Amazon A2I を使用するDescribeHumanLoopオペレーション. 呼び出しに使用したヒューマンループ名を指定します。AnalyzeDocument。

次のSDK for Python (Boto3) の例ではを呼び出します。describe_human_loop。

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

ヒューマンループを停止する

人間のループが開始されたら、Amazon A2I コンソールと API を使用してループを停止できます。

人間のループを停止するには (コンソール)

1. で Amazon A2I コンソールを開きます。https://console.aws.amazon.com/a2iにアクセスするにはヒューマンレビューワークフローページで.
2. 設定に使用したヒューマンレビューワークフローを選択します。HumanLoopConfigのAnalyzeDocumentオペレーション.
3. 左ヒューマンループ[] セクションで、停止するヒューマンループを選択します。
4. [停止] を選択します。

人間のループ (API) を停止するには

Amazon A2I を使用するStopHumanLoopオペレーション. 呼び出しに使用したヒューマンループの名前を指定します。AnalyzeDocument。

次の SDK for Python (Boto3) の例の呼び出しstop_human_loop。

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

出力データとワーカーメトリクスの表示

ワーカーによってヒューマンレビュータスクが完了すると、Amazon A2I はヒューマンレビューワークフローで指定した Amazon S3 バケットに出力データを格納します。

プライベートワークフォースを使用する場合、出力データには、個々の作業者のアクティビティを追跡するために使用できる作業者のメタデータが含まれます。

Amazon S3 内の出力データの検索

Amazon A2I は、ヒューマンレビューワークフロー名を、そのヒューマンレビューワークフローを使用して作成されたヒューマンループの出力データを格納するファイルの名前の接頭辞として使用します。

ヒューマンループ出力へのパスは、次のパターンを使用します。*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*人間のループ作成日を年 () で表します。YYYY)、月 (MM)、日 (DD) と作成時間 (時間)hh)、分 (mm)、第二 (ss).

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

ヒューマンループの出力を表示するには、Amazon A2I コンソールを使用します。

ヒューマンループ出力を表示するには

1. で Amazon A2I コンソールを開きます。https://console.aws.amazon.com/a2iにアクセスするにはヒューマンレビューワークフローページで.
2. 設定に使用するヒューマンレビューワークフローを選択します。HumanLoopConfigにAnalyzeDocument。
3. 左ヒューマンループセクションで、出力を確認するヒューマンループを選択します。
4. []出力場所[] を選択し、出力データへのリンクを選択します。

プライベートワーカーアクティビティの追跡

ヒューマンレビュータスクにプライベートワークフォースを使用する場合、出力データには、レビューを完了した作業者に関する次の情報が含まれます。

- 。workerId
- Eclipse workerMetadata:
  - identityProviderType— プライベートワークフォースを管理するために使用されるサービス。
  - issuer— このヒューマンレビュータスクに割り当てられたワークチームに関連付けられた Amazon Cognito ユーザープールまたは OIDC アイデンティティプロバイダー (IdP) 発行者。
  - sub— ワーカーを参照する一意の識別子。Amazon Cognito を使用してワークフォースを作成した場合は、Amazon Cognito を使用して、この ID を使用して、このワーカーに関する詳細 (名前やユーザー名など) を取得できます。その方法については、「」を参照してください。ユーザーアカウントの管理と検索にAmazon Cognito 開発者ガイド。

以下は、Amazon Cognito を使用してプライベートワークフォースを作成した場合に表示される出力の例です。

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
```

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

次に、独自の OIDC IdP を使用してプライベートワークフォースを作成した場合に表示される出力の例を示します。

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

プライベートワークフォースの使用の詳細については、「」を参照してください。プライベートワークフォースを使用するのAmazon SageMaker 開発者ガイド。

Amazon Textract のセキュリティ

AWS では、クラウドのセキュリティが最優先事項です。AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャから利点を得られます。

以下のトピックを使用して、Amazon Textract のリソースをセキュリティで保護する方法について説明します。

トピックス

Amazon Textract におけるデータ保護

Amazon Textract は、AWS 責任共有モデルは、データ保護の規制とガイドラインが含まれています。AWSは、すべてを実行するグローバルインフラストラクチャの保護を担いますAWSのサービス。AWSでは、カスタマーコンテンツと個人データを処理するためのセキュリティ構成管理など、このインフラストラクチャでホストされているデータの制御が維持されます。AWS顧客とAPNパートナーは、データコントローラーまたはデータプロセッサの役割を果たし、AWSクラウド。

データ保護の目的で、AWS アカウントの認証情報を保護し、個々のユーザーアカウントを AWS Identity and Access Management (IAM) で設定することをお勧めします。これにより、それぞれの職務を遂行するために必要なアクセス許可のみを各ユーザーに付与することができます。また、以下の方法でデータを保護することをお勧めします:

- 各アカウントで多要素認証(MFA)を使用します。
- SSL/TLS を使用して AWS リソースと通信します。

- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS暗号化ソリューションをAWSサービス内のすべてのデフォルトのセキュリティ管理と一緒に使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これにより、Amazon S3 に保存される個人データの検出と保護が支援されます。

顧客のアカウント番号などの機密の識別情報は、名前フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。Amazon Textract などを使用する場合も同様です。AWSコンソール、API、を使用したサービスAWS CLI, またはAWSSDK。Amazon Textract、またはその他のサービスに入力したデータはすべて、診断ログの内容として取得される可能性があります。外部サーバーへの URL を指定するときは、そのサーバーへのリクエストを検証するための認証情報を URL に含めないでください。

データ保護の詳細については、AWS セキュリティブログのブログ投稿「AWS の責任共有モデルと GDPR」を参照してください。

Amazon Textract での暗号化

データ暗号化とは、転送中および保管時にデータを保護することです。Amazon S3 マネージドキーを使用して、データを保護できます。AWS KMS key安静時、転送中の標準のトランスポート層セキュリティと並んで。

保管時の暗号化

Amazon Textract でデータを暗号化する主な方法は、サーバー側の暗号化です。Amazon S3 バケットから渡された入力ドキュメントは、Amazon S3 によって暗号化され、アクセスすると復号化されます。リクエストが認証され、お客様がアクセス許可を持っていれば、オブジェクトが暗号化されているかどうかに関係なく同じ方法でアクセスできます。例えば、署名付き URL を使用してオブジェクトを共有する場合、その署名付き URL は、オブジェクトが暗号化されているかどうかに関係なく同じように動作します。さらに、バケット内のオブジェクトをリスト化すると、`List`API は、オブジェクトが暗号化されているかどうかに関係なく、すべてのオブジェクトのリストを返します。

Amazon Textract は、サーバー側での暗号化に 2 つの相互に排他的な方法を使用します。

Amazon S3 が管理するキーによるサーバー側の暗号化 (SSE-S3)

Amazon S3 が管理するキー (SSE-S3) によるサーバー側の暗号化 (SSE-S3) を使用すると、各オブジェクトは一意のキーで暗号化されます。さらにセキュリティを強化するために、このメソッドは、キー自体が、定期的に更新されるマスターキーで暗号化されます。Amazon S3 のサーバー側の暗

号化では、最強のブロック暗号の一つである、256 ビットの高度暗号化規格 (AES-256) を使用してデータを暗号化します。詳細については、「Amazon S3 で管理された暗号化キーによるサーバー側の暗号化 (SSE-S3) を使用したデータの保護」を参照してください。

AWS Key Management Service (SSE-KMS) に保存されている KMS キーによるサーバー側の暗号化

AWS Key Management Service (SSE-KMS) に保存されている KMS キーによるサーバー側の暗号化は、SSE-S3 と似ていますが、このサービスを使用した場合はいくつかの追加の利点があり、追加の料金がかかります。Amazon S3 のオブジェクトへの不正アクセスに対する追加の保護を提供する KMS キーを使用するための個別の許可があります。また、SSE-KMS は、KMS キーがいつ誰によって使用されたかを示す監査証跡も提供します。さらに、KMS キーを作成および管理したり、AWS マネージドキーこれは、ユーザー、サービス、およびリージョンに固有です。詳細については、「」を参照してください。サーバー側の暗号化を使用したデータの保護AWS Key Management Service (SSE-KMS) に保存されている KMS キーを使用します。

転送時の暗号化

Amazon Textract は、転送中の Transport Layer Security (TLS) を使用して、サービスとエージェント間で送受信されるデータを暗号化します。さらに、Amazon Textract は VPC エンドポイントを使用して、Amazon Textract がドキュメントを処理するときに使用されるさまざまなマイクロサービス間でデータを送信します。

インターネットトラフィックのプライバシー

Amazon Textract は HTTPS エンドポイントを介して排他的に通信します。HTTPS エンドポイントは、Amazon Textract でサポートされているすべてのリージョンでサポートされています。

Amazon Textract のIdentity and Access Management

AWS Identity and Access Management(IAM) は、管理者がリソースへのアクセスを安全に制御するために役立つAWSのサービスです。AWSIAM 管理者は、誰にできるかを制御します。認証済み(サインイン) と認可Amazon Textract リソースを使用するには (権限を持つ)。IAM は、無料で使用できる AWS のサービスです。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理

- Amazon Textract で IAM を使用する方法
- Amazon Textract のアイデンティティベースのポリシーの例
- Amazon Textract の ID とアクセスのトラブルシューティング

対象者

の使用方法AWS Identity and Access Management(IAM) は、Amazon Textract で行う作業に応じて異なります。

サービスユーザー— ジョブを実行するために Amazon Textract サービスを使用する場合は、管理者が必要なアクセス許可と認証情報を用意します。さらに多くの Amazon Textract 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするために役に立ちます。Amazon Textract の機能にアクセスできない場合は、「」を参照してください。Amazon Textract の ID とアクセスのトラブルシューティング。

サービス管理者— 社内の Amazon Textract リソースを担当している場合は、通常、Amazon Textract へのフルアクセスがあります。管理者は、従業員がアクセスできる Amazon Textract 機能とリソースを決定します。その後、IAM 管理者にリクエストを送信して、サービスユーザーの許可を変更する必要があります。このページの情報を確認して、IAM の基本概念を理解してください。お客様の会社で Amazon Textract で IAM を利用する方法の詳細については、「」を参照してください。Amazon Textract で IAM を使用する方法。

IAM 管理者— IAM 管理者は、Amazon Textract へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用できる Amazon Textract アイデンティティベースのポリシーの例を表示するには、「」を参照してください。Amazon Textract のアイデンティティベースのポリシーの例。

アイデンティティを使用した認証

認証は、アイデンティティ認証情報を使用して AWS にサインインする方法です。AWS マネジメントコンソールを使用したサインインの詳細については、『IAM ユーザーガイド』の「IAM ユーザーまたはルートユーザーとしての AWS マネジメントコンソール へのサインイン」を参照してください。

AWS アカウント のルートユーザーもしくは IAM ユーザーとして、または IAM ロールを引き受けることによって、認証を受ける (AWS にサインインする) 必要があります。会社のシングルサインオン認証を使用することも、Google や Facebook を使用してサインインすることもできます。このよう

な場合、管理者は以前に IAM ロールを使用して ID フェデレーションを設定しました。他の会社の認証情報を使用して AWS にアクセスした場合、ロールを間接的に割り当てられています。

AWS マネジメントコンソールに直接サインインするには、パスワードとルートユーザーの E メールまたは IAM ユーザー名を使用します。ルートユーザーまたは IAM ユーザーのアクセスキーを使用して AWS にプログラム的にアクセスできます。AWS は、ユーザーの認証情報を使用してリクエストに暗号的で署名するための SDK とコマンドラインツールを提供します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。これには、インバウンド API リクエストを認証するためのプロトコル、署名バージョン 4 を使用します。リクエストの認証の詳細については、「AWS 全般のリファレンス」の「署名バージョン 4 の署名プロセス」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供を要求される場合もあります。例えば、AWS では多要素認証 (MFA) を使用してアカウントのセキュリティを高めることを推奨しています。詳細については、『IAM ユーザーガイド』の「AWS での多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

AWS アカウント を初めて作成する場合は、このアカウントのすべての AWS サービスとリソースに対して完全なアクセス権限を持つシングルサインインアイデンティティで始めます。このアイデンティティは AWS アカウント ルートユーザー と呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。強くお勧めするのは、日常的なタスクには、それが管理者タスクであっても、ルートユーザーを使用しないことです。代わりに、初期の IAM ユーザーを作成するためにのみ、ルートユーザーを使用するというベストプラクティスに従います。その後、ルートユーザーの認証情報を安全な場所に保管し、それらを使用して少数のアカウントおよびサービス管理タスクのみを実行します。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対する特定の許可を持つ AWS アカウント 内のアイデンティティです。IAM ユーザーは、ユーザー名とパスワード、アクセスキーのセットなど、長期的な認証情報を持つことができます。アクセスキーの生成方法の詳細については、『IAM ユーザーガイド』の「IAM ユーザーのアクセスキーの管理」を参照してください。IAM ユーザーにアクセスキーを生成するとき、必ずキーペアを表示して安全に保存してください。後になって、シークレットアクセスキーを回復することはできません。新しいアクセスキーペアを生成する必要があります。

IAM グループ は、IAM ユーザーのコレクションを指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、一度に複数のユーザーに対してアクセス許

可を指定できます。多数の組のユーザーがある場合、グループを使用すると管理が容易になります。例えば、IAMAdminsという名前のグループを設定して、そのグループに IAM リソースを管理するアクセス許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の特定の人またはアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールではテンポラリ認証情報が利用できます。詳細については、『IAM ユーザーガイド』の「IAM ユーザーの作成が適している場合 (ロールではなく)」を参照してください。

IAM ロール

IAM ロール は、特定のアクセス許可を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールを切り替えることによって、AWS マネジメントコンソールで IAM ロールをテンポラリに引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API 演算を呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、『IAM ユーザーガイド』の「IAM ロールの使用」を参照してください。

IAM ロールとテンポラリ認証情報は、次の状況で役立ちます。

- テンポラリ IAM ユーザーアクセス許可 - IAM ユーザーは、特定のタスクに対して複数の異なるアクセス許可をテンポラリに IAM ロールで引き受けることができます。
- フェデレーティッドユーザーアクセス - IAM ユーザーを作成する代わりに、Directory Service、エンタープライズユーザーディレクトリ、またはウェブ ID プロバイダーからの既存のアイデンティティを使用できます。このようなユーザーは フェデレーティッドユーザー と呼ばれます。AWS では、ID プロバイダーを通じてアクセスがリクエストされたとき、フェデレーティッドユーザーにロールを割り当てます。フェデレーティッドユーザーの詳細については、『IAM ユーザーガイド』の「フェデレーティッドユーザーとロール」を参照してください。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを別のアカウントの人物 (信頼済みプリンシパル) に許可できます。ロールは、クロスアカウントアクセスを許可する主な方法です。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスでのロールとリソースベースのポリシーの違いの詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- [Cross-service access] (クロスサービスアクセス) – 一部の AWS のサービスは、AWS の他のサービスの機能を使用します。例えば、サービスで呼び出しを行う場合、そのサービスでは Amazon

EC2 でアプリケーションを実行したり、Amazon S3 にオブジェクトを保存したりするのが一般的です。サービスは、呼び出し元プリンシパルのアクセス許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- [Principal permissions] (プリンシパル許可) – IAM ユーザーまたはロールを使用して AWS でアクションを実行する場合、そのユーザーはプリンシパルとみなされます。ポリシーは、プリンシパルにアクセス許可を付与します。一部のサービスを使用する場合、別のサービスで別のアクションをトリガーするアクションを実行することがあります。この場合、両方のアクションを実行するための許可が必要です。アクションがポリシーで追加の依存アクションを必要とするかどうかを確認するには、を参照してください。Amazon Textract のアクション、リソース、および条件コンテキストキーのサービス認証リファレンス。
- [Service role] (サービスロール) – サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロール です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス権限を委任するロールの作成」を参照してください。
- [Service-linked role] (サービスリンクロール) – サービスリンクロールは、AWS のサービスにリンクされたサービスロールの一種です。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは、IAM アカウント内に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – EC2 インスタンスで実行され、AWS CLI または AWS API リクエストを作成しているアプリケーションの一時的な認証情報を管理するために、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムはテンポラリ認証情報を取得することができます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、IAM ユーザーガイドの「(IAMユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

AWS でのアクセスは、ポリシーを作成し、それらを IAM アイデンティティまたは AWS リソースに添付することで制御できます。ポリシーは AWS のオブジェクトであり、ID やリソースに関連付け

て、これらのアクセス許可を定義します。ルートユーザーまたは IAM ユーザーとしてサインインすることも、IAM ロールを引き受けることもできます。その後リクエストを行うと、AWS が関連するアイデンティティベースまたはリソースベースのポリシーを評価します。ポリシーでのアクセス許可により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWS に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、『IAM ユーザーガイド』の「JSON ポリシー概要」を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

すべての IAM エンティティ (ユーザーまたはロール) は、アクセス許可のない状態からスタートします。言い換えると、デフォルト設定では、ユーザーは何もできず、自分のパスワードを変更することすらできません。何かを実行するアクセス許可をユーザーに付与するには、管理者がユーザーにアクセス許可ポリシーを添付する必要があります。また、管理者は、必要なアクセス許可があるグループにユーザーを追加できます。管理者がグループにアクセス許可を付与すると、そのグループ内のすべてのユーザーにこれらのアクセス許可が付与されます。

IAM ポリシーは、演算の実行方法を問わず、アクションのアクセス許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS マネジメントコンソール、AWS CLI、または AWS API からロールの情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM user ユーザー、ユーザーのグループ、ロールなど、アイデンティティに添付できる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、AWS アカウント 内の複数のユーザー、グループ、およびロールに添付できるスタンドアロンポリシーです。管理ポリシーには、AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーの例は、IAM ロールの信頼ポリシー および Amazon S3 バケットポリシーです。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーが添付されているリソースの場合、ポリシーは、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件を定義します。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または AWS のサービスを含めることができます。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーで IAM の AWS 管理ポリシーを使用することはできません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」 の 「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS では、別のあまり一般的ではないポリシータイプもサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の許可を設定できます。

- 許可の境界 - 許可の境界は、ID ベースのポリシーが IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティの許可の境界を設定できます。結果として得られる許可は、エンティティの アイデンティティベースポリシーとその許可の境界の共通部分です。`Principal`フィールドでユーザーまたはロールを指定するリソースベースのポリシーは、許可の境界では制限されません。これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になります。許可の境界の詳細については、『IAM ユーザーガイド』の「IAM エンティティの許可の境界」を参照してください。
- [Service control policies (SCPs)] (サービスコントロールポリシー SCP) – SCP は、AWS Organizations で組織や組織単位 (OU) に最大アクセス許可を指定する JSON ポリシーです。AWS Organizations は、お客様のビジネスが所有する複数の AWS アカウントをグループ化し、一元的

に管理するサービスです。組織内のすべての機能を有効にすると、サービス制御ポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティ (各 AWS アカウント ルートユーザーなど) に対するアクセス許可を制限します。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。

- セッションポリシー - セッションポリシーは、ロールまたはフェデレーティッドユーザーのテンポラリセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果として得られるセッションの許可は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分です。また、リソースベースのポリシーから許可が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になります。詳細については、「IAM ユーザーガイド」 の 「セッションポリシー」 を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される許可を理解するのがさらに複雑になります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWSが決定する方法の詳細については、『IAM ユーザーガイド』のポリシーの評価ロジックを参照してください。

Amazon Textract で IAM を使用する方法

IAM を使用して Amazon Textract へのアクセスを管理する前に、Amazon Textract で使用できる IAM 機能について理解しておく必要があります。Amazon Textract などの概要を説明します。AWSIAM と連携するサービスについては、「」を参照してください。AWSIAM と連携するサービスのIAM ユーザーガイド。

トピック

- Amazon Textract のアイデンティティベースのポリシー
- Amazon Textract のリソースベースのポリシー
- Amazon Textract タグに基づく認可
- Amazon Textract IAM ロール

Amazon Textract のアイデンティティベースのポリシー

IAM のアイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションが許可または拒否される条件を指定できます。Amazon Textract は、特定のアクショ

ン、リソース、および条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

アクション

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どの プリンシパル がどの リソース に対してどのような 条件 下で アクション を実行できるかということです。

JSON ポリシーの `Action` 要素は、ポリシー内のアクセスを許可または拒否するために使用できるアクションを記述します。ポリシーアクションの名前は通常、関連する AWS API 演算と同じです。一致する API 演算を持たないアクセス許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要な演算もあります。これらの追加アクションは、依存アクション と呼ばれます。

このアクションは、関連付けられた操作を実行するための許可を付与するポリシーで使用されます。

Amazon Textract の非同期アクションには、開始アクションと Get アクションの 2 つのアクション権限が必要です。さらに、Amazon S3 バケットを使用してドキュメントを渡す場合は、アカウントに読み取りアクセス権を付与する必要があります。

Amazon Textract では、すべてのポリシーアクションは次のように始まります。`textract:`。 たとえば、Amazon Textract で Amazon Textract オペレーションを実行するためのアクセス許可を付与するには、`AnalyzeDocument`オペレーションでは、`textract:AnalyzeDocument`彼らのポリシーでの行動。ポリシーステートメントには、`Action` または `NotAction` 要素を含める必要があります。Amazon Textract は、このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一のステートメントに複数のアクションを指定するには、以下のようにコンマで区切ります。

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

ワイルドカード (*) を使用して複数のアクションを指定することができます。例えば、`Describe` という単語で始まるすべてのアクションを指定するには、以下のアクションを含めます。

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

Amazon Textract のアクション一覧については、「」を参照してください。Amazon Textract で定義されるアクションのIAM ユーザーガイド。

リソース

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシーエレメントは、オブジェクトあるいはアクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource エレメントを含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルのアクセス許可 と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

操作のリスト化など、リソースレベルの許可をサポートしないアクションの場合は、ワイルドカード (*) を使用して、ステートメントがすべてのリソースに適用されることを示します。

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

Amazon Textract では、ポリシーでリソース ARN を指定することはできません。

条件キー

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition エレメント (またはCondition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition エレメントはオプションです。イコールや以下などの 条件演算子 を使用する条件式を作成して、リクエスト内に値のあるポリシーの条件に一致させることができます。

1 つのステートメントに複数の Condition エレメントを指定する場合、または 1 つの Condition エレメントに複数のキーを指定する場合、AWS が論理 AND 演算を使用してそれらを評価します。単一の条件キーに複数の値を指定する場合、AWSが論理的な OR 演算を使用して条件を評価します。ステートメントのアクセス許可が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー可変も使用できます。例えば、IAM ユーザー名でタグ付けされている場合のみ、リソースにアクセスする IAM ユーザーアクセス許可を付与できます。詳細につ

いては、『IAM ユーザーガイド』の「IAM ポリシーエレメント: 変数およびタグ」を参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、『IAM ユーザーガイド』の「AWS グローバル条件コンテキストキー」を参照してください。

Amazon Textract にはサービス固有条件キーがありませんが、いくつかのグローバル条件キーの使用をサポートしています。すべてのリストについてはAWSグローバル条件キー、「」を参照してください。AWSグローバル条件コンテキストキーのIAM ユーザーガイド。

例

Amazon Textract アイデンティティベースのポリシーの例を表示するには、「」を参照してください。Amazon Textract のアイデンティティベースのポリシーの例。

Amazon Textract のリソースベースのポリシー

Amazon Textract では、リソースベースのポリシーはサポートされていません。

Amazon Textract タグに基づく認可

Amazon Textract は、リソースのタグ付けやタグに基づいたアクセスの制御をサポートしていません。

Amazon Textract IAM ロール

IAM ロールは AWS アカウント内のエンティティで、特定の許可を持っています。

Amazon Textract での一時的な認証情報の使用

テンポラリ認証情報を使用して、フェデレーションでサインイン、IAM ロールを引き受ける、またはクロスアカウントロールを引き受けることができます。テンポラリセキュリティ認証情報を取得するには、AWS STSAssumeRole または GetFederationToken などの API オペレーションを呼び出します。

Amazon Textract は、一時認証情報の使用をサポートしています。

サービスリンクロール

サービスリンクロールを使用すると、AWS のサービスがユーザーに代わって他のサービスのリソースにアクセスし、アクションを完了することができます。サービスリンクロールは IAM アカウント

内に表示され、サービスによって所有されます。IAM 管理者はサービスリンクロールの許可を表示できますが、編集することはできません。

Amazon Textract は、サービスにリンクされたロールをサポートしていません。

> **Note**
>
> Amazon Textract は、サービスにリンクされたロールをサポートしていないため、AWS のサービスプリンシパルをサポートしていません。サービスプリンシパルの詳細については、「」を参照してください。AWS サービスプリンシパルのIAM ユーザーガイド

サービスロール

この機能では、サービスロールをユーザーに代わって引き受けることをサービスに許可します。このロールにより、サービスはユーザーに代わって他のサービスのリソースにアクセスし、アクションを実行できます。サービスロールは、IAM アカウントに表示され、サービスによって所有されます。つまり、IAM 管理者は、このロールのアクセス許可を変更できます。ただし、これを行うことにより、サービスの機能が損なわれる場合があります。

Amazon Textract ではサービスロールがサポートされています。

Amazon Textract のアイデンティティベースのポリシーの例

IAM ユーザーおよびロールには、Amazon Textract リソースを作成または変更するアクセス許可は付与されていません。AWS マネジメントコンソール、AWS CLI、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行するアクセス許可をユーザーとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者はそれらのアクセス許可が必要な IAM ユーザーまたはグループにそのポリシーを添付します。

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーのベストプラクティス
- ユーザーに自分のアクセス許可の表示を許可
- Amazon Textract での同期オペレーションへのアクセスを許可する

- Amazon Textract での非同期オペレーションへのアクセスを許可する

ポリシーのベストプラクティス

アイデンティティベースポリシーは非常に強力です。これらは、アカウント内で Amazon Textract リソースを作成、アクセス、または削除できるユーザーを決定します。これらのアクションを実行すると、AWS アカウント に追加料金が発生する可能性があります。アイデンティティベースポリシーを作成または編集するときは、以下のガイドラインと推奨事項に従います：

- の使用を開始します。AWSマネージドポリシー— Amazon Textract をすぐに使い始めるには、AWS従業員に必要なアクセス許可を付与するための管理ポリシー。これらのポリシーはアカウントで既に有効になっており、AWS によって管理および更新されています。詳細については、IAM ユーザーガイドの「AWS 管理ポリシーを使用したアクセス許可の使用スタート」を参照してください。
- 最小権限を付与する – カスタムポリシーを作成するときは、タスクの実行に必要な許可のみを付与します。最小限のアクセス許可からスタートし、必要に応じて追加のアクセス許可を付与します。この方法は、寛容なアクセス許可で始め、後でそれらを強化しようとするよりも安全です。詳細については、『IAM ユーザーガイド』の「最小特権を認める」を参照してください。
- 機密性の高い操作に MFA を有効にする – 追加セキュリティとして、機密性の高いリソースまたは API 操作にアクセスするために IAM ユーザーに対して、多要素認証 (MFA) の使用を要求します。詳細については、IAM ユーザーガイドの「AWS での多要素認証 (MFA) の使用」を参照してください。
- 追加のセキュリティとしてポリシー条件を使用する – 実行可能な範囲内で、ID ベースのポリシーでリソースへのアクセスを許可する条件を定義します。例えば、要求が発生しなければならない許容 IP アドレスの範囲を指定するための条件を記述できます。指定された日付または時間範囲内でのみリクエストを許可する条件を書くことも、SSL や MFA の使用を要求することもできます。詳細については、「」を参照してください。IAM JSON ポリシー要素: 条件のIAM ユーザーガイド。

ユーザーに自分のアクセス許可の表示を許可

この例では、ユーザー ID にアタッチされたインラインおよび管理ポリシーの表示を IAM ユーザーに許可するポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了するアクセス許可が含まれています。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Textract での同期オペレーションへのアクセスを許可する

この例のポリシーは、Amazon Textract の同期アクションへのアクセスをの IAM ユーザーに付与します。AWSアカウント.

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:DetectDocumentText",
```

```
                "textract:AnalyzeDocument"
            ],
            "Resource": "*"
        }
    ]
```

Amazon Textract での非同期オペレーションへのアクセスを許可する

次のポリシー例では、IAM ユーザーにAWSAmazon Textract で使用されるすべての非同期オペレーションへのアカウントアクセス。

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

Amazon Textract の ID とアクセスのトラブルシューティング

次の情報は、Amazon Textract と IAM の使用に伴い、発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- Amazon Textract でアクションを実行する権限がない
- iam:PassRole を実行する権限がない
- マイアクセスキーを表示したい
- 管理者として Amazon Textract へのアクセスを他のユーザーに許可したい
- 自分の外の人を許可したいAWSAmazon Textract リソースにアクセスするためのアカウント

Amazon Textract でアクションを実行する権限がない

AWS マネジメントコンソール から、アクションを実行する権限がないと通知された場合、管理者に問い合わせ、サポートを依頼する必要があります。お客様のユーザー名とパスワードを発行したのが、担当の管理者です。

次のエラー例は、mateojacksonIAM ユーザーが実行しようとするDetectDocumentTextテストイメージ上ではあるが、持っていないtextract:DetectDocumentTextアクセス許可。

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

この場合、Mateo は、textimage.png アクションを使用して textract:DetectDocumentText リソースにアクセスできるように、ポリシーの更新を管理者に依頼します。

iam:PassRole を実行する権限がない

iam:PassRole アクションを実行する権限がないというエラーが表示された場合、管理者に問い合わせ、サポートを依頼する必要があります。お客様のユーザー名とパスワードを発行したのが、担当の管理者です。Amazon Textract にロールを渡すことを許可するようにポリシーを更新するようユーザーに依頼します。

一部の AWS サービスでは、新しいサービスロールまたはサービスにリンクされたロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

以下の例のエラーは、という IAM ユーザーがする場合に発生します。marymajorは、コンソールを使用して Amazon Textract でアクションを実行しようとします。ただし、アクションでは、サービスロールによって付与されたアクセス許可がサービスにある必要があります。Mary には、ロールをサービスに渡すアクセス許可がありません。

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

この場合、メアリーは担当の管理者に iam:PassRole アクションを実行できるようにポリシーの更新を依頼します。

マイアクセスキーを表示したい

IAM ユーザーアクセスキーを作成した後は、いつでもアクセスキー ID を表示できます。ただし、シークレットアクセスキーをもう一度表示することはできません。シークレットアクセスキーを紛失した場合は、新しいキーペアを作成する必要があります。

アクセスキーは、アクセスキー ID (例：AKIAIOSFODNN7EXAMPLE) とシークレットアクセスキー (例：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY) の 2 つの部分から構成されます。ユーザー名とパスワードと同様に、リクエストを認証するために、アクセスキー ID とシークレットアクセスキーの両方を使用する必要があります。ユーザー名とパスワードと同様に、アクセスキーをしっかり管理してください。

> **Important**
>
> 正規ユーザー ID を確認するためであっても、アクセスキーをサードパーティーに提供しないでください。提供すると、第三者がアカウントへの永続的アクセスを取得する場合があります。

アクセスキーペアを作成する場合、アクセスキー ID とシークレットアクセスキーを安全な場所に保存するように求めるプロンプトが表示されます。このシークレットアクセスキーは、作成時にのみ使用できます。シークレットアクセスキーを紛失した場合、新しいアクセスキーを IAM ユーザーに追加する必要があります。最大 2 つのアクセスキーを持つことができます。すでに 2 つある場合は、新しいキーペアを作成する前に、いずれかを削除する必要があります。手順を表示するには、『IAM ユーザーガイド』の「アクセスキーの管理」を参照してください。

管理者として Amazon Textract へのアクセスを他のユーザーに許可したい

Amazon Textract へのアクセスを他のユーザーに許可するには、アクセスを必要とする人またはアプリケーション用に IAM エンティティ (ユーザーまたはロール) を作成する必要があります。ユーザーまたはアプリケーションは、このエンティティの認証情報を使用して AWS にアクセスします。次に、Amazon Textract の適切なアクセス許可を付与するポリシーを、そのエンティティにアタッチする必要があります。

すぐにスタートするには、IAM ユーザーガイドの「IAM が委任した初期のユーザーおよびグループの作成」を参照してください。 。

自分の外の人を許可したいAWSAmazon Textract リソースにアクセスするためのアカウント

他のアカウントのユーザーや組織外のユーザーが、リソースへのアクセスに使用できるロールを作成できます。ロールを引き受けるように信頼されたユーザーを指定することができます。リソースベースのポリシーまたはアクセス制御リスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください：

- Amazon Textract がこれらの機能をサポートしているかどうかを確認するには、「」を参照してください。Amazon Textract で IAM を使用する方法。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、『IAM ユーザーガイド』の「所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、『IAM ユーザーガイド』の「第三者が所有する AWS アカウント アカウントへのアクセス権を付与する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、IAM ユーザーガイドの IAM ロールとリソースベースのポリシーとの相違点を参照してください。

ログ記録とモニタリング

Amazon Textract をモニタリングするには、Amazon CloudWatch を使用します。このセクションでは、Amazon Textract のモニタリングを設定する方法について説明します。また、Amazon Textract 指標のリファレンスコンテンツも提供します。

トピック

- Amazon Textract モニタリング
- Amazon Textract 用の CloudWatch メトリクス

Amazon Textract モニタリング

CloudWatch を使用すると、アカウントの個々の Amazon Textract オペレーションまたはグローバルなAmazon Textract メトリクスのメトリクスを取得できます。メトリクスを使用すると、Amazon TextTract ベースのソリューションの状態をトラッキングし、定義されているしきい値を 1 つ以上のメトリクスが超えた場合に通知するようにアラームをセットアップできます。たとえば、サーバーエラーの発生数のメトリクスを確認することができます。特定の Amazon Textract オペレーションが成功した回数を確認することもできます。メトリクスを表示するには、Amazon CloudWatchとすると、AWS CLI、またはCloudWatch API。

Amazon Textract での CloudWatch メトリクスの使用

メトリクスを使用するには、以下の情報を指定する必要があります。

- メトリクスディメンション、またはディメンションなし。ディメンションは、メトリクスを一意に識別するための名前と値のペアです。Amazon Textract にはディメンションが 1 つあり、オペレーション。オペレーション別のメトリクスを提供します。ディメンションを指定しないと、アカウント内のすべての Amazon Textract オペレーションがメトリクスの対象範囲になります。
- メトリクス名 (`UserErrorCount` など)。

Amazon Textract のモニタリングデータは、AWS マネジメントコンソールとすると、AWS CLI、CloudWatch API。CloudWatch API は、いずれかの Amazon AWS Software Development Kit (SDK) または CloudWatch API ツールでも使用できます。コンソールには、CloudWatch API の raw データに基づいて一連のグラフが表示されます。必要に応じて、コンソールに表示されるグラフまたは API から取得したグラフを使用できます。

以下のリストは、メトリクスの一般的な利用方法をいくつか示しています。ここで紹介するのは開始するための提案事項です。すべてを網羅しているわけではありません。

目的	関連するメトリクス
アプリケーションが 1 秒あたりの最大リクエスト数に達したかどうかを確認する	`ThrottledCount` メトリクスの `Sum` 統計をモニタリングします。
リクエストエラーをモニタリングする	`UserErrorCount` メトリクスの `Sum` 統計を使用します。

目的	関連するメトリクス
リクエストの総数を確認する	`ResponseTime` メトリクスの `SampleCount` 統計を使用します。これには、エラーになったリクエストも含まれます。オペレーションの呼び出しが成功した回数のみを確認する場合は、`SuccessfulRequestCount` メトリクスを使用します。
Amazon Textract オペレーションの呼び出しのレイテンシーをモニタリングするにはどうすればよいですか。	`ResponseTime` メトリクスを使用します。

CloudWatch で Amazon Textract をモニタリングするには、適切な CloudWatch アクセス許可が必要です。詳細については、「Amazon CloudWatch に対する認証とアクセスコントロール」を参照してください。

Amazon Textract メトリクスへのアクセス

以下の例では、CloudWatch コンソールを使用して Amazon Textract メトリクスにアクセスする方法を示しています。AWS CLI、CloudWatch API。

メトリクスを表示するには (コンソール)

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 選択のメトリクスで、すべてのメトリクス[] タブで、[Amazon Textract。
3. 選択オペレーションによるメトリクスを選びます。

   例えば、[] を選択します。StartDocumentAnalysis非同期ドキュメント分析が開始された回数を測定するメトリック。
4. 日付範囲の値を選択します。メトリクスのカウントがグラフに表示されます。

成功のメトリクスを表示するには**`StartDocumentAnalysis`**一定期間中に実行されたオペレーションコール (CLI)

- AWS CLI を開き、以下のコマンドを入力します。

```
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
```

この例では、一定期間に StartDocumentAnalysis オペレーションの呼び出しが成功した回数を示しています。詳細については、「get-metric-statistics」を参照してください。

メトリクスにアクセスするには (CloudWatch API)

- GetMetricStatistics を呼び出します。詳細については、「Amazon CloudWatch API リファレンス」を参照してください。

アラームの作成

アラームの状態が変わったときに Amazon Simple Notification Service (Amazon SNS) メッセージを送信する CloudWatch のアラームを作成することができます。指定した期間中、1 つのアラームが 1 つのメトリクスを監視します。このアラームは、複数の期間にわたる一定のしきい値とメトリクスの値の関係性に基づき、1 つ以上のアクションを実行します。アクションは、Amazon SNS のトピックまたは Auto Scaling のポリシーに送信される通知です。

アラームは、持続している状態変化に対してのみアクションを呼び出します。CloudWatch アラームは、特定の状態にあるというだけの理由ではアクションを呼び出しません。状態が変わって、変わった状態が指定期間にわたって維持される必要があります。

アラームを設定するには (コンソール)

1. AWS マネジメントコンソール にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[アラーム] を選択し、[アラームの作成] を選択します。これにより、[アラームウィザードの作成] が起動します。
3. [メトリクスの選択] を選択します。
4. 左すべてのメトリクス[] タブで、[Textract。

5. 選択オペレーション別をクリックし、メトリクスを選びます。

   たとえば、[] を選択します。StartDocumentAnalysisをクリックし、ドキュメントの非同期分析オペレーションの最大数のアラームを設定します。

6. [グラフ化したメトリクス] タブを選択します。
7. [統計] で、[合計] を選択します。
8. [メトリクスの選択] を選択します。
9. [名前] と [説明] を入力します。[次の時] で、[>=] を選択し、任意の最大値を入力します。
10. アラーム状態に達したときに CloudWatch から E メールを受け取るには、このアラームはいつでも次のようになります。で、状態はALARM。既存の Amazon SNS トピックにアラームを送信するには、[通知の送信先:] で既存の SNS トピックを選択します。新しいメールサブスクリプションリスト用の名前とメールアドレスを設定するには、[新しいリスト] を選択します。CloudWatch はリストを保存してフィールドに表示されるため、以降のアラーム設定に利用できます。

    > **Note**
    >
    > [新しいリスト] を使用して新しい Amazon SNS トピックを作成する場合は、宛先に通知を送信する前にメールアドレスを検証する必要があります。Amazon SNS は、アラームがアラーム状態になったときにのみメールを送信します。アラーム状態になったときに E メールアドレスの検証がまだ完了していない場合、宛先には通知が届きません。

11. [Create Alarm] を選択します。

アラームを設定するには (AWS CLI)

- AWS CLI を開き、以下のコマンドを入力します。の値を変更します。`alarm-actions`パラメータを使用して、作成済みの Amazon SNS トピックを参照します。

```
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
```

次の例では、へのコールに対して 10 回を超えるユーザエラーが 5 分以内に発生した場合のアラームの作成方法を示しています。`StartDocumentAnalysis`。詳細については、「put-metric-alarm」を参照してください。

アラームを設定するには (CloudWatch API)

- を呼び出します。`PutMetricAlarm`詳細については、「Amazon CloudWatch API リファレンス」を参照してください。

Amazon Textract 用の CloudWatch メトリクス

このセクションでは、Amazon CloudWatch メトリクスとオペレーションAmazon Textract で使用できるディメンション。

Amazon Textract コンソールから Amazon Textract の集計メトリクスを確認することもできます。

Amazon Textract 用の CloudWatch メトリクス

Amazon Textract メトリクスを次の表に示します。

メトリクス	説明
SuccessfulRequestCount	成功したリクエストの数。成功したリクエストのレスポンスコード範囲は 200〜299 です。 単位: カウント 有効な統計: `Sum,Average`
ThrottledCount	スロットルされたリクエストの数。Amazon Textract は、1 秒あたりに受け取るトランザクションの数がアカウントに設定された制限数を超えると、リクエストをスロットルします。アカウントに設定された制限を頻

メトリクス	説明
	繁に超える場合は、制限の引き上げをリクエストできます。引き上げをリクエストするには、「AWS サービス制限」を参照してください。 単位: カウント 有効な統計: Sum,Average
ResponseTime	Amazon Textract でレスポンスを計算する時間 (ミリ秒)。 単位 : 1. Data Samples 統計のカウント 2. Average 統計のミリ秒 有効な統計: Data Samples,Average Note -ResponseTime メトリックスは Amazon Textract メトリックスペインには含まれません。
ServerErrorCount	サーバーエラーの数。サーバーエラーのレスポンスコード範囲は 500〜599 です。 単位: カウント 有効な統計: Sum,Average
UserErrorCount	ユーザーエラーの数 (無効なパラメータ、無効なイメージ、アクセス許可なしなど)。ユーザーエラーのレスポンスコード範囲は 400〜499 です。 単位: カウント 有効な統計: Sum,Average

Amazon Textract 用の CloudWatch ディメンション

オペレーション別のメトリクスを取得するには、AWS/Textract 名前空間を使用して、オペレーションディメンションを指定します。ディメンションの詳細については、「」を参照してください。ディメンションのAmazon CloudWatch ユーザーガイド。

での Amazon Textract API コールのログ記録AWS CloudTrail

Amazon Textract はと統合されていますAWS CloudTrailは、ユーザー、ロール、または、によって実行されたアクションの記録を提供するサービスAWSAmazon Textract のサービス。CloudTrail は、Amazon Textract のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、Amazon Textract コンソールからの呼び出しと、Amazon Textract API オペレーションへのコード呼び出しが含まれます。

証跡を作成する場合は、Amazon Textract のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールの [Event history (イベント履歴)] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、Amazon Textract に対するリクエスト、そのリクエストが発信された IP アドレス、リクエストの作成者、リクエスト作成日時、その他の詳細情報などを確認できます。

CloudTrailの詳細については、AWS CloudTrail ユーザーガイドを参照してください。

Amazon Textract の CloudTrail での情報

CloudTrailは、アカウントを作成すると AWS アカウントで有効になります。Amazon Textract でアクティビティが発生すると、そのアクティビティは [CloudTrail] (その他) と共に CloudTrail イベントに記録されます。AWSのサービスイベントイベント履歴。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

でのイベントの継続的な記録については、AWSAmazon Textract のイベントなど、アカウントが証跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで作成した追跡がすべての AWS リージョンに適用されます。追跡は、AWS パーティションのすべてのリージョンからのイベントを ログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、その他の設定もできます。AWSサービス:CloudTrail ログで収集されたイベントデータをより詳細に分析し、それに基づく対応を行います。詳細については、以下を参照してください:

- 追跡を作成するための概要

- CloudTrailのサポート対象サービスと統合
- Amazon SNSのCloudTrailの通知の設定
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての Amazon Textract オペレーションは CloudTrail によって記録され、API リファレンス。例えば、`DetectDocumentText` 、`AnalyzeDocument`、および `GetDocumentText` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。同一性情報は以下の判断に役立ちます:

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか.
- リクエストが、ロールとフェデレーティッドユーザーの一時的なセキュリティ認証情報で行われたか。
- リクエストが、別の AWS サービスによって送信されたかどうか。

詳細については、CloudTrail userIdentity エレメントを参照してください。

ログに記録されないリクエストパラメータとレスポンスフィールド

プライバシーの目的で、リクエストイメージバイトやレスポンスバウンディングボックス情報など、特定のリクエストパラメータとレスポンスフィールドはログに記録されません。リクエストパラメータで指定された Amazon S3 バケット名とファイル名は CloudTrail ログエントリで提供されます。CloudTrail ログには、リクエストで渡されたイメージバイトに関する情報は提供されません。次の表は、Amazon Textract オペレーションごとにログに記録されない入力パラメータとレスポンスパラメータを示しています。

オペレーション	リクエストパラメータ	レスポンスのフィールド
AnalyzeDocument	`Bytes`	すべて
DetectDocumentText	`Bytes`	すべて
StartDocumentAnalysis	なし	なし

オペレーション	リクエストパラメータ	レスポンスのフィールド
GetDocumentAnalysis	なし	すべて
StartDocumentTextDetection	なし	なし
GetDocumentTextDetection	なし	すべて

Amazon Textract ログファイルエントリの概要

追跡は、指定したAmazon S3バケットにイベントをログファイルとして配信するように設定できるものです。CloudTrailのログファイルには、単一か複数のログエントリがあります。各イベントは任意の送信元からの単一のリクエストを表し、リクエストされたオペレーション、オペレーションの日時、リクエストパラメーターなどに関する情報を含みます。CloudTrail· ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

`AnalyzeDocument` オペレーションを示す CloudTrail ログエントリの例は、次のとおりです。入力のイメージバイトdocument分析結果 (`responseElements`) はログに記録されません。

```
{
    "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"
}
```

次の例は、の CloudTrail ログエントリを示しています。StartDocumentAnalysisオペレーション. ログエントリには、の Amazon S3 バケット名とイメージファイル名が含まれています。documentLocation。ログにはオペレーションレスポンスも含まれます。

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111111111111:user/janedoe",
                "accountId": "11111111111",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "janedoe"
            },
            "eventTime": "2019-04-04T01:42:24Z",
            "eventSource": "textract.amazonaws.com",
            "eventName": "StartDocumentAnalysis",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "",
            "requestParameters": {
                "documentLocation": {
                    "s3Object": {
                        "bucket": "bucket",
                        "name": "document.png"
                    }
                },
                "featureTypes": [
                    "TABLES"
                ]
            },
            "responseElements": {
                "jobId":
 "f3c718b444fa603d5d625ab967008f4b620d4650c9db8ca1cae01ef7efe51373"
```

```
            },
            "requestID": "9ae352e8-9de1-41ad-b77b-85aa348c2e82",
            "eventID": "f741bca0-c3cb-4805-82ea-baf76439deef",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111111111111"
        }

    ]
}
```

Amazon Textract のコンプライアンス検証

サードパーティーの監査者は、複数の Amazon Textract のセキュリティとコンプライアンスを評価します。AWSコンプライアンスプログラム。これには、HIPAA、SOC、ISO、および PCI が含まれます。

> **Note**
>
> PCI DSS 準拠の対象となる Textract サービスを通じてデータを処理する場合は、AWS Support に連絡し、提供されたプロセスに従ってアカウントをオプトアウトする必要があります。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。一般的な情報については、AWS[Compliance Programs]』コンプライアンス プログラム)(コンプライアンス プログラム)を参照してください。

サードパーティーの監査レポートをダウンロードするには、AWS Artifact を使用します。詳細については、Downloading Reports in AWS および を参照してください。

Amazon Textract を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用可能な法律および規制によって決定されます。AWSでは、コンプライアンスに役立つ、次のリソースを提供しています。

- セキュリティ & コンプライアンスクイックリファレンスガイド – これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、AWS でセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするためのステップが記載されています。

- HIPAA セキュリティおよびコンプライアンスのためのアーキテクチャの設計ホワイトペーパー – このホワイトペーパーは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法を説明します。
- AWS コンプライアンスのリソース - このワークブックおよびガイドのコレクションは、ユーザーの業界や地域で使用できる場合があります。
- AWS Config デベロッパーガイドの Evaluating Resources with Rules – AWS Config のサービスでは、リソースの設定が自社プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。
- AWS Security Hub CSPM— これAWSサービスは、内のセキュリティ状態の包括的なビューを提供します。AWS。セキュリティハブは、セキュリティ業界の標準およびベストプラクティスへの準拠を確認できます。

Amazon Textract の耐障害性

AWSのグローバルインフラストラクチャはAWSリージョンとアベイラビリティーゾーンを中心に構築されます。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立· 隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、[AWS Global Infrastructure] (グローバルインフラストラクチャ) を参照してください。

> **Note**
>
> 一般データ保護規則 (GDPR) により、地域間でのデータの転送は許可されていません。

Amazon Textract のインフラストラクチャセキュリティ

マネージドサービスとして、Amazon Textract はAWSで説明されているグローバルネットワークセキュリティ手順Amazon Web Services: セキュリティプロセスの概要ホワイトペーパー。

あなたは使うAWSが公開している API 呼び出しは、ネットワーク経由で Amazon Textract にアクセスします。クライアントで Transport Layer Security (TLS) 1.0 以降がサポートされている必要

があります。TLS 1.2 以降を推奨します。また、Ephemeral Diffie-Hellman (DHE)や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)などの Perfect Forward Secrecy (PFS)を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、テンポラリセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon Textract での設定と脆弱性の分析

設定および IT 管理は、AWS とお客様の間で共有される責任です。詳細については、AWS 責任共有モデルを参照してください。

Amazon Textract とインターフェイス VPC エンドポイント (AWS PrivateLink)

VPC と Amazon Textract との間にプライベート接続を確立するには、インターフェイス VPC エンドポイント。インターフェイスエンドポイントはAWS PrivateLinkは、インターネットゲートウェイ、NAT デバイス、VPN 接続、AWS Direct Connect 接続なしで、Amazon Textract API にプライベートにアクセスできるテクノロジーです。VPC のインスタンスは、パブリック IP アドレスがなくても Amazon Textract API と通信できます。VPC と Amazon Textract との間のトラフィックは、AWS ネットワークを離れません。

各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。

詳細については、Amazon VPC ユーザーガイドの「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。

Amazon Textract VPC エンドポイントに関する考慮事項

Amazon Textract のインターフェイス VPC エンドポイントを設定する前に、次の項目を確認してください。インターフェイスエンドポイントのプロパティと制限のAmazon VPC User Guide。

Amazon Textract は、VPC からのすべての API アクションの呼び出しをサポートしています。

Amazon Textract 用のインターフェイス VPC エンドポイントの作成

Amazon Textract サービス用の VPC エンドポイントを作成するには、Amazon VPC コンソールまたはAWS Command Line Interface(AWS CLI). 詳細については、Amazon VPC ユーザーガイドのインターフェイスエンドポイントの作成を参照してください。

Amazon Textract 用の VPC エンドポイントを作成するには、次のサービス名を使用します。

- com.amazonaws.##.textract-ほとんどの Amazon Textract オペレーションのエンドポイントを作成します。
- com.amazonaws.##.texttract-fips-連邦情報処理標準 (FIPS) 刊行物 140-2 米国政府標準に準拠した Amazon Textract のエンドポイントを作成するため。

エンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名 (など) を使用して、Amazon Textract への API リクエストを実行できます。`textract.us-east-1.amazonaws.com`。

詳細については、Amazon VPC ユーザーガイドのインターフェイスエンドポイントを介したサービスへのアクセスを参照してください。

Amazon Textract 用の VPC エンドポイントポリシーの作成

Amazon Textract へのアクセスをコントロールするVPC エンドポイントにエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、Amazon VPC ユーザーガイドの「VPC エンドポイントによるサービスのアクセス制御」を参照してください。

例: Amazon Textract アクション用の VPC エンドポイントポリシー

Amazon Textract のエンドポイントポリシーの例を次に示します。このポリシーは、エンドポイントにアタッチされると、すべてのリソースのすべてのプリンシパルに対して、登録されている Amazon Textract アクションへのアクセスを許可します。

このポリシーの例では、オペレーションのみへのアクセスが許可されます。DetectDocumentTextそしてAnalyzeDocument。ただし、ユーザーは VPC エンドポイントの外部から Amazon Textract オペレーションを呼び出すことができます。

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

API リファレンス

このセクションでは、Amazon Textract API オペレーションについて説明します。

トピック

アクション

以下のアクションがサポートされています:

AnalyzeDocument

入力ドキュメントで検出されたアイテム間の関係を分析します。

返される情報のタイプは次のとおりです。

- フォームデータ (キーと値のペア)。2 つの関連情報が返されますBlockオブジェクト、各タイプKEY_VALUE_SET: KEYBlockオブジェクトと VALUEBlockオブジェクト。たとえば、名前: アナ・シルバ・カロライナキーと値が含まれます。名前:が鍵です。アナ・シルバ・カロライナは値です。
- テーブルとテーブルのセルデータ。テーブルBlockオブジェクトには、検出されたテーブルに関する情報が含まれています。セルのあるBlockオブジェクトは、テーブル内の各セルに対して返されます。
- テキストの行と単語。行線Blockオブジェクトのある WORD が 1 つ以上含まれていますBlockオブジェクト。ドキュメント内で検出されたすべての行と単語が返されます (の値と関係のないテキストを含む)。FeatureTypes).

チェックボックスやオプションボタン (ラジオボタン) などの選択要素は、フォームデータやテーブル内で検出できます。セレクション工レメントBlockオブジェクトには、選択状態など、選択要素に関する情報が含まれます。

実行する解析のタイプは、FeatureTypesリスト。

出力は、のリストに返されます。Blockオブジェクト。

AnalyzeDocument は同期演算です。ドキュメントを非同期的に分析するには、StartDocumentAnalysis。

詳細については、「」を参照してください。ドキュメントテキスト分析。

リクエストの構文

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

```
    },
    "FeatureTypes": [ "string" ],
    "HumanLoopConfig": {
       "DataAttributes": {
          "ContentClassifiers": [ "string" ]
       },
       "FlowDefinitionArn": "string",
       "HumanLoopName": "string"
    }
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

Document

base64 でエンコードされたバイトまたは Amazon S3 オブジェクトとしての入力ドキュメント。AWS CLI を使用して Amazon Textract オペレーションを呼び出す場合、イメージバイトを渡すことはできません。ドキュメントは、JPEG、PNG、PDF、または TIFF 形式の画像である必要があります。

AWS SDK を使用して Amazon Textract を呼び出す場合は、を使用して渡されるイメージバイトを base64 エンコードする必要がない場合があります。`Bytes`フィールド。

型: Document オブジェクト

: 必須 はい

FeatureTypes

実行する解析のタイプのリスト。TABLES をリストに追加して、入力ドキュメントで検出されたテーブルに関する情報を返します。FORMS を追加して、検出されたフォームデータを返します。両方のタイプの分析を実行するには、TABLES と FORMS を`FeatureTypes`。ドキュメント内で検出されたすべての行と単語が応答に含まれます (次の値に関連しないテキストを含む) 。`FeatureTypes`).

Type: 文字列の配列

有効な値: `TABLES | FORMS`

: 必須 はい

HumanLoopConfig

ドキュメントを分析するためのループワークフローで、人間の構成を設定します。

型: HumanLoopConfig オブジェクト

: 必須 いいえ

レスポンスの構文

```
{
   "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"
      }
   }
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

AnalyzeDocumentModelVersion

ドキュメントの分析に使用されたモデルのバージョン。

Type: 文字列

Blocks

によって検出および分析される項目AnalyzeDocument。

Type: の配列Blockオブジェクト

DocumentMetadata

分析されたドキュメントに関するメタデータ。一例はページ数です。

型: DocumentMetadata オブジェクト

HumanLoopActivationOutput

ループ評価で人間の結果を表示します。

型: HumanLoopActivationOutput オブジェクト

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

BadDocumentException

Amazon Textract はドキュメントを読み取ることができません。Amazon Textract のドキュメント制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

HTTP ステータスコード: 400

DocumentTooLargeException

文書が大きすぎるため処理できません。10 MB の同期オペレーションの最大ドキュメントサイズ。非同期操作の最大ドキュメントサイズは、PDF ファイルの場合 500 MB です。

HTTP ステータスコード: 400

HumanLoopQuotaExceededException

使用可能なループワークフローでアクティブな人間の最大数を超過したことを示します

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作では、`InvalidParameterException`例外が発生するのは、どちらも`S3Object`または`Bytes`値は、`Document`リクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

UnsupportedDocumentException

入力ドキュメントの形式はサポートされていません。操作用のドキュメントは、PNG、JPEG、PDF、または TIFF 形式にすることができます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby v3

AnalyzeExpense

AnalyzeExpenseテキスト間の財政的関連関係について、入力ドキュメントを同期的に分析します。

情報は次のように返されます。ExpenseDocumentsとし、以下のように分離した。

- LineItemGroups-を含むデータセットLineItems購入したアイテムや領収書の価格など、テキスト行に関する情報を格納する場所。
- SummaryFields-ヘッダー情報や仕入先名など、領収書のその他のすべての情報が含まれます。

リクエストの構文

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

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

Document

バイトまたは S3 オブジェクトとしての入力ドキュメント。

イメージのバイトを Amazon Textract API オペレーションに渡すには、Bytesプロパティ。たとえば、Bytesローカルファイルシステムからロードされたドキュメントを渡すプロパティです。を使用して渡されるイメージバイト数Bytesプロパティは base64 でエンコードされている必要があります。AWS SDK を使用してAmazon Textract API オペレーションを呼び出す場合、コードでは、ドキュメントファイルのバイトをエンコードする必要がない場合があります。

S3 バケットに保存されたイメージを Amazon Textract API オペレーションに渡すには、S3Objectプロパティ。S3 バケットに保存されたドキュメントは base64 でエンコードする必要はありません。

S3 オブジェクトが含まれている S3 バケットの AWS リージョンと Amazon Textract オペレーションで使用する AWS リージョンが一致している必要があります。

AWS CLI を使用して Amazon Textract オペレーションを呼び出す場合、Bytes プロパティを使用したイメージのバイトを渡すことはサポートされていません。最初に Amazon S3 バケットにドキュメントをアップロードし、次に S3Object プロパティを使用してオペレーションを呼び出します。

Amazon Textract が S3 オブジェクトを処理するには、ユーザーが S3 オブジェクトにアクセスするためのアクセス許可が必要です。

型: Document オブジェクト

必須 はい

レスポンスの構文

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

```
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

DocumentMetadata

入力ドキュメントに関する情報。

型: DocumentMetadata オブジェクト

ExpenseDocuments

Amazon Textract によって検出された経費。

Type: 配列のExpenseDocumentオブジェクト

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

BadDocumentException

Amazon Textract はドキュメントを読み取ることができません。Amazon Textract のドキュメント制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

HTTP ステータスコード: 400

DocumentTooLargeException

文書が大きすぎるため処理できません。10 MB の同期オペレーションの最大ドキュメントサイズ。非同期操作の最大ドキュメントサイズは、PDF ファイルの場合 500 MB です。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作では、InvalidParameterException例外が発生するのは、どちらもS3ObjectまたはBytes値は、Documentリクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

UnsupportedDocumentException

入力ドキュメントの形式はサポートされていません。操作用のドキュメントは、PNG、JPEG、PDF、または TIFF 形式にすることができます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

AnalyzeID

アイデンティティドキュメントを分析し、関連情報を確認します。この情報は抽出され、次のように返されます。IdentityDocumentFields。抽出されたテキストの正規化されたフィールドと値の両方を記録します。他の Amazon Textract オペレーションとは異なり、AnalyzeIDは Geometry データを返しません。

リクエストの構文

```
{
   "DocumentPages": [
      {
         "Bytes": blob,
         "S3Object": {
            "Bucket": "string",
            "Name": "string",
            "Version": "string"
         }
      }
   ]
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

DocumentPages

AnalyzeID に渡されるドキュメント。

Type: 配列のDocumentオブジェクト

配列メンバー: 最小数は 1 項目です。最大数は 2 項目です。

必須 はい

レスポンスの構文

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

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

AnalyzeIDModelVersion

ドキュメントの処理に使用されている AnalyzeIdentity API のバージョン。

Type: 文字列

DocumentMetadata

入力ドキュメントに関する情報。

型: DocumentMetadata オブジェクト

IdentityDocuments

AnalyzeID によって処理されるドキュメントのリスト。リスト内での位置を示す数字と、ドキュメントの応答構造が含まれます。

Type: 配列のIdentityDocumentオブジェクト

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

BadDocumentException

Amazon Textract はドキュメントを読み取ることができません。Amazon Textract のドキュメント制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

HTTP ステータスコード: 400

DocumentTooLargeException

文書が大きすぎるため処理できません。10 MB 同期オペレーションの最大ドキュメントサイズ。非同期操作の最大ドキュメントサイズは、PDF ファイルの場合 500 MB です。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作で はInvalidParameterException例外が発生するのは、どちらもS3ObjectまたはBytes値は、Documentリクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

UnsupportedDocumentException

入力ドキュメントの形式はサポートされていません。操作用のドキュメントは、PNG、JPEG、PDF、または TIFF 形式にすることができます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

DetectDocumentText

入力ドキュメント内のテキストを検出します。Amazon Textract では、テキスト行とテキスト行を構成する単語を検出できます。入力ドキュメントは、JPEG、PNG、PDF、または TIFF 形式の画像である必要があります。DetectDocumentText検出されたテキストを次の配列で返します。Blockオブジェクト。

各ドキュメントページには、Blockタイプの PAGE。各ページBlockオブジェクトはLINEの親ですBlockページ上で検出されたテキストの行を表すオブジェクト。[1 行]Blockオブジェクトは、行を構成する各単語の親です。単語は次のように表されます。BlockWORD 型のオブジェクト。

DetectDocumentText は同期演算です。ドキュメントを非同期的に分析するには、StartDocumentTextDetection。

詳細については、「」を参照してください。ドキュメントのテキストの検出。

リクエストの構文

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

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

Document

base64 でエンコードされたバイトまたは Amazon S3 オブジェクトとしての入力ドキュメント。AWS CLI を使用して Amazon Textract オペレーションを呼び出す場合、イメージバイトを渡すことはできません。ドキュメントは、JPEG または PNG 形式であることが必要です。

AWS SDK を使用して Amazon Textract を呼び出す場合は、を使用して渡されるイメージバイトを base64 エンコードする必要がない場合があります。Bytesフィールド。

型: Document オブジェクト

: 必須 はい

レスポンスの構文

```
{
   "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
   }
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

Blocks

の配列Block文書内で検出されたテキストを含むオブジェクト。

Type: の配列Blockオブジェクト

DetectDocumentTextModelVersion

Type: 文字列

DocumentMetadata

ドキュメントに関するメタデータ。ドキュメント内で検出されたページ数が含まれます。

型: DocumentMetadata オブジェクト

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

BadDocumentException

Amazon Textract はドキュメントを読み取ることができません。Amazon Textract のドキュメント制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

HTTP ステータスコード: 400

DocumentTooLargeException

文書が大きすぎるため処理できません。10 MB の同期オペレーションの最大ドキュメントサイズ。非同期操作の最大ドキュメントサイズは、PDF ファイルの場合 500 MB です。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作では、`InvalidParameterException`例外が発生するのは、どちらも`S3Object`または`Bytes`値は、`Document`リクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

UnsupportedDocumentException

入力ドキュメントの形式はサポートされていません。操作用のドキュメントは、PNG、JPEG、PDF、または TIFF 形式にすることができます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

GetDocumentAnalysis

ドキュメント内のテキストを分析する Amazon Textract 非同期オペレーションの結果を取得します。

非同期テキスト分析を開始するには、を呼び出します。StartDocumentAnalysis。ジョブ識別子 (JobId). テキスト分析操作が完了すると、Amazon Textract は、への最初の呼び出しに登録された Amazon Simple Notification Service (Amazon SNS) トピックに完了ステータスを公開します。StartDocumentAnalysis。テキスト検出操作の結果を取得するには、まずAmazon SNS トピックに発行されたステータス値がSUCCEEDED。もしそうなら、電話してくださいGetDocumentAnalysisをクリックし、ジョブ識別子を渡します (JobId) の最初の呼び出しからStartDocumentAnalysis。

GetDocumentAnalysisの配列を返します。Blockオブジェクト。次のタイプの情報が返されます。

- フォームデータ (キーと値のペア)。関連情報は 2 つで返されます。Blockオブジェクト、各タイプKEY_VALUE_SET: KeyBlockオブジェクトと VALUEBlockオブジェクト。たとえば、名前: アナ・シルバ・カロライナキーと値が含まれます。名前:が鍵です。アナ・シルバ・カロライナは値です。
- 表および表のセルデータ。テーブルBlockオブジェクトには、検出されたテーブルに関する情報が含まれています。セルのあるBlockオブジェクトは、テーブル内の各セルに対して返されます。
- テキストの行と単語。[1 行]Blockオブジェクトには、1 つ以上の WORD が含まれていますBlockオブジェクト。ドキュメント内で検出されたすべての行と単語が返されます (文字列の値と関係のないテキストを含む)。StartDocumentAnalysis FeatureTypes入力パラメータ)。

チェックボックスやオプションボタン (ラジオボタン) などの選択要素は、フォームデータやテーブル内で検出できます。セレクションエレメントBlockオブジェクトには、選択状態など、選択要素に関する情報が含まれます。

を使用するMaxResults返されるブロックの数を制限するパラメータです。で指定した数を超える結果がある場合MaxResultsとすると、の値NextTokenオペレーションのレスポンスは、次の結果セットを取得するためのページ割りトークンが含まれています。次の結果ページを取得したい場合GetDocumentAnalysisを設定し、NextToken前回の呼び出しから返されたトークン値を持つリクエストパラメータGetDocumentAnalysis。

詳細については、「」を参照してください。ドキュメントテキスト分析。

リクエストの構文

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

JobId

テキスト検出ジョブの一意の識別子。-JobIdから返されましたStartDocumentAnalysis。あるJobId値は 7 日間だけ有効です。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: ^[a-zA-Z0-9-_]+$

必須: はい

MaxResults

ページ割りコールごとに返す結果の最大数。指定できる最大値は 1,000 です。1,000 より大きい値を指定した場合、最大 1,000 件の結果が返されます。デフォルト値は 1,000 です。

Type: 整数

有効範囲: 最小値は 1 です。

必須: いいえ

NextToken

前のレスポンスが不完全だった場合 (取得するブロックが多いため)、Amazon Textract はレスポンスでページ割りトークンを返します。このページ割りトークンを使用して、次のブロックのセットを取得できます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `.*\S.*`

必須: いいえ

レスポンスの構文

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

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

AnalyzeDocumentModelVersion

Type: 文字列

Blocks

テキスト分析操作の結果。

Type: の配列Blockオブジェクト

DocumentMetadata

Amazon Textract が処理したドキュメントに関する情報。DocumentMetadataは、Amazon Textract ビデオオペレーションからページ分割されたレスポンスの各ページに返されます。

型: DocumentMetadata オブジェクト

JobStatus

テキスト検出ジョブの現在のステータス。

Type: 文字列

有効な値: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

レスポンスが切り捨てられた場合、Amazon Textract はこのトークンを返します。次のリクエストでこのトークンを使用して、次のテキスト検出結果セットを取得できます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `.*\S.*`

StatusMessage

検出ジョブを完了できなかった場合に返します。発生したエラーの説明が含まれています。

Type: 文字列

Warnings

ドキュメント分析操作中に発生した警告のリスト。

Type: の配列Warningオブジェクト

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidJobIdException

無効なジョブ識別子が渡されましたGetDocumentAnalysisまたはGetDocumentAnalysis。

HTTP ステータスコード: 400

InvalidKMSKeyException

KMS キーが入力された状態で復号化権限がないか、KMS キーが誤って入力されたことを示します。

HTTP ステータスコード: 400

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作では、`InvalidParameterException`例外が発生するのは、どちらも`S3Object`または`Bytes`値は、`Document`リクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

GetDocumentTextDetection

ドキュメント内のテキストを検出する Amazon Textract 非同期オペレーションの結果を取得します。Amazon Textract では、テキスト行とテキスト行を構成する単語を検出できます。

非同期テキスト検出を開始するには、を呼び出します。StartDocumentTextDetection。ジョブ識別子 (JobId). テキスト検出操作が完了すると、Amazon Textract は、への最初の呼び出しに登録された Amazon Simple Notification Service (Amazon SNS) トピックに完了ステータスを公開します。StartDocumentTextDetection。テキスト検出操作の結果を取得するには、まずAmazon SNS トピックに発行されたステータス値がSUCCEEDED。もしそうなら、電話してくださいGetDocumentTextDetectionを選択し、ジョブ識別子を渡します (JobId) の最初の呼び出しからStartDocumentTextDetection。

GetDocumentTextDetectionの配列を返します。Blockオブジェクト。

各ドキュメントページには、Blockタイプの PAGE。各ページBlockオブジェクトは LINE の親ですBlockページ上で検出されたテキストの行を表すオブジェクト。行ですBlockオブジェクトは、行を構成する各単語の親です。単語は次のように表されます。BlockWORD 型のオブジェクト。

MaxResults パラメーターを使用して、返されるブロックの数を制限します。で指定した数を超える結果がある場合MaxResultsとすると、の値NextTokenオペレーションのレスポンスには、次の結果セットを取得するためのページ割りトークンが含まれています。次の結果ページを取得するには、GetDocumentTextDetectionを入力し、NextToken前回の呼び出しから返されたトークン値を持つリクエストパラメータGetDocumentTextDetection。

詳細については、「」を参照してください。ドキュメントのテキストの検出。

リクエストの構文

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

JobId

テキスト検出ジョブの一意の識別子。-JobIdから返されるStartDocumentTextDetection。あるJobId値は 7 日間だけ有効です。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: ^[a-zA-Z0-9-_]+$

: 必須 はい

MaxResults

ページ割りコールごとに返す結果の最大数。指定できる最大値は 1,000 です。1,000 より大きい値を指定した場合、最大 1,000 件の結果が返されます。デフォルト値は 1,000 です。

Type: 整数

有効範囲: 最小値は 1 です。

: 必須 いいえ

NextToken

前のレスポンスが不完全だった場合 (取得するブロックが多いため)、Amazon Textract はレスポンスでページ割りトークンを返します。このページ割りトークンを使用して、次のブロックのセットを取得できます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: .*\S.*

: 必須 いいえ

レスポンスの構文

```
{
   "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 ]
      }
```

```
    ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

Blocks

テキスト検出操作の結果。

Type: の配列Blockオブジェクト

DetectDocumentTextModelVersion

Type: 文字列

DocumentMetadata

Amazon Textract が処理したドキュメントに関する情報。DocumentMetadataは、Amazon Textract ビデオオペレーションからページ分割されたレスポンスの各ページに返されます。

型: DocumentMetadata オブジェクト

JobStatus

テキスト検出ジョブの現在のステータス。

Type: 文字列

有効な値: IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

レスポンスが切り捨てられると、Amazon Textract はこのトークンを返します。後続のリクエストでこのトークンを使用して、次のテキスト検出結果セットを取得できます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: .*\S.*

StatusMessage

検出ジョブを完了できなかった場合に返します。発生したエラーの説明が含まれています。

Type: 文字列

Warnings

ドキュメントのテキスト検出操作中に発生した警告のリスト。

Type: の配列Warningオブジェクト

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidJobIdException

無効なジョブ識別子が渡されましたGetDocumentAnalysisまたはGetDocumentAnalysis。

HTTP ステータスコード: 400

InvalidKMSKeyException

KMS キーが入力された状態で復号化権限がないか、KMS キーが誤って入力されたことを示します。

HTTP ステータスコード: 400

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作で
はInvalidParameterException例外が発生するのは、どちらもS3ObjectまたはBytes値

は、Documentリクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby v3

GetExpenseAnalysis

請求書と領収書を分析する Amazon Textract 非同期オペレーションの結果を取得します。Amazon Textract は、入力された請求書と領収書から、連絡先情報、購入した商品、およびベンダー名を検索します。

非同期請求書/領収書の分析を開始するには、StartExpenseAnalysis。ジョブ識別子 (JobId). 請求書/領収分析が完了すると、Amazon Simple Notification Service (Amazon SNS) トピックに完了ステータスが発行されます。このトピックは、への最初の呼び出しで登録する必要があります。StartExpenseAnalysis。請求書/領収分析オペレーションの結果を取得するには、Amazon SNS トピックに発行されたステータス値がSUCCEEDED。もしそうなら、電話してくださいGetExpenseAnalysisを選択し、ジョブ識別子を渡します (JobId) を最初の呼び出しからStartExpenseAnalysis。

MaxResults パラメーターを使用して、返されるブロックの数を制限します。で指定した数を超える結果がある場合MaxResultsとすると、の値NextTokenオペレーションレスポンスには、次の結果セットを取得するためのページ割りトークンが含まれています。次の結果ページを取得するには、GetExpenseAnalysisを入力し、NextToken前回の呼び出しから返されたトークン値を持つリクエストパラメータGetExpenseAnalysis。

詳細については、「」を参照してください。請求書と領収書の分析。

リクエストの構文

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

JobId

テキスト検出ジョブの一意の識別子。-JobIdから返されるStartExpenseAnalysis。あるJobId値は 7 日間だけ有効です。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: ^[a-zA-Z0-9-_]+$

必須: はい

MaxResults

ページ割りコールごとに返す結果の最大数。指定できる最大値は 20 です。20 より大きい値を指定した場合、最大 20 件の結果が返されます。デフォルト値は 20 です。

Type: 整数

有効範囲: 最小値は 1 です。

必須: いいえ

NextToken

前のレスポンスが不完全だった場合 (取得するブロックが多いため)、Amazon Textract はレスポンスでページ割りトークンを返します。このページ割りトークンを使用して、次のブロックのセットを取得できます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: .*\S.*

必須: いいえ

レスポンスの構文

```
{
   "AnalyzeExpenseModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
               "LineItems": [
```

```
                {
                   "LineItemExpenseFields": [
                      {
                         "LabelDetection": {
                            "Confidence": number,
                            "Geometry": {
                               "BoundingBox": {
                                  "Height": number,
                                  "Left": number,
                                  "Top": number,
                                  "Width": number
                               },
                               "Polygon": [
                                  {
                                     "X": number,
                                     "Y": number
                                  }
                               ]
                            },
                            "Text": "string"
                         },
                         "PageNumber": number,
                         "Type": {
                            "Confidence": number,
                            "Text": "string"
                         },
                         "ValueDetection": {
                            "Confidence": number,
                            "Geometry": {
                               "BoundingBox": {
                                  "Height": number,
                                  "Left": number,
                                  "Top": number,
                                  "Width": number
                               },
                               "Polygon": [
                                  {
                                     "X": number,
                                     "Y": number
                                  }
                               ]
                            },
                            "Text": "string"
                         }
```

```
                        }
                    ]
                }
            ]
        }
    ],
    "SummaryFields": [
        {
            "LabelDetection": {
                "Confidence": number,
                "Geometry": {
                    "BoundingBox": {
                        "Height": number,
                        "Left": number,
                        "Top": number,
                        "Width": number
                    },
                    "Polygon": [
                        {
                            "X": number,
                            "Y": number
                        }
                    ]
                },
                "Text": "string"
            },
            "PageNumber": number,
            "Type": {
                "Confidence": number,
                "Text": "string"
            },
            "ValueDetection": {
                "Confidence": number,
                "Geometry": {
                    "BoundingBox": {
                        "Height": number,
                        "Left": number,
                        "Top": number,
                        "Width": number
                    },
                    "Polygon": [
                        {
                            "X": number,
                            "Y": number
```

```
                        }
                     ]
                  },
                  "Text": "string"
               }
            }
         ]
      }
   ],
   "JobStatus": "string",
   "NextToken": "string",
   "StatusMessage": "string",
   "Warnings": [
      {
         "ErrorCode": "string",
         "Pages": [ number ]
      }
   ]
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

AnalyzeExpenseModelVersion

AnalyzeExpense の現在のモデルバージョンです。

Type: 文字列

DocumentMetadata

Amazon Textract が処理したドキュメントに関する情報。DocumentMetadataは、Amazon Textract オペレーションからページ分割されたレスポンスの各ページに返されます。

型: DocumentMetadata オブジェクト

ExpenseDocuments

Amazon Textract によって検出された経費。

Type: 配列のExpenseDocumentオブジェクト

JobStatus

テキスト検出ジョブの現在のステータス。

Type: 文字列

有効な値: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

レスポンスが切り捨てられると、Amazon Textract はこのトークンを返します。後続のリクエストでこのトークンを使用して、次のテキスト検出結果セットを取得できます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 255 です。

パターン: `.*\S.*`

StatusMessage

検出ジョブを完了できなかった場合に返します。発生したエラーの説明が含まれています。

Type: 文字列

Warnings

ドキュメントのテキスト検出操作中に発生した警告のリスト。

Type: 配列のWarningオブジェクト

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidJobIdException

無効なジョブ識別子が渡されましたGetDocumentAnalysisまたはGetDocumentAnalysis。

HTTP ステータスコード: 400

InvalidKMSKeyException

KMS キーが入力された状態で復号化権限がないか、KMS キーが誤って入力されたことを示します。

HTTP ステータスコード: 400

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作で はInvalidParameterException例外が発生するのは、どちらもS3ObjectまたはBytes値は、Documentリクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

StartDocumentAnalysis

キーと値のペア、表、選択要素など、検出された項目間のリレーションシップについて、入力ドキュメントの非同期分析を開始します。

StartDocumentAnalysisでは、JPEG、PNG、TIFF、および PDF 形式のドキュメント内のテキストを分析できます。ドキュメントは Amazon S3 バケットに格納されます。を使用するDocumentLocationを使用して、ドキュメントのバケット名とファイル名を指定します。

StartDocumentAnalysisジョブ識別子を返します (JobId) を使用して、オペレーションの結果を取得します。テキスト分析が完了すると、Amazon Textract は、で指定した Amazon Simple Notification Service (Amazon SNS) トピックに完了ステータスを発行します。NotificationChannel。テキスト分析操作の結果を取得するには、まず Amazon SNS トピックに発行されたステータス値がSUCCEEDED。もしそうなら、電話してくださいGetDocumentAnalysisをクリックし、ジョブ識別子を渡します (JobId) の最初の呼び出しからStartDocumentAnalysis。

詳細については、「」を参照してください。ドキュメントテキスト分析。

リクエストの構文

```
{
   "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"
   }
```

```
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

ClientRequestToken

開始リクエストを識別するために使用するべき等トークン。同じトークンを複数で使用する場合StartDocumentAnalysisリクエスト、同じJobIdが返されます。を使用するClientRequestToken同じジョブが誤って複数回開始されないようにするためです。詳細については、「」を参照してください。Amazon Textract 非同期オペレーションを呼び出す。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: `^[a-zA-Z0-9-_]+$`

必須: いいえ

DocumentLocation

処理されるドキュメントの場所。

型: DocumentLocation オブジェクト

必須: はい

FeatureTypes

実行する解析のタイプのリスト。TABLES をリストに追加して、入力ドキュメントで検出されたテーブルに関する情報を返します。FORMS を追加して、検出されたフォームデータを返します。両方のタイプの分析を実行するには、TABLES と FORMS をFeatureTypes。ドキュメント内で検出されたすべての行と単語が応答に含まれます (次の値に関連しないテキストを含む) FeatureTypes).

Type: 文字列の配列

有効な値: `TABLES | FORMS`

必須: はい

JobTag

Amazon SNS トピックに対して発行される完了通知に含まれる、指定する識別子。例えば、次を使用できます。JobTag完了通知が対応する文書のタイプ (納税フォームや領収書など) を識別します。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: `[a-zA-Z0-9_.\-:]+`

必須: いいえ

KMSKeyId

推論結果の暗号化に使用される KMS キー。これは、キー ID 形式またはキーエイリアス形式のいずれかになります。KMS キーが提供されると、KMS キーは、カスタマーバケット内のオブジェクトのサーバー側の暗号化に使用されます。このパラメータが有効でない場合、結果は SSE-S3 を使用してサーバー側で暗号化されます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 2,048 です。

パターン: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

必須: いいえ

NotificationChannel

Amazon Textract でオペレーションの完了ステータスを公開する Amazon SNS トピック ARN。

型: NotificationChannel オブジェクト

必須: いいえ

OutputConfig

出力が顧客定義のバケットに送られるかどうかを設定します。デフォルトでは、Amazon Textract は内部的に結果を保存し、GetDocumentAnalysis s オペレーションによってアクセスされます。

型: OutputConfig オブジェクト

必須: いいえ

レスポンスの構文

```
{
   "JobId": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

JobId

ドキュメントテキスト検出ジョブの識別子。を使用するJobId以降の呼び出しでジョブを識別するにはGetDocumentAnalysis。あるJobId値は 7 日間だけ有効です。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: ^[a-zA-Z0-9-_]+$

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

BadDocumentException

Amazon Textract はドキュメントを読み取ることができません。Amazon Textract のドキュメント制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

HTTP ステータスコード: 400

DocumentTooLargeException

文書が大きすぎるため処理できません。同期オペレーションの最大ドキュメントサイズ 10 MB。非同期操作の最大ドキュメントサイズは、PDF ファイルの場合 500 MB です。

HTTP ステータスコード: 400

IdempotentParameterMismatchException

あるClientRequestToken入力パラメータがオペレーションに再利用されましたが、他の入力パラメータの少なくとも 1 つが、オペレーションに対する前回の呼び出しとは異なります。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidKMSKeyException

KMS キーが入力された状態で復号化権限がないか、KMS キーが誤って入力されたことを示します。

HTTP ステータスコード: 400

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作で はInvalidParameterException例外が発生するのは、どちらもS3ObjectまたはBytes値は、Documentリクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、を参照してください。Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

LimitExceededException

Amazon Textract サービスの制限を超えました。たとえば、同時起動する非同期ジョブが多すぎると、オペレーションの開始を呼び出します (StartDocumentTextDetectionたとえば、) 同時に実行されるジョブの数が Amazon Textract のサービスの制限を下回るまで、limitExceptionException の例外 (HTTP ステータスコード:400) を受け取ります。

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この上限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

UnsupportedDocumentException

入力ドキュメントの形式はサポートされていません。操作用のドキュメントは、PNG、JPEG、PDF、または TIFF 形式にすることができます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

StartDocumentTextDetection

ドキュメント内のテキストの非同期検出を開始します。Amazon Textract では、テキスト行とテキスト行を構成する単語を検出できます。

StartDocumentTextDetectionでは、JPEG、PNG、TIFF、および PDF 形式のドキュメント内のテキストを分析できます。ドキュメントは Amazon S3 バケットに格納されます。を使用するDocumentLocationを選択して、ドキュメントのバケット名とファイル名を指定します。

StartTextDetectionジョブ識別子を返します (JobId) を使用して、オペレーションの結果を取得します。テキストの検出が完了すると、Amazon Textract は、で指定した Amazon Simple Notification Service (Amazon SNS) トピックに完了ステータスを発行します。NotificationChannel。テキスト検出操作の結果を取得するには、まず Amazon SNS トピックに発行されたステータス値がSUCCEEDED。もしそうなら、電話してくださいGetDocumentTextDetectionをクリックし、ジョブ識別子を渡します (JobId) の最初の呼び出しからStartDocumentTextDetection。

詳細については、「」を参照してください。ドキュメントのテキストの検出。

リクエストの構文

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

ClientRequestToken

開始リクエストを識別するために使用されるべき等トークン。複数で同じトークンを使用する場合StartDocumentTextDetectionリクエスト、同じJobIdが返されました。を使用するClientRequestToken同じジョブが誤って複数回開始されないようにする。詳細については、「」を参照してください。Amazon Textract 非同期オペレーションを呼び出す。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: ^[a-zA-Z0-9-_]+$

必須: いいえ

DocumentLocation

処理されるドキュメントの場所。

型: DocumentLocation オブジェクト

必須: はい

JobTag

Amazon SNS トピックに対して発行される完了通知に含まれる、指定した ID。例えば、JobTag完了通知が対応する文書のタイプ（納税フォームや領収書など）を識別します。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: [a-zA-Z0-9_.\-:]+

必須: いいえ

KMSKeyId

推論結果の暗号化に使用される KMS キーです。これは、キー ID またはキーエイリアスの形式のいずれかになります。KMS キーが提供されると、KMS キーは、カスタマーバケット内のオブジェクトのサーバー側の暗号化に使用されます。このパラメータが有効でない場合、結果は SSE-S3 を使用してサーバー側で暗号化されます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 2,048 です。

パターン: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

必須: いいえ

NotificationChannel

Amazon Textract でオペレーションの完了ステータスを公開する Amazon SNS トピック ARN。

型: NotificationChannel オブジェクト

必須: いいえ

OutputConfig

出力が顧客定義のバケットに送られるかどうかを設定します。デフォルトでは、Amazon Textract は、GetDocumentTextDetection オペレーションでアクセスするために内部的に結果を保存します。

型: OutputConfig オブジェクト

必須: いいえ

レスポンスの構文

```
{
   "JobId": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

JobId

ドキュメントのテキスト検出ジョブの識別子。を使用するJobId以降の呼び出しでジョブを識別するにはGetDocumentTextDetection。あるJobId値は 7 日間有効です。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: `^[a-zA-Z0-9-_]+$`

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

BadDocumentException

Amazon Textract はドキュメントを読み取ることができません。Amazon Textract のドキュメント制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

HTTP ステータスコード: 400

DocumentTooLargeException

文書が大きすぎるため処理できません。同期オペレーションの最大ドキュメントサイズは 10 MB です。非同期操作の最大ドキュメントサイズは、PDF ファイルの場合 500 MB です。

HTTP ステータスコード: 400

IdempotentParameterMismatchException

あるClientRequestToken入力パラメータがオペレーションで再利用されましたが、他の入力パラメータの少なくとも 1 つが、オペレーションに対する前回の呼び出しとは異なります。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidKMSKeyException

KMS キーが入力された状態で復号化権限がないか、KMS キーが誤って入力されたことを示します。

HTTP ステータスコード: 400

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作では、`InvalidParameterException`例外が発生するのは、どちらも`S3Object`または`Bytes`値は、`Document`リクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。詳細については、Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

LimitExceededException

Amazon Textract のサービスの制限を超えました。たとえば、同時起動する非同期ジョブが多すぎると、オペレーションの開始を呼び出します (`StartDocumentTextDetection`たとえば、LimitExceededException の例外 (HTTP ステータスコード:400) を受け取ります。同時に実行されるジョブの数が Amazon Textract のサービスの制限を下回るまで、limitException の例外 (HTTP ステータスコード:400) を受け取ります。

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この制限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

UnsupportedDocumentException

入力ドキュメントの形式はサポートされていません。操作用のドキュメントは、PNG、JPEG、PDF、または TIFF 形式にすることができます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

StartExpenseAnalysis

連絡先情報、購入品目、仕入先名などのデータの請求書または領収書の非同期分析を開始します。

StartExpenseAnalysisでは、JPEG、PNG、および PDF 形式のドキュメント内のテキストを分析できます。ドキュメントは Amazon S3 バケットに格納する必要があります。を使用するDocumentLocationパラメータを使用して、S3 バケットの名前と、バケット内のドキュメントの名前を指定します。

StartExpenseAnalysisジョブ識別子を返します (JobId) に提供することとGetExpenseAnalysisを選択して、オペレーションの結果を取得します。入力請求書/領収書の分析が完了すると、Amazon Textract は、指定した Amazon Simple Notification Service (Amazon SNS) トピックに完了ステータスを発行します。NotificationChannel。請求書および領収書の分析操作の結果を取得するには、Amazon SNS トピックに発行されたステータス値がSUCCEEDED。もしそうなら、電話してくださいGetExpenseAnalysisをクリックし、ジョブ識別子を渡します (JobId) への呼び出しによって返されたStartExpenseAnalysis。

詳細については、「」を参照してください。請求書と領収書の分析。

リクエストの構文

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

リクエストパラメータ

リクエストは以下の JSON 形式のデータを受け入れます。

ClientRequestToken

開始リクエストを識別するために使用されるべき等トークン。同じトークンを複数で使用する場合StartDocumentTextDetectionリクエスト、同じJobIdが返されます。を使用するClientRequestToken同じジョブが誤って複数回開始されないようにするためです。詳細については、「」を参照してください。Amazon Textract 非同期オペレーションを呼び出す

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: ^[a-zA-Z0-9-_]+$

必須: いいえ

DocumentLocation

処理されるドキュメントの場所。

型: DocumentLocation オブジェクト

必須: はい

JobTag

Amazon SNS トピックに対して発行される完了通知に含まれる、指定した ID。たとえば、を使用できます。JobTag完了通知が対応する文書のタイプ（納税フォームや領収書など）を識別します。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: [a-zA-Z0-9_.\-:]+

必須: いいえ

KMSKeyId

推論結果の暗号化に使用される KMS キー。これは、キー ID またはキーエイリアスの形式のいずれかになります。KMS キーが提供されると、KMS キーは、カスタマーバケット内のオブジェク

トのサーバー側の暗号化に使用されます。このパラメータが有効でない場合、結果は SSE-S3 を使用してサーバー側で暗号化されます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 2,048 です。

パターン: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

必須: いいえ

NotificationChannel

Amazon Textract でオペレーションの完了ステータスを公開する Amazon SNS トピック ARN。

型: NotificationChannel オブジェクト

必須: いいえ

OutputConfig

出力が顧客定義のバケットに送られるかどうかを設定します。デフォルトでは、Amazon Textract は内部的に結果を保存し、`GetExpenseAnalysis`オペレーション.

型: OutputConfig オブジェクト

必須: いいえ

レスポンスの構文

```
{
   "JobId": "string"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

サービスから以下のデータが JSON 形式で返されます。

JobId

テキスト検出ジョブの一意の識別子。-`JobId`から返される`StartExpenseAnalysis`。ある`JobId`値は 7 日間有効です。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 64 です。

パターン: `^[a-zA-Z0-9-_]+$`

エラー

AccessDeniedException

そのアクションを実行する権限がありません。そのオペレーションを実行するには、承認されたユーザーまたは IAM ロールの Amazon リソースネーム (ARN) を使用します。

HTTP ステータスコード: 400

BadDocumentException

Amazon Textract はドキュメントを読み取ることができません。Amazon Textract のドキュメント制限の詳細については、「」を参照してください。Amazon Textract のハード制限。

HTTP ステータスコード: 400

DocumentTooLargeException

文書が大きすぎるため処理できません。同期オペレーションの最大ドキュメントサイズ 10 MB。非同期操作の最大ドキュメントサイズは、PDF ファイルの場合 500 MB です。

HTTP ステータスコード: 400

IdempotentParameterMismatchException

あるClientRequestToken入力パラメータがオペレーションで再利用されましたが、他の入力パラメータの少なくとも 1 つが、オペレーションに対する前回の呼び出しとは異なります。

HTTP ステータスコード: 400

InternalServerError

Amazon Textract でサービスの問題が発生しました。もう一度やり直してください。

HTTP ステータスコード: 500

InvalidKMSKeyException

KMS キーが入力された状態で復号化権限がないか、KMS キーが誤って入力されたことを示します。

HTTP ステータスコード: 400

InvalidParameterException

入力パラメータが制約に違反しています。たとえば、同期操作では、InvalidParameterException例外が発生するのは、どちらもS3ObjectまたはBytes値は、Documentリクエストパラメータ。API オペレーションを再度呼び出す前にパラメータを検証します。

HTTP ステータスコード: 400

InvalidS3ObjectException

Amazon Textract は、リクエストで指定された S3 オブジェクトにアクセスできません。Amazon S3 へのアクセスの設定トラブルシューティング情報については、「」を参照してください。Amazon S3 トラブルシューティング

HTTP ステータスコード: 400

LimitExceededException

Amazon Textract サービスの制限を超えました。たとえば、同時起動する非同期ジョブが多すぎると、オペレーションの開始を呼び出します (StartDocumentTextDetectionたとえば、) 同時に実行されるジョブの数が Amazon Textract のサービスの制限を下回るまで、limitExceptedException の例外 (HTTP ステータスコード:400) を受け取ります。

HTTP ステータスコード: 400

ProvisionedThroughputExceededException

お客様のスループット制限を超えたリクエストの数。この制限を引き上げる場合は、Amazon Textract までお問い合わせください。

HTTP ステータスコード: 400

ThrottlingException

Amazon Textract は一時的にリクエストを処理できませんでした。もう一度やり直してください。

HTTP ステータスコード: 500

UnsupportedDocumentException

入力ドキュメントの形式はサポートされていません。操作用のドキュメントは、PNG、JPEG、PDF、または TIFF 形式にすることができます。

HTTP ステータスコード: 400

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

データ型

以下のデータ型がサポートされています。

- AnalyzeIDDetections
- Block
- BoundingBox
- Document
- DocumentLocation
- DocumentMetadata
- ExpenseDetection
- ExpenseDocument
- ExpenseField
- ExpenseType
- Geometry
- HumanLoopActivationOutput

- HumanLoopConfig
- HumanLoopDataAttributes
- IdentityDocument
- IdentityDocumentField
- LineItemFields
- LineItemGroup
- NormalizedValue
- NotificationChannel
- OutputConfig
- Point
- Relationship
- S3Object
- Warning

AnalyzeIDDetections

AnalyzeID 操作で検出された情報を格納するために使用します。

内容

Confidence

検出されたテキストの信頼スコア。

Type: 浮動小数点

値の範囲: 最小値は 0 です。最大値は 100 です。

: 必須 いいえ

NormalizedValue

日付に対してのみ返され、検出された値のタイプと、より機械で読みやすい方法で書かれた日付を返します。

型: NormalizedValue オブジェクト

: 必須 いいえ

Text

正規化されたフィールドまたはそれに関連付けられた値のテキスト。

Type: 文字列

: 必須 はい

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

Block

あるBlockは、互いに近いピクセルのグループ内で文書内で認識される項目を表します。で返された情報Blockオブジェクトは、操作のタイプに応じて異なります。ドキュメントのテキスト検出 (例:DetectDocumentText) をクリックすると、検出された単語とテキスト行に関する情報が取得されます。テキスト分析 (例 : AnalyzeDocument) では、ドキュメント内で検出されたフィールド、テーブル、および選択要素に関する情報を取得することもできます。

の配列Blockオブジェクトは、同期操作と非同期操作の両方によって返されます。同期操作では、DetectDocumentTextの配列Blockオブジェクトは、結果のセット全体です。非同期操作では、GetDocumentAnalysisの場合、配列は 1 つ以上の応答に対して返されます。

詳細については、「」を参照してください。Amazon Textract 仕組み。

内容

BlockType

認識されるテキスト項目のタイプ。テキスト検出の操作では、次のタイプが返されます。

- ページ-LINEのリストが含まれます。Blockドキュメントページで検出されたオブジェクト。
- 単語-文書ページで検出された単語。単語とは、スペースで区切られていない、1 個以上の ISO 基本ラテンアルファベットです。
- ライン-文書ページで検出された、タブ区切りの連続した単語の文字列。

テキスト分析操作では、次のタイプが返されます。

- ページ-子のリストが含まれます。Blockドキュメントページで検出されたオブジェクト。
- KEY_VALUE_SET-キーと値を格納するBlockドキュメントページで検出されたリンクされたテキストのオブジェクト。を使用するEntityTypeフィールドを使用して、KEY_VALUE_SETオブジェクトがキーかどうかを判別します。Blockオブジェクトまたは VALUEBlockオブジェクト。
- 単語-文書ページで検出された単語。単語とは、スペースで区切られていない、1 個以上の ISO 基本ラテンアルファベットです。
- ライン-文書ページで検出された、タブ区切りの連続した単語の文字列。
- テーブル-ドキュメントページで検出されたテーブル。テーブルは、2 つ以上の行または列を持つグリッドベースの情報で、セル範囲はそれぞれ 1 行と 1 列です。
- 細胞-検出されたテーブル内のセル。セルは、セル内のテキストを含むブロックの親です。

- ELEMENT-ドキュメントページで検出されるオプションボタン (ラジオボタン) やチェックボックスなどの選択要素。の値を使うSelectionStatus選択要素のステータスを確認することができます。

Type: 文字列

有効な値: KEY_VALUE_SET | PAGE | LINE | WORD | TABLE | CELL | SELECTION_ELEMENT

: 必須 いいえ

ColumnIndex

表のセルが表示される列です。最初の列の位置は 1 です。ColumnIndexが返されませんDetectDocumentTextそしてGetDocumentTextDetection。

Type: 整数

有効範囲: 最小値は 0 です。

: 必須 いいえ

ColumnSpan

表のセルがまたがる列の数。現在、この値は、スパンされる列数が 1 より大きい場合でも、常に 1 です。ColumnSpanが返されませんDetectDocumentTextそしてGetDocumentTextDetection。

Type: 整数

有効範囲: 最小値は 0 です。

: 必須 いいえ

Confidence

Amazon Textract の信頼スコアは、認識されたテキストの精度と、認識されたテキストの周囲にあるジオメトリの精度を示します。

Type: 浮動小数点

有効範囲: 最小値は 0 です。最大値は 100 です。

: 必須 いいえ

EntityTypes

エンティティのタイプ。次のものが返されます。

- キー-ドキュメント上のフィールドの識別子。
- 値-フィールドテキスト。

`EntityTypes`が返されません`DetectDocumentText`そして`GetDocumentTextDetection`。

Type: 文字列の配列

有効な値: `KEY | VALUE`

: 必須 いいえ

Geometry

イメージ上の認識されたテキストの位置。これには、テキストを囲む軸揃えの粗い境界ボックスと、より正確な空間情報を得るための、細かい粒度のポリゴンが含まれています。

型: Geometry オブジェクト

: 必須 いいえ

Id

認識されたテキストの識別子。この識別子は、1 つの操作に対してのみ一意です。

Type: 文字列

パターン: `.*\S.*`

: 必須 いいえ

Page

ブロックが検出されたページ。`Page`非同期操作によって返されます。1 より大きいページ値は、PDF または TIFF 形式の複数ページのドキュメントに対してのみ返されます。スキャンした画像 (JPEG/PNG) は、複数のドキュメントページを含む場合でも、単一ページのドキュメントと見なされます。の価値`Page`は常に1です。同期操作は返されない`Page`なぜなら、すべての入力ドキュメントは単一ページのドキュメントと見なされるからです。

Type: 整数

有効範囲: 最小値は 0 です。

: 必須 いいえ

Relationships

現在のブロックの子ブロックのリスト。たとえば、LINE オブジェクトには、テキスト行の一部である各 WORD ブロックの子ブロックがあります。現在のブロックに子ブロックがない場合など、存在しないリレーションシップについては、リストに Relationship オブジェクトがありません。リストのサイズは、次のようになります。

- 0-ブロックには子ブロックがありません。
- 1-ブロックに子ブロックがあります。

Type: の配列Relationshipオブジェクト

: 必須 いいえ

RowIndex

表のセルが配置されている行。最初の行の位置は 1 です。RowIndexが返されませんDetectDocumentTextそしてGetDocumentTextDetection。

Type: 整数

有効範囲: 最小値は 0 です。

: 必須 いいえ

RowSpan

テーブルのセルがまたがる行数。現在、この値は、スパンされる行数が 1 より大きい場合でも、常に 1 です。RowSpanが返されませんDetectDocumentTextそしてGetDocumentTextDetection。

Type: 整数

有効範囲: 最小値は 0 です。

: 必須 いいえ

SelectionStatus

オプションボタンやチェックボックスなど、選択要素の選択ステータス。

Type: 文字列

有効な値: SELECTED | NOT_SELECTED

: 必須 いいえ

Text

Amazon Textract で認識される単語またはテキスト行。

Type: 文字列

: 必須 いいえ

TextType

Amazon Textract が検出したテキストの種類です。手書きのテキストと印刷されたテキストをチェックできます。

Type: 文字列

有効な値: `HANDWRITING | PRINTED`

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

BoundingBox

文書ページ上で検出されたページ、テキスト、キーと値のペア、表、表のセル、または選択要素の周囲の境界ボックス。-`left`(X 座標) と`top`(y 座標) は、境界ボックスの上端と左端を表す座標。画像の左上隅が原点 (0,0) であることに注意してください。

-`top`そして`left`返される値は、ドキュメント全体のページサイズの比率です。たとえば、入力イメージが 700 x 200 ピクセルの場合、境界ボックスの左上の座標は 350 x 50 ピクセルで、API は `left` 値 0.5 (350/700) および `top` 値 0.25 (50/200) を返します。

-`width`そして`height`値は、文書ページ全体の大きさの比率としての境界ボックスの大きさを表します。たとえば、ドキュメントのページサイズが 700 x 200 ピクセルで、境界ボックスの幅が 70 ピクセルの場合、返される幅は 0.1 になります。

内容

Height

境界ボックスの高さ。

Type: 浮動小数点

必須 いいえ

Left

Left ドキュメント全体の幅の比率としての境界ボックスの左座標。

Type: 浮動小数点

必須 いいえ

Top

Top 文書ページ全体の高さの比率としての境界ボックスの上端座標。

Type: 浮動小数点

必須 いいえ

Width

境界ボックスの幅。文書ページ全体の幅の比率としての境界ボックスの幅。

Type: 浮動小数点

必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

Document

バイトまたは S3 オブジェクトとしての入力ドキュメント。

イメージのバイトを Amazon Textract API オペレーションに渡すには、Bytesプロパティ。たとえば、Bytesローカルファイルシステムからロードされたドキュメントを渡すプロパティです。を使用して渡されるイメージバイト数Bytesプロパティは base64 でエンコードされている必要があります。AWS SDK を使用して Amazon Textract API オペレーションを呼び出している場合、コードでは、ドキュメントファイルのバイトをエンコードする必要がない場合があります。

S3 バケットに保存されたイメージを Amazon Textract API オペレーションに渡すには、S3Objectプロパティ。S3 バケットに保存されたドキュメントは base64 でエンコードする必要はありません。

S3 オブジェクトが含まれている S3 バケットの AWS リージョンと Amazon Textract オペレーションで使用する AWS リージョンが一致している必要があります。

AWS CLI を使用して Amazon Textract オペレーションを呼び出す場合、Bytes プロパティを使用したイメージのバイトを渡すことはサポートされていません。最初に Amazon S3 バケットにドキュメントをアップロードし、次に S3Object プロパティを使用してオペレーションを呼び出します。

Amazon Textract が S3 オブジェクトを処理するには、ユーザーが S3 オブジェクトにアクセスするためのアクセス許可が必要です。

内容

Bytes

base64 でエンコードされたドキュメントバイトの BLOB。BLOB バイトで提供されるドキュメントの最大サイズは 5 MB です。ドキュメントバイトは PNG または JPEG 形式である必要があります。

AWS SDK を使用して Amazon Textract を呼び出す場合は、を使用して渡されたイメージバイトを base64 エンコードする必要がない場合があります。Bytesフィールド。

Type: Base64 でエンコードされたバイナリデータオブジェクト

長さの制約: 最小長は 1 です。最大長は 10485760 です。

必須: いいえ

S3Object

S3 オブジェクトをドキュメントソースとして識別します。S3 バケットに保存されたドキュメントの最大サイズは 5 MB です。

型: S3Object オブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

DocumentLocation

処理するドキュメントが含まれる Amazon S3 バケット。これは、次のような非同期操作で使用されます。StartDocumentTextDetection。

入力ドキュメントは、JPEG または PNG 形式の画像ファイルになります。また、PDF 形式のファイルにすることもできます。

内容

S3Object

入力ドキュメントが含まれる Amazon S3 バケット。

型: S3Object オブジェクト

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

DocumentMetadata

入力ドキュメントに関する情報。

内容

Pages

ドキュメント内で検出されたページ数。

Type: 整数

有効範囲: 最小値は 0 です。

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

ExpenseDetection

Amazon Textract によって検出された値またはラベルに関する情報を格納するために使用されるオブジェクトです。

内容

Confidence

検出の信頼度 (パーセンテージ)

Type: 浮動小数点

有効範囲: 最小値は 0 です。最大値は 100 です。

: 必須 いいえ

Geometry

検出ページ、テキスト、キーと値のペア、表、表のセル、選択要素です。

型: Geometry オブジェクト

: 必須 いいえ

Text

Amazon Textract で認識される単語またはテキスト行

Type: 文字列

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

ExpenseDocument

AnalyzeExpense によって返されるすべての情報を保持する構造体

内容

ExpenseIndex

情報の取得元が文書内のどの請求書または領収書かを示します。最初のドキュメントは 1、2 番目のドキュメントは 2 番目のドキュメントです。

Type: 整数

有効範囲: 最小値は 0 です。

: 必須 いいえ

LineItemGroups

ドキュメントの各テーブルで検出された情報。LineItems。

Type: の配列LineItemGroupオブジェクト

: 必須 いいえ

SummaryFields

Amazon Textract のテーブル外で見つかった情報。

Type: の配列ExpenseFieldオブジェクト

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

ExpenseField

タイプ、ラベル検出、および値検出のカテゴリに分類された、検出された情報の内訳

内容

LabelDetection

検出された要素の明示的に記述されたラベル。

型: ExpenseDetection オブジェクト

: 必須 いいえ

PageNumber

値が検出されたページ番号。

Type: 整数

値の範囲: 最小値は 0 です。

: 必須 いいえ

Type

検出された要素の暗黙のラベル。明示的な要素については labelDetection と一緒に提示します。

型: ExpenseType オブジェクト

: 必須 いいえ

ValueDetection

検出された要素の値。明示的要素と暗黙的な要素に存在する。

型: ExpenseDetection オブジェクト

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください :

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

ExpenseType

Amazon Textract によって検出されたタイプに関する情報を格納するために使用されるオブジェクトです。

内容

Confidence

精度の信頼度 (パーセンテージ)。

Type: 浮動小数点

有効範囲: 最小値は 0 です。最大値は 100 です。

: 必須 いいえ

Text

Amazon Textract によって検出された単語またはテキスト行。

Type: 文字列

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

Geometry

検出ページ、テキスト、キーと値のペア、表、表のセル、選択要素など、ドキュメントページ上の項目についてドキュメントページ上の位置に関する情報。

内容

BoundingBox

ドキュメントページ上の認識されたアイテムの位置を表す軸揃えの粗い表現。

型: BoundingBox オブジェクト

: 必須 いいえ

Polygon

バウンディングボックス内では、認識されたアイテムの周囲をきめ細かなポリゴン。

Type: 配列配列Pointオブジェクト

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

HumanLoopActivationOutput

ループ評価で人間の結果を表示します。HumanLooParn がない場合、入力はヒューマンレビューをトリガーしませんでした。

内容

HumanLoopActivationConditionsEvaluationResults

人間のレビューを活性化した条件を含む、状態評価の結果を表示します。

Type: 文字列

長さの制約: 最大長は 10240 です。

必須: いいえ

HumanLoopActivationReasons

人間のレビューが必要だったかどうか、そして理由を示します。

Type: 文字列の配列

配列メンバー: 最小数は 1 項目です。

必須: いいえ

HumanLoopArn

新たに作成された HumanLoop の Amazon リソースネーム (ARN)。

Type: 文字列

長さの制約: 最大長は 256 です。

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++

- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

HumanLoopConfig

いずれかの条件が満たされた場合にドキュメントが送信されるヒューマンレビューワークフローを設定します。レビューの前に画像の特定の属性を設定することもできます。

内容

DataAttributes

入力データの属性を設定します。

型: HumanLoopDataAttributes オブジェクト

必須 いいえ

FlowDefinitionArn

フロー定義の Amazon リソースネーム (ARN)。

Type: 文字列

長さの制約: 最大長は 256 です。

必須 はい

HumanLoopName

このイメージに使用されるヒューマンワークフローの名前。これは、リージョン内で一意である必要があります。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 63 です。

Pattern: `^[a-z0-9](-*[a-z0-9])*`

必須 はい

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++

- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

HumanLoopDataAttributes

イメージの属性を設定できます。現在、個人を特定できる情報やアダルトコンテンツを含まない画像を宣言できます。

内容

ContentClassifiers

入力イメージに個人を特定できる情報やアダルトコンテンツが含まれているかどうかを設定します。

Type: 文字列の配列

配列メンバ: 最大数は 256 項目です。

有効な値: `FreeOfPersonallyIdentifiableInformation | FreeOfAdultContent`

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

IdentityDocument

AnalyzeID 操作で処理された各ドキュメントをリストする構造体。

内容

DocumentIndex

IdentityDocument リスト内のドキュメントの配置を示します。最初のドキュメントは 1、2 番目のドキュメントは 2 とマークされます。

Type: 整数

値の範囲: 最小値は 0 です。

必須: いいえ

IdentityDocumentFields

アイデンティティドキュメントから抽出された情報を記録するために使用される構造。正規化されたフィールドと抽出されたテキストの値の両方が含まれます。

Type: 配列のIdentityDocumentFieldオブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

IdentityDocumentField

抽出された情報の正規化されたタイプと、それに関連付けられたテキストの両方を含む構造体。これらはそれぞれ「タイプ」と「値」として抽出されます。

内容

Type

AnalyzeID 操作で検出された情報を格納するために使用します。

型: AnalyzeIDDetections オブジェクト

: 必須 いいえ

ValueDetection

AnalyzeID 操作で検出された情報を格納するために使用します。

型: AnalyzeIDDetections オブジェクト

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

LineItemFields

ドキュメントのテーブルにあるさまざまな行に関する情報を保持する構造体。

内容

LineItemExpenseFields

ExpenseFields は、テーブル上で検出された行からの情報を表示するために使用します。

Type: 配列のExpenseFieldオブジェクト

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

LineItemGroup

lineItems を含むテーブルのグループで、各テーブルがテーブルで識別されます。`LineItemGroupIndex`。

内容

LineItemGroupIndex

ドキュメント内の特定の表を識別するために使用される番号。最初に検出されたテーブルには lineItemGroupIndex が 1、2 番目の 2 などになります。

Type: 整数

値の範囲: 最小値は 0 です。

: 必須 いいえ

LineItems

テーブルの特定の行に関する情報の内訳。

Type: 配列のLineItemFieldsオブジェクト

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

NormalizedValue

値のタイプや値など、ドキュメント内の日付に関する情報が含まれます。

内容

Value

日付の値で、年-月-日時:分:秒と記述されます。

Type: 文字列

: いいえ

ValueType

検出された値の正規化されたタイプ。この場合、DATE。

Type: 文字列

有効な値: DATE

: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

NotificationChannel

Amazon Textract が非同期ドキュメントオペレーションの完了ステータスを発行する先の Amazon Simple Notification Service (Amazon SNS) トピックは次のとおりです。StartDocumentTextDetection。

内容

RoleArn

Amazon SNS トピックへの Amazon Textract パブリッシングアクセス権限を付与する、IAM ロールの Amazon リソースネーム (ARN)。

Type: 文字列

長さの制約: 最小長は 20 です。最大長は 2,048 です。

パターン: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必須: はい

SNSTopicArn

Amazon Textract が完了ステータスを投稿する Amazon SNS トピック。

Type: 文字列

長さの制約: 最小長は 20 です。最大長は 1,024 です。

Pattern: `(^arn:([a-z\d-]+):sns:[a-zA-Z\d-]{1,20}:\w{12}:.+$)`

必須: はい

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2

- AWS SDK for Ruby v3

OutputConfig

ユーザーが作成したバケットに出力を送信するかどうかを設定します。バケットの名前と出力ファイルのプレフィックスを設定するために使用します。

`OutputConfig`は、出力の配置場所を調整できるオプションのパラメータです。デフォルトでは、Amazon Textract は結果を内部的に保存し、Get API オペレーションによってのみアクセスできます。OutputConfig を有効にすると、出力が送信されるバケットの名前と、結果をダウンロードできる結果のファイルプレフィックスを設定できます。さらに、`KMSKeyID`出力を暗号化するには、のパラメータをカスタマーマスターキー (CMK) に設定します。このパラメータを設定しないと、Amazon Textract は AWS マネージド CMK for Amazon S3 を使用してサーバーサイドを暗号化します。

Amazon Textract によるドキュメントの処理には、カスタマーコンテンツの解読が必要です。お客様のアカウントが AI サービスのオプトアウトポリシーに基づいてオプトアウトされた場合、カスタマーコンテンツがサービスによって処理された後、暗号化されていないすべてのカスタマーコンテンツが即時かつ完全に削除されます。Amazon Textract では、出力のコピーは保持されません。オプトアウトの方法の詳細については、「」を参照してください。AI サービスのオプトアウトポリシーの管理。

データプライバシーの詳細については、「」を参照してください。データプライバシーのよくある質問。

内容

S3Bucket

出力先のバケットの名前。

Type: 文字列

長さの制約: 最小長は 3 です。最大長は 255 です。

パターン: `[0-9A-Za-z\.\-_]*`

必須 はい

S3Prefix

出力を保存するオブジェクトキーのプレフィックス。有効でない場合、プレフィックスは「texttract_output」になります。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 1,024 です。

Pattern: `.*\S.*`

必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

Point

ドキュメントページ上のポイントの X 座標と Y 座標。返される X および Y 値は、ドキュメント全体のページサイズの比率です。たとえば、入力ドキュメントが 700 x 200 で、X=0.5 および Y = 0.25 を返す場合、ポイントはドキュメントページの (350,50) ピクセル座標になります。

の配列Pointオブジェクト、Polygon「」の一部として、が返りますGeometryで返されるオブジェクトBlockオブジェクト。あるPolygonオブジェクトは、検出されたテキスト、キーと値のペア、テーブル、テーブルセル、または選択エレメントの周りのきめ細かなポリゴンを表します。

内容

X

上の点の X 座標の値Polygon。

Type: 浮動小数点

必須: いいえ

Y

上の点の Y 座標の値Polygon。

Type: 浮動小数点

必須: いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

Relationship

ブロックが互いにどのように関連しているかに関する情報。あるBlockオブジェクトには、0 個以上が含まれていますRelationリスト内のオブジェクト、Relationships。詳細については、「Block」を参照してください。

-Type要素は、すべてのブロックに対する関係の型を提供します。IDs配列。

内容

Ids

関連ブロックの ID の配列。関係のタイプは、Type要素。

Type: 文字列の配列

パターン: .*\S.*

: 必須 いいえ

Type

IDs 配列のブロックが現在のブロックと持つ関係のタイプ。この関係はVALUEまたはCHILD。VALUE 型の関係は、キーと値のペアの KEY に関連付けられている VALUE ブロックの ID を含むリストです。CHILD 型の関係は、行の場合は WORD ブロックを識別する ID のリストです。テーブルの場合は Cell ブロック、選択要素の場合は WORD ブロックを識別します。

Type: 文字列

有効な値: VALUE | CHILD | COMPLEX_FEATURES

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2

- AWS SDK for Ruby v3

S3Object

ドキュメントを識別する S3 バケット名とファイル名。

ドキュメントを含む S3 バケットの AWS リージョンは、Amazon Textract オペレーションに使用するリージョンと一致する必要があります。

Amazon Textract バケット内のファイルを処理するには、S3 バケットとファイルにアクセスするための許可が必要です。

内容

Bucket

S3 バケットの名前。# 文字はファイル名には有効ではないことに注意してください。

Type: 文字列

長さの制約: 最小長は 3 です。最大長は 255 です。

パターン: `[0-9A-Za-z\.\-_]*`

: 必須 いいえ

Name

入力ドキュメントのファイル名。同期操作では、JPEG または PNG 形式のイメージファイルを使用できます。非同期操作は、PDF および TIFF 形式のファイルもサポートします。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 1,024 です。

Pattern: `.*\S.*`

: 必須 いいえ

Version

バケットのバージョニングが有効化されている場合には、オブジェクトのバージョンを指定できます。

Type: 文字列

長さの制約: 最小長は 1 です。最大長は 1,024 です。

Pattern: `.*\S.*`

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

Warning

非同期テキスト分析時に発生した問題に関する警告 (StartDocumentAnalysis) または非同期ドキュメントテキスト検出 (StartDocumentTextDetection).

内容

ErrorCode

警告のエラーコード。

Type: 文字列

: 必須 いいえ

Pages

警告が適用されるページのリスト。

Type: 整数の配列

値の範囲: 最小値は 0 です。

: 必須 いいえ

以下の資料も参照してください。

言語固有の AWS SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください：

- AWS SDK for C++
- AWS SDK for Go
- AWSSDK for Java V2
- AWS SDK for Ruby v3

Amazon Textract のハード制限

Amazon Textract のハード制限を次に示します。これは変更できません。変更できる場所の制限と制限については、「」を参照してください。Amazon Textract エンドポイントとクォータ。変更できる制限については、「AWS サービスの制限」を参照してください。制限を変更するには、「ケースの作成」を参照してください。

Amazon Textract

制限	説明
受け入れられるファイル形式	操作は、JPEG、PNG、PDF、および TIFF ファイルをサポートしています。（PDF 内の JPEG 2000 でエンコードされたイメージがサポートされています）。
ファイルサイズとページ数の制限	同期操作の場合、JPEG、PNG、PDF、および TIFF ファイルのサイズ制限は 10 MB です。PDF ファイルと TIFF ファイルも 1 ページの制限があります。非同期操作の場合、JPEG ファイルと PNG ファイルのサイズ制限は 10 MB です。PDF ファイルと TIFF ファイルの制限は 500 MB です。PDF ファイルと TIFF ファイルの制限は 3,000 ページです。
PDF 固有の制限	最大の高さと幅は40インチ、2880ポイントです。PDF はパスワードで保護できません。PDF には JPEG 2000 形式の画像を含めることができます。
ドキュメントの回転とイメージサイズ	Amazon Textract では、45 度の平面内回転など、すべての面内ドキュメントの回転がサポートされています。 Amazon Textract は、すべての面で 10000 ピクセル以下の解像度の画像をサポートしています。
テキストの整列	テキストは、ドキュメント内で水平方向に整列させることができます。Amazon Textract は、ドキュメント内の縦書きテキストの配置をサポートしていません。
言語	Amazon Textract は、英語、フランス語、ドイツ語、イタリア語、ポルトガル語、スペイン語のテキストの検出をサポートし

制限	説明
	ています。Amazon Textract は、出力で検出された言語を返しません。
文字サイズ	検出されるテキストの最小高さは 15 ピクセルです。150 DPI では、これは 8 ポイントフォントと同じになります。
文字型	Amazon Textract は、手書き文字認識と印刷文字認識の両方をサポートしています。
CHARACT	Amazon Textract は次の文字を検出します。 • 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 Û à À è È ù Ù ë Ë ï Ï ü Ü á Á é É í Í ó Ó ú Ú ü Ü ñ Ñ ì Ì ò Ò ã Ã õ Õ • ! 「# $% '& () * +,-./:; =? @ [\] ^ _ `{|} ~ > <° € ¥ ¥ ß ¿¡€ £ ¥ Ø Ø ©®™ § ¹ ² ³ '
AnalyzeID 固有の制限	AnalyzeID は、米国のパスポートと米国の運転免許証のみをサポートしています。

Amazon Textract のドキュメント履歴

次の表に、『』の各リリースにおける重要な変更点を示します。Amazon Textract 開発者ガイド。このドキュメントの更新に関する通知については、RSS フィードでサブスクライブできます。

- ドキュメントの最終更新: 2019 年 5 月 29 日

update-history-change	update-history-description	update-history-date
からのコード例を統合する AWSドキュメント SDK コード例 GitHub レポ	Amazon Textract ガイドに追加のコード例が含まれるようになりました。前の例セクションの名前を「チュートリアル」に変更しました。	2022 年 1 月 30 日
AnalyzeExpense	Amazon Textract は、Analyze Expense API を使用した請求書および領収書ドキュメントの分析をサポートするようになりました。この機能は、アジアパシフィック（ムンバイ）、アジアパシフィック（ソウル）、アジアパシフィック（シンガポール）、アジアパシフィック（シドニー）、カナダ（中部）、ヨーロッパ（フランクフルト）、ヨーロッパ（アイルランド）、ヨーロッパ（ロンドン）、米国東部（オハイオ）、米国西部（北カリフォルニア）、米国西部（オレゴン）の各リージョンでのみ利用できます。	2021 年 7 月 26 日

Augmented AI Support	Amazon Textract は Amazon Augmented AI がヒューマンレビューの実装をサポートするようになりました。	2019 年 12 月 3 日
新しいサービスとガイド	Amazon Textract が一般的に利用できるようになりました。	2019 年 5 月 29 日
選択エレメントのSupport	Amazon Textract で選択要素 (ラジオボタンとチェックボックス) を検出できるようになりました。	2019 年 4 月 24 日
Amazon Textract のリリース	これは Amazon Textract のドキュメントの最初のリリースです。	2018年11月28日

AWS 用語集

最新の AWS の用語については、AWS 全般のリファレンスの AWS 用語集を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。