aws

Panduan Developer

Amazon Textract

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Textract: Panduan Developer

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon adalah milik dari pemiliknya masing-masing, yang mungkin berafiliasi atau tidak berafiliasi dengan, terkait, atau disponsori oleh Amazon.

Table of Contents

Apa itu Amazon Textract?

Amazon Textract membuat kemudahan untuk menambahkan deteksi dan analisis teks dokumen ke aplikasi Anda. Menggunakan pelanggan Amazon Textract dapat:

- Mendeteksi teks diketik dan tulisan tangan dalam berbagai dokumen, termasuk laporan keuangan, catatan medis, dan formulir pajak.
- Ekstrak teks, formulir, dan tabel dari dokumen dengan data terstruktur, menggunakan Amazon Texact Document Analysis API.
- Proses faktur dan tanda terima dengan AnalyzeExpense API.
- Memproses dokumen ID seperti SIM dan paspor yang dikeluarkan oleh pemerintah AS, menggunakan AnalyzeID API.

Amazon Textract didasarkan pada teknologi deep learning yang terbukti, sangat dapat diskalakan, dan dikembangkan oleh para ilmuwan penglihatan komputer Amazon untuk menganalisis miliaran citra dan video setiap hari. Anda tidak perlu keahlian machine learning untuk menggunakannya. Amazon Textract menyertakan API sederhana dan mudah digunakan yang dapat menganalisis file gambar dan file PDF. Amazon Textract selalu belajar dari data baru, dan Amazon terus menambahkan fitur baru ke layanan ini.

Berikut ini adalah kasus penggunaan Amazon Textract:

- Membuat indeks pencarian cerdas— Menggunakan Amazon Textract Anda dapat membuat pustaka teks yang terdeteksi dalam file gambar dan PDF.
- Menggunakan ekstraksi teks cerdas untuk pemrosesan bahasa alami (NLP)— Amazon Textract memberi Anda kontrol atas bagaimana teks dikelompokkan sebagai masukan untuk aplikasi NLP. Hal ini dapat mengekstrak teks sebagai kata dan garis. Ini juga mengelompokkan teks berdasarkan sel tabel jika analisis tabel dokumen Amazon Textract diaktifkan.
- Mempercepat penangkapan dan normalisasi data dari berbagai sumber— Amazon Textract memungkinkan ekstraksi data teks dan tabular dari berbagai dokumen, seperti dokumen keuangan, laporan penelitian, dan catatan medis. Dengan Amazon Textract Analyze Document API, Anda dapat dengan mudah dan cepat mengekstrak data yang tidak terstruktur dan terstruktur dari dokumen Anda.
- Mengotomatisasi pengambilan data dari formulir— Amazon Textract memungkinkan data terstruktur diekstraksi dari formulir. Dengan Amazon Textract Texact Analysis API, Anda dapat

membangun kemampuan ekstraksi ke dalam alur kerja bisnis yang ada sehingga data pengguna yang dikirimkan melalui formulir dapat diekstraksi ke dalam format yang dapat digunakan.

Beberapa manfaat menggunakan Amazon Textract meliputi:

- Integrasi deteksi teks dokumen ke dalam aplikasi Anda— Amazon Textract menghilangkan kompleksitas membangun kemampuan deteksi teks ke dalam aplikasi Anda dengan membuat analisis yang kuat dan akurat tersedia dengan API sederhana. Anda tidak memerlukan visi komputer atau keahlian pembelajaran mendalam untuk menggunakan Amazon Textract untuk mendeteksi teks dokumen. Dengan Amazon Text API, Anda dapat dengan mudah membangun deteksi teks ke dalam aplikasi web, seluler, atau perangkat yang terhubung.
- Analisis dokumen yang dapat diskalakan— Amazon Textract memungkinkan Anda menganalisis dan mengekstrak data dengan cepat dari jutaan dokumen, yang dapat mempercepat pengambilan keputusan.
- Biaya rendah— Dengan Amazon Textract, Anda hanya membayar dokumen yang Anda analisis. Tidak ada biaya minimum atau pun komitment di muka Anda dapat memulai secara gratis, dan simpan lebih banyak saat Anda membutuhkannya dengan model harga bertingkat kami.

Dengan pemrosesan sinkron, Amazon Textract dapat menganalisis dokumen satu halaman untuk aplikasi yang sangat penting latensi. Amazon Textract juga menyediakan operasi asinkron untuk memperluas dukungan ke dokumen multihalaman.

Pengguna Amazon Textract Pertama Kali

Jika ini pertama kali menggunakan Amazon Textract, kami menyarankan agar Anda membaca bagian-bagian berikut secara berurutan:

1. Cara Amazon Textract Bekerja— Bagian ini memperkenalkan komponen Amazon Textract dan cara kerjanya bersama untuk pengalaman menyeluruh.
2. Memulai dengan Amazon Textract— Pada bagian ini, Anda mengatur akun Anda dan menguji Amazon Textract API.

Menggunakan Amazon Textract Texact denganAWSSDK

AWSperangkat lunak pengembangan kit (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan pengembang untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK untuk C++	AWS SDK untuk C++Contoh kode
AWS SDK untuk Go	AWS SDK untuk GoContoh kode
AWS SDK untuk Java	AWS SDK untuk JavaContoh kode
AWS SDK untuk JavaScript	AWS SDK untuk JavaScriptContoh kode
AWS SDK untuk .NET	AWS SDK untuk .NETContoh kode
AWS SDK untuk PHP	AWS SDK untuk PHPContoh kode
AWS SDK untuk Python (Boto3)	AWS SDK untuk Python (Boto3)Contoh kode
AWS SDK untuk Ruby	AWS SDK untuk RubyContoh kode

Contoh ketersediaan

Tidak dapat menemukan apa yang Anda butuhkan? Meminta contoh kode dengan menggunakanBerikan umpan baliklink di bagian bawah halaman ini.

Cara Amazon Textract Bekerja

Amazon Textract memungkinkan Anda mendeteksi dan menganalisis teks dalam dokumen input tunggal atau multihalaman (lihatDokumen Input).

Amazon Textract menyediakan operasi untuk tindakan berikut.

- Mendeteksi teks saja. Untuk informasi selengkapnya, lihatMendeteksi teks.
- Mendeteksi dan menganalisis hubungan antar teks. Untuk informasi selengkapnya, lihatMenganalisis Dokumen.
- Mendeteksi dan menganalisis teks dalam faktur dan tanda terima. Untuk informasi selengkapnya, lihatMenganalisis Faktur dan Penerimaan.
- Mendeteksi dan menganalisis teks dalam dokumen identitas pemerintah. Untuk informasi selengkapnya, lihatMenganalisis Dokumen Identitas.

Amazon Textract menyediakan operasi sinkron untuk memproses dokumen kecil, satu halaman, dan dengan respons real-time yang dekat. Untuk informasi selengkapnya, lihat Memproses Dokumen dengan Operasi Sinkron. Amazon Textract juga menyediakan operasi asinkron yang dapat Anda gunakan untuk memproses dokumen multihalaman yang lebih besar. Tanggapan asinkron tidak dalam waktu nyata. Untuk informasi selengkapnya, lihat Memproses Dokumen dengan Operasi Asynchronous.

Ketika operasi Amazon Textract memproses dokumen, hasilnya akan dikembalikan dalam arraythe section called “Block”objek atau arraythe section called “ExpenseDocument”objek. Kedua objek berisi informasi yang terdeteksi tentang item, termasuk lokasinya pada dokumen dan hubungannya dengan item lain pada dokumen. Untuk informasi selengkapnya, lihat Objek Tanggapan Amazon Textract. Untuk contoh yang menunjukkan cara menggunakan`Block`benda, lihatTutorial.

Topik

- Mendeteksi teks
- Menganalisis Dokumen
- Menganalisis Faktur dan Penerimaan
- Menganalisis Dokumen Identitas
- Dokumen Input
- Objek Tanggapan Amazon Textract

- Lokasi Item pada Halaman Dokumen

Mendeteksi teks

Amazon Textract menyediakan operasi sinkron dan asinkron yang hanya mengembalikan teks yang terdeteksi dalam dokumen. Untuk kedua set operasi, informasi berikut dikembalikan dalam beberapathe section called “Block”objek.

- Garis dan kata-kata teks yang terdeteksi
- Hubungan antara garis dan kata-kata teks yang terdeteksi
- Halaman yang terdeteksi teks muncul di
- Lokasi baris dan kata-kata teks pada halaman dokumen

Untuk informasi selengkapnya, lihat the section called “Baris dan Kata-kata Teks”.

Untuk mendeteksi teks secara serentak, gunakanDetectDocumentTextOperasi API, dan lulus file dokumen sebagai masukan. Seluruh rangkaian hasil dikembalikan oleh operasi. Untuk informasi lebih lanjut dan contoh, lihat Memproses Dokumen dengan Operasi Sinkron.

> **Note**
>
> Operasi Amazon Rekognition API`DetectText`berbeda dari`DetectDocumentText`. Anda menggunakan`DetectText`untuk mendeteksi teks dalam adegan langsung, seperti poster atau rambu jalan.

Untuk mendeteksi teks secara asinkron, gunakanStartDocumentTextDetectionuntuk mulai memproses file dokumen input. Untuk mendapatkan hasilnya, hubungiGetDocumentTextDetection. Hasilnya dikembalikan dalam satu atau lebih tanggapan dari`GetDocumentTextDetection`. Untuk informasi lebih lanjut dan contoh, lihat Memproses Dokumen dengan Operasi Asynchronous.

Menganalisis Dokumen

Amazon Textract menganalisis dokumen dan formulir untuk hubungan antara teks yang terdeteksi. Operasi analisis Amazon Textract Texact mengembalikan 3 kategori ekstraksi dokumen — teks, formulir, dan tabel. Analisis faktur dan tanda terima ditangani melalui proses yang berbeda, untuk informasi lebih lanjut lihatMenganalisis Faktur dan Penerimaan.

Ekstraksi teks

Teks mentah diekstrak dari dokumen. Untuk informasi selengkapnya, lihatBaris dan kata-kata teks.

Ekstraksi formulir

Data formulir terkait dengan item teks yang diekstrak dari dokumen. Amazon Textract mewakili data formulir sebagai pasangan nilai kunci. Pada contoh berikut, salah satu baris teks yang terdeteksi oleh Amazon Textract adalahNama: Doe. Amazon Textract juga mengidentifikasi kunci (Nama:) dan nilai (Doe). Untuk informasi selengkapnya, lihatData formulir (Pasangan kunci/nilai).

Nama: Doe

Alamat: 123 Any Street, Anytown, Amerika Serikat

Tanggal lahir: 12-26-1980

Pasangan kunci-nilai juga digunakan untuk mewakili kotak centang atau tombol opsi (tombol radio) yang diekstraksi dari bentuk.

Laki-Laki:☑

Untuk informasi selengkapnya, lihatElemen seleksi.

Ekstraksi Tabel

Amazon Textract dapat mengekstrak tabel, sel tabel, dan item dalam sel tabel dan dapat diprogram untuk mengembalikan hasil dalam file JSON, .csv, atau .txt.

Nama	Alamat
Carolina	123 Kota mana pun

Untuk informasi selengkapnya, lihatTabel. Elemen seleksi juga bisa diekstraksi dari tabel. Untuk informasi selengkapnya, lihatElemen seleksi.

Untuk item yang dianalisis, Amazon Textract mengembalikan yang berikut dalam beberapathe section called “Block”Objek:

- Garis dan kata-kata teks yang terdeteksi
- Isi item yang terdeteksi

- Hubungan antara item yang terdeteksi
- Halaman yang item terdeteksi pada
- Lokasi item pada halaman dokumen

Anda dapat menggunakan operasi sinkron atau asinkron untuk menganalisis teks dalam dokumen. Untuk menganalisis teks secara serentak, gunakanAnalyzeDocumentoperasi, dan lulus dokumen sebagai masukan.`AnalyzeDocument`mengembalikan seluruh rangkaian hasil. Untuk informasi selengkapnya, lihat Menganalisis Teks Dokumen dengan Amazon Textract.

Untuk mendeteksi teks secara asinkron, gunakanStartDocumentAnalysisuntuk memulai pemrosesan. Untuk mendapatkan hasilnya, hubungiGetDocumentAnalysis. Hasilnya dikembalikan dalam satu atau lebih tanggapan dari`GetDocumentAnalysis`. Untuk informasi lebih lanjut dan contoh, lihat Mendeteksi atau Menganalisis Teks dalam Dokumen Multipage.

Untuk menentukan jenis analisis yang akan dilakukan, Anda dapat menggunakan`FeatureTypes`daftar parameter masukan. Tambahkan TABLES ke daftar untuk mengembalikan informasi tentang tabel yang terdeteksi dalam dokumen input — misalnya, sel tabel, teks sel, dan elemen seleksi dalam sel. Tambahkan FORM untuk mengembalikan hubungan kata, seperti pasangan kunci-nilai dan elemen seleksi. Untuk melakukan kedua jenis analisis, tambahkan TABLES dan FORMS ke`FeatureTypes`.

Semua baris dan kata-kata yang terdeteksi dalam dokumen termasuk dalam respon (termasuk teks yang tidak terkait dengan nilai`FeatureTypes`).

Menganalisis Faktur dan Penerimaan

Amazon Textract mengekstrak data yang relevan seperti informasi kontak, item yang dibeli, dan nama vendor, dari hampir semua faktur atau tanda terima tanpa memerlukan templat atau konfigurasi apa pun. Faktur dan tanda terima sering menggunakan berbagai tata letak, sehingga sulit dan memakan waktu untuk mengekstrak data secara manual dalam skala besar. Amazon Textract menggunakan ML-nya untuk memahami konteks faktur dan tanda terima serta secara otomatis mengekstrak data seperti tanggal faktur atau tanda terima, nomor faktur atau tanda terima, harga barang, jumlah total, dan persyaratan pembayaran yang sesuai dengan kebutuhan bisnis Anda.

Amazon Textract juga mengidentifikasi nama vendor yang sangat penting untuk alur kerja Anda tetapi mungkin tidak diberi label secara eksplisit. Misalnya, Amazon Textract dapat menemukan nama vendor pada tanda terima meskipun hanya ditunjukkan dalam logo di bagian atas halaman

tanpa kombinasi pasangan nilai kunci eksplisit. Amazon Textract juga memudahkan Anda untuk mengkonsolidasikan masukan dari beragam tanda terima dan faktur yang menggunakan kata berbeda untuk konsep yang sama. Misalnya, Amazon Textract memetakan hubungan antara nama lapangan dalam dokumen yang berbeda seperti nomor pelanggan, nomor pelanggan, dan ID akun, menampilkan taksonomi standar sebagai`INVOICE_RECEIPT_ID`. Dalam hal ini, Amazon Textract mewakili data secara konsisten di berbagai jenis dokumen. Bidang yang tidak sejajar dengan taksonomi standar dikategorikan sebagai`OTHER`.

Berikut ini adalah daftar bidang standar yang AnalyzeExpense saat ini mendukung:

- Nama Vendor:`VENDOR_NAME`
- Total:`TOTAL`
- Alamat Penerima:`RECEIVER_ADDRESS`
- Tanggal Faktur/Tanda Terima:`INVOICE_RECEIPT_DATE`
- Faktur/Tanda Terima ID:`INVOICE_RECEIPT_ID`
- Ketentuan Pembayaran:`PAYMENT_TERMS`
- Subtotal:`SUBTOTAL`
- Tanggal jatuh tempo:`DUE_DATE`
- Pajak:`TAX`
- ID Pembayar Pajak Faktur (SSN/ITIN atau EIN):`TAX_PAYER_ID`
- Nama Item:`ITEM_NAME`
- Harga Item:`PRICE`
- Kuantitas Item:`QUANTITY`

AnalyzeExpense API mengembalikan elemen berikut untuk halaman dokumen yang diberikan:

- Jumlah penerimaan atau faktur dalam halaman diwakili sebagai`ExpenseIndex`
- Nama standar untuk bidang individu direpresentasikan sebagai`Type`
- Nama sebenarnya dari bidang seperti yang muncul pada dokumen, direpresentasikan sebagai`LabelDetection`
- Nilai bidang yang sesuai direpresentasikan sebagai`ValueDetection`
- Jumlah halaman dalam dokumen yang diajukan direpresentasikan sebagai`Pages`
- Nomor halaman di mana bidang, nilai, atau item baris terdeteksi, direpresentasikan sebagai`PageNumber`

- Geometri, yang mencakup kotak pembatas dan mengkoordinasikan lokasi bidang individu, nilai, atau item baris pada halaman, direpresentasikan sebagai`Geometry`
- Skor kepercayaan yang terkait dengan setiap bagian dari data yang terdeteksi pada dokumen, direpresentasikan sebagai`Confidence`
- Seluruh baris item baris individu yang dibeli, direpresentasikan sebagai`EXPENSE_ROW`

Berikut ini adalah sebagian dari output API untuk tanda terima yang diproses oleh AnalyzeExpense yang menunjukkan Total: $55.64 dalam dokumen yang diekstrak sebagai bidang standar`TOTAL`, teks aktual pada dokumen sebagai “Total”, Keyakinan Skor “97.1”, Halaman Nomor “1", Nilai total sebagai “$55.64" dan kotak pembatas dan koordinat poligon:


```
{
    "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
    }
```


Anda dapat menggunakan operasi sinkron untuk menganalisis faktur atau tanda terima. Untuk menganalisis dokumen-dokumen ini, Anda menggunakan operasi AnalyzeExpense dan memberikan tanda terima atau faktur untuk itu.`AnalyzeExpense`mengembalikan seluruh rangkaian hasil. Untuk informasi selengkapnya, lihat Menganalisis Faktur dan Penerimaan dengan Amazon Textract.

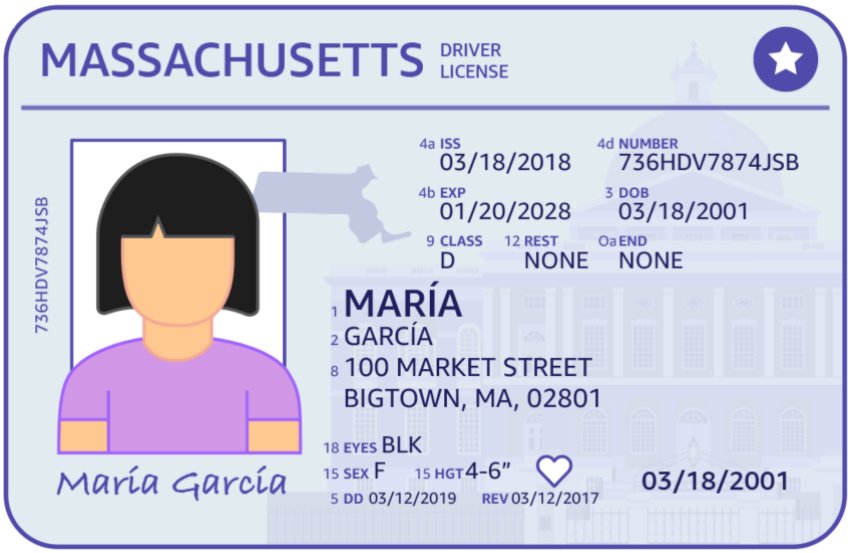
Untuk menganalisis faktur dan tanda terima secara asinkron, gunakanStartExpenseAnalysisuntuk mulai memproses file dokumen input. Untuk mendapatkan hasilnya, hubungiGetExpenseAnalysis. Hasil untuk panggilan yang diberikan keStartExpenseAnalysisdikembalikan oleh`GetExpenseAnalysis`. Untuk informasi lebih lanjut dan contoh, lihat Memproses Dokumen dengan Operasi Asynchronous.

Menganalisis Dokumen Identitas

Amazon Textract dapat mengekstrak informasi yang relevan dari paspor, SIM, dan dokumentasi identitas lainnya yang dikeluarkan oleh Pemerintah AS menggunakan AnalyzeID API. Dengan Analyze ID, bisnis dapat dengan cepat, dan akurat mengekstrak informasi dari ID seperti SIM AS, ID negara, dan paspor yang memiliki template atau format yang berbeda. AnalyzeID API mengembalikan dua kategori tipe data:

- Pasangan nilai kunci tersedia pada ID seperti Tanggal Lahir, Tanggal Terbitan, ID #, Kelas, dan Pembatasan.
- Bidang tersirat pada dokumen yang mungkin tidak memiliki kunci eksplisit yang terkait dengannya seperti Nama, Alamat, dan Dikeluarkan Oleh.

Nama kunci distandarisasi dalam respon. Misalnya, jika SIM Anda mengatakan LIC # (nomor lisensi) dan paspor mengatakan Paspor No, Analyze ID response akan mengembalikan kunci standar sebagai “ID Dokumen” bersama dengan kunci mentah (misalnya LIC #). Standarisasi ini memungkinkan pelanggan dengan mudah menggabungkan informasi di banyak ID yang menggunakan istilah yang berbeda untuk konsep yang sama.



Menganalisis ID mengembalikan informasi dalam struktur yang disebut`IdentityDocumentFields`. Ini adalah`JSON`struktur yang berisi dua buah informasi: Jenis dinormalisasi dan Nilai yang terkait dengan Type. Keduanya juga memiliki skor kepercayaan diri. Untuk informasi selengkapnya, lihat Identitas Dokumentasi Respon Ob.

Anda dapat menggunakan operasi sinkron untuk menganalisis SIM atau paspor. Untuk menganalisis dokumen-dokumen ini, Anda menggunakan operasi AnalyzeID dan meneruskan dokumen identitas ke dalamnya.`AnalyzeID`mengembalikan seluruh rangkaian hasil. Untuk informasi selengkapnya, lihat Menganalisis Dokumentasi Identitas dengan Amazon Textract.

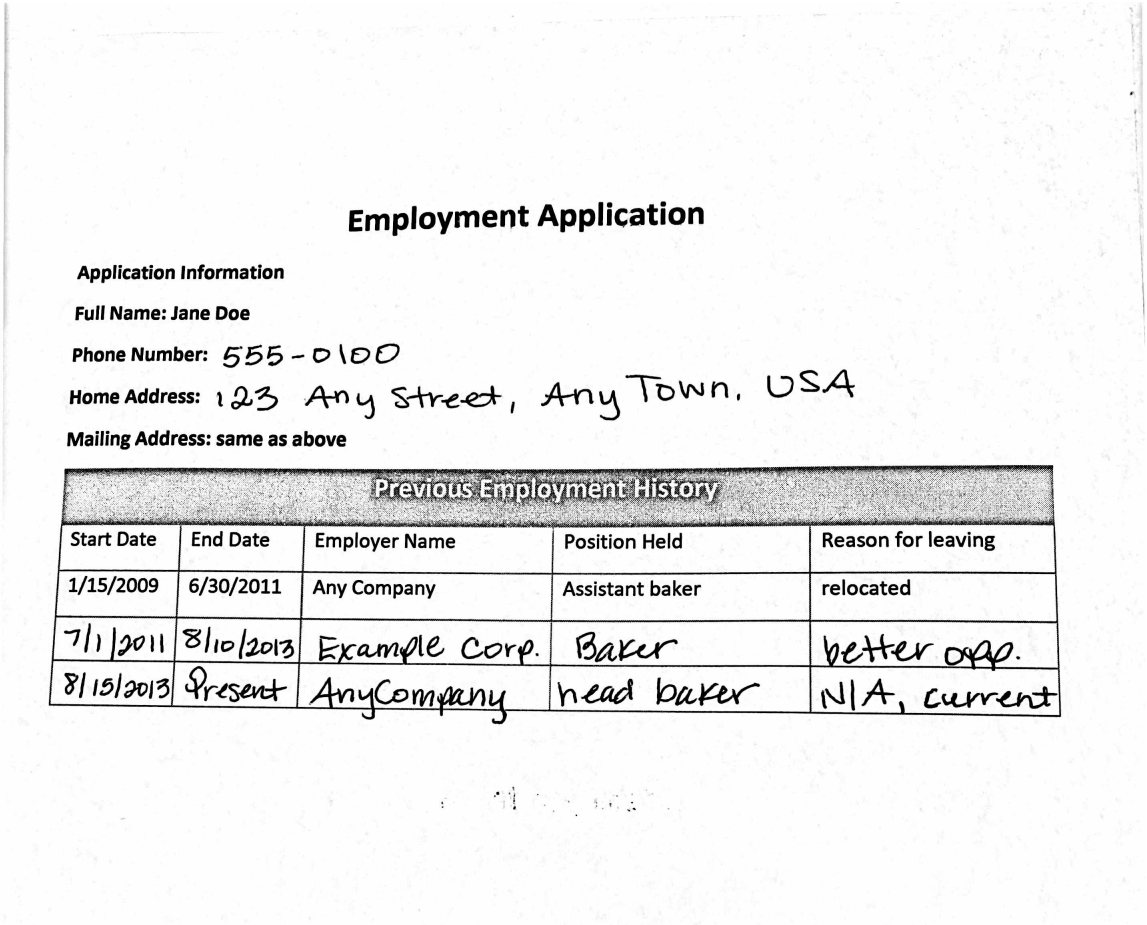
> **Note**
>
> Beberapa dokumen identitas, seperti SIM, memiliki dua sisi. Anda dapat melewati gambar depan dan belakang lisensi driver sebagai gambar terpisah dalam permintaan Analyze ID API yang sama.

Dokumen Input

Input yang sesuai untuk operasi Amazon Textract adalah dokumen tunggal atau multipage. Beberapa contoh adalah dokumen hukum, formulir, ID, atau surat. Formulir adalah dokumen dengan pertanyaan atau petunjuk bagi pengguna untuk memberikan jawaban. Beberapa contoh adalah formulir pendaftaran pasien, formulir pajak, atau formulir klaim asuransi.

Dokumen dapat dalam format JPEG, PNG, PDF atau TIFF. Dengan file format PDF dan TIFF, Anda dapat memproses dokumen multipage. Untuk informasi tentang cara Amazon Textract mewakili dokumen sebagai`Block`benda, lihatDeteksi Teks dan Dokumen Analisis Respon Objek.

Berikut ini adalah contoh dokumen input yang dapat diterima.



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

Untuk informasi tentang batas dokumen, lihatBatas Keras di Amazon Textract.

Untuk operasi sinkron Amazon Textract, Anda dapat menggunakan dokumen input yang disimpan di bucket Amazon S3, atau Anda dapat meneruskan byte citra yang dikodekan base64. Untuk informasi selengkapnya, lihat Memanggil Operasi Sinkronisasi Amazon Texact. Untuk operasi asinkron, Anda perlu memasok dokumen input di bucket Amazon S3. Untuk informasi selengkapnya, lihat Memanggil Operasi Asinkron Amazon Texact.

Objek Tanggapan Amazon Textract

Operasi Amazon Textract Texact mengembalikan berbagai jenis objek tergantung pada operasi yang dijalankan. Untuk mendeteksi teks, dan menganalisis dokumen generik, operasi mengembalikan objek Block. Untuk menganalisis faktur atau tanda terima, operasi mengembalikan objek expenseDocuments. Untuk menganalisis dokumentasi identitas, operasi mengembalikan objek IdentityDocumentFields. Untuk informasi selengkapnya tentang objek respons ini, lihat bagian berikut:

Topik

- Deteksi Teks dan Dokumen Analisis Respon Objek
- Faktur dan Respon Tanda Terima Objek
- Identitas Dokumentasi Respon Ob

Deteksi Teks dan Dokumen Analisis Respon Objek

Saat Amazon Textract memproses dokumen, dokumen akan membuat daftarBlockobjek untuk teks terdeteksi atau dianalisis. Setiap blok berisi informasi tentang item yang terdeteksi, di mana letaknya, dan keyakinan yang dimiliki Amazon Textract dalam keakuratan pemrosesan.

Sebuah dokumen terdiri dari jenis berikut`Block`objek.

- Halaman
- Baris dan kata-kata teks
- Data Formulir (Pasangan kunci/nilai)
- Tabel dan Sel
- Elemen seleksi

Isi blok tergantung pada operasi yang Anda panggil. Jika Anda memanggil salah satu operasi deteksi teks, halaman, baris, dan kata-kata teks yang terdeteksi akan dikembalikan. Untuk informasi selengkapnya, lihat Mendeteksi teks. Jika Anda memanggil salah satu operasi analisis dokumen, informasi tentang halaman terdeteksi, pasangan kunci-nilai, tabel, elemen seleksi, dan teks dikembalikan. Untuk informasi selengkapnya, lihat Menganalisis Dokumen.

Beberapa`Block`bidang objek umum untuk kedua jenis pengolahan. Misalnya, setiap blok memiliki pengenal yang unik.

Untuk contoh yang menunjukkan cara menggunakan`Block`benda, lihatTutorial.

Tata Letak Dokumen

Amazon Textract mengembalikan representasi dokumen sebagai daftar berbagai jenis`Block`objek yang terkait dalam hubungan orangtua-ke-anak atau pasangan kunci-nilai. Metadata yang menyediakan jumlah halaman dalam dokumen juga dikembalikan. Berikut ini adalah JSON untuk khas`Block`objek tipe`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
    }
}
```

Dokumen dibuat dari satu atau lebihPAGEblok. Setiap halaman berisi daftar blok anak untuk item utama yang terdeteksi pada halaman, seperti baris teks dan tabel. Untuk informasi selengkapnya, lihat Halaman.

Anda dapat menentukan jenisBlockobjek dengan memeriksaBlockTypeBidang.

SEBUAHBlockobjek berisi daftar terkaitBlockbenda-benda diRelationshipslapangan, yang merupakan arrayRelationshipobjek. SEBUAHRelationshipsarray adalah salah satu dari jenis ANAK atau jenis NILAI. Array jenis ANAK digunakan untuk daftar item yang anak-anak dari blok saat ini. Misalnya, jika blok saat ini adalah tipe LINE,Relationshipsberisi daftar ID untuk blok WORD yang membentuk baris teks. Array tipe VALUE digunakan untuk mengandung pasangan nilai kunci. Anda dapat menentukan jenis hubungan dengan memeriksaTypebidangRelationshipobjek.

Blok anak tidak memiliki informasi tentang objek Blokir induknya.

Untuk contoh yang menunjukkanBlockinformasi, lihatMemproses Dokumen dengan Operasi Sinkron.

Kepercayaan

Operasi Amazon Textract mengembalikan kepercayaan persentase yang dimiliki Amazon Textract dalam keakuratan item yang terdeteksi. Untuk mendapatkan kepercayaan diri, gunakanConfidencebidangBlockobjek. Nilai yang lebih tinggi menunjukkan kepercayaan yang lebih tinggi. Tergantung pada skenario, deteksi dengan kepercayaan rendah mungkin memerlukan konfirmasi visual oleh manusia.

Geometry

Operasi Amazon Textract Texact, dengan pengecualian analisis identitas, mengembalikan informasi lokasi tentang lokasi item yang terdeteksi pada halaman dokumen. Untuk mendapatkan lokasi, gunakanGeometrybidangBlockobjek. Untuk informasi selengkapnya, lihatLokasi Item pada Halaman Dokumen

Halaman

Sebuah dokumen terdiri dari satu atau beberapa halaman. SEBUAHthe section called “Block”objek tipePAGEada untuk setiap halaman dokumen. SEBUAHPAGEblok objek berisi daftar ID anak untuk baris teks, pasangan kunci-nilai, dan tabel yang terdeteksi pada halaman dokumen.

JSON untukPAGEterlihat seperti berikut ini.

```
{

    "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
},
```

Jika Anda menggunakan operasi asinkron dengan dokumen multipage yang ada dalam format PDF, Anda dapat menentukan halaman tempat blok berada di dengan memeriksaPagebidangBlockobjek. Gambar yang dipindai (gambar dalam format JPEG, PNG, PDF, atau TIFF) dianggap sebagai dokumen satu halaman, bahkan jika ada lebih dari satu halaman dokumen pada gambar. Operasi asinkron selalu mengembalikanPagenilai 1 untuk gambar yang dipindai.

Jumlah total halaman dikembalikan dalamPagesbidangDocumentMetadata.DocumentMetadatadikembalikan dengan setiap daftarBlockobjek yang dikembalikan oleh operasi Amazon Textract.

Baris dan Kata-kata Teks

Teks yang terdeteksi yang dikembalikan oleh operasi Amazon Textract Texact dikembalikan dalam daftarthe section called “Block”objek. Benda-benda ini mewakili baris teks atau kata-kata tekstual

yang terdeteksi pada halaman dokumen. Teks berikut menunjukkan dua baris teks yang dibuat dari beberapa kata.

Ini adalah teks.

Dalam dua baris terpisah.

Teks yang terdeteksi dikembalikan dalam`Text`bidang a`Block`objek. Parameter`BlockType`bidang menentukan apakah teks adalah baris teks (LINE) atau kata (WORD). SEBUAHKATAadalah satu atau lebih karakter skrip Latin dasar ISO yang tidak dipisahkan oleh spasi. SEBUAHLINIadalah string kata-kata tab-delimited dan bersebelahan.

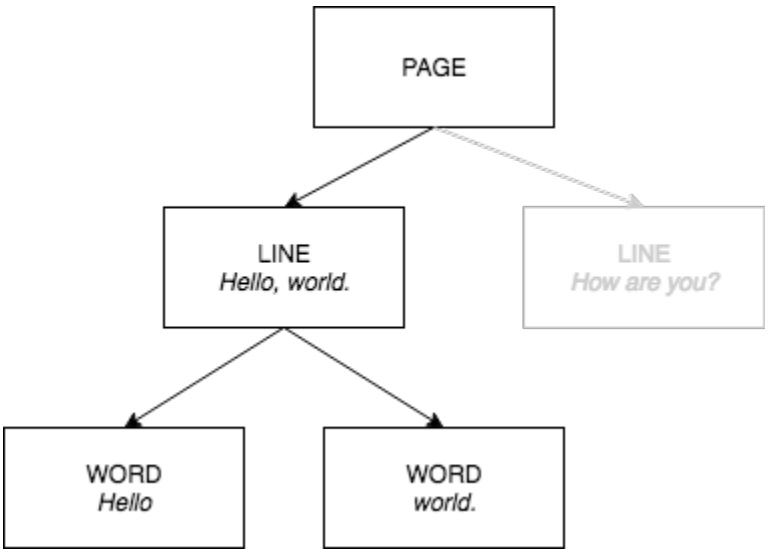
Selain itu, Amazon Textract akan menentukan apakah sepotong teks ditulis tangan atau dicetak menggunakan`TextTypes`Bidang. Ini kembali sebagai TANGAN TANGAN dan DICETAKAN masing-masing.

Yang lainnya`Block`properti umum untuk semua jenis blok, seperti ID, kepercayaan diri, dan informasi geometri. Untuk informasi selengkapnya, lihat the section called “Deteksi Teks dan Dokumen Analisis Respon Objek”.

Untuk mendeteksi hanya baris dan kata-kata, Anda dapat menggunakanDetectDocumentTextatauStartDocumentTextDetection. Untuk informasi selengkapnya, lihat Mendeteksi teks. Untuk mendapatkan teks yang terdeteksi (baris dan kata-kata) dan informasi tentang bagaimana hal itu berhubungan dengan bagian lain dari dokumen, seperti tabel, Anda dapat menggunakanAnalyzeDocumentatauStartDocumentAnalysis. Untuk informasi selengkapnya, lihat Menganalisis Dokumen.

`PAGE`,`LINE`, dan`WORD`blok terkait satu sama lain dalam hubungan orang tua-ke-anak. SEBUAH`PAGE`blok adalah induk untuk semua`LINE`blok objek pada halaman dokumen. Karena GARIS dapat memiliki satu atau beberapa kata, maka`Relationships`array untuk blok LINE menyimpan ID untuk blok WORD anak yang membentuk baris teks.

Diagram berikut menunjukkan bagaimana garisHalo, dunia.dalam teksHalo, dunia. Bagaimana kabarmu?diwakili oleh`Block`objek.



Berikut ini adalah output JSON dari`DetectDocumentText`ketika kalimat`Halo, dunia. Bagaimana kabarmu?`terdeteksi. Contoh pertama adalah JSON untuk halaman dokumen. Perhatikan bagaimana ID ANAK memungkinkan Anda untuk menavigasi dokumen.

```
{
    "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"
},
```

Berikut ini adalah JSON untuk blok LINE yang membentuk baris “Hello, World”:

```
{
    "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"
},
```

Berikut ini adalah JSON untuk blok WORD untuk kataHalo,:

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

JSON terakhir adalah blok WORD untuk katadunia.:

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

Data Formulir (Pasangan Nilai Kunci)

Amazon Textract dapat mengekstrak data formulir dari dokumen sebagai pasangan nilai kunci. Misalnya, dalam teks berikut, Amazon Textract dapat mengidentifikasi kunci (Nama:) dan nilai (Carolina).

Nama: Carolina

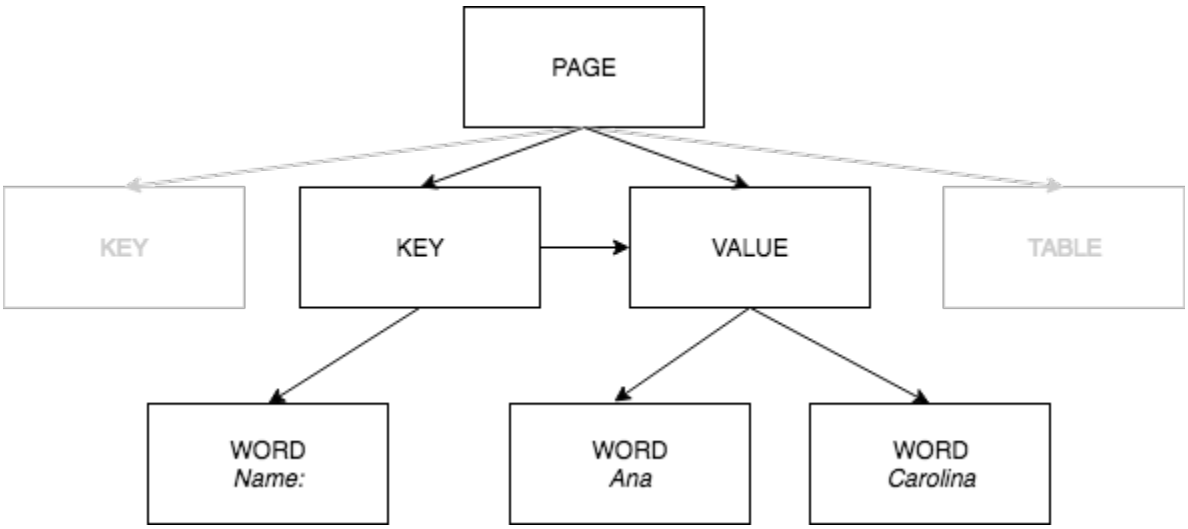
Pasangan nilai kunci dikembalikan sebagaiBlockobjek dalam tanggapan dariAnalyzeDocumentdanGetDocumentAnalysis. Anda dapat menggunakanFeatureTypesparameter masukan untuk mengambil informasi tentang pasangan kunci-nilai, tabel, atau keduanya. Untuk pasangan kunci-nilai saja, gunakan nilaiFORMS. Sebagai contoh, lihat Mengekstrak Pasangan Ky-Value dari Dokumen Formulir. Untuk informasi umum tentang bagaimana dokumen diwakili olehBlockbenda, lihatDeteksi Teks dan Dokumen Analisis Respon Objek.

Blokir objek dengan tipe KEY_VALUE_SET adalah wadah untuk objek KEY atau VALUE Block yang menyimpan informasi tentang item teks terkait yang terdeteksi dalam dokumen. Anda dapat menggunakanEntityTypeatribut untuk menentukan apakah blok adalah KEY atau NILAI.

- SEBUAHKUNCIobjek berisi informasi tentang kunci untuk teks terkait. Misalnya,Nama:. Sebuah blok KEY memiliki dua daftar hubungan. Sebuah hubungan jenis VALUE adalah daftar yang berisi ID dari blok VALUE terkait dengan kunci. Hubungan tipe ANAK adalah daftar ID untuk blok WORD yang membentuk teks kunci.
- SEBUAHNILAIobjek berisi informasi tentang teks yang terkait dengan kunci. Dalam contoh sebelumnya,Carolinaadalah nilai untuk kunciNama:. Sebuah blok NILAI memiliki hubungan dengan daftar blok ANAK yang mengidentifikasi blok WORD. Setiap blok WORD berisi salah satu kata yang membentuk teks nilai. SEBUAHVALUEobjek juga dapat berisi informasi tentang elemen yang dipilih. Untuk informasi selengkapnya, lihat Elemen Seleksi.

Setiap contoh dari KEY_VALUE_SETBlockobjek adalah anak dari PAGEBlockobjek yang sesuai dengan halaman saat ini.

Diagram berikut menunjukkan bagaimana pasangan nilai kunciNama: Carolinadiwakili olehBlockobjek.



Contoh berikut menunjukkan bagaimana pasangan nilai kunciNama: Carolinadiwakili oleh JSON.

Blok PAGE memiliki blok ANAK tipeKEY_VALUE_SETuntuk setiap KEY dan blok NILAI terdeteksi dalam dokumen.

```
{
    "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
},
```

JSON berikut menunjukkan bahwa blok KEY (52be1777-53f7-42f6-a7cf-6d09bdc15a30) memiliki hubungan dengan blok NILAI (7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c). Ini juga memiliki blok ANAK untuk blok WORD (c734fca6-c4c4-415c-b6c1-30f7510b72ea) yang berisi teks untuk kunci (Nama:).

```
{
```

```
    "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
},
```

JSON berikut menunjukkan bahwa VALUE blok 7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c memiliki daftar ANAK ID untuk blok WORD yang membentuk teks dari nilai (AnadanCarolina).

```
{
    "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
```

```
}
```

JSON berikut menunjukkan`Block`objek untuk kata-kataNama:,Ana, danCarolina.

```
{
    "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"
},
```

Tabel

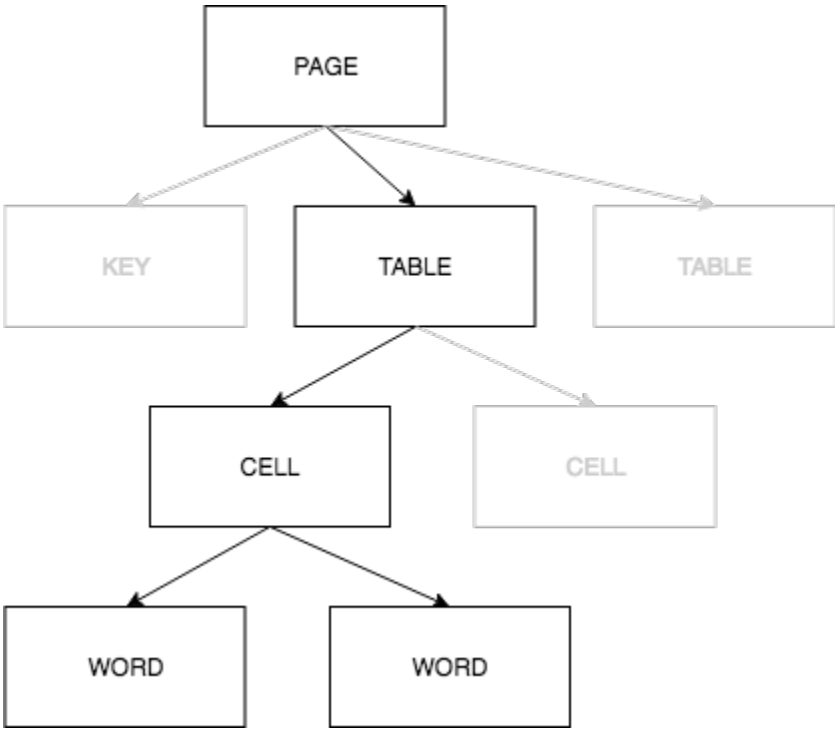
Amazon Textract dapat mengekstrak tabel dan sel dalam tabel. Misalnya, ketika tabel berikut terdeteksi pada formulir, Amazon Textract mendeteksi tabel dengan empat sel.

Nama	Alamat
Carolina	123 Kota mana pun

Tabel yang terdeteksi dikembalikan sebagaiBlockobjek dalam tanggapan dariAnalyzeDocumentdanGetDocumentAnalysis. Anda dapat

menggunakan`FeatureTypes`parameter masukan untuk mengambil informasi tentang pasangan kunci-nilai, tabel, atau keduanya. Untuk tabel saja, gunakan nilainya`TABLES`. Sebagai contoh, lihat Mengekspor Tabel ke File CSV. Untuk informasi umum tentang bagaimana dokumen diwakili oleh`Block`benda, lihatDeteksi Teks dan Dokumen Analisis Respon Objek.

Diagram berikut menunjukkan bagaimana sel tunggal dalam tabel diwakili oleh`Block`objek.



Sel mengandung`WORD`blok untuk kata-kata yang terdeteksi, dan`SELECTION_ELEMENT`blok untuk elemen seleksi seperti kotak centang.

Berikut ini adalah JSON partial untuk tabel sebelumnya, yang memiliki empat sel.

Objek PAGE Block memiliki daftar ID Blok ANAK untuk blok TABLE dan setiap LINE teks yang terdeteksi.

```
{
    "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"
},
```

Blok TABLE mencakup daftar ID anak untuk sel-sel dalam tabel. Sebuah blok TABLE juga mencakup informasi geometri untuk lokasi tabel dalam dokumen. JSON berikut menunjukkan bahwa tabel memiliki empat sel, yang tercantum dalam`Ids`array.

```
{
    "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"
},
```

Jenis Blok untuk sel tabel adalah CELL. Parameter`Block`objek untuk setiap sel mencakup informasi tentang lokasi sel dibandingkan dengan sel-sel lain dalam tabel. Ini juga mencakup informasi geometri untuk lokasi sel pada dokumen. Dalam contoh sebelumnya,`505e9581-0d1c-42fb-a214-6ff736822e8c`adalah ID anak untuk sel yang berisi kataNama. Contoh berikut adalah informasi untuk sel.

```
{
    "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"
},
```

Setiap sel memiliki lokasi di meja, dengan sel pertama menjadi 1,1. Pada contoh sebelumnya, sel dengan nilaiNamaadalah di baris 1, kolom 1. Sel dengan nilai123 Kota mana punadalah di baris 2, kolom 2. Sebuah objek blok sel berisi informasi ini di`RowIndex`dan`ColumnIndex`bidang. Daftar anak berisi ID untuk objek WORD Block yang berisi teks yang ada di dalam sel. Kata-kata dalam daftar berada dalam urutan di mana mereka terdeteksi, dari kiri atas sel ke kanan bawah sel. Pada contoh sebelumnya, sel memiliki ID anak dengan nilai e9108c8e-0167-4482-989e-8b6cd3c3653e. Output berikut adalah untuk Blok WORD dengan nilai ID e9108c8e-0167-4482-989e-8b6cd3c3653e:

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

Elemen Seleksi

Amazon Textract dapat mendeteksi elemen pilihan seperti tombol opsi (tombol radio) dan kotak centang pada halaman dokumen. elemen seleksi dapat dideteksi didata formulirdan ditabel. Misalnya, ketika tabel berikut terdeteksi pada formulir, Amazon Textract mendeteksi kotak centang di sel tabel.

	Setuju	Netral	Tidak Setuju
Pelayanan yang Baik	☑	☐	☐

Mudah digunakan	☐	☑	☐
Harga yang adil	☑	☐	☐

elemen seleksi terdeteksi dikembalikan sebagaiBlockobjek dalam tanggapan dariAnalyzeDocumentdanGetDocumentAnalysis.

> **Note**
>
> Anda dapat menggunakanFeatureTypesparameter masukan untuk mengambil informasi tentang pasangan kunci-nilai, tabel, atau keduanya. Misalnya, jika Anda memfilter pada tabel, respon mencakup elemen seleksi yang terdeteksi dalam tabel. Elemen seleksi yang terdeteksi dalam pasangan kunci-nilai tidak termasuk dalam respon.

Informasi tentang elemen seleksi terkandung dalamBlockobjek tipeSELECTION_ELEMENT. Untuk menentukan status elemen yang dapat dipilih, gunakanSelectionStatusbidangSELECTION_ELEMENTblok. Status dapat berupaTERPILIHatauNOT_SELECTED. Misalnya, nilaiSelectionStatusuntuk gambar sebelumnya adalahTERPILIH.

SEBUAHSELECTION_ELEMENT Blockobjek terkait dengan pasangan nilai kunci atau sel tabel. SEBUAHSELECTION_ELEMENT Blockobjek berisi informasi kotak pembatas untuk elemen seleksi diGeometryBidang. SEBUAHSELECTION_ELEMENT Blockobjek bukan anakPAGE Blockobjek.

Data Formulir (Pasangan Nilai Kunci)

Pasangan kunci-nilai digunakan untuk mewakili elemen seleksi yang terdeteksi pada formulir. ParameterKEYblok berisi teks untuk elemen seleksi. ParameterVALUEblok berisi blok SELECTION_ELEMENT. Diagram berikut menunjukkan bagaimana elemen seleksi diwakili olehthe section called “Block”objek.

Untuk informasi selengkapnya tentang pasangan nilai kunci, lihatData Formulir (Pasangan Nilai Kunci).

Cuplikan JSON berikut menunjukkan kunci untuk pasangan kunci-nilai yang berisi elemen seleksi (Laki-Laki ☑). ID anak (Id bd14cfd5-9005-498b-a7f3-45ceb171f0ff) adalah ID dari blok WORD yang

berisi teks untuk elemen seleksi (jantan). ID nilai (Id 24aaac7f-fcce-49c7-a4f0-3688b05586d4) adalah IDVALUEblok yang berisi`SELECTION_ELEMENT`objek blok.

```
{
    "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"
}
```

Potongan JSON berikut adalah blok WORD untuk kataLaki-laki. Blok WORD juga memiliki blok LINE induk.

```
{
    "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"
},
```

Blok VALUE memiliki anak (Id f2f5e8cd-e73a-4e99-a095-053acd3b6bfb) yang merupakan blok SELECTION_ELEMENT.

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "f2f5e8cd-e73a-4e99-a095-053acd3b6bfb"  // Selection element
            ]
        }
    ],
    "Confidence": 94.15619659423828,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.017281491309404373,
            "Top": 0.07643391191959381,
            "Left": 0.2271782010793686,
            "Height": 0.026274094358086586
        },
        "Polygon": [
            {
                "Y": 0.07643391191959381,
                "X": 0.2271782010793686
            },
            {
                "Y": 0.07643391191959381,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.2271782010793686
            }
        ]
    },
    "BlockType": "KEY_VALUE_SET",
```

```
    "EntityTypes": [
        "VALUE"
    ],
    "Id": "24aaac7f-fcce-49c7-a4f0-3688b05586d4"
},
}
```

JSON berikut adalah blok SELECTION_ELEMENT. Nilai dari`SelectionStatus`menunjukkan bahwa kotak centang dipilih.

```
{
    "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"
}
```

Sel tabel

Amazon Textract dapat mendeteksi elemen seleksi di dalam sel tabel. Misalnya, sel-sel dalam tabel berikut memiliki kotak centang.

	Setuju	Netral	Tidak Setuju
Pelayanan yang Baik	☑	☐	☐
Mudah digunakan	☐	☑	☐
Harga yang adil	☑	☐	☐

SEBUAH`CELL`blok dapat berisi anak`SELECTION_ELEMENT`objek untuk elemen seleksi, serta anak`WORD`blok untuk teks terdeteksi.

Untuk informasi selengkapnya, lihatTabel.

TABLE`Block`objek untuk tabel sebelumnya terlihat mirip dengan ini.

```
{
    "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"
}
```

CELBL0CKobjek (Id c63ad40d-5a14-4646-a8df-2d4304213dbc) untuk sel yang berisi kotak centangPelayanan yang Baikterlihat seperti berikut ini. Ini termasuk seorang anak`Block`(Id = 26d122fd-c5f4-4b53-92c4-0ae92730ee1e) itu adalah`SELECTI0N_ELEMENT` `Block`untuk kotak centang.

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

SELECTION_ELEMENT`Block`objek untuk kotak centang adalah sebagai berikut. Nilai dari`SelectionStatus`menunjukkan bahwa kotak centang dipilih.

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

```
}
```

Faktur dan Respon Tanda Terima Objek

Ketika Anda mengirimkan faktur atau tanda terima ke AnalyzeExpense API, ia mengembalikan serangkaian objek ExpenseDocuments. Setiap expenseDocument selanjutnya dipisahkan menjadi`LineItemGroups`dan`SummaryFields`. Sebagian besar faktur dan tanda terima berisi informasi seperti nama vendor, nomor tanda terima, tanggal penerimaan, atau jumlah total. AnalyzeExpense mengembalikan informasi ini di bawah`SummaryFields`. Tanda terima dan faktur juga berisi rincian tentang barang yang dibeli. AnalyzeExpense API mengembalikan informasi ini di bawah`LineItemGroups`. Parameter`ExpenseIndex`lapangan unik mengidentifikasi biaya, dan mengaitkan yang sesuai`SummaryFields`dan`LineItemGroups`terdeteksi dalam biaya itu.

Tingkat data yang paling terperinci dalam respons AnalyzeExpense terdiri dari`Type`,`ValueDetection`, dan`LabelDetection`(Opsional). Entitas individu adalah:

- Tipe: Mengacu pada jenis informasi apa yang terdeteksi pada tingkat tinggi.
- LabelDetection: Mengacu pada label nilai terkait dalam teks dokumen.`LabelDetection`adalah opsional dan hanya dikembalikan jika label ditulis.
- ValueDetection: Mengacu pada nilai label atau jenis dikembalikan.

AnalyzeExpense API juga mendeteksi`ITEM`,`QUANTITY`, dan`PRICE`dalam item baris sebagai bidang dinormalisasi. Jika ada teks lain dalam item baris pada gambar tanda terima seperti SKU atau deskripsi rinci, maka akan dimasukkan dalam JSON sebagai`EXPENSE_ROW`seperti yang ditunjukkan pada contoh di bawah ini:

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

Contoh di atas menunjukkan bagaimana AnalyzeExpense API mengembalikan seluruh baris pada tanda terima yang berisi informasi item baris tentang 5 pisang dijual seharga $2.5.

Tipe

Berikut ini adalah contoh tipe standar atau normalisasi dari pasangan nilai kunci:

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

Tanda terima tidak memiliki “Nama Penjual” secara eksplisit terdaftar. Namun, Analyze Expense API mengakui dokumen sebagai tanda terima dan mengkategorikan nilai “AMAZON” sebagai Type`VENDOR_NAME`.

LabelDetection

Berikut ini adalah contoh teks seperti yang ditunjukkan pada halaman dokumen pelanggan:

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

```
                "Confidence": 88.19171142578125
            },
            "ValueDetection": {
                "Geometry": { ... },
                "Text": "Mina",
                "Confidence": 87.89806365966797
            }
        }
```

Contoh dokumen berisi “CASHIER Mina”. Analyze Expense API mengekstrak apa adanya nilai dan mengembalikannya di bawah`LabelDetection`. Untuk nilai tersirat seperti “Nama Penjual”, di mana “kunci” tidak ditampilkan secara eksplisit dalam tanda terima,`LabelDetection`tidak akan disertakan dalam elemen AnalyzeExpense. Dalam kasus tersebut, AnalyzeExpense API tidak kembali`LabelDetection`.

ValueDetection

Berikut ini adalah contoh menunjukkan “nilai” dari pasangan nilai kunci.

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

Dalam contoh, dokumen berisi “CASHIER Mina”. AnalyzeExpense API mendeteksi nilai Kasir sebagai Mina dan mengembalikannya di bawah`ValueDetection`.

Identitas Dokumentasi Respon Ob

Ketika Anda mengirimkan dokumen identitas ke AnalyzeID API, ia mengembalikan serangkaian`IdentityDocumentField`objek. Masing-masing benda-benda ini berisi`Type`, dan`Value`.`Type`mencatat bidang dinormalisasi yang Amazon Textract mendeteksi, dan`Value`mencatat teks yang terkait dengan bidang dinormalisasi.

Di bawah ini adalah contoh`IdentityDocumentField`, dipersingkat untuk singkatnya.

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

Ini adalah dua contoh IdentityDocumentFields dipotong dari respon yang lebih panjang. Ada pemisahan antara jenis terdeteksi dan nilai untuk jenis itu. Di sini, itu adalah nama pertama dan belakang masing-masing. Struktur ini berulang dengan semua informasi yang terkandung. Jika tipe tidak diakui sebagai bidang dinormalisasi, maka akan terdaftar sebagai “lainnya”.

Berikut ini adalah daftar bidang dinormalisasi untuk SIM:

- Nama Depan
- Nama belakang
- Nama Tengah
- akhiran
- alamat kota
- kode pos di alamat
- alamat
- county
- nomor dokumen
- Tanggal kedaluwarsa
- tanggal lahir
- nama negara
- tanggal penerbitan
- kelas
- pembatasan
- dukungan
- Tipe id
- kawakan
- menegur

Berikut ini adalah daftar bidang dinormalisasi untuk Paspor AS:

- Nama Depan
- Nama belakang
- Nama Tengah
- nomor dokumen
- Tanggal kedaluwarsa
- tanggal lahir
- tempat lahir
- tanggal penerbitan
- Tipe id

Lokasi Item pada Halaman Dokumen

Operasi Amazon Textract Texact mengembalikan lokasi dan geometri item yang ditemukan di halaman dokumen.DetectDocumentTextdanGetDocumentTextDetectionmengembalikan lokasi dan geometri untuk garis dan kata-kata, sementaraAnalyzeDocumentdanGetDocumentAnalysismengembalikan lokasi dan geometri pasangan kunci-nilai, tabel, sel, dan elemen seleksi.

Untuk menentukan di mana item berada di halaman dokumen, gunakan kotak pembatas (Geometry) informasi yang dikembalikan oleh operasi Amazon Textract Texact dalamBlockobjek. Parameter`Geometry`objek berisi dua jenis lokasi dan informasi geometris untuk item terdeteksi:

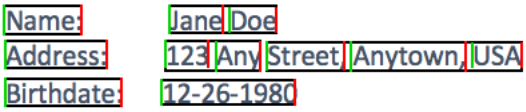
- Sumbu sejajarBoundingBoxobjek yang berisi koordinat kiri atas dan lebar dan tinggi item.
- Sebuah objek poligon yang menggambarkan garis besar item, ditentukan sebagai arrayPointbenda-benda yang mengandung`X`(sumbu horizontal) dan`Y`(sumbu vertikal) koordinat halaman dokumen setiap titik.

JSON untuk`Block`terlihat seperti berikut ini. Catatan`BoundingBox`dan`Polygon`bidang.

```
{
    "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"
},
```


Anda dapat menggunakan informasi geometri untuk menggambar kotak pembatas di sekitar item yang terdeteksi. Untuk contoh yang menggunakan`BoundingBox`dan`Polygon`informasi untuk menggambar kotak di sekitar garis dan garis vertikal pada awal dan akhir setiap kata, lihatMendeteksi Teks Dokumen dengan Amazon Textract. Output contoh ini mirip dengan yang berikut ini.



Kotak Pembatas

Sebuah kotak pembatas (`BoundingBox`) memiliki sifat sebagai berikut:

- Tinggi — Tinggi kotak pembatas sebagai rasio tinggi halaman dokumen secara keseluruhan.
- Kiri — Koordinat X titik kiri atas kotak pembatas sebagai rasio lebar halaman dokumen secara keseluruhan.
- Atas — Koordinat Y dari titik kiri atas kotak pembatas sebagai rasio tinggi halaman dokumen secara keseluruhan.
- Lebar — Lebar kotak pembatas sebagai rasio lebar halaman dokumen secara keseluruhan.

Setiap properti BoundingBox memiliki nilai antara 0 dan 1. Nilai adalah rasio lebar citra secara keseluruhan (berlaku untuk`Left`dan`Width`) atau tinggi (berlaku untuk`Height`dan`Top`). Misalnya, jika citra input berukuran 700 x 200 piksel, dan koordinat kiri atas kotak batas adalah (350,50) piksel, maka API mengembalikan a`Left`nilai 0,5 (350/700) dan`Top`nilai 0,25 (50/200).

Diagram berikut menunjukkan kisaran halaman dokumen yang mencakup setiap properti BoundingBox.

Untuk menampilkan kotak pembatas dengan lokasi dan ukuran yang benar, Anda harus mengalikan nilai-nilai BoundingBox dengan lebar halaman dokumen atau tinggi (tergantung pada nilai yang Anda inginkan) untuk mendapatkan nilai-nilai piksel. Anda menggunakan nilai-nilai piksel untuk menampilkan kotak pembatas. Contohnya adalah menggunakan halaman dokumen dengan lebar 608 piksel x tinggi 588 piksel, dan nilai kotak pembatas berikut untuk teks yang dianalisis:

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

Lokasi kotak pembatas teks dalam piksel dihitung sebagai berikut:

```
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
```

Anda menggunakan nilai-nilai ini untuk menampilkan kotak pembatas di sekitar teks yang dianalisis. Contoh Java dan Python berikut menunjukkan bagaimana menampilkan kotak pembatas.

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

Contoh Python ini mengambil di`response`dikembalikan olehDetectDocumentTextOperasi API.

```
def process_text_detection(response):

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

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

        draw = ImageDraw.Draw(image)

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

    # Display the image
    image.show()

    return len(blocks)
```

Polygon

Poligon yang dikembalikan oleh`AnalyzeDocument`adalah arrayPointobjek. Masing-masing`Point`memiliki koordinat X dan Y untuk lokasi tertentu pada halaman dokumen. Seperti koordinat BoundingBox, koordinat poligon dinormalisasi dengan lebar dan tinggi dokumen, dan antara 0 dan 1.

Anda dapat menggunakan poin dalam array poligon untuk menampilkan kotak pembatas butir halus di sekitar`Block`objek. Anda menghitung posisi setiap titik poligon pada halaman dokumen dengan menggunakan teknik yang sama digunakan untuk`BoundingBoxes`. Kalikan koordinat X dengan lebar halaman dokumen, dan kalikan koordinat Y dengan tinggi halaman dokumen.

Contoh berikut menunjukkan cara menampilkan garis vertikal poligon.

```
   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));

   }
```

Memulai dengan Amazon Textract

Bagian ini menyediakan topik untuk membantu Anda memulai menggunakan Amazon Textract. Jika Anda baru mengenal Amazon Textract, sebaiknya Anda terlebih dahulu meninjau konsep dan terminologi diCara Amazon Textract Bekerja.

Anda dapat mencoba API dengan menggunakan demonstrasi di konsol Amazon Textract Texact. Untuk informasi selengkapnya, lihathttps://console.aws.amazon.com/textract/.

Topik

- Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM
- Langkah 2: MenyiapkanAWS CLIdanAWSSDK
- Langkah 3: Memulai menggunakanAWS CLIdanAWSSDK API

Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM

Sebelum Anda menggunakan Amazon Textract untuk pertama kali, selesaikan tugas berikut:

1. Mendaftar ke AWS.
2. Membuat Pengguna IAM.

Mendaftar ke AWS

Saat Anda mendaftar untuk Amazon Web Services (AWS), akun AWS Anda secara otomatis terdaftar untuk semua layanan yang dirilis di AWS. Anda hanya akan dikenakan biaya untuk layanan yang digunakan.

Dengan Amazon Textract, Anda hanya membayar untuk sumber daya yang Anda gunakan. Untuk informasi selengkapnya tentang tarif penggunaan Amazon Textract, lihatHarga Amazon Textract. Jika Anda adalah pelanggan baru AWS, Anda dapat memulai Amazon Textract secara gratis. Untuk informasi selengkapnya, lihat Tingkatan Gratis AWS.

Jika Anda sudah memiliki akun AWS, lewati ke tugas berikutnya. Jika Anda belum memiliki akun AWS, lakukan langkah-langkah berikut untuk membuatnya.

Untuk membuat akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran adalah menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

Catat ID akun AWS Anda, karena Anda akan membutuhkannya nanti.

Membuat Pengguna IAM

Layanan di AWS, seperti Amazon Textract, mewajibkan Anda memberikan kredensia saat Anda mengaksesnya. Layanan kemudian dapat menentukan apakah Anda memiliki izin untuk mengakses sumber daya yang dimiliki oleh layanan tersebut. Konsol tersebut memerlukan kata sandi Anda. Anda dapat membuat access key untuk akun AWS Anda untuk mengakses AWS CLI atau API. Namun, kami tidak merekomendasikan Anda untuk mengakses AWS dengan menggunakan kredensional untuk akun AWS Anda. Sebagai gantinya, kami merekomendasikan Anda:

- Gunakan (IAM) AWS Identity and Access Management untuk membuat pengguna IAM.
- Tambahkan pengguna ke grup IAM dengan izin administratif.

Anda kemudian dapat mengakses AWS menggunakan URL khusus dan kredensial pengguna IAM.

Jika Anda mendaftar ke AWS tetapi belum membuat pengguna IAM untuk Anda sendiri, Anda dapat membuatnya menggunakan konsol IAM. Ikuti prosedur untuk membuat pengguna IAM di akun Anda.

Untuk membuat pengguna IAM dan masuk ke konsol tersebut

1. Buat pengguna IAM dengan izin administrator di akun AWS Anda. Untuk instruksi, buka Membuat Grup Pengguna dan Administrator IAM Pertama Anda di Panduan Pengguna IAM.
2. Sebagai pengguna IAM, masuk ke Konsol Manajemen AWS dengan menggunakan URL khusus. Untuk informasi selengkapnya, lihat Cara Pengguna Masuk ke Akun Anda dalam Panduan Pengguna IAM.

> **Note**
>
> IAM pengguna dengan izin administrator memiliki akses tak terbatas ke layanan AWS di akun Anda. Contoh kode dalam panduan ini mengasumsikan Anda memiliki pengguna dengan`AmazonTextractFullAccess`izin.`AmazonS3ReadOnlyAccess`Diperlukan untuk contoh yang mengakses dokumen yang tersimpan di bucket Amazon S3. Tergantung pada persyaratan keamanan, Anda mungkin ingin menggunakan grup IAM yang terbatas pada izin ini. Untuk informasi selengkapnya, lihat Membuat Grup IAM.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- AWS Identity and Access Management(IAM)
- Memulai
- Panduan Pengguna IAM

Langkah Selanjutnya

Langkah 2: MenyiapkanAWS CLIdanAWSSDK

Langkah 2: MenyiapkanAWS CLIdanAWSSDK

Langkah-langkah berikut menunjukkan cara menginstal SDK AWS Command Line Interface (AWS CLI) dan AWS yang menggunakan contoh dalam dokumentasi ini.

Ada sejumlah cara yang berbeda untuk mengautentikasi panggilan SDK AWS. Contoh dalam panduan ini menganggap bahwa Anda menggunakan profil kredensial default untuk menelepon perintah operasi API SDK AWS CLI dan AWS. Kredensyal default Anda akan berfungsi di seluruh layanan, jadi jika Anda telah mengonfigurasi kredensyal Anda, Anda tidak perlu melakukannya lagi. Namun, jika Anda ingin membuat set kredensyal lain untuk layanan ini, Anda dapat membuat profil nama. Untuk informasi lebih lanjut tentang membuat profil,lihat Profil Bernama.

Untuk daftar lengkap Wilayah AWS, lihat Wilayah dan Titik Akhir dalam Referensi Umum Amazon Web Services.

Untuk mengatur SDK AWS CLI dan AWS

1. Unduh dan instal SDK AWS CLI dan AWS yang ingin Anda gunakan. Panduan ini memberikan contoh untukAWS CLI, Java, dan Python. Untuk informasi tentang AWS SDK lainnya, lihat Alat untuk Amazon Web Services.

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

2. Buat access key untuk pengguna yang Anda buat diMembuat Pengguna IAM.

   a. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.

   b. Di panel navigasi, pilih Pengguna.

   c. Pilih nama pengguna yang Anda buat diMembuat Pengguna IAM.

   d. Pilih tab Kredensial keamanan.

   e. Pilih Buat access key. Lalu pilih Unduh file .csv untuk menyimpan access key ID dan secret access key ke file CSV dalam komputer Anda. Simpan file di lokasi aman. Anda tidak akan memiliki akses ke secret access key lagi setelah menutup kotak dialog ini. Setelah mengunduh file CSV, pilihtutup.

3. Tetapkan kredensyal dalam file profil kredensyal AWS di sistem lokal Anda, yang terletak di:

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

   Parameter`.aws`folder tidak ada sebelum konfigurasi awal pertama Anda dari instans AWS Anda. Pertama kali Anda mengkonfigurasi kredensyal Anda dengan CLI, folder ini akan dibuat. Untuk informasi selengkapnya tentang kredensial AWS, lihatKonfigurasi dan Pengaturan File Kredenal.

   File ini harus berisi baris dalam format berikut:

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

Ganti access key ID dan secret access key Anda dengan your_access_key_id dan your_secret_access_key.

4. Mengatur Wilayah AWS default di AWS`config`file di sistem lokal Anda, terletak di:

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

   Parameter`.aws`folder tidak ada sebelum konfigurasi awal pertama Anda dari instans AWS Anda. Pertama kali Anda mengkonfigurasi kredensyal Anda dengan CLI, folder ini akan dibuat. Untuk informasi selengkapnya tentang kredensial AWS, lihatKonfigurasi dan Pengaturan File Kredenal.

   File ini harus berisi baris berikut:

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

   Ganti Wilayah AWS (misalnya, “us-west-2") untukyour_aws_region.

   > **Note**
   >
   > Jika Anda tidak memilih Wilayah, maka us-east-1 digunakan secara default.

Langkah Selanjutnya

Langkah 3: Memulai menggunakanAWS CLIdanAWSSDK API

Langkah 3: Memulai menggunakanAWS CLIdanAWSSDK API

Setelah Anda mengaturAWS CLIdanAWSSDK yang ingin Anda gunakan, Anda dapat membangun aplikasi yang menggunakan Amazon Textract. Topik-topik berikut ini akan menunjukkan kepada Anda cara memulai Amazon Textract.

- Menganalisis Teks Dokumen dengan Amazon Textract

FormatAWS CLIContoh

Contoh AWS CLI dalam panduan ini diformat untuk sistem operasi Linux. Untuk menggunakan sampel dengan Microsoft Windows, Anda perlu mengubah format parameter `--document` JSON, dan mengubah jeda baris dari garis miring terbalik (\) menjadi tanda sisipan (^). Untuk informasi selengkapnya tentang pemformatan JSON, lihat Penentuan Nilai Parameter untuk Antarmuka Baris Perintah AWS.

Memproses Dokumen dengan Operasi Sinkron

Amazon Textract dapat mendeteksi dan menganalisis teks dalam dokumen satu halaman yang disediakan sebagai gambar dalam format JPEG, PNG, PDF, dan TIFF. Operasi yang sinkron dan kembali menghasilkan hampir secara waktu nyata. Untuk informasi lebih lanjut tentang dokumen, lihat Deteksi Teks dan Dokumen Analisis Respon Objek.

Bagian ini mencakup bagaimana Anda dapat menggunakan Amazon Textract untuk mendeteksi dan menganalisis teks dalam dokumen satu halaman secara serentak. Untuk mendeteksi dan menganalisis teks dalam dokumen multipage, atau untuk mendeteksi dokumen JPEG dan PNG secara asinkron, lihatMemproses Dokumen dengan Operasi Asynchronous.

Anda dapat menggunakan operasi sinkron Amazon Texact untuk tujuan berikut:

- Deteksi teks - Anda dapat mendeteksi baris dan kata-kata pada gambar dokumen satu halaman dengan menggunakanDetectDocumentTextoperasi. Untuk informasi selengkapnya, lihat Mendeteksi teks.
- Analisis teks - Anda dapat mengidentifikasi hubungan antara teks yang terdeteksi pada dokumen satu halaman dengan menggunakanAnalyzeDocumentoperasi. Untuk informasi selengkapnya, lihat Menganalisis Dokumen.
- Analisis Faktur dan Tanda Terima — Anda dapat mengidentifikasi hubungan keuangan antara teks yang terdeteksi pada faktur atau tanda terima satu halaman menggunakan operasi AnalyzeExpense. Untuk informasi selengkapnya, lihatMenganalisis Faktur dan Penerimaan
- Analisis Dokumen Identitas - Anda dapat menganalisis dokumen identitas yang dikeluarkan oleh Pemerintah AS dan mengekstrak informasi bersama dengan jenis informasi umum yang ditemukan pada dokumen identitas. Untuk informasi selengkapnya, lihatMenganalisis Dokumen Identitas.

Topik

- Memanggil Operasi Sinkronisasi Amazon Texact
- Mendeteksi Teks Dokumen dengan Amazon Textract
- Menganalisis Teks Dokumen dengan Amazon Textract
- Menganalisis Faktur dan Penerimaan dengan Amazon Textract
- Menganalisis Dokumentasi Identitas dengan Amazon Textract

Memanggil Operasi Sinkronisasi Amazon Texact

Operasi Amazon Textract memproses citra dokumen yang disimpan di sistem file lokal, atau citra dokumen yang disimpan dalam bucket Amazon S3. Anda menentukan di mana dokumen input berada dengan menggunakanDocumentparameter input. Gambar dokumen dapat berupa format PNG, JPEG, PDF, atau TIFF. Hasil untuk operasi sinkron dikembalikan segera dan tidak disimpan untuk pengambilan.

Untuk contoh lengkap, lihatMendeteksi Teks Dokumen dengan Amazon Textract.

Permintaan

Berikut ini menjelaskan cara kerja permintaan di Amazon Textract.

Dokumen Lulus sebagai Image Bytes

Anda dapat meneruskan gambar dokumen ke operasi Amazon Textract dengan meneruskan gambar sebagai array byte berkode base64. Contohnya adalah citra dokumen yang dimuat dari sistem file lokal. Kode Anda mungkin tidak perlu mengodekan byte file dokumen jika Anda menggunakanAWSSDK untuk memanggil operasi Amazon Textract API.

Byte gambar ditentukan dalam`Bytes`bidang`Document`parameter input. Contoh berikut menunjukkan masukan JSON untuk operasi Amazon Textract yang meneruskan byte gambar di`Bytes`parameter input.

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

> **Note**
>
> Jika Anda menggunakanAWS CLI, Anda tidak dapat meneruskan byte citra ke operasi Amazon Textract. Sebagai gantinya, Anda harus mereferensikan citra yang disimpan di bucket Amazon S3.

Kode Java berikut menunjukkan cara memuat citra dari sistem file lokal dan memanggil operasi Amazon Textract.

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

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

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


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

Dokumen yang Disimpan di Amazon S3 Bucket

Amazon Textract Texact dapat menganalisis citra dokumen yang disimpan di bucket Amazon S3. Anda menentukan bucket dan nama file dengan menggunakanS3Objectbidang`Document`parameter input. Contoh berikut menunjukkan JSON input untuk operasi Amazon Textract yang memproses dokumen yang disimpan dalam bucket Amazon S3.

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

Contoh berikut menunjukkan cara memanggil operasi Amazon Textract menggunakan citra yang tersimpan di bucket Amazon S3.

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

AmazonTextract client = AmazonTextractClientBuilder.defaultClient();

DetectDocumentTextRequest request = new DetectDocumentTextRequest()
        .withDocument(new Document()
                .withS3Object(new S3Object()
```

```
                        .withName(document)
                        .withBucket(bucket)));

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

Response

Contoh berikut ini adalah respons JSON dari panggilan ke`DetectDocumentText`. Untuk informasi selengkapnya, lihat Mendeteksi teks.

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


Mendeteksi Teks Dokumen dengan Amazon Textract

Untuk mendeteksi teks dalam dokumen, Anda menggunakanDetectDocumentTextoperasi, dan lulus file dokumen sebagai masukan.`DetectDocumentText`mengembalikan struktur JSON yang berisi garis dan kata-kata teks yang terdeteksi, lokasi teks dalam dokumen, dan hubungan antara teks yang terdeteksi. Untuk informasi selengkapnya, lihat Mendeteksi teks.

Anda dapat menyediakan dokumen input sebagai array bit citra (bit citra yang dikodekan base64), atau sebagai objek Amazon S3. Dalam prosedur ini, Anda mengunggah file citra ke bucket S3 Anda dan menentukan nama file.

Untuk mendeteksi teks dalam dokumen (API)

1. Jika belum:

   a. Buat atau perbarui pengguna IAM dengan izin `AmazonTextractFullAccess` dan `AmazonS3ReadOnlyAccess`. Untuk informasi selengkapnya, lihat Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM.

   b. Instal dan konfigurasikan SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

2. Unggah dokumen ke bucket S3.

   Untuk instruksi, lihatMengunggah Objek ke Amazon S3diPanduan Pengguna Amazon Simple Storage Service.

3. Gunakan contoh berikut untuk memanggil operasi `DetectDocumentText`.

   Java

   Kode contoh berikut menampilkan dokumen dan kotak di sekitar baris teks terdeteksi.

   Fungsi`main`, ganti nilai`bucket`dan`document`dengan nama bucket Amazon S3 dan dokumen yang Anda gunakan pada langkah 2.

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

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

```
import com.amazonaws.services.textract.model.DetectDocumentTextResult;
import com.amazonaws.services.textract.model.Document;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
import com.amazonaws.services.textract.model.Relationship;

public class DocumentText extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    DetectDocumentTextResult result;

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

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

    }

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

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

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

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

        // Iterate through blocks and display polygons around lines of detected
 text.
        List<Block> blocks = result.getBlocks();
        for (Block block : blocks) {
            DisplayBlockInfo(block);
            if ((block.getBlockType()).equals("LINE")) {
                ShowPolygon(height, width, block.getGeometry().getPolygon(),
 g2d);
                /*
```

```
                ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d);
               */
           } else { // its a word, so just show vertical lines.
               ShowPolygonVerticals(height, width,
block.getGeometry().getPolygon(), g2d);
           }
       }
   }

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

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

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

   }

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

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

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

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

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

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

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

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

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

        }

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

```
                System.out.println("        IDs: " +
 relationship.getIds().toString());
            }
        } else {
            System.out.println("        No related Blocks");
        }

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

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

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

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

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


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


        // Get the document from S3
```

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

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


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

        DetectDocumentTextResult result = client.detectDocumentText(request);

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

    }
}
```

AWS CLI

Perintah AWS CLI ini menampilkan output JSON untuk operasi CLI `detect-document-text`.

Ganti nilai`Bucket`dan`Name`dengan nama bucket Amazon S3 dan dokumen yang Anda gunakan pada langkah 2.

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

Python

Kode contoh berikut menampilkan dokumen dan kotak di sekitar baris terdeteksi teks.

Fungsi`main`, ganti nilai`bucket`dan`document`dengan nama bucket Amazon S3 dan dokumen yang Anda gunakan pada langkah 2.

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

import psutil
import time

import math
from PIL import Image, ImageDraw, ImageFont


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

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

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

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

```
    print('        Polygon: {}'.format(block['Geometry']['Polygon']))

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

def process_text_detection(bucket, document):


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

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

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


    # Detect text in the document

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

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

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

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:
            print('Type: ' + block['BlockType'])
            if block['BlockType'] != 'PAGE':
                print('Detected: ' + block['Text'])
```

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

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

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


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

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

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

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


   # Display the image
```

```
    image.show()
    # display image for 10 seconds


    return len(blocks)

def main():

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

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

Node.js

Berikut Node.js contoh kode menampilkan dokumen dan kotak di sekitar baris terdeteksi teks, output gambar hasil ke direktori Anda menjalankan kode dari. Itu membuat penggunaan`image-size`dan`images`paket.

Fungsi`main`, ganti nilai`bucket`dan`document`dengan nama bucket Amazon S3 dan dokumen yang Anda gunakan pada langkah 2. Ganti nilai`regionConfig`dengan nama wilayah akun Anda berada.

```
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. Jalankan contoh. Contoh Python dan Java menampilkan gambar dokumen. Kotak hitam mengelilingi setiap baris teks yang terdeteksi. Garis vertikal hijau adalah awal dari kata yang terdeteksi. Garis vertikal merah adalah akhir dari kata yang terdeteksi. ParameterAWS CLIcontoh hanya menampilkan output JSON untuk`DetectDocumentText`operasi.

Menganalisis Teks Dokumen dengan Amazon Textract

Untuk menganalisis teks dalam dokumen, Anda menggunakanAnalyzeDocumentoperasi, dan lulus file dokumen sebagai masukan.`AnalyzeDocument`mengembalikan struktur JSON yang berisi teks yang dianalisis. Untuk informasi selengkapnya, lihat Menganalisis Dokumen.

Anda dapat menyediakan dokumen input sebagai array bit citra (bit citra yang dikodekan base64), atau sebagai objek Amazon S3. Dalam prosedur ini, Anda mengunggah file citra ke bucket S3 Anda dan menentukan nama file.

Untuk menganalisis teks dalam dokumen (API)

1. Jika belum:

   a. Buat atau perbarui pengguna IAM dengan izin `AmazonTextractFullAccess` dan `AmazonS3ReadOnlyAccess`. Untuk informasi selengkapnya, lihat Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM.

   b. Instal dan konfigurasikan SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

2. Unggah citra yang berisi dokumen ke bucket S3 Anda.

   Untuk instruksi, lihatMengunggah Objek ke Amazon S3diPanduan Pengguna Amazon Simple Storage Service.

3. Gunakan contoh berikut untuk memanggil operasi `AnalyzeDocument`.

   Java

   Contoh kode berikut menampilkan dokumen dan kotak di sekitar item yang terdeteksi.

   Fungsi`main`, ganti nilai`bucket`dan`document`dengan nama bucket Amazon S3 dan citra dokumen yang Anda gunakan pada langkah 2.

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

import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
```

```
import javax.imageio.ImageIO;
import javax.swing.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.AnalyzeDocumentRequest;
import com.amazonaws.services.textract.model.AnalyzeDocumentResult;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.BoundingBox;
import com.amazonaws.services.textract.model.Document;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
import com.amazonaws.services.textract.model.Relationship;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;

public class AnalyzeDocument extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;

    AnalyzeDocumentResult result;

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

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

    }

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

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

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

        // Draw the image.
```

```
        g2d.drawImage(image, 0, 0, image.getWidth(this), image.getHeight(this),
this);

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

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

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

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

```
   }

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

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

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

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

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

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

   }

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

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

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

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

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

       }

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

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

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

       System.out.println("    Entity Types");
       if(entityTypes!=null) {
```

```
            for (String entityType : entityTypes) {
                System.out.println("        Entity Type: " + entityType);
            }
        } else {
            System.out.println("        No entity type");
        }

        if(block.getBlockType().equals("SELECTION_ELEMENT")) {
            System.out.print("    Selection element detected: ");
            if (block.getSelectionStatus().equals("SELECTED")){
                System.out.println("Selected");
            }else {
                System.out.println(" Not selected");
            }
        }

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

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

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

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


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

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

```
                .withEndpointConfiguration(endpoint).build();


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


        AnalyzeDocumentResult result = client.analyzeDocument(request);

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

    }
}
```

AWS CLI

Perintah AWS CLI ini menampilkan output JSON untuk operasi CLI `detect-document-text`.

Ganti nilai`Bucket`dan`Name`dengan nama bucket Amazon S3 dan dokumen yang Anda gunakan pada langkah 2.

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

Python

Contoh kode berikut menampilkan dokumen dan kotak di sekitar item yang terdeteksi.

Fungsimain, ganti nilaibucketdandocumentdengan nama bucket Amazon S3 dan dokumen yang Anda gunakan pada langkah 2.

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

import math
from PIL import Image, ImageDraw, ImageFont

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

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

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

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

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

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

    if block['BlockType'] == 'CELL':
        print("    Cell information")
        print("        Column:" + str(block['ColumnIndex']))
```

```
            print("        Row:" + str(block['RowIndex']))
            print("        Column Span:" + str(block['ColumnSpan']))
            print("        RowSpan:" + str(block['ColumnSpan']))

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

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

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

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

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

    def process_text_analysis(bucket, document):

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

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

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

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

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

        ### Alternatively, process using S3 object ###
```

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

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


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

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

        DisplayBlockInformation(block)

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

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

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

```
                ShowSelectedElement(draw, block['Geometry']
['BoundingBox'],width,height, 'blue')

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

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


def main():

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

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

Node.js

Contoh kode berikut menampilkan dokumen dan kotak di sekitar item yang terdeteksi.

Pada kode di bawah ini, ganti nilai`bucket`dan`photo`dengan nama bucket Amazon S3 dan dokumen yang Anda gunakan pada langkah 2. Ganti nilai`region`dengan wilayah yang terkait dengan akun Anda.

```
// 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. Jalankan contoh. Contoh Python dan Java menampilkan gambar dokumen dengan kotak pembatas berwarna berikut:

   - Merah - objek Blok KUNCI
   - Hijau - NILAI Blok objek
   - Biru - TABLE Blok benda
   - Kuning - SELL Blok benda

   Elemen seleksi yang dipilih diisi dengan warna biru.

   ParameterAWS CLIcontoh hanya menampilkan output JSON untuk`AnalyzeDocument`operasi.

Menganalisis Faktur dan Penerimaan dengan Amazon Textract

Untuk menganalisis dokumen faktur dan tanda terima, Anda menggunakan AnalyzeExpense API, dan meneruskan file dokumen sebagai masukan.`AnalyzeExpense`adalah operasi sinkron yang mengembalikan struktur JSON yang berisi teks dianalisis. Untuk informasi selengkapnya, lihat Menganalisis Faktur dan Penerimaan.

Untuk menganalisis faktur dan tanda terima secara asinkron, gunakan`StartExpenseAnalysis`untuk mulai memproses file dokumen input dan menggunakan`GetExpenseAnalysis`untuk mendapatkan hasilnya.

Anda dapat menyediakan dokumen input sebagai array bit citra (bit citra yang dikodekan base64), atau sebagai objek Amazon S3. Dalam prosedur ini, Anda mengunggah file citra ke bucket S3 Anda dan menentukan nama file.

Untuk menganalisis faktur atau tanda terima (API)

1. Jika belum:

   a. Buat atau perbarui pengguna IAM dengan izin `AmazonTextractFullAccess` dan `AmazonS3ReadOnlyAccess`. Untuk informasi selengkapnya, lihat Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM.

   b. Instal dan konfigurasikan SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

2. Unggah citra yang berisi dokumen ke bucket S3 Anda.

   Untuk instruksi, lihatMengunggah Objek ke Amazon S3diPanduan Pengguna Amazon Simple Storage Service.

3. Gunakan contoh berikut untuk memanggil operasi `AnalyzeExpense`.

   CLI

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

   Python

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

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

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

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

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

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

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

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

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

    # Set width and height to display image and draw bounding boxes
```

```
    # Create drawing object
    width, height = image.size
    draw = ImageDraw.Draw(image)

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

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

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

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

    # Display the image
    image.show()

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

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

Java

```
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.*;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseRequest;
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseResponse;
import software.amazon.awssdk.services.textract.model.BoundingBox;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.ExpenseDocument;
import software.amazon.awssdk.services.textract.model.ExpenseField;
import software.amazon.awssdk.services.textract.model.LineItemFields;
import software.amazon.awssdk.services.textract.model.LineItemGroup;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.model.Point;

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

 private static final long serialVersionUID = 1L;
```

```
BufferedImage image;
static AnalyzeExpenseResponse result;

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

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

}

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

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

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

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

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

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

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

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

    }
```

```
    }

   }

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

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

    for (LineItemGroup lineitemgroup : lineitemgroups) {

     if (lineitemgroup.hasLineItems()) {

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

       if (lineitemfield.hasLineItemExpenseFields()) {

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

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

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

        }
       }

      }

     }

    }
```

```
  }
 }

}

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

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

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

}

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

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

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

}

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

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

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

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

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

  for (ExpenseField summaryfield : summaryfields) {

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

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

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

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

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

```
      "        Polygon: " +
summaryfield.valueDetection().geometry().polygon().toString());

   }

  }

 }

}

private void DisplayAnalyzeExpenseLineItemGroupsInfo(ExpenseDocument
expensedocument) {

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

 if (expensedocument.hasLineItemGroups()) {

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

  for (LineItemGroup lineitemgroup : lineitemgroups) {

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

   if (lineitemgroup.hasLineItems()) {

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

    for (LineItemFields lineitemfield : lineItems) {

     if (lineitemfield.hasLineItemExpenseFields()) {

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

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

       }

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

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

       }

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

      }
     }

    }

   }
  }
 }

}

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

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


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

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

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

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

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


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

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

      AnalyzeExpenseResponse result = textractclient.analyzeExpense(request);

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

      ByteArrayInputStream bais = new
 ByteArrayInputStream(getObjectResponse.asByteArray());
      try {
       BufferedImage image = ImageIO.read(bais);
       System.out.println("Successfully read the image");
       JFrame frame = new JFrame("Expense Image");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       TextractAnalyzeExpense panel = new TextractAnalyzeExpense(result, image);
```

```
       panel.setPreferredSize(new Dimension(image.getWidth(),
 image.getHeight()));
       frame.setContentPane(panel);
       frame.pack();
       frame.setVisible(true);
      } catch (IOException e) {
       throw new RuntimeException(e);
      } catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
     } else {
      // Handle the error
      exception.printStackTrace();
     }
    });

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

 }
}
```

Node.Js

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

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

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

// Set params
const params = {
```

```
    Document: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
  }

const process_text_detection = async () => {
    try {
        const aExpense = new AnalyzeExpenseCommand(params);
        const response = await textractClient.send(aExpense);
        //console.log(response)
        response.ExpenseDocuments.forEach(doc => {
            doc.LineItemGroups.forEach(items => {
                items.LineItems.forEach(fields => {
                    fields.LineItemExpenseFields.forEach(expenseFields =>{
                        console.log(expenseFields)
                    })
                }
                )}
                )
            }
        )
        return response; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

process_text_detection()
```

4. Ini akan memberi Anda output JSON untuk`AnalyzeExpense`operasi.

Menganalisis Dokumentasi Identitas dengan Amazon Textract

Untuk menganalisis dokumen identitas, Anda menggunakan AnalyzeID API, dan meneruskan file dokumen sebagai masukan.`AnalyzeID`mengembalikan struktur JSON yang berisi teks yang dianalisis. Untuk informasi selengkapnya, lihat Menganalisis Dokumen Identitas.

Anda dapat menyediakan dokumen input sebagai array bit citra (bit citra yang dikodekan base64), atau sebagai objek Amazon S3. Dalam prosedur ini, Anda mengunggah file citra ke bucket S3 Anda dan menentukan nama file.

Untuk menganalisis dokumen identitas (API)

1. Jika belum:

   a. Buat atau perbarui pengguna IAM dengan izin `AmazonTextractFullAccess` dan `AmazonS3ReadOnlyAccess`. Untuk informasi selengkapnya, lihat Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM.

   b. Instal dan konfigurasikan SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

2. Unggah citra yang berisi dokumen ke bucket S3 Anda.

   Untuk instruksi, lihatMengunggah Objek ke Amazon S3diPanduan Pengguna Amazon Simple Storage Service.

3. Gunakan contoh berikut untuk memanggil operasi `AnalyzeID`.

   CLI

   Contoh berikut mengambil dalam file input dari bucket S3 dan menjalankan`AnalyzeID`operasi di atasnya. Pada kode di bawah ini, ganti nilai*ember*dengan nama bucket S3 Anda, nilai*fail*dengan nama file dalam bucket Anda, dan nilai*daerah*dengan nama`region`terkait dengan akun Anda.

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

   Anda juga dapat memanggil API dengan bagian depan dan belakang SIM dengan menambahkan objek S3 lain ke input.

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

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda bukan tanda kutip tunggal dan melarikan diri tanda kutip ganda dalam dengan garis miring terbalik (yaitu\) untuk mengatasi kesalahan parser yang mungkin Anda hadapi. Sebagai contoh, lihat di bawah ini:

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

Python

Contoh berikut mengambil dalam file input dari bucket S3 dan menjalankan`AnalyzeID`operasi di atasnya, mengembalikan pasangan nilai kunci yang terdeteksi. Pada kode di bawah ini, ganti nilai*bucket_name*dengan nama bucket S3 Anda, nilai*file_name*dengan nama file dalam bucket Anda, dan nilai*daerah*dengan nama`region`terkait dengan akun Anda.

```
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

Contoh berikut mengambil dalam file input dari bucket S3 dan menjalankan`AnalyzeID`operasi di atasnya, mengembalikan data yang terdeteksi. Dalam fungsi utama, ganti nilai-nilais3bucketdan`sourceDoc`dengan nama bucket Amazon S3 dan citra dokumen yang Anda gunakan pada langkah 2.

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

package com.amazonaws.samples;

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

public class AnalyzeIdentityDocument {

    public static void main(String[] args) {

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

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

        String s3bucket = "bucket-name"; //args[0];
        String sourceDoc = "sourcedoc-name";  //args[1];
```

```
       AmazonTextractClient textractClient = (AmazonTextractClient)
AmazonTextractClientBuilder.standard()
               .withRegion(Regions.US_EAST_1)
               .build();

       getDocDetails(textractClient, s3bucket, sourceDoc);
   }

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

      try {

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

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

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

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

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

               List<IdentityDocumentField>idFields =
doc.getIdentityDocumentFields();
               for (IdentityDocumentField field: idFields) {
                   System.out.println("Field type is "+
field.getType().getText());
                   System.out.println("Field value is "+
field.getValueDetection().getText());
               }
           }

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

```
    }
}
```

4. Ini akan memberi Anda output JSON untuk`AnalyzeID`operasi.

Memproses Dokumen dengan Operasi Asynchronous

Amazon Textract dapat mendeteksi dan menganalisis teks dalam dokumen multihalaman yang dalam format PDF atau TIFF. Ini termasuk faktur dan tanda terima. Pengolahan dokumen multipage merupakan operasi asinkron. Pengolahan dokumen asinkron berguna untuk memproses dokumen multipage yang besar. Misalnya, file PDF dengan lebih dari 1.000 halaman membutuhkan waktu untuk diproses. Memproses file PDF secara asinkron memungkinkan aplikasi Anda menyelesaikan tugas lain sementara menunggu proses selesai.

Bagian ini mencakup bagaimana Anda dapat menggunakan Amazon Textract untuk mendeteksi dan menganalisis teks secara asinkron pada dokumen multihalaman atau satu halaman. Dokumen multipage harus dalam format PDF atau TIFF. Dokumen satu halaman yang diproses dengan operasi asinkron dapat dalam format JPEG, PNG, TIFF atau PDF.

Anda dapat menggunakan operasi asinkron Amazon Texact untuk tujuan berikut:

- Deteksi teks - Anda dapat mendeteksi baris dan kata-kata pada dokumen multipage. Operasi asinkronStartDocumentTextDetectiondanGetDocumentTextDetection. Untuk informasi selengkapnya, lihat Mendeteksi teks.
- Analisis teks - Anda dapat mengidentifikasi hubungan antara teks yang terdeteksi pada dokumen multipage. Operasi asinkronStartDocumentAnalysisdanGetDocumentAnalysis. Untuk informasi selengkapnya, lihat Menganalisis Dokumen.
- Analisis pengeluaran — Anda dapat mengidentifikasi hubungan data pada faktur dan tanda terima multipage. Amazon Textract memperlakukan setiap faktur atau halaman penerimaan dokumen multi-halaman sebagai tanda terima individu atau faktur. Ini tidak mempertahankan konteks dari satu halaman ke halaman lain dari dokumen multi-halaman. Operasi asinkronStartExpenseAnalysisdanGetExpenseAnalysis. Untuk informasi selengkapnya, lihat Menganalisis Faktur dan Penerimaan.

Topik

- Memanggil Operasi Asinkron Amazon Texact
- Mengkonfigurasi Amazon Textract untuk Operasi Asynchronous
- Mendeteksi atau Menganalisis Teks dalam Dokumen Multipage
- Pemberitahuan Amazon Textract

Memanggil Operasi Asinkron Amazon Texact

Amazon Textract menyediakan API asinkron yang dapat Anda gunakan untuk memproses dokumen multipage dalam format PDF atau TIFF. Anda juga dapat menggunakan operasi asinkron untuk memproses dokumen satu halaman yang ada dalam format JPEG, PNG, TIFF, atau PDF.

Informasi dalam topik ini menggunakan operasi pendeteksian teks untuk menunjukkan cara menggunakan operasi asinkron Amazon Texact. Pendekatan yang sama bekerja dengan operasi analisis teksthe section called “StartDocumentAnalysis”danthe section called “GetDocumentAnalysis”. Ini juga bekerja sama denganthe section called “StartExpenseAnalysis”danthe section called “GetExpenseAnalysis”.

Sebagai contoh, lihat Mendeteksi atau Menganalisis Teks dalam Dokumen Multipage.

Amazon Textract Texact secara tidak sinkron memproses dokumen yang tersimpan di bucket Amazon S3. Anda mulai memproses dengan memanggil`Start`operasi, seperti StartDocumentTextDetection. Status penyelesaian untuk permintaan diterbitkan ke topik Amazon Simple Notification Service (Amazon SNS). Untuk mendapatkan status penyelesaian dari topik Amazon SNS, Anda dapat menggunakan antrean Amazon Simple Queue Service (Amazon SQS) atau fungsi AWS Lambda. Setelah Anda memiliki status penyelesaian, Anda perlu memanggil operasi `Get`, seperti GetDocumentTextDetection, untuk mendapatkan hasil permintaan.

Hasil panggilan asinkron dienkripsi dan disimpan selama 7 hari dalam bucket milik Amazon Textract Texact secara default, kecuali jika Anda menetapkan bucket Amazon S3 menggunakan operasi`OutputConfig`argumen.

Tabel berikut menunjukkan operasi Start and Get yang sesuai untuk berbagai jenis pemrosesan asinkron yang didukung oleh Amazon Textract:

Mulai/Dapatkan Operasi API untuk Operasi Asinkron Amazon Texact

Jenis Pengolahan	Mulai API	Dapatkan API
Pendeteksi teks	StartDocumentTextDetection	GetDocumentTextDetection
Analisis Teks	StartDocumentAnalysis	GetDocumentAnalysis
Analisis Beban	StartExpenseAnalysis	GetExpenseAnalysis

Untuk contoh yang menggunakanAWS Lambdafungsi, lihatPemrosesan dokumen berskala besar dengan Amazon Textract.

Diagram berikut menunjukkan proses untuk mendeteksi teks dokumen dalam citra dokumen yang tersimpan di bucket Amazon S3. Dalam diagram, antrean Amazon SQS mendapatkan status penyelesaian dari topik Amazon SNS.

Proses yang ditampilkan oleh diagram sebelumnya adalah sama untuk menganalisis teks dan faktur/tanda terima. Anda mulai menganalisis teks dengan meneleponthe section called "StartDocumentAnalysis"dan mulai menganalisis faktur/tanda terima dengan meneleponthe section called "StartExpenseAnalysis"Anda mendapatkan hasilnya dengan meneleponthe section called "GetDocumentAnalysis"atauthe section called "GetExpenseAnalysis"masing-masing.

Pendeteksi teks

Anda memulai permintaan pendeteksi teks Amazon Texact dengan meneleponStartDocumentTextDetection. Berikut ini adalah contoh permintaan JSON yang diberikan oleh `StartDocumentTextDetection`.

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

Parameter input`DocumentLocation`menyediakan nama file dokumen dan bucket Amazon S3 untuk mengambilnya.`NotificationChannel`berisi Amazon Resource Name (ARN) dari topik Amazon SNS yang memberi tahu Amazon Textract saat permintaan pendeteksi teks selesai. Topik Amazon SNS harus berada di Wilayah AWS yang sama dengan titik akhir Amazon Textract Texact yang Anda hubungi.`NotificationChannel`juga berisi ARN untuk peran yang memungkinkan Amazon

Textract untuk mempublikasikan ke topik Amazon SNS. Anda memberikan izin penerbitan Amazon Textract Texact untuk topik Amazon SNS Anda dengan menciptakan peran layanan IAM. Untuk informasi selengkapnya, lihat Mengkonfigurasi Amazon Textract untuk Operasi Asynchronous.

Anda juga dapat menentukan parameter input opsional,`JobTag`, yang mengizinkan Anda untuk mengidentifikasi tugas, atau kelompok tugas, dalam status penyelesaian yang diterbitkan untuk topik Amazon SNS. Misalnya, Anda dapat menggunakan`JobTag`untuk mengidentifikasi jenis dokumen yang sedang diproses, seperti formulir pajak atau tanda terima.

Untuk mencegah duplikasi pekerjaan analisis yang tidak disengaja, Anda dapat memberikan token idempotensi , `ClientRequestToken`. Jika Anda memberikan nilai untuk`ClientRequestToken`, yang`Start`operasi mengembalikan yang sama`JobId`untuk beberapa panggilan identik ke`Start`operasi, seperti`StartDocumentTextDetection`. Token `ClientRequestToken` memiliki masa pakai 7 hari. Setelah 7 hari, Anda bisa menggunakannya kembali. Jika Anda menggunakan kembali token selama masa token aktif, hal berikut akan terjadi:

- Jika Anda menggunakan kembali token dengan sama`Start`operasi dan parameter input yang sama, sama`JobId`dikembalikan. Tugas tidak akan dilakukan lagi dan Amazon Textract Texact tidak mengirimkan status penyelesaian ke topik Amazon SNS yang terdaftar.
- Jika Anda menggunakan kembali token dengan operasi `Start` dan perubahan pada input parameter kecil, Anda mendapatkan pengecualian yang ditimbulkan `idempotentparametermismatchexception` (Kode status HTTP: 400).
- Jika Anda menggunakan kembali token dengan yang berbeda`Start`operasi, operasi berhasil.

Parameter opsional lain yang tersedia adalah`OutputConfig`, yang memungkinkan Anda menyesuaikan di mana output Anda akan ditempatkan. Secara default, Amazon Textract akan menyimpan hasil secara internal, dan hanya dapat diakses oleh operasi Get API. Dengan`OutputConfig`diaktifkan, Anda dapat mengatur nama bucket output akan dikirim ke, dan awalan file hasil, di mana Anda dapat men-download hasil Anda. Selain itu, Anda dapat mengatur`KMSKeyID`parameter ke kunci yang dikelola pelanggan untuk mengenkripsi output Anda. Tanpa parameter ini, Amazon Textract akan mengenkripsi sisi server menggunakan Kunci yang dikelola AWS untuk Amazon S3

> **Note**
>
> Sebelum menggunakan parameter ini, pastikan Anda memiliki izin PutObject untuk bucket output. Selain itu, pastikan Anda memiliki izin Decrypt, ReEncrypt, GenerateDataKey, dan DescribeKey untukAWS KMSkunci jika Anda memutuskan untuk menggunakannya.

Respons terhadap operasi `StartDocumentTextDetection` adalah pengidentifikasi tugas (`JobId`). GunakanJobIduntuk melacak permintaan dan mendapatkan hasil analisis setelah Amazon Textract Texact menerbitkan status penyelesaian ke topik Amazon SNS. Berikut ini adalah contoh:

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

Jika Anda memulai terlalu banyak pekerjaan secara bersamaan, panggilan keStartDocumentTextDetectionmenaikkanLimitExceededExceptionexception (Kode status HTTP: 400) hingga jumlah tugas yang berjalan bersamaan di bawah batas layanan Amazon Textract Texact.

Jika Anda menemukan bahwa pengecualian LimitExceededException dimunculkan dengan rongsokan aktivitas, mohon pertimbangkan untuk menggunakan antrean Amazon SQS untuk mengelola permintaan yang masuk. KontakAWSSupport jika Anda menemukan bahwa jumlah rata-rata permintaan bersamaan tidak dapat dikelola oleh antrean Amazon SQS dan Anda masih menerimaLimitExceededExceptionpengecualian.

Mendapatkan Status Penyelesaian Permintaan Analisis Amazon Textract

Amazon Textract Texact mengirimkan notifikasi penyelesaian analisis ke topik Amazon SNS yang terdaftar. Pemberitahuan tersebut mencakup pengindentifikasi tugas dan status penyelesaian operasi dalam string JSON. Permintaan deteksi teks yang sukses memilikiSUCCEEDEDstatus. Misalnya, hasil berikut menunjukkan berhasilnya pengolahan tugas pendeteksi teks.

```
{
    "JobId": "642492aea78a86a40665555dc375ee97bc963f342b29cd05030f19bd8fd1bc5f",
    "Status": "SUCCEEDED",
    "API": "StartDocumentTextDetection",
    "JobTag": "Receipt",
    "Timestamp": 1543599965969,
    "DocumentLocation": {
```

```
        "S3ObjectName": "document",
        "S3Bucket": "bucket"
    }
}
```

Untuk informasi selengkapnya, lihat Pemberitahuan Amazon Textract.

Untuk mendapatkan status informasi yang dipublikasikan ke topik Amazon SNS, gunakan salah satu opsi berikut:

- AWS Lambda — Anda dapat berlangganan fungsi AWS Lambda yang Anda tulis untuk topik Amazon SNS. Fungsi ini dipanggil saat Amazon Textract memberi tahu topik Amazon SNS bahwa permintaannya telah selesai. Gunakan fungsi Lambda jika Anda menginginkan kode sisi server untuk memproses hasil permintaan deteksi teks. Misalnya, Anda mungkin ingin menggunakan kode sisi server untuk menganotasi gambar atau membuat laporan tentang teks yang terdeteksi sebelum mengembalikan informasi ke aplikasi klien.
- Amazon SQS— Anda dapat berlangganan antrean Amazon SQS ke topik Amazon SNS. Anda kemudian dapat melakukan polling antrean Amazon SQS untuk mengambil status penyelesaian yang diterbitkan oleh Amazon Textract saat permintaan pendeteksi teks selesai. Untuk informasi selengkapnya, lihat Mendeteksi atau Menganalisis Teks dalam Dokumen Multipage. Gunakan antrean Amazon SQS jika Anda ingin memanggil operasi Amazon Textract Texact hanya dari aplikasi klien.

> **Important**
>
> Kami tidak merekomendasikan untuk mendapatkan status penyelesaian permintaan dengan berulang kali memanggil Amazon Textract`Get`operasi. Hal ini karena Amazon Textract Texact throttles`Get`operasi jika terlalu banyak permintaan yang dibuat. Jika Anda memproses beberapa dokumen secara bersamaan, akan lebih mudah dan lebih efisien untuk memantau satu antrean SQS untuk notifikasi penyelesaian daripada membuat polling Amazon Textract untuk status setiap pekerjaan secara individual.

Mendapatkan Hasil Deteksi Amazon Textract Texact

Untuk mendapatkan hasil permintaan pendeteksi teks, pertama pastikan bahwa status penyelesaian yang diambil dari topik Amazon SNS adalah`SUCCEEDED`. Kemudian panggil

`GetDocumentTextDetection`, yang memberikan nilai `JobId` yang dikembalikan dari `StartDocumentTextDetection`. Permintaan JSON serupa dengan contoh berikut:

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

`JobId`adalah pengidentifikasi untuk operasi deteksi teks. Karena deteksi teks dapat menghasilkan data dalam jumlah besar, gunakan`MaxResults`untuk menentukan jumlah hasil maksimum yang akan dikembalikan dalam satu`Get`operasi. Nilai default untuk`MaxResults`adalah 1.000. Jika Anda menentukan nilai yang lebih besar dari 1.000, hanya 1.000 hasil yang dikembalikan. Jika operasi tidak mengembalikan semua hasil, token pagination untuk halaman berikutnya dikembalikan. Untuk mendapatkan halaman hasil berikutnya, tentukan token di`NextToken`parameter.

> **Note**
>
> Amazon Textract Texact mempertahankan hasil operasi asinkron selama 7 hari. Anda tidak dapat mengambil hasilnya setelah waktu ini.

Parameter`GetDocumentTextDetection`respon operasi JSON serupa dengan yang berikut ini. Jumlah total halaman yang terdeteksi dikembalikan`DocumentMetadata`. Teks yang terdeteksi dikembalikan dalam`Blocks`array. Untuk informasi tentang`Block`benda, lihatDeteksi Teks dan Dokumen Analisis Respon Objek.

```
{
    "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
        }
    ]
}
```


Mengkonfigurasi Amazon Textract untuk Operasi Asynchronous

Prosedur berikut menunjukkan kepada Anda cara mengonfigurasi Amazon Textract yang akan digunakan dengan topik Amazon Simple Notification Service (Amazon SNS) dan antrean Amazon Simple Queue Service (Amazon SQS).

> **Note**
>
> Jika Anda menggunakan instruksi ini untuk mengaturMendeteksi atau Menganalisis Teks dalam Dokumen Multipageexample, Anda tidak perlu melakukan langkah 3 — 6. Contoh tersebut mencakup kode untuk membuat dan mengonfigurasi topik Amazon SNS dan antrean Amazon SQS.

Untuk mengonfigurasi Amazon Textract

1. Menyiapkan sebuahAWSakun untuk mengakses Amazon Textract. Untuk informasi selengkapnya, lihat Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM.

   Pastikan bahwa pengguna setidaknya memiliki izin berikut:

   - AmazonTextractFullAccess
   - AmazonS3ReadOnlyAccess
   - AmazonSNSFullAccess

- AmazonSQSFullAccess

2. Instal dan konfigurasikan SDK AWS yang diperlukan. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.
3. Membuat topik Amazon SNS. Tambahkan nama topik denganAmazontExtract. Perhatikan topik Amazon Resource Name (ARN). Pastikan bahwa topik berada di Wilayah yang sama denganAWSendpoint yang Anda gunakan dengan akun AWS Anda.
4. Buat antrean standar Amazon SQS dengan menggunakan Konsol Amazon SQS. Perhatikan antrean ARN.
5. Berlangganan antrean ke topik yang Anda buat di langkah 3.
6. Memberikan izin untuk topik Amazon SNS untuk mengirim pesan ke antrean Amazon SQS.
7. Buat peran layanan IAM untuk memberikan akses Amazon Textract Texact ke topik Amazon SNS Anda. Catat Amazon Resource Name (ARN) dari peran layanan tersebut. Untuk informasi selengkapnya, lihat Memberikan Amazon Textract Akses ke Topik Amazon SNS Anda.
8. Tambahkan kebijakan inline berikutuntuk pengguna IAM yang Anda buat pada langkah 1.

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

   Berikan kebijakan inline sebuah nama.
9. Sekarang Anda dapat menjalankan contoh diMendeteksi atau Menganalisis Teks dalam Dokumen Multipage.

Memberikan Amazon Textract Akses ke Topik Amazon SNS Anda

Amazon Textract memerlukan izin untuk mengirim pesan ke topik Amazon SNS Anda saat operasi asinkron selesai. Anda menggunakan peran layanan IAM untuk memberikan akses Amazon Textract Texact ke topik Amazon SNS.

Ketika Anda membuat topik Amazon SNS, Anda harus menambahkan nama topik dengan**`AmazonTextract`**—misalnya,**`AmazonTextractMyTopicName`**.

1. Masuk ke konsol IAM (https://console.aws.amazon.com/iam).
2. Di panel navigasi, pilih Peran.
3. Pilih Buat peran.
4. Untuk Pilih jenis entitas tepercaya, pilih Layanan AWS.
5. UntukPilih layanan yang akan menggunakan peran ini, pilihTextract.
6. Pilih Berikutnya: Izin.
7. Verifikasi bahwaAmazonTextractServiceRolekebijakan telah dimasukkan dalam daftar kebijakan terlampir. Untuk menampilkan kebijakan dalam daftar, masukkan bagian dari nama kebijakan diKebijakan filter.
8. Pilih Berikutnya: Tanda.
9. Anda tidak perlu menambahkan tag, jadi pilihSelanjutnya: Tinjau.
10. DiTinjaubagian, untukNama peran, masukkan nama untuk peran tersebut (example,`TextractRole`). MasukDeskripsi peran, perbarui deskripsi untuk peran, dan kemudian pilihBuat peran.
11. Pilih peran baru untuk membuka halaman detail peran.
12. DiRingkasan, salinARN Perannilai dan menyimpannya.
13. MemiilihHubungan kepercayaan.
14. MemiilihEdit hubungan kepercayaan, dan memastikan bahwa kebijakan kepercayaan terlihat sebagai berikut.

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

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

15. Pilih Perbarui Kebijakan Kepercayaan.

Mendeteksi atau Menganalisis Teks dalam Dokumen Multipage

Prosedur ini menunjukkan kepada Anda cara untuk mendeteksi atau menganalisis teks dalam dokumen multihalaman dengan menggunakan operasi pendeteksi Amazon Textract, dokumen yang disimpan dalam bucket Amazon S3, topik Amazon SNS, dan antrean Amazon SQS. Pengolahan dokumen multipage merupakan operasi asinkron. Untuk informasi selengkapnya, lihat Memanggil Operasi Asinkron Amazon Texact.

Anda dapat memilih jenis pemrosesan yang ingin Anda lakukan kode: deteksi teks, analisis teks, atau analisis biaya.

Hasil pengolahan dikembalikan dalam arraythe section called “Block”objek, yang berbeda tergantung pada jenis pengolahan yang Anda gunakan.

Untuk mendeteksi teks atau menganalisis dokumen multipage, Anda melakukan hal berikut:

1. Buat topik Amazon SNS dan antrean Amazon SQS.
2. Berlangganan antrean topik.
3. Berikan izin topik untuk mengirim pesan ke antrean.
4. Mulai memproses dokumen. Gunakan operasi yang sesuai untuk jenis analisis yang Anda pilih:
   - StartDocumentTextDetectionuntuk tugas deteksi teks.
   - StartDocumentAnalysisuntuk tugas analisis teks.
   - StartExpenseAnalysisuntuk tugas analisis biaya.
5. Dapatkan status penyelesaian dari antrean Amazon SQS. Contoh kode melacak pengenal pekerjaan (`JobId`) yang dikembalikan oleh`Start`operasi. Ini hanya mendapatkan hasil untuk mencocokkan pengidentifikasi tugas yang dibaca dari status penyelesaian. Hal ini penting jika aplikasi lain menggunakan antrean dan topik yang sama. Untuk kesederhanaan, contoh penghapusan tugas yang tidak cocok. Pertimbangkan untuk menambahkan tugas yang dihapus ke antrean surat mati Amazon SQS untuk penyelidikan lebih lanjut.

6. Dapatkan dan tampilkan hasil pemrosesan dengan memanggil operasi yang sesuai untuk jenis analisis yang Anda pilih:
   - GetDocumentTextDetectionuntuk tugas deteksi teks.
   - GetDocumentAnalysisuntuk tugas analisis teks.
   - GetExpenseAnalysisuntuk tugas analisis biaya.
7. Hapus topik Amazon SNS dan antrean Amazon SQS.

Melakukan Operasi Asinkron

Contoh kode untuk prosedur ini disediakan di Java, Python, danAWS CLI. Sebelum memulai, pasang yang sesuaiAWSSDK. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

Untuk mendeteksi atau menganalisis teks dalam dokumen multipage

1. Konfigurasikan akses pengguna ke Amazon Textract Texact, dan konfigurasikan akses Amazon Textract Texact ke Amazon SNS. Untuk informasi selengkapnya, lihat Mengkonfigurasi Amazon Textract untuk Operasi Asynchronous. Untuk menyelesaikan prosedur ini, Anda memerlukan file dokumen multipage dalam format PDF. Lewati langkah 3 — 6 karena contoh kode membuat dan mengonfigurasi topik Amazon SNS dan antrean Amazon SQS. Jika completDalam contoh CLI, Anda tidak perlu mengatur antrian SQS.
2. Unggah file dokumen multipage dalam format PDF atau TIFF ke bucket Amazon S3. (Dokumen satu halaman dalam format JPEG, PNG, TIFF, atau PDF juga dapat diproses).

   Untuk instruksi, lihatMengunggah objek ke Amazon S3diPanduan Pengguna Amazon Simple Storage Service.
3. Gunakan hal berikutAWS SDK untuk Java, SDK for Python (Boto3), atauAWS CLIkode untuk mendeteksi teks atau menganalisis teks dalam dokumen multipage. `Dimain`Fungsi:
   - Ganti nilai`roleArn`dengan peran IAM ARN yang Anda simpanMemberikan Amazon Textract Akses ke Topik Amazon SNS Anda.
   - Ganti nilai`bucket`dan`document`dengan nama file bucket dan dokumen yang Anda tentukan pada langkah 2.
   - Ganti nilai`type`parameter masukan dari`ProcessDocument`berfungsi dengan jenis pengolahan yang ingin Anda lakukan. Gunakan`ProcessType.DETECTIO`Untuk mendeteksi teks. Gunakan`ProcessType.ANALYSI`Suntuk menganalisis teks.

- Untuk contoh Python, ganti nilai`region_name`dengan wilayah klien Anda beroperasi di.

UntukAWS CLIcontoh, lakukan hal berikut:

- Saat menelepon[StartDocumentTextDetection](), ganti nilai`bucket-name`dengan nama bucket S3 Anda, dan ganti`file-name`dengan nama file yang Anda tentukan pada langkah 2. Tentukan wilayah bucket Anda dengan menggantiregion-namedengan nama wilayah Anda. Perhatikan bahwa contoh CLI tidak menggunakan SQS.
- Saat menelepon[GetDocumentTextDetection]()menggantikan`job-id-number`dengan`job-id`dikembalikan oleh[StartDocumentTextDetection](). Tentukan wilayah bucket Anda dengan menggantiregion-namedengan nama wilayah Anda.

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

IniAWS CLImemulai deteksi asinkron teks dalam dokumen tertentu. Ini menghasilkan`job-id`yang dapat digunakan untuk retreive hasil deteksi.

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

IniAWS CLIperintah mengembalikan hasil untuk operasi asinkron Amazon Textract bila disediakan dengan`job-id`.

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

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda bukan tanda kutip tunggal dan melarikan diri tanda kutip ganda dalam dengan garis miring terbalik (yaitu\) untuk mengatasi kesalahan parser yang mungkin Anda hadapi. Sebagai contoh, lihat di bawah

```
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

Dalam contoh ini, ganti nilai`roleArn`dengan peran IAM ARN yang Anda simpanMemberikan Amazon Textract Akses ke Topik Amazon SNS Anda. Ganti nilai`bucket`dan`document`dengan nama file bucket dan dokumen yang Anda tentukan pada langkah 2 di atas. Ganti nilai`processType`dengan jenis pemrosesan yang ingin Anda gunakan pada dokumen input. Akhirnya, ganti nilai`REGION`dengan wilayah klien Anda beroperasi di.

```
 // 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. Jalankan kode tersebut. Operasi mungkin membutuhkan waktu beberapa saat untuk menyelesaikan. Setelah selesai, daftar blok untuk teks yang terdeteksi atau dianalisis akan ditampilkan.

Pemberitahuan Amazon Textract

Amazon Textract menerbitkan hasil permintaan analisis Amazon Textract, termasuk status penyelesaian, ke topik Amazon Simple Notification Service (Amazon SNS). Untuk mendapatkan notifikasi dari topik Amazon SNS, gunakan antrean Amazon SQS atauAWS Lambdafungsi. Untuk informasi selengkapnya, lihat Memanggil Operasi Asinkron Amazon Texact. Sebagai contoh, lihat Mendeteksi atau Menganalisis Teks dalam Dokumen Multipage.

Hasilnya memiliki format JSON berikut:

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

Tabel ini menjelaskan parameter yang berbeda dalam respons Amazon Textract.

Parameter	Deskripsi
JobId	Pengenal unik yang diberikan Amazon Textract untuk pekerjaan itu. Cocok dengan pengidentifikasi tugas yang dikembalikan dari`Start`operasi, seperti StartDocumentTextDetection.
Status	Status tugas. Nilai yang valid adalah Berhasil, Gagal, atau Kesalahan.

Parameter	Deskripsi
API	Operasi Amazon Textract Texact yang digunakan untuk menganalisis dokumen input, seperti StartDocumentTextDetection atau StartDocumentAnalysis.
JobTag	Pengidentifikasi yang ditentukan pengguna untuk tugas tersebut. Anda menentukan `JobTag` dalam panggilan ke `Start` operasi, seperti StartDocumentTextDetection.
Timestamp	The Unix timestamp yang menunjukkan ketika pekerjaan selesai, kembali dalam milidetik.
DokumenLokasi	Detail tentang dokumen yang telah diproses. Mencakup nama file dan bucket Amazon S3 tempat file disimpan.

Penanganan Throttled Call dan Dropped Connections

Operasi Amazon Textract Texact dapat gagal jika Anda melebihi jumlah maksimum transaksi per detik (TPS), yang menyebabkan layanan untuk throttle aplikasi Anda, atau ketika koneksi Anda turun. Misalnya, jika Anda membuat terlalu banyak panggilan ke operasi Amazon Texact dalam waktu singkat, panggilan akan mengurangi panggilan Anda dan mengirim`ProvisionedThroughputExceededException`kesalahan dalam respon operasi. Untuk informasi selengkapnya tentang kuota Amazon Textract TPS, lihatAmazon Textract.

Anda dapat mengelola throttling dan drop koneksi dengan secara otomatis mencoba kembali operasi. Anda dapat menentukan jumlah retries dengan memasukkan`Config`parameter saat Anda membuat klien Amazon Textract. Kami merekomendasikan hitungan coba lagi dari 5. ParameterAWSSDK mencoba ulang operasi jumlah tertentu kali sebelum gagal dan melempar pengecualian. Untuk informasi selengkapnya, lihatPengulangan Kesalahan dan Backoff Eksponensial di AWS.

> **Note**
>
> Coba ulang otomatis bekerja untuk operasi sinkron dan asinkron. Sebelum menentukan percobaan ulang otomatis, pastikan Anda memiliki SDK AWS versi terbaru. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

Contoh berikut menunjukkan cara mencoba ulang operasi Amazon Textract secara otomatis saat Anda memproses beberapa dokumen.

Prasyarat

- Jika belum:

  a. Buat atau perbarui pengguna IAM dengan izin `AmazonTextractFullAccess` dan `AmazonS3ReadOnlyAccess`. Untuk informasi selengkapnya, lihat Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM.

  b. Instal dan konfigurasikan SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

Untuk secara otomatis mencoba lagi operasi

1. Unggah beberapa gambar dokumen ke bucket S3 Anda untuk menjalankan contoh Synchronous. Unggah dokumen multi-halaman ke bucket S3 Anda dan jalankan`StartDocumentTextDetection`di atasnya untuk menjalankan contoh Asynchronous.

   Untuk instruksi, lihatMengunggah Objek ke Amazon S3di dalamPanduan Pengguna Amazon Simple Storage.
2. Contoh-contoh berikut ini menunjukkan cara menggunakan`Config`parameter untuk secara otomatis mencoba lagi operasi. Contoh Synchronous memanggil`DetectDocumentText`operasi, sedangkan contoh Asynchronous memanggil`GetDocumentTextDetection`operasi.

   Sync Example

   Gunakan contoh berikut untuk memanggil`DetectDocumentText`operasi pada dokumen dalam bucket Amazon S3. Masuk`main`, mengubah nilai`bucket`ke bucket S3 Anda. Ubah nilai`documents`untuk nama-nama gambar dokumen yang Anda upload pada langkah 2.

```
import boto3
from botocore.client import Config
# Documents

def process_multiple_documents(bucket, documents):

    config = Config(retries = dict(max_attempts = 5))

    # Amazon Textract client
    textract = boto3.client('textract', config=config)

    for documentName in documents:

        print("\nProcessing:
 {}\n=========================================".format(documentName))

        # Call Amazon Textract
        response = textract.detect_document_text(
            Document={
                'S3Object': {
                    'Bucket': bucket,
                    'Name': documentName
                }
            })
```

```
        # Print detected text
        for item in response["Blocks"]:
            if item["BlockType"] == "LINE":
                print ('\033[94m' +  item["Text"] + '\033[0m')


def main():
    bucket = ""
    documents = ["document-image-1.png",
    "document-image-2.png", "document-image-3.png",
    "document-image-4.png", "document-image-5.png" ]
    process_multiple_documents(bucket, documents)



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

Async Example

Gunakan contoh berikut untuk memanggil operasi `GetDocumentTextDetection`. Ini mengasumsikan Anda telah meneleponStartDocumentTextDetectionpada dokumen di bucket Amazon S3 dan memperolehJobId. Masukmain, mengubah nilaibucketbucket S3 Anda dan nilairoleArnke Arn ditugaskan untuk peran Textract Anda. Anda juga harus mengubah nilaidocumentdengan nama dokumen multi-halaman di bucket Amazon S3. Akhirnya, ganti nilairegion_namedengan nama wilayah Anda dan memberikanGetResultsberfungsi dengan namajobId.

```
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()
```

Praktik Terbaik untuk Amazon Textract

Amazon Textract menggunakan machine learning untuk membaca dokumen seperti yang dilakukan seseorang. Ini ekstrak teks, tabel, dan bentuk dari dokumen. Gunakan praktik terbaik berikut untuk mendapatkan hasil terbaik dari dokumen Anda.

Menyediakan Dokumen Input Optimal

Berikut ini adalah daftar beberapa cara agar Anda dapat mengoptimalkan dokumen masukan Anda untuk hasil yang lebih baik.

- Pastikan bahwa teks dokumen Anda dalam bahasa yang didukung Amazon Textract. Saat ini, Amazon Textract Texact mendukung bahasa Inggris, Spanyol, Jerman, Italia, Prancis, dan Portugis.
- Berikan gambar berkualitas tinggi, idealnya setidaknya 150 DPI.
- Jika dokumen Anda sudah berada dalam salah satu format file yang didukung Amazon Textract (PDF, TIFF, JPEG, dan PNG), jangan mengonversi atau downsample dokumen sebelum mengunggahnya ke Amazon Textract.

Untuk hasil terbaik saat mengekstrak teks dari tabel dalam dokumen, pastikan bahwa:

- Tabel dalam dokumen Anda secara visual dipisahkan dari elemen sekitarnya pada halaman. Misalnya, tabel tidak dilapis ke gambar atau pola kompleks.
- Teks dalam tabel adalah tegak. Misalnya, teks tidak diputar relatif terhadap teks lain pada halaman.

Saat mengekstrak teks dari tabel, Anda mungkin melihat hasil yang tidak konsisten saat:

- Sel tabel gabungan yang menjangkau beberapa kolom.
- Tabel dengan sel, baris, atau kolom yang berbeda dari bagian lain dari tabel yang sama.

Kami merekomendasikan penggunaanPendeteksi teksebagai solusi.

Gunakan Skor Keyakinan

Anda harus mempertimbangkan skor kepercayaan yang dikembalikan oleh operasi Amazon Textract API dan sensitivitas kasus penggunaannya. Skor kepercayaan adalah angka antara 0 dan 100 yang menunjukkan probabilitas bahwa prediksi yang diberikan benar. Ini membantu Anda membuat keputusan tentang bagaimana Anda menggunakan hasilnya.

Dalam aplikasi yang sensitif terhadap kesalahan deteksi (false positives), menegakkan ambang batas skor kepercayaan minimum. Aplikasi harus membuang hasil di bawah ambang batas atau situasi bendera yang membutuhkan tingkat pengawasan manusia yang lebih tinggi.

Ambang optimal tergantung pada aplikasi. Untuk tujuan arsip, seperti mendokumentasikan catatan tulisan tangan, mungkin serendah 50%. Proses bisnis yang melibatkan keputusan keuangan mungkin memerlukan ambang batas 90% atau lebih tinggi.

Pertimbangkan Menggunakan Tinjauan Manusia

Juga pertimbangkan untuk memasukkan tinjauan manusia ke dalam alur kerja Anda. Hal ini sangat penting untuk aplikasi sensitif, seperti proses bisnis yang melibatkan keputusan keuangan.

Tutorial

the section called “Block”objek yang dikembalikan dari operasi Amazon Textract Texact berisi hasil deteksi teks dan operasi analisis teks, sepertithe section called “AnalyzeDocument”. Tutorial Python berikut menunjukkan beberapa cara yang berbeda yang dapat Anda pakai untuk menggunakan objek Block. Misalnya, Anda dapat mengekspor informasi tabel ke file nilai yang dipisahkan koma (CSV).

Tutorial menggunakan operasi Amazon Textract sinkron yang mengembalikan semua hasil. Jika Anda ingin menggunakan operasi asinkron sepertithe section called “StartDocumentAnalysis”, Anda perlu mengubah kode contoh untuk mengakomodasi beberapa batch yang dikembalikan`Block`benda. Untuk memanfaatkan contoh operasi asinkron, pastikan bahwa Anda telah mengikuti instruksi yang diberikan diMengkonfigurasi Amazon Textract untuk Operasi Asynchronous.

Untuk contoh yang menunjukkan cara lain untuk menggunakan Amazon Textract, lihatSampel Kode Tambahan.

Topik

- Prasyarat
- Mengekstrak Pasangan Ky-Value dari Dokumen Formulir
- Mengekspor Tabel ke File CSV
- MembuatAWS LambdaFungsi
- Sampel Kode Tambahan

Prasyarat

Sebelum Anda dapat menjalankan contoh di bagian ini, Anda harus mengonfigurasi lingkungan Anda.

Untuk mengonfigurasi lingkungan

1. Buat atau perbarui pengguna IAM dengan izin `AmazonTextractFullAccess`. Untuk informasi selengkapnya, lihat Langkah 1: Siapkan Akun AWS dan Buat Pengguna IAM.
2. Instal dan konfigurasikan SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat Langkah 2: MenyiapkanAWS CLIdanAWSSDK.

Mengekstrak Pasangan Ky-Value dari Dokumen Formulir

Contoh Python berikut menunjukkan bagaimana untuk mengekstrak pasangan kunci-nilai dalam dokumen formulir dari the section called “Block” benda yang disimpan dalam peta. Blok objek dikembalikan dari panggilan ke the section called “AnalyzeDocument”. Untuk informasi selengkapnya, lihat Data Formulir (Pasangan Nilai Kunci).

Anda menggunakan fungsi-fungsi berikut:

- `get_kv_map` — Panggilan AnalyzeDocument, dan menyimpan objek KEY dan VALUE BLOCK dalam peta.
- `get_kv_relationship` dan `find_value_block` — Membangun hubungan kunci-nilai dari peta.

Untuk mengekstrak pasangan kunci-nilai dari dokumen formulir

1. Konfigurasikan lingkungan Anda. Untuk informasi selengkapnya, lihat Prasyarat.
2. Simpan kode contoh berikut ini ke sebuah file dengan nama `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. Di prompt perintah, masukkan perintah berikut. Ganti`file`dengan file gambar dokumen yang ingin Anda analisis.

```
textract_python_kv_parser.py file
```

4. Saat Anda diminta, masukkan kunci yang ada di dokumen input. Jika kode mendeteksi kunci, itu akan menampilkan nilai kunci.

Mengekspor Tabel ke File CSV

Contoh Python ini menunjukkan cara untuk mengekspor tabel dari gambar dokumen ke file nilai yang dipisahkan koma (CSV).

Contoh untuk analisis dokumen sinkron mengumpulkan informasi tabel dari panggilan kethe section called “AnalyzeDocument”. Contoh untuk analisis dokumen asinkron membuat panggilan kethe section called “StartDocumentAnalysis”dan kemudian mengambil hasil darithe section called “GetDocumentAnalysis”sebagai`Block`benda.

Informasi tabel dikembalikan sebagaithe section called “Block”objek dari panggilan kethe section called “AnalyzeDocument”. Untuk informasi selengkapnya, lihat Tabel. Parameter`Block`objek disimpan dalam struktur peta yang digunakan untuk mengekspor data tabel ke dalam file CSV.

Synchronous

Dalam contoh ini, Anda akan menggunakan fungsi:

- `get_table_csv_results`— PanggilanAnalyzeDocument, dan membangun peta tabel yang terdeteksi dalam dokumen. Menciptakan representasi CSV dari semua tabel terdeteksi.
- `generate_table_csv`— Menghasilkan file CSV untuk tabel individu.
- `get_rows_columns_map`— Mendapat baris dan kolom dari peta.
- `get_text`Mendapat teks dari sel.

Mengekspor tabel ke dalam file CSV

1. Konfigurasikan lingkungan Anda. Untuk informasi selengkapnya, lihat Prasyarat.
2. Simpan kode contoh berikut ini ke sebuah file dengan namatextract_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. Di prompt perintah, masukkan perintah berikut. Ganti`file`dengan nama file gambar dokumen yang ingin Anda analisis.

```
python textract_python_table_parser.py file
```

Ketika Anda menjalankan contoh, output CSV disimpan dalam file bernama`output.csv`.

Asynchronous

Dalam contoh ini, Anda akan menggunakan make menggunakan dua script yang berbeda. Script pertama memulai proses asynchronoulsy menganalisis dokumen dengan`StartDocumentAnalysis`dan mendapat`Block`informasi yang dikembalikan oleh`GetDocumentAnalysis`. Skrip kedua mengambil kembali`Block`informasi untuk setiap halaman, memformat data sebagai tabel, dan menyimpan tabel ke file CSV.

Mengekspor tabel ke dalam file CSV

1. Konfigurasikan lingkungan Anda. Untuk informasi selengkapnya, lihat Prasyarat.
2. Pastikan bahwa Anda telah mengikuti instruksi yang diberikan di lihatMengkonfigurasi Amazon Textract untuk Operasi Asynchronous. Proses yang didokumentasikan pada halaman tersebut memungkinkan Anda mengirim dan menerima pesan tentang status penyelesaian pekerjaan asinkron.
3. Pada contoh kode berikut, ganti nilai`roleArn`dengan Arn yang ditetapkan untuk peran yang Anda buat di Langkah 2. Ganti nilai`bucket`dengan nama bucket S3 yang mengandung dokumen Anda. Ganti nilai`document`dengan nama dokumen di bucket S3. Ganti nilai`region_name`Dengan nama wilayah bucket Anda.

Simpan kode contoh berikut ini ke sebuah file dengan namastart_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. Jalankan kode tersebut. Kode akan mencetak JobId. Salin JobId ini ke bawah.
5. Tunggu sampai pekerjaan Anda selesai diproses, dan setelah selesai, salin kode berikut ke file bernamaget_doc_analysis_for_table_extraction.py. Ganti nilai`jobId`dengan ID Job yang Anda salin sebelumnya. Ganti nilai`region_name`dengan nama wilayah yang terkait dengan peran Textract Anda. Ganti nilai`file_name`dengan nama yang ingin Anda berikan output CSV.

```
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. Jalankan kode tersebut.

   Setelah Anda memperoleh hasil, pastikan untuk menghapus sumber daya SNS dan SQS terkait, atau Anda dapat memperoleh biaya untuk mereka.

MembuatAWS LambdaFungsi

Anda dapat memanggil operasi Amazon Textract API dari dalamAWS Lambdafungsi. Petunjuk berikut menunjukkan cara membuat fungsi Lambda dengan Python yang memanggilthe section called “DetectDocumentText”. Ia mengembalikan daftarthe section called “Block”benda. Untuk menjalankan contoh ini, Anda memerlukan bucket Amazon S3 yang berisi dokumen dalam format PNG atau JPEG. Untuk membuat fungsi, Anda menggunakan konsol.

Untuk contoh yang menggunakan fungsi Lambda untuk memproses dokumen dalam skala besar, lihatPemrosesan dokumen berskala besar dengan Amazon Textract.

Untuk memanggil operasi DetectDocumentText dari fungsi Lambda:

Langkah 1: Membuat paket deployment Lambda

1. Buka jendela perintah.
2. Masukkan perintah berikut ini untuk membuat paket penyebaran dengan versi terbaruAWSSDK.

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

Langkah 2: Buat fungsi Lambda

1. Masuk ke Konsol Manajemen AWS dan buka konsol AWS Lambda di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi.
3. Tentukan hal berikut.

   - Pilih Penulis dari scratch.

- UntukNama fungsi, masukkan sebuah nama.
- UntukWaktu pengoperasian, pilihPython 3.7atauPython 3.6.
- UntukMemiilih atau membuat peran eksekusi, pilihMembuat peran baru dengan izin Lambda dasar.

4. MemiilihMembuat fungsiuntuk membuat fungsi Lambda.
5. Buka konsol IAM di https://console.aws.amazon.com/iam/.
6. Pilih panel navigasiPeran.
7. Dari daftar sumber daya, pilih peran IAM yang dibuat Lambda untuk Anda. Nama peran dimulai dengan nama fungsi Lambda Anda.
8. PilihIzintab, lalu pilihLampirkan kebijakan.
9. Pilih Kebijakan AmazonTextractFullAccess dan Amazons3ReadOnlyAccess.
10. PilihLampirkan kebijakan.

Untuk informasi selengkapnya, lihatMembuat Fungsi Lambda dengan Konsol

Langkah 3: Membuat dan menambahkan layer

1. Buka konsol AWS Lambda tersebut di https://console.aws.amazon.com/lambda/.
2. Di panel navigasi, pilih Layers (Lapisan).
3. Pilih Buat lapisan.
4. UntukNama, masukkan sebuah nama.
5. Untuk Deskripsi, masukkan deskripsi.
6. UntukJenis entri kode, pilihUnggah file .zipdan pilihUnggah.
7. Di kotak dialog, pilih file zip (boto3-layer.zip), zip yang Anda buatLangkah 1: Membuat paket deployment Lambda.
8. UntukRuntime yang kompatibel, pilih versi runtime yang Anda pilihLangkah 2: Buat fungsi Lambda.
9. MemiilihBuatuntuk membuat lapisan.
10. Pilih ikon menu panel navigasi.
11. Di panel navigasi, pilih Fungsi.
12. Dalam daftar sumber daya, pilih fungsi yang Anda buatLangkah 2: Buat fungsi Lambda.
13. MemiilihKonfigurasidan diDesainerbagian, pilihLapisan(di bawah nama fungsi Lambda Anda).

14. DiLapisanbagian, pilihTambahkan lapisan.
15. MemiilihPilih dari daftar layer yang kompatibel runtime.
16. MasukLapisan yang kompatibel, pilihNamadanVersilapisan yang Anda buat pada langkah 3.
17. Pilih Tambahkan.

Langkah 4: Tambahkan kode python ke Fungsi

1. MasukDesainer, pilih fungsi Anda.
2. Dalam editor kode fungsi, tambahkan yang berikut ke filelambda_function.py. Mengubah nilai-nilai`bucket`dan`document`ke ember dan dokumen.

```
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. MemiilihSimpanuntuk menyimpan fungsi Lambda Anda.

Langkah 5: Pengujian Lambda

1. PilihPengujian.

2. Masukkan nilai untukNama peristiwa.
3. Pilih Create (Buat).
4. Output, daftarthe section called “Block”objek, muncul di panel hasil Eksekusi.

JikaAWS Lambdafungsi mengembalikan kesalahan batas waktu, panggilan operasi API Amazon Textract mungkin penyebabnya. Untuk informasi tentang memperpanjang periode timout untukAWS Lambdafungsi, lihatKonfigurasi Fungsi AWS Lambda.

Untuk informasi tentang meminta fungsi Lambda dari kode Anda, lihatMemanggilAWS LambdaFungsi.

Sampel Kode Tambahan

Tabel berikut menyediakan tautan ke contoh kode Amazon Textract.

Contoh	Deskripsi
Sampel Kode Amazon Textract	Tampilkan berbagai cara di mana Anda dapat menggunakan Amazon Textract.
Pemrosesan dokumen berskala besar dengan Amazon Textract	Menunjukkan arsitektur referensi tanpa server yang memproses dokumen dalam skala besar.
Parser Amazon Textract	Menunjukkan bagaimana menguraithe section called “Block”objek yang dikembalikan oleh operasi Amazon Textract Texact.
Contoh Kode Dokumentasi Amazon Textract	Contoh kode yang digunakan dalam panduan ini.
Textractor	Menunjukkan cara mengonversi output Amazon Textract menjadi beberapa format.
Hasilkan dokumen PDF yang dapat dicari dengan Amazon Textract	Menunjukkan cara membuat dokumen PDF yang dapat dicari dari berbagai jenis dokumen masukan seperti gambar format JPG/PNG dan dokumen PDF yang dipindai.

Contoh kode untuk Amazon Textract

Contoh kode berikut ini menunjukkan cara menggunakan Amazon TextractAWSperangkat lunak pengembangan kit (SDK).

Contoh dibagi ke dalam kategori berikut:

Tindakan

Kutipan kode yang menunjukkan cara memanggil fungsi layanan individual.

Contoh lintas layanan

Contoh aplikasi yang bekerja di beberapaAWSlayanan.

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Contoh kode

- Tindakan untuk Amazon Textract
  - Menganalisis dokumen menggunakan Amazon Textract Texact danAWSSDK
  - Mendeteksi teks dalam dokumen menggunakan Amazon Textract Texact danAWSSDK
  - Mendapatkan data tentang pekerjaan analisis Amazon Textract Texact menggunakanAWSSDK
  - Mulai analisis asinkron dokumen menggunakan Amazon Textract Texact danAWSSDK
  - Mulai deteksi teks asinkron menggunakan Amazon Textract Texact danAWSSDK
- Contoh lintas layanan untuk Amazon Textract
  - Membuat aplikasi penjelajah Amazon Textract Texact
  - Mendeteksi entitas dalam teks yang diekstrak dari gambar menggunakanAWSSDK

Tindakan untuk Amazon Textract

Contoh kode berikut ini mendemonstrasikan cara melakukan tindakan Amazon TextractAWSSDK. Kutipan ini memanggil API Amazon Textract dan tidak dimaksudkan untuk dijalankan secara terpisah. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan kode dalam konteks.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkap, lihatReferensi API Amazon Textract.

Contoh

- Menganalisis dokumen menggunakan Amazon Textract Texact danAWSSDK
- Mendeteksi teks dalam dokumen menggunakan Amazon Textract Texact danAWSSDK
- Mendapatkan data tentang pekerjaan analisis Amazon Textract Texact menggunakanAWSSDK
- Mulai analisis asinkron dokumen menggunakan Amazon Textract Texact danAWSSDK
- Mulai deteksi teks asinkron menggunakan Amazon Textract Texact danAWSSDK

Menganalisis dokumen menggunakan Amazon Textract Texact danAWSSDK

Contoh kode berikut ini menunjukkan cara menganalisis dokumen menggunakan Amazon Textract.

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);
        }
    }
```

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatAnalyzeDocumentdiAWS SDK for Java 2.xReferensi API.

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
```

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatAnalyzeDocumentdiAWSReferensi API SDK for Python (Boto3).

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Mendeteksi teks dalam dokumen menggunakan Amazon Textract Texact danAWSSDK

Contoh kode berikut ini menunjukkan cara mendeteksi teks dalam dokumen menggunakan Amazon Textract.

Java

SDK for Java 2.x

Mendeteksi teks dari dokumen masukan.

```
   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);
        }
    }
```

Mendeteksi teks dari dokumen yang terletak di bucket Amazon S3.

```
    public static void detectDocTextS3 (TextractClient textractClient, String
bucketName, String docName) {

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

            // Create a Document object and reference the s3Object instance
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            // Create a DetectDocumentTextRequest object
            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

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

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

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

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

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

        } catch (TextractException e) {

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

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatDetectDocumentTextdiAWS SDK for Java 2.xReferensi API.

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
```

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatDetectDocumentTextdiAWSReferensi API SDK for Python (Boto3).

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Mendapatkan data tentang pekerjaan analisis Amazon Textract Texact menggunakanAWSSDK

Contoh kode berikut ini menunjukkan cara mendapatkan data tentang pekerjaan analisis dokumen Amazon Textract.

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
```

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatGetDocumentAnalysisdiAWSReferensi API SDK for Python (Boto3).

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Mulai analisis asinkron dokumen menggunakan Amazon Textract Texact danAWSSDK

Contoh kode berikut ini menunjukkan cara memulai analisis asinkron dari dokumen menggunakan Amazon Textract.

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 "";
    }
```

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatStartDocumentAnalysisdiAWS SDK for Java 2.xReferensi API.

Python

SDK for Python (Boto3)

Mulai pekerjaan asinkron untuk menganalisis dokumen.

```
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
```

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatStartDocumentAnalysisdiAWSReferensi API SDK for Python (Boto3).

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Mulai deteksi teks asinkron menggunakan Amazon Textract Texact danAWSSDK

Contoh kode berikut menunjukkan cara memulai deteksi teks asinkron dalam dokumen menggunakan Amazon Textract.

Python

SDK for Python (Boto3)

Mulai pekerjaan asinkron untuk mendeteksi teks dalam dokumen.

```
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
```

- Temukan instruksi dan kode lainnyaGitHub.
- Untuk rincian selengkapnya, lihatStartDocumentTextDetectiondiAWSReferensi API SDK for Python (Boto3).

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Contoh lintas layanan untuk Amazon Textract

Contoh aplikasi berikut digunakanAWSSDK untuk menggabungkan Amazon Textract dengan yang lainAWSlayanan. Setiap contoh termasuk link ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan aplikasi.

Contoh

- Membuat aplikasi penjelajah Amazon Textract Texact
- Mendeteksi entitas dalam teks yang diekstrak dari gambar menggunakanAWSSDK

Membuat aplikasi penjelajah Amazon Textract Texact

Contoh kode berikut ini menunjukkan cara mengeksplorasi output Amazon Textract melalui aplikasi interaktif.

JavaScript

SDK for JavaScript V3

Menunjukkan cara menggunakanAWS SDK untuk JavaScriptuntuk membangun aplikasi React yang menggunakan Amazon Textract untuk mengekstrak data dari gambar dokumen dan menampilkannya di halaman web interaktif. Contoh ini berjalan di browser web dan memerlukan identitas Amazon Cognito yang diautentikasi untuk kredensi. Amazon Simple Storage Service (Amazon S3) untuk penyimpanan, dan untuk notifikasi, Amazon Simple Queue Service (Amazon SQS) yang berlangganan topik Amazon Simple Notification Service (Amazon SNS).

Untuk kode sumber lengkap dan petunjuk tentang cara mengatur dan menjalankan, lihat contoh lengkap diGitHub.

Layanan yang digunakan dalam contoh ini

- Identitas Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

Menunjukkan cara menggunakanAWS SDK untuk Python (Boto3)dengan Amazon Textract untuk mendeteksi elemen teks, formulir, dan tabel dalam gambar dokumen. Gambar input dan output Amazon Textract ditampilkan dalam aplikasi Tkinter yang memungkinkan Anda menjelajahi elemen yang terdeteksi.

- Kirim gambar dokumen ke Amazon Textract dan jelajahi output elemen yang terdeteksi.
- Mengirimkan gambar langsung ke Amazon Textract atau melalui bucket Amazon Simple Storage Service (Amazon S3).
- Gunakan API asinkron untuk memulai tugas yang menerbitkan notifikasi ke topik Amazon Simple Notification Service (Amazon SNS) saat tugas selesai.
- Antrian Amazon Simple Queue Service (Amazon SQS) untuk pesan penyelesaian pekerjaan dan menampilkan hasilnya.

Untuk kode sumber lengkap dan petunjuk tentang cara mengatur dan menjalankan, lihat contoh lengkap diGitHub.

Layanan yang digunakan dalam contoh ini

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

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Mendeteksi entitas dalam teks yang diekstrak dari gambar menggunakanAWSSDK

Contoh kode berikut menunjukkan cara menggunakan Amazon Comprehend untuk mendeteksi entitas dalam teks yang diekstraksi oleh Amazon Textract Texact dari gambar yang disimpan di Amazon S3.

Python

SDK for Python (Boto3)

Menunjukkan cara menggunakanAWS SDK untuk Python (Boto3)dalam notebook Jupyter untuk mendeteksi entitas dalam teks yang diekstraksi dari gambar. Contoh ini menggunakan Amazon Textract untuk mengekstrak teks dari gambar yang disimpan di Amazon Simple Storage Service (Amazon S3) dan Amazon Comprehend untuk mendeteksi entitas dalam teks yang diekstraksi.

Contoh ini adalah notebook Jupyter dan harus dijalankan di lingkungan yang dapat menjadi tuan rumah notebook. Untuk petunjuk tentang cara menjalankan contoh menggunakan Amazon SageMaker, lihat petunjukTextAndComprehendNotebook.ipynb.

Untuk kode sumber lengkap dan petunjuk tentang cara mengatur dan menjalankan, lihat contoh lengkap diGitHub.

Layanan yang digunakan dalam contoh ini

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Untuk daftar lengkapAWSPanduan pengembang SDK dan contoh kode, lihatMenggunakan Amazon Textract Texact denganAWSSDK. Topik ini juga mencakup informasi tentang memulai dan rincian tentang versi SDK sebelumnya.

Menggunakan Amazon Augmented AI untuk Menambahkan Tinjauan Manusia ke Output Amazon Textract

Amazon Augmented AI (Amazon A2I) adalah layanan machine learning (ML) yang memudahkan pembuatan alur kerja untuk peninjauan manual analisis ML-nya.

Amazon Textract Texact terintegrasi dengan Amazon A2I. Anda dapat menggunakannya untuk merutekan hasil analisis dokumen yang memiliki skor kepercayaan rendah kepada pengulas manusia.

Anda dapat menggunakan Amazon Textract`AnalyzeDocument`API untuk mengekstrak data dari formulir dan konsol Amazon A2I. Anda dapat menentukan kondisi di mana Amazon A2I merutekan prediksi kepada pengulas. Anda menetapkan kondisi berdasarkan ambang kepercayaan kunci bentuk penting. Misalnya, Anda dapat mengirim dokumen ke peninjau manusia jika kuncinya`Nama`atau nilai yang terkait`Jane`terdeteksi dengan kepercayaan rendah.

Topik

Konsep Inti Amazon A2I

Tinjau persyaratan berikut untuk membiasakan diri dengan konsep inti Amazon A2I.

Kondisi Aktivasi Tinjauan Manusia

Anda dapat menggunakan Amazon A2I*kondisi aktivasi*untuk menentukan kapan dokumen dikirim ke manusia untuk ditinjau dan bentuk-konten yang pekerja diminta untuk meninjau.

Misalnya, Anda dapat mengatur kondisi aktivasi agar formulir rute Amazon Textract ke Amazon A2I ketika kunci penting terdeteksi dengan tingkat kepercayaan rendah, seperti`Nomor Telepon`. Dalam contoh ini, pengulas manusia diminta untuk meninjau`Nomor Telepon`bidang dan nilai yang terkait terdeteksi oleh Amazon Textract.

Anda dapat menggunakan kondisi aktivasi berikut untuk menentukan kapan formulir dikirim ke manusia untuk ditinjau:

- Memicu tinjauan manusia untuk kunci formulir tertentu berdasarkan skor kepercayaan kunci formulir. Peninjau manusia diminta untuk meninjau kunci formulir ini dan nilai terkait.
- Memicu tinjauan manusia ketika kunci formulir tertentu hilang. Peninjau manusia diminta untuk mengidentifikasi kunci formulir ini dan nilai-nilai terkait.
- Memicu tinjauan manusia untuk semua kunci formulir yang diidentifikasi oleh Amazon Textract dengan skor percaya diri dalam kisaran tertentu.
- Secara acak mengirim sampel formulir ke manusia untuk ditinjau. Peninjau manusia diminta untuk meninjau semua kunci formulir dan nilai yang terdeteksi oleh Amazon Textract.

Ketika kondisi aktivasi Anda tergantung pada skor kepercayaan kunci bentuk, Anda dapat menggunakan dua jenis ambang kepercayaan prediksi untuk memicu tinjauan manusia:

- kepercayaan identifikasi— Skor kepercayaan untuk pasangan kunci-nilai terdeteksi dalam bentuk.
- kepercayaan kualifikasi— Skor kepercayaan untuk teks yang terkandung dalam pasangan kunci-nilai dalam bentuk.

Jika Anda menentukan ambang kepercayaan, Amazon A2I hanya merutekan prediksi yang termasuk dalam ambang batas untuk pengulas manusia. Anda dapat menyesuaikan ambang batas ini kapan saja untuk mencapai keseimbangan yang tepat antara akurasi dan efektivitas biaya. Hal ini dapat membantu Anda menerapkan audit untuk secara teratur memantau akurasi prediksi.

> **Note**
>
> Anda dapat menyesuaikan lebih lanjut kondisi di mana dokumen dikirim ke manusia untuk ditinjau menggunakan jenis tugas kustom Amazon A2I. Dengan jenis tugas ini, Anda menentukan kondisi untuk tinjauan manusia secara langsung di aplikasi Anda. Untuk informasi, lihatMenggunakan Amazon Augmented AI dengan Jenis Tugas Kustomdi Panduan Pengembang Amazon SageMaker.

Alur kerja tinjauan manusia (definisi aliran)

Anda menggunakanalur kerja tinjauan manusia, juga disebut sebagaidefinisi alur, untuk menentukan sumber daya yang digunakan untuk membuat alur kerja tinjauan manusia Anda, dan untuk menentukan kondisi aktivasi Anda.

Sumber daya yang Anda tentukan adalah:

- Peran IAM dengan izin untuk memanggil operasi API Amazon A2I
- Bucket Amazon S3 tempat Anda ingin menyimpan output peninjauan manual
- Manusia Andatim kerja
- SEBUAHTemplat tugas pekerjayang mencakup instruksi dan contoh untuk membantu pekerja menyelesaikan tugas peninjauan

Anda juga menggunakan alur kerja tinjauan manusia untuk menentukan kondisi aktivasi. Untuk informasi selengkapnya, lihat Kondisi Aktivasi Tinjauan Manusia.

Anda dapat menggunakan alur kerja tinjauan manusia tunggal untuk membuat beberapaLoop manusia.

Anda dapat membuat alur kerja tinjauan manusia di konsol SageMaker atau dengan SageMaker API. Untuk informasi, lihat Buat Alur Kerja Tinjauan Manusia.

Templat tugas pekerja

Anda menggunakanTemplat tugas pekerjauntuk membuat UI pekerja yang digunakan untuk tugas peninjauan manusia Anda.

UI pekerja menampilkan dokumen dan instruksi pekerja Anda. Ini juga menyediakan alat yang digunakan pekerja untuk menyelesaikan tugas Anda.

Anda dapat menggunakan konsol SageMaker untuk mengonfigurasi templat tugas pekerja saat Anda membuat alur kerja peninjauan manual. Untuk informasi, lihat Buat Alur Kerja Tinjauan Manusia.

Tim kerja

SEBUAHtim kerjaadalah sekelompok pekerja manusia yang Anda kirimkan tugas peninjauan manusia Anda.

Ketika Anda membuat alur kerja tinjauan manusia, Anda menentukan satu tim kerja.

Dengan Amazon A2I, Anda dapat menggunakan kolam penampil dalam organisasi Anda sendiri. Anda juga dapat mengakses tenaga kerja yang terdiri dari lebih dari 500.000 kontraktor independen yang sudah melakukan tugas machine learning melalui Amazon Mechanical Turk. Pilihan lainnya adalah menggunakan vendor tenaga kerja yang diprasaring oleh AWS untuk kualitas dan ketaatan terhadap prosedur keamanan.

Amazon A2I juga menyediakan pengulas antarmuka web yang terdiri dari semua instruksi dan alat yang mereka butuhkan untuk menyelesaikan tugas peninjauan mereka.

Untuk setiap jenis tenaga kerja (swasta, vendor, dan Mechanical Turk), Anda dapat membuat beberapa tim kerja. Anda dapat menggunakan setiap tim kerja dalam beberapa alur kerja tinjauan manusia. Untuk belajar membuat tenaga kerja dan tim kerja, lihatMembuat dan Mengelola Tenaga Kerjadi Panduan Pengembang Amazon SageMaker.

> **Important**
>
> Klikkemariuntuk melihat program kepatuhan yang mencakup Amazon Augmented AI saat ini. Jika Anda menggunakan Amazon Augmented AI bersama dengan layanan AWS lainnya (seperti Amazon Rekognition dan Amazon Textract Texact), harap perhatikan bahwa Amazon Augmented AI mungkin tidak berada dalam lingkup program kepatuhan yang sama dengan layanan lainnya. Anda bertanggung jawab atas cara Anda menggunakan Amazon Augmented AI, termasuk memahami bagaimana layanan akan memproses atau menyimpan data pelanggan, dan dampak apa pun terhadap kepatuhan terhadap lingkungan data Anda. Anda harus mendiskusikan tujuan dan sasaran beban kerja Anda dengan tim akun AWS Anda; mereka dapat membantu Anda mengevaluasi apakah layanan ini cocok untuk kasus penggunaan dan arsitektur yang diusulkan.
> Saat ini, Amazon Augmented AI mematuhi PCI kecuali untuk kasus tenaga kerja publik dan vendor.
> Untuk informasi mengenai kepatuhan HIPAA Augmented AI Amazon, klikkemari.

Loop manusia

Anda menggunakan loop manusia untuk membuat tugas tinjauan manusia.

Anda menetapkan tugas peninjauan manusia kepada pekerja di tim pekerja manusia Anda. Pekerja diminta untuk meninjau pasangan nilai kunci yang terdeteksi oleh Amazon Textract dalam dokumen masukan yang Anda tentukan dalam kondisi aktivasi Anda.

Misalnya, katakanlah gambar jatuh antara lima puluh dan enam puluh persen keyakinan bahwa itu mengandung apel. Ini termasuk dalam ambang kepercayaan Anda untuk tinjauan manusia dan dikirim ke pekerja. Pekerja memeriksa dokumen untuk apel, menandainya sebagai mengandung atau tidak mengandung apel. Kemudian, Amazon A2I mengirimkan dokumen kembali ke alur kerja.

Saat Anda memanggil Amazon Textract`AnalyzeDocument`dan tentukan alur kerja tinjauan manusia (definisi aliran) dan nama loop manusia, tugas tinjauan manusia dibuat ketika kondisi aktivasi yang ditentukan dalam alur kerja tinjauan manusia Anda terpenuhi. Tugas-tugas ini dibuat menggunakan sumber daya yang Anda tentukan dalam alur kerja tinjauan manusia Anda.

Memulai Menggunakan Amazon A2I

Langkah-langkah berikut membantu Anda mengintegrasikan Amazon A2I ke dalam tugas analisis dokumen satu halaman Amazon Texact. Anda melakukan hal berikut:

1. Buat alur kerja peninjauan manusia menggunakan konsol Amazon A2I (direkomendasikan untuk pengguna baru) atau API Amazon A2I.
2. Untuk menganalisis bentuk dan menyertakan tinjauan manusia bila diperlukan, gunakan`AnalyzeDocument`operasi dan tentukan Amazon Resource Name (ARN) alur kerja peninjauan manual. Tanggapan memberitahu Anda jika peninjauan manual diperlukan.
3. Pantau loop manusia Anda menggunakan konsol Amazon A2I dan API.
4. Tinjau hasil peninjauan manual di bucket Amazon S3 tempat hasilnya dikirim.

Untuk mengatur instance notebook SageMaker dan gunakan contoh notebook lihatDemo end-to-end Menggunakan Amazon Textract dan AI Augmenteddi Panduan Developer Amazon SageMaker

> **Note**
>
> Bagian ini menjelaskan cara membuat alur kerja tinjauan manusia untuk jenis tugas Amazon A2I, Amazon Textract Texact. Untuk lebih menyesuaikan integrasi Amazon A2I dan Amazon Textract, Anda dapat menggunakan jenis tugas kustom Amazon A2I. Dengan opsi ini, Anda menyediakan template tugas pekerja khusus dan menentukan kondisi di mana dokumen dikirim untuk tinjauan manusia secara langsung dalam aplikasi Anda. Untuk informasi, lihatMenggunakan Amazon Augmented AI dengan Jenis Tugas Kustomdi Panduan Developer Amazon SageMaker.

Topik

- Buat Alur Kerja Tinjauan Manusia
- Menganalisis Dokumen
- Loop Manusia
- Lihat Data Output dan Metrik Pekerja

Buat Alur Kerja Tinjauan Manusia

Anda dapat membuat alur kerja peninjauan manusia menggunakan konsol Amazon A2I (direkomendasikan untuk pengguna baru) atau Amazon A2I`CreateFlowDefinition`operasi.

Topik

- Membuat Alur Kerja Tinjauan Manusia (Konsol)
- Membuat Workflow Tinjauan Manusia (API)

Membuat Alur Kerja Tinjauan Manusia (Konsol)

Anda dapat menyelesaikan contoh ini menggunakan dokumen Anda sendiri di Amazon S3, atau Anda dapat mengunduhdokumen contoh inidan letakkan di bucket Amazon S3.

Pastikan bucket S3 Anda samaAWSWilayah yang Anda gunakan Amazon Textract. Untuk membuat bucket, lihatBuat BucketdiPanduan Pengguna Amazon Simple Storage Service.

> **Note**
>
> Konsol Amazon A2I disematkan di konsol SageMaker. Untuk menggunakan konsol, Anda memerlukan izin untuk mengakses konsol SageMaker, dan untuk membuat tim kerja. Untuk memulai, Anda dapat menggunakanAmazonSageMakerFullAccessKebijakan terkelola IAM yang mencakup semua izin yang diperlukan untuk melakukan sebagian besar tindakan di SageMaker. Untuk informasi selengkapnya, lihatIdentity and Access Management untuk Amazon SageMakerdiPanduan Developer Amazon SageMaker.

Topik

- Langkah 1: Buat Tim Kerja (Konsol)
- Langkah 2: Membuat Alur Kerja Tinjauan Manusia (Konsol)

Langkah 1: Buat Tim Kerja (Konsol)

Pertama, buat tim kerja di konsol Amazon A2I dan tambahkan diri Anda sebagai pekerja sehingga Anda dapat melihat pratinjau tugas peninjauan manusia di portal pekerja, anggota tim kerja dapat melihat berbagai tugas dan dokumen yang diberikan kepada mereka.

Untuk membuat tenaga kerja pribadi menggunakan email pekerja (Konsol)

1. Buka konsol SageMakerhttps://console.aws.amazon.com/sagemaker/.
2. Di panel navigasi, di bawahGround Truth, pilihTenaga kerja pelabelan.
3. MemiilihPrivat, kemudian pilihMembuat tim pribadi.
4. MemiilihUndang pekerja baru melalui email.
5. Untuk contoh ini, masukkan alamat email Anda dan alamat email orang lain yang ingin Anda dapat melihat pratinjau portal pekerja. Anda dapat menempelkan atau mengetik daftar hingga 50 alamat email, dipisahkan dengan koma, ke dalamAlamat Emailkotak.
6. Masukkan nama oneorganization dan email kontak.
7. MemiilihMembuat tim pribadi.

Jika Anda menambahkan diri Anda ke tim kerja pribadi, Anda menerima email dari`no-reply@verificationemail.com`dengan informasi login. Gunakan tautan di email ini untuk mengatur ulang kata sandi Anda dan masuk ke portal pekerja. Di sinilah tugas peninjauan manusia Anda akan muncul setelah Anda meneleponAnalyzeDocument.

Langkah 2: Membuat Alur Kerja Tinjauan Manusia (Konsol)

Pada langkah ini, Anda membuat alur kerja peninjauan manusia Amazon Textract.

Membuat alur kerja tinjauan manusia (konsol)

1. Buka konsol Amazon A2Ihttps://console.aws.amazon.com/a2iuntuk mengaksesAlur kerja manusiahalaman.
2. MemiilihBuat alur kerja tinjauan manusia.
3. UntukNama, masukkan nama alur kerja.
4. UntukBucket S3, pilih bucket tempat Anda ingin Amazon A2I menyimpan hasil tugas peninjauan manual Anda. Jika Anda tidak memilih bucket, ubah itu untuk memasukkan nama bucket.
5. Di bawahPeran IAM, pilihMembuat peran baru. Sebuah jendela muncul dengan judulBuat IAM role. Gunakan jendela ini untuk menentukan bucket Amazon S3 yang Anda inginkan agar peran ini memiliki akses. Jika Anda tidak memilihBucket S3, tentukan bucket output yang Anda tentukan pada langkah 4, dan bucket yang berisi dokumen input Anda.
6. UntukJenis tugas, pilihTextract - Ekstraksi pasangan kunci-nilai.
7. MasukEkstraksi bentuk Amazon Texact - Ketentuan untuk meminta tinjauan manusia, tentukan kondisi aktivasi. Kami menyarankan Anda menetapkan ambang skor kepercayaan tinggi untuk

setidaknya satu kunci dalam dokumen Anda untuk memicu tinjauan manusia sehingga Anda dapat melihat pratinjau tugas pekerja di portal pekerja.

Jika Anda menggunakan dokumen sampel yang disediakan dalam panduan ini, tentukan kondisi aktivasi sebagai berikut:

a. MemiilihMemicu tinjauan manusia untuk kunci formulir tertentu berdasarkan skor kepercayaan kunci formulir atau ketika kunci formulir tertentu hilang.
b. UntukNama Kunci, masukkan**`Mail Address`**.
c. Mengaturkepercayaan identifikasiambang batas antara0dan99.
d. Mengaturkepercayaan kualifikasiambang batas antara0dan99.
e. MemiilihMemicu tinjauan manusia untuk semua kunci formulir yang diidentifikasi oleh Amazon Textract dengan skor percaya diri dalam kisaran tertentu.
f. Untuk**`identification confidence`**, pilih nomor antara 0 dan 90.
g. Untuk**`qualification confidence`**, pilih nomor antara 0 dan 90.

Ini memicu tinjauan manusia jika Amazon Textract mengembalikan skor kepercayaan yang kurang dari 99 untukAlamat suratdan nilainya, atau jika ia mengembalikan skor kepercayaan kurang dari 90 untuk setiap pasangan kunci-nilai terdeteksi dalam dokumen.

8. Di bawahPenciptaan Template Tugas Pekerja, pilihMembuat dari template default.
9. UntukNama templat, masukkan nama deskriptif..
10. UntukDeskripsi tugas, tambahkan sesuatu yang serupa dengan yang berikut ini:

    `Read the instructions and review the document.`
11. UntukPekerjapilihPrivat.
12. Dari menu pilih tim pribadi yang Anda buat.
13. Pilih Buat.

Setelah alur kerja peninjauan manusia Anda dibuat, itu muncul di tabel padaAlur kerja manusiahalaman. SaatStatusadalahAktif, menyalin dan menyimpan alur kerja ARN.

Membuat Workflow Tinjauan Manusia (API)

Anda dapat membuat alur kerja tinjauan manusia, ataudefinisi alur, menggunakan Amazon A2I,`CreateFlowDefinition`operasi.

Untuk contoh ini, Anda dapat menggunakan dokumen Anda sendiri di Amazon S3, atau Anda dapat mengunduhdokumen contoh inidan menyimpannya dalam bucket S3 Anda.

Pastikan bucket Amazon S3 berada di tempat yang samaAWSWilayah yang akan Anda gunakan untuk meneleponAnalyzeDocument. Untuk membuat bucket, ikuti petunjuk diBuat BucketdiPanduan Pengguna Amazon Simple Storage Service.

Prasyarat

Untuk menggunakan API Amazon A2I untuk membuat alur kerja tinjauan manusia, Anda harus menyelesaikan prasyarat berikut:

- Konfigurasikan peran IAM dengan izin untuk memanggil operasi Amazon A2I dan Amazon Textract Texact API. Untuk memulai, Anda dapat melampirkan kebijakan AWS, AmazonAugmentedaiFullAccess, dan AmazonTextractFullAccess ke peran IAM. Rekam peran IAM Amazon Resources Name (ARN) karena Anda akan membutuhkannya nanti.

  Untuk izin granular lainnya saat menggunakan Amazon Textract, lihatContoh Kebijakan Berbasis Identitas Amazon Textract. Untuk Amazon A2I, lihatIzin dan Keamanan di Amazon Augmented AIdiPanduan Developer Amazon SageMaker.
- Buat tim kerja pribadi dan rekam tim kerja ARN. Jika Anda adalah pengguna baru Amazon A2I, ikuti petunjuk diLangkah 1: Buat Tim Kerja (Konsol).
- Membuat template tugas pekerja. Ikuti petunjuknya diBuat Template Tugas Pekerjauntuk membuat template menggunakan konsol Amazon A2I. Saat Anda membuat template, pilihEkstraksi Form TeksturuntukJenis templat. Dalam template, gantis3_arndengan Amazon S3 ARN dokumen Anda. Tambahkan instruksi pekerja tambahan di<full-instructions header="Instructions"></full-instructions>.

  Jika Anda ingin melihat pratinjau template Anda, pastikan peran IAM Anda memiliki izin yang dijelaskan dalamAktifkan Pratinjau Template Tugas Pekerja.

  Setelah Anda membuat template Anda, rekam template tugas pekerja ARN.

Anda menggunakan sumber daya yang Anda buatPrasyaratuntuk mengonfigurasiCreateFlowDefinitionpermintaan. Dalam permintaan ini, Anda juga menentukan kondisi aktivasi dalam format JSON. Untuk mempelajari cara mengonfigurasi kondisi aktivasi Anda, lihatGunakan Kondisi Aktivasi Loop Manusia Skema JSON dengan Amazon Textract.

Membuat Alur Kerja Tinjauan Manusia (AWS SDK for Python (Boto3))

Untuk menggunakan contoh ini, ganti*merah*teks dengan spesifikasi dan sumber daya Anda.

Pertama, encode kondisi aktivasi Anda menjadi objek JSON menggunakan kode berikut. Ini memicu tinjauan manusia jika Amazon Textract mengembalikan skor kepercayaan yang kurang dari 99 untukAlamat suratdan nilainya, atau jika ia mengembalikan skor kepercayaan kurang dari 90 untuk setiap pasangan kunci-nilai terdeteksi dalam dokumen. Jika Anda menggunakan dokumen sampel yang disediakan dalam contoh ini, kondisi aktivasi ini membuat tugas tinjauan manusia.

```
import json

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

Gunakan`humanLoopActivationConditions`untuk mengonfigurasi`create_flow_definition`permintaan. Contoh berikut menggunakan SDK for Python (Boto3) untuk memanggil`create_flow_definition`Wilayah AWS us-west-2. Ini menentukan menggunakan tim kerja pribadi.

```
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"
)
```

Menganalisis Dokumen

Untuk memasukkan Amazon A2I ke dalam alur kerja analisis dokumen Amazon Textract, Anda mengonfigurasi`HumanLoopConfig`di`AnalyzeDocument`operasi.

Masuk`HumanLoopConfig`, Anda menentukan alur kerja tinjauan manusia Anda (definisi aliran) ARN di`FlowDefinitionArn`, dan memberikan lingkaran manusia Anda nama di`HumanLoopName`.

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

Contoh berikut menggunakan SDK for Python (Boto3) untuk memanggil`analyze_document`di us-2. Ganti*merah, miring*teks dengan sumber daya Anda. Untuk informasi selengkapnya, lihatanalyze_documentdiAWSReferensi API SDK for Python (Boto).

```
client.analyze_document(Document={'S3Object': {"Bucket": "DOC-EXAMPLE-BUCKET",
 "Name": "document-name.png"}},
        HumanLoopConfig={"FlowDefinitionArn":"arn:aws:sagemaker:us-
west-2:111122223333:flow-definition/flow-definition-name",
                         "HumanLoopName":"human-loop-name",
                         "DataAttributes":{"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent",]}},
        FeatureTypes=["FORMS"])
```

Analyze the Document (AWS CLI)

Contoh berikut menggunakanAWSCLI untuk meneleponanalyze_document. Contoh-contoh ini kompatibel denganAWSVersi CLI 2. Yang pertama adalah sintaks singkatan, yang kedua dalam sintaks JSON. Untuk informasi selengkapnya, lihatmenganalisis-dokumendiAWS CLIReferensi Perintah.

```
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**
>
> Hindari spasi putih dalam parameter —human-loop-config Anda, karena hal ini dapat menyebabkan masalah pemrosesan untuk kode Anda.

Tanggapan atas permintaan ini berisiHumanLoopActivationOutput, yang menunjukkan apakah lingkaran manusia diciptakan, dan jika memang demikian, mengapa. Jika loop manusia diciptakan, objek ini juga berisiHumanLoopArn.

Untuk informasi lebih lanjut tentang dan contoh menggunakanAnalyzeDocumentoperasi, lihatMenganalisis Teks Dokumen dengan Amazon Textract.

Loop Manusia

Anda dapat melihat detail tentang loop manusia Anda dan menghentikan loop manusia yang aktif jika terjadi kesalahan menggunakan konsol Amazon A2I dan API.

Lihat Detail Loop Manusia

Anda dapat melihat status loop manusia di konsol Amazon A2I dan dengan menggunakanAPI Runtime Amazon A2I.

Untuk menemukan detail tentang loop manusia Anda (konsol)

1. Buka konsol Amazon A2Ihttps://console.aws.amazon.com/a2iuntuk mengaksesAlur kerja manusiahalaman.
2. Pilih alur kerja peninjauan manusia yang Anda gunakan juga untuk mengonfigurasiHumanLoopConfigdiAnalyzeDocument.
3. DiLoop manusiabagian, pilih loop manusia yang detailnya ingin Anda lihat.

Untuk menemukan detail tentang loop manusia (API) Anda:

Menggunakan Amazon A2IDescribeHumanLoopoperasi. Tentukan nama loop manusia yang digunakan untuk memanggilAnalyzeDocument.

Panggilan contoh FollowingSDK for Python (Boto3)describe_human_loop.

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

Hentikan Loop Manusia

Setelah loop manusia dimulai, Anda dapat menghentikannya menggunakan konsol Amazon A2I dan API.

Untuk menghentikan loop manusia Anda (konsol)

1. Buka konsol Amazon A2Ihttps://console.aws.amazon.com/a2iuntuk mengaksesAlur kerja manusiahalaman.
2. Pilih alur kerja tinjauan manusia yang digunakan untuk mengonfigurasi`HumanLoopConfig`di`AnalyzeDocument`operasi.
3. DiLoop manusiabagian, pilih loop manusia yang ingin Anda hentikan.
4. Pilih Berhenti.

Untuk menghentikan loop manusia Anda (API)

Menggunakan Amazon A2I`StopHumanLoop`operasi. Tentukan nama loop manusia yang digunakan untuk menelepon`AnalyzeDocument`.

SDK berikut untuk Python (Boto3) contoh panggilan`stop_human_loop`.

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

Lihat Data Output dan Metrik Pekerja

Ketika tugas peninjauan manusia diselesaikan oleh pekerja, Amazon A2I menyimpan data output Anda di bucket Amazon S3 yang Anda tentukan dalam alur kerja peninjauan manusia.

Jika Anda menggunakan tenaga kerja pribadi, data output berisi metadata pekerja yang dapat Anda gunakan untuk melacak aktivitas pekerja individu.

Temukan Data Output di Amazon S3

Amazon A2I menggunakan nama alur kerja tinjauan manusia Anda sebagai awalan untuk nama file yang menyimpan data output dari loop manusia yang dibuat menggunakan pekerjaan peninjauan manusia tersebut.

Jalan ke output loop manusia menggunakan pola berikut di mana*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*mewakili tanggal penciptaan loop manusia dengan tahun (YYYY), bulan (MM), dan hari (DD) dan waktu pembuatan dengan jam (hh), menit (mm), dan kedua (ss).

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

Untuk melihat output untuk loop manusia, gunakan konsol Amazon A2I.

Untuk melihat output loop manusia

1. Buka konsol Amazon A2Ihttps://console.aws.amazon.com/a2iuntuk mengaksesAlur kerja manusiahalaman.
2. Pilih alur kerja tinjauan manusia yang Anda gunakan untuk mengonfigurasi`HumanLoopConfig`di`AnalyzeDocument`.
3. DiLoop manusiabagian, pilih loop manusia yang outputnya ingin Anda tinjau.
4. Di bawahLokasi output, pilih tautan ke data output.

Melacak aktivitas pekerja pribadi

Saat Anda menggunakan tenaga kerja pribadi untuk tugas peninjauan manusia, data output mencakup informasi berikut tentang pekerja yang menyelesaikan peninjauan:

- Parameter`workerId`.
- Dalam `workerMetadata`:
  - `identityProviderType`— Layanan yang digunakan untuk mengelola tenaga kerja swasta.
  - `issuer`— Amazon Cognito pengguna pool atau OIDC Identity Provider (iDP) penerbit terkait dengan tim kerja yang ditugaskan untuk tugas tinjauan manusia ini.
  - `sub`— Sebuah pengenal unik yang mengacu pada pekerja. Jika Anda membuat tenaga kerja menggunakan Amazon Cognito, Anda dapat mengambil rincian tentang pekerja ini (seperti nama atau nama pengguna) menggunakan ID ini menggunakan Amazon Cognito. Untuk mempelajari caranya, lihatMengelola dan Mencari Akun PenggunadiPanduan Developer Amazon Cognito.

Berikut ini adalah contoh output yang mungkin Anda lihat jika Anda menggunakan Amazon Cognito untuk membuat tenaga kerja pribadi.

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

Berikut ini adalah contoh output yang mungkin Anda lihat jika Anda menggunakan IDP OIDC Anda sendiri untuk membuat tenaga kerja pribadi:


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


Untuk mempelajari selengkapnya tentang penggunaan tenaga kerja pribadi, lihatMenggunakan Tenaga Kerja PribadidiPanduan Developer Amazon SageMaker.

Keamanan di Amazon Textract

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Gunakan topik berikut untuk mempelajari cara mengamankan sumber daya Amazon Textract Anda.

Topik

Perlindungan Data di Amazon Textract

Amazon Textract sesuai denganAWS Model tanggung jawab bersama, yang mencakup peraturan dan pedoman untuk perlindungan data.AWSbertanggung jawab untuk melindungi infrastruktur global yang menjalankan semuaAWSlayanan.AWSmempertahankan kontrol atas data yang dihosting di infrastruktur ini, termasuk kontrol konfigurasi keamanan untuk menangani konten pelanggan dan data pribadi.AWSpelanggan danAPNmitra, bertindak baik sebagai pengontrol data atau pengolah data, bertanggung jawab atas data pribadi apa pun yang mereka masukkan ke dalamAWSAwan.

Untuk tujuan perlindungan data, kami merekomendasikan agar Anda melindungi kredensial akun AWS dan menyiapkan akun pengguna individu dengan AWS Identity and Access Management (IAM). Hal ini memastikan bahwa setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tugas pekerjaan mereka. Kami juga menyarankan agar Anda mengamankan data Anda dengan cara-cara berikut:

- Gunakan autentikasi multifaktor (MFA) pada setiap akun.

- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya AWS.
- Atur API dan log aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama semua kontrol keamanan default dalam layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon S3.

Kami sangat menyarankan agar Anda tidak memasukkan informasi identifikasi sensitif apapun, seperti nomor rekening pelanggan Anda, ke dalam kolom isian teks bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Amazon Textract atau lainnyaAWSlayanan menggunakan konsol, API,AWS CLI, atauAWSSDK. Data apa pun yang Anda masukkan ke dalam Amazon Textract atau layanan lainnya mungkin akan diambil untuk disertakan dalam log diagnostik. Saat Anda memberikan URL ke server eksternal, jangan menyertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Untuk informasi selengkapnya tentang perlindungan data, lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS.

Enkripsi di Amazon Textract

Enkripsi data mengacu pada perlindungan data saat transit dan saat tidak digunakan. Anda dapat melindungi data Anda dengan menggunakan Amazon S3-Managed Keys atauAWS KMS keysaat istirahat, bersama standar Transport Layer Security saat transit.

Enkripsi saat Data Tidak Berpindah

Metode utama untuk mengenkripsi data di Amazon Textract enkripsi sisi server. Dokumen input yang diteruskan dari bucket Amazon S3 dienkripsi oleh Amazon S3 dan didekripsi saat Anda mengaksesnya. Selama Anda mengautentikasi permintaan Anda dan memiliki izin akses, tidak ada perbedaan dalam cara Anda mengakses objek terenkripsi atau tidak terenkripsi. Misalnya, jika Anda berbagi objek menggunakan URL yang ditandatangani sebelumnya, URL tersebut bekerja dengan cara yang sama untuk objek terenkripsi maupun tidak terenkripsi. Selain itu, ketika Anda daftar objek di bucket Anda,`List`API mengembalikan daftar semua objek, terlepas dari apakah mereka dienkripsi.

Amazon Textract Texact menggunakan dua metode enkripsi sisi server yang saling eksklusif.

Enkripsi Sisi Server dengan Amazon S3-Managed Keys (SSE-S3)

Saat Anda menggunakan enkripsi sisi server dengan Amazon S3-Managed Keys (SSE-S3), setiap objek dienkripsi dengan kunci unik. Sebagai pelindung tambahan, metode ini mengenkripsi kunci itu sendiri dengan kunci utama yang diputar secara rutin. Enkripsi sisi server Amazon S3 menggunakan salah satu cipher blok terkuat yang tersedia, Advanced Encryption Standard 256-bit (AES-256), untuk mengenkripsi data Anda. Untuk informasi lebih lanjut,lihat Melindungi Data Menggunakan Enkripsi Sisi Serverdengan Kunci Enkripsi Terkelola Amazon S3 (SSE-S3).

Enkripsi Sisi Server dengan kunci KMS yang Disimpan di AWS Key Management Service (SSE-KMS)

Enkripsi sisi server dengan kunci KMS yang tersimpan di AWS Key Management Service (SSE-KMS) serupa dengan SSE-S3, tetapi dengan beberapa manfaat dan biaya tambahan untuk penggunaan layanan ini. Ada izin terpisah untuk penggunaan kunci KMS yang memberikan perlindungan tambahan terhadap akses tidak sah dari objek Anda di Amazon S3. SSE-KMS juga menyediakan jejak audit yang menunjukkan kapan kunci KMS Anda digunakan dan oleh siapa. Selain itu, Anda dapat membuat dan mengelola kunci KMS atau menggunakanKunci yang dikelola AWSyang unik bagi Anda, dan Wilayah Anda. Untuk informasi selengkapnya, lihatMelindungi Data Menggunakan Enkripsi Sisi Serverdengan kunci KMS Disimpan di AWS Key Management Service (SSE-KMS).

Enkripsi Saat Data Berpindah

Untuk data yang sedang transit, Amazon Textract menggunakan Transport Layer Security (TLS) untuk mengenkripsi data yang dikirim antara layanan dan agen. Selain itu, Amazon Textract menggunakan endpoint VPC untuk mengirim data antara berbagai layanan mikro yang digunakan saat Amazon Textract memproses dokumen.

Privasi Lalu Lintas Kerja Internet

Amazon Textract berkomunikasi secara eksklusif melalui titik akhir HTTPS, yang didukung di semua Wilayah yang didukung oleh Amazon Textract Texact

Identity and Access Management untuk Amazon Textract

AWS Identity and Access Management (IAM) adalah layanan AWS yang membantu administrator mengendalikan akses ke sumber daya AWS dengan aman. Administrator IAM mengontrol siapa yang bisamengonfirmasi(masuk) danresmi(memiliki izin) untuk menggunakan sumber daya Amazon Textract. IAM adalah layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Penonton

- Mengautentikasi Menggunakan Identitas
- Mengelola Akses Menggunakan Kebijakan
- Cara Amazon Textract Bekerja dengan IAM
- Contoh Kebijakan Berbasis Identitas Amazon Textract
- Pemecahan masalah identitas dan akses Amazon Textract

Penonton

Cara menggunakanAWS Identity and Access Management(IAM) berbeda-beda, tergantung pada pekerjaan yang Anda lakukan di Amazon Textract.

Pengguna layanan— Jika Anda menggunakan layanan Amazon Textract untuk melakukan tugas Anda, administrator Anda akan memberikan kredensyal dan izin yang dibutuhkan. Saat Anda menggunakan lebih banyak fitur Amazon Textract Texact untuk melakukan pekerjaan, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda untuk meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Amazon Textract, lihatPemecahan masalah identitas dan akses Amazon Textract.

Administrator layanan— Jika Anda bertanggung jawab atas sumber daya Amazon Textract di perusahaan Anda, maka Anda mungkin memiliki akses penuh ke Amazon Textract. Anda bertanggung jawab untuk menentukan fitur Amazon Textract Texact dan sumber daya mana yang dapat diakses karyawan Anda. Anda kemudian harus mengirimkan permintaan ke administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari selengkapnya cara perusahaan Anda dapat menggunakan IAM dengan Amazon Textract, lihatCara Amazon Textract Bekerja dengan IAM.

Administrator IAM— Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang bagaimana Anda dapat menulis kebijakan untuk mengelola akses ke Amazon Textract. Untuk melihat contoh kebijakan berbasis identitas Amazon Texact yang dapat Anda gunakan di IAM, lihatContoh Kebijakan Berbasis Identitas Amazon Textract.

Mengautentikasi Menggunakan Identitas

Autentikasi adalah cara Anda untuk masuk ke AWS menggunakan kredensial identitas Anda. Untuk informasi selengkapnya tentang cara masuk menggunakan Konsol Manajemen AWS, lihat Masuk ke Konsol Manajemen AWS sebagai pengguna IAM atau pengguna root dalam Panduan Pengguna IAM.

Anda harus terautentikasi (masuk ke AWS) sebagai pengguna root Akun AWS, pengguna IAM, atau dengan menggunakan IAM role. Anda juga dapat menggunakan autentikasi single sign-on milik perusahaan Anda, atau bahkan masuk menggunakan Google atau Facebook. Dalam kasus ini, administrator Anda sebelumnya telah menyiapkan federasi identitas menggunakan IAM role. Saat Anda mengakses AWS menggunakan kredensial dari perusahaan lain, secara tidak langsung Anda mengambil sebuah peran.

Untuk masuk secara langsung ke Konsol Manajemen AWS, gunakan kata sandi Anda dengan alamat email pengguna asal atau nama pengguna IAM Anda. Anda dapat mengakses AWS secara terprogram menggunakan kunci akses pengguna asal atau pengguna IAM Anda. AWS menyediakan SDK dan alat baris perintah untuk menandatangani permintaan Anda secara kriptografis menggunakan kredensial Anda. Jika Anda tidak menggunakan peralatan AWS, Anda harus menandatangani sendiri permintaan tersebut. Lakukan ini menggunakan Versi Tanda Tangan 4, sebuah protokol untuk mengautentikasi permintaan API masuk. Untuk informasi selengkapnya tentang autentikasi permintaan, lihat proses penandatanganan Versi Tanda Tangan 4 dalam Referensi Umum AWS.

Terlepas dari metode autentikasi yang Anda gunakan, Anda mungkin juga diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS menyarankan supaya Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Pengguna root Akun AWS

Saat pertama kali membuat akun Akun AWS, Anda memulai dengan satu identitas masuk yang memiliki akses lengkap ke semua layanan dan sumber daya AWS dalam akun tersebut. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk menggunakan alamat email dan kata sandi yang Anda gunakan saat membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari, bahkan tugas administratif. Sebagai gantinya, patuhi praktik terbaik dalam menggunakan pengguna root saja untuk membuat pengguna IAM pertama Anda. Kemudian, kunci kredensial pengguna root dengan aman dan gunakan kredensial itu untuk melakukan beberapa tugas manajemen akun dan layanan saja.

Pengguna dan Grup IAM

Pengguna IAM adalah identitas dalam akun Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Pengguna IAM dapat memiliki kredensial jangka panjang, seperti nama pengguna dan kata sandi atau satu set access key. Untuk mempelajari cara membuat kunci akses,

lihat Mengelola access key untuk pengguna IAM dalam Panduan Pengguna IAM. Saat Anda membuat access key untuk pengguna IAM, pastikan bahwa Anda melihat pasangan kunci dan menyimpannya dengan aman. Anda tidak dapat memulihkan secret access key di masa mendatang. Sebaliknya, Anda harus membuat pasangan access key baru.

Grup IAM adalah identitas yang menentukan kumpulan dari para pengguna IAM. Anda tidak dapat masuk sebagai kelompok. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk set besar pengguna. Misalnya, Anda dapat memiliki grup yang diberi nama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Para pengguna berbeda dari peran. Seorang pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran ini dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

IAM Role

IAM role adalah identitas dalam akun Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat menggunakan IAM role untuk sementara di dalam Konsol Manajemen AWS dengan cara berganti peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL khusus. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Menggunakan IAM roles dalam Panduan Pengguna IAM.

IAM role dengan kredensial sementara berguna dalam situasi berikut:

- Izin pengguna IAM sementara – Pengguna IAM dapat menggunakan IAM role untuk sementara dan mendapatkan izin yang berbeda untuk tugas tertentu.
- Akses pengguna gabungan – Alih-alih membuat pengguna IAM, Anda dapat menggunakan identitas yang sudah ada dari Directory Service, direktori pengguna korporasi Anda, atau penyedia identitas web. Ini dikenal sebagai pengguna gabungan. AWS menugaskan peran kepada pengguna gabungan saat akses diminta melalui penyedia identitas. Untuk informasi selengkapnya tentang pengguna gabungan, lihat Pengguna dan peran gabungan dalam Panduan Pengguna IAM.
- Akses lintas akun – Anda dapat menggunakan IAM role agar seseorang (principal tepercaya) di akun lain diizinkan untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa layanan AWS, Anda dapat

melampirkan kebijakan langsung ke sumber daya (alih-alih menggunakan peran sebagai proksi). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara IAM role dan kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.

- Akses lintas layanan – Beberapa layanan AWS menggunakan fitur di layanan AWS lainnya. Sebagai conoth, ketika Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Layanan mungkin melakukan ini menggunakan izin panggilan principal, menggunakan peran layanan, atau peran tertaut layanan.
  - Izin prinsipal – Saat Anda menggunakan pengguna IAM atau IAM role untuk melakukan tindakan di AWS, Anda dianggap sebagai principal. Kebijakan memberikan izin kepada principal. Saat Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memicu tindakan lain di layanan yang berbeda. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk melihat apakah suatu tindakan memerlukan tindakan dependen tambahan dalam suatu kebijakan, lihatTindakan, Sumber Daya, dan Kunci Kondisi untuk Amazon TextractdiReferensi Otorisasi Layanan.
  - Peran layanan – Peran layanan adalah IAM role yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke layanan AWS dalam Panduan Pengguna IAM.
  - Peran tertaut layanan – Peran tertaut layanan adalah tipe peran layanan yang tertaut dengan layanan AWS. Layanan dapat menggunakan peran untuk melakukan tindakan atas nama Anda. Peran tertaut layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan IAM role untuk mengelola kredensial sementara untuk aplikasi yang berjalan pada instans EC2, dan membuat permintaan API AWS CLI atau AWS. Menyimpan access key di dalam instans EC2 lebih disarankan. Untuk menugaskan sebuah peran AWS ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda dapat membuat sebuah profil instans yang dilampirkan ke instans. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 untuk mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM role untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari kapan waktunya menggunakan IAM role atau pengguna IAM, lihat Kapan harus membuat IAM role (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola Akses Menggunakan Kebijakan

Anda mengendalikan akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas IAM atau sumber daya AWS. Kebijakan adalah objek di AWS, yang saat terkait dengan identitas atau sumber daya, akan menentukan izinnya. Anda dapat masuk sebagai pengguna root atau pengguna IAM, atau Anda dapat menggunakan IAM role. Ketika Anda kemudian membuat permintaan, AWS mengevaluasi kebijakan berbasis identitas atau kebijakan berbasis sumber daya yang terkait. Izin dalam kebijakan dapat menentukan permintaan yang diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran Umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Setiap entitas IAM (pengguna atau peran) dimulai tanpa izin. Dengan kata lain, secara default, pengguna tidak dapat melakukan apa pun, termasuk mengubah kata sandi mereka sendiri. Untuk memberikan izin kepada pengguna untuk melakukan sesuatu, administrator harus melampirkan kebijakan izin kepada pengguna. Atau administrator dapat menambahkan pengguna ke grup yang memiliki izin yang dimaksudkan. Ketika administrator memberikan izin untuk grup, semua pengguna dalam grup tersebut diberikan izin tersebut.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk pengoperasiannya. Misalnya, Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari API Konsol Manajemen AWS, the AWS CLI, or the AWS.

Kebijakan Berbasis Identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke identitas, seperti pengguna IAM, grup pengguna, atau peran. Kebijakan ini mengontrol tipe tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan dalam syarat. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan secara langsung ke satu pengguna, grup, atau peran.

Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam akun Akun AWS Anda. Kebijakan terkelola meliputi kebijakan yang dikelola AWS dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan Berbasis Sumber Daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan tepercaya IAM role dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan menetapkan tindakan apa yang dapat dilakukan oleh principal tertentu di sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Principal dapat meliputi akun, pengguna, peran, pengguna gabungan, atau layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan yang dikelola AWS dari IAM dalam kebijakan berbasis sumber daya.

Daftar Kontrol Akses (ACL)

Daftar kontrol akses (ACL) mengendalikan principal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Developer Amazon Simple Storage Service.

Tipe Kebijakan Lainnya

AWS mendukung tipe kebijakan tambahan, yang kurang umum. Tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau IAM role). Anda dapat menetapkan batas izin untuk suatu entitas. Izin yang dihasilkan adalah persimpangan antara

kebijakan berbasis identitas milik entitas dan batas izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini dapat membatalkan izin. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.

- Kebijakan Kontrol Layanan (SCPs) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa akun Akun AWS secara terpusat yang dimiliki oleh bisnis Anda. Jika Anda mengaktifkan semua fitur di sebuah organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau ke semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap pengguna root Akun AWS. Untuk informasi selengkapnya tentang Organizations dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter saat Anda membuat sesi sementara secara terprogram bagi peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah persimpangan kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga dapat berasal dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai Tipe Kebijakan

Ketika beberapa tipe kebijakan berlaku untuk sebuah permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan untuk mengizinkan permintaan ketika beberapa tipe kebijakan dilibatkan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Cara Amazon Textract Bekerja dengan IAM

Sebelum menggunakan IAM untuk mengelola akses ke Amazon Textract, Anda harus memahami fitur IAM apa saja yang tersedia untuk digunakan dengan Amazon Textract. Untuk mendapatkan tampilan tingkat tinggi tentang bagaimana Amazon Textract dan lainnyaAWSLayanan bekerja dengan IAM, lihatAWSLayanan yang Bekerja dengan IAMdiPanduan Pengguna IAM.

Topik

- Kebijakan Berbasis Identitas Amazon Textract
- Kebijakan Berbasis Sumber Daya Amazon Textract

- Otorisasi Berdasarkan Tag Amazon Textract
- IAM Ran Amazon Textract

Kebijakan Berbasis Identitas Amazon Textract

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak dan ketentuan di mana tindakan tersebut diperbolehkan atau ditolak. Amazon Textract Texact mendukung tindakan, sumber daya, dan kunci syarat tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat Referensi Elemen Kebijakan JSON IAM dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam syarat apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan-tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama sebagai operasi API AWS terkait. Ada beberapa pengecualian, misalnya tindakan hanya dengan izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Sertakan tindakan dalam kebijakan untuk memberikan izin guna melakukan operasi yang terkait.

Tindakan asinkron di Amazon Textract memerlukan dua izin tindakan yang akan diberikan, satu untuk tindakan Mulai dan satu untuk tindakan Get. Selain itu, jika Anda menggunakan bucket Amazon S3 untuk meneruskan dokumen, Anda harus memberikan akses baca akun Anda.

Di Amazon Textract, semua tindakan kebijakan dimulai dengan: `textract:`. Misalnya, untuk memberikan izin kepada seseorang untuk menjalankan operasi Amazon Textract dengan Amazon Textract Texact`AnalyzeDocument`operasi, Anda termasuk`textract:AnalyzeDocument`tindakan dalam kebijakan mereka. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. Amazon Textract menentukan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut.

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

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Misalnya, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut.

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

Untuk daftar tindakan Amazon Textract, lihatTindakan Ditetapkan oleh Amazon TextractdiPanduan Pengguna IAM.

Sumber daya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam syarat apa.

Elemen kebijakan JSON `Resource` menentukan objek atau objek-objek yang menjadi target penerapan tindakan. Pernyataan harus mencakup elemen `Resource` atau `NotResource`. Sebagai praktik terbaik, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung tipe sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin tingkat sumber daya, misalnya operasi pencantuman, gunakan karakter wildcart (*) untuk menunjukkan bahwa pernyataan tersebut berlaku bagi semua sumber daya.

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

Amazon Textract tidak mendukung penentuan ARN sumber daya dalam kebijakan.

Kunci Syarat

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, prinsipal mana yang dapat melakukan tindakan pada sumber daya apa, dan menurut persyaratan apa.

Elemen `Condition` (atau `Condition` blok) memungkinkan Anda menentukan syarat di mana suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator syarat, seperti sama dengan atau kurang dari, untuk mencocokkan syarat dalam kebijakan dengan nilai dalam permintaan.

Jika Anda menentukan beberapa elemen `Condition` dalam pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya denga menggunakan operasi logika `AND`. Jika Anda menetapkan beberapa nilai untuk kunci syarat tunggal, AWS akan mengevaluasi syarat tersebut dengan menggunakan operasi logika `OR`. Semua persyaratan harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan syarat. Sebagai contoh, Anda dapat memberikan izin pengguna IAM untuk mengakses sumber daya hanya jika ditandai dengan nama pengguna IAM mereka. Untuk informasi lebih lanjut, lihat Elemen kebijakan IAM: variabel dan tag dalam Panduan Pengguna IAM.

AWS mendukung kunci syarat global dan kunci syarat khusus layanan. Untuk melihat semua kunci syarat global AWS, lihat Kunci konteks syarat global AWS dalam Panduan Pengguna IAM.

Amazon Textract tidak menyediakan kunci syarat khusus layanan, tetapi mendukung penggunaan beberapa kunci syarat global. Untuk daftar dari semuaAWSKunci kondisi global, lihatAWSKunci Konteks Kondisi GlobaldiPanduan Pengguna IAM.

Contoh

Untuk melihat contoh kebijakan berbasis identitas Amazon Textract, lihatContoh Kebijakan Berbasis Identitas Amazon Textract.

Kebijakan Berbasis Sumber Daya Amazon Textract

Amazon Textract tidak mendukung kebijakan berbasis sumber daya.

Otorisasi Berdasarkan Tag Amazon Textract

Amazon Textract Texact tidak mendukung penandaan sumber daya atau pengendalian akses berdasarkan tag.

IAM Ran Amazon Textract

IAM role adalah entitas di dalam akun AWS Anda yang memiliki izin tertentu.

Menggunakan kredensi sementara dengan Amazon Textract

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda mendapatkan kredensial keamanan sementara dengan memanggil operasi API AWS STS, seperti AssumeRole atau GetFederationToken.

Amazon Textract mendukung penggunaan kredensi sementara.

Peran Tertaut Layanan

Peran terkait layanan mengizinkan layanan AWS untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran terkait layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Amazon Textract tidak mendukung peran terkait layanan.

> **Note**
>
> Karena Amazon Textract Texact tidak mendukung peran terkait layanan, Amazon Texact tidak mendukung prinsip layanan AWS. Untuk informasi lebih lanjut tentang prinsipal layanan, lihatPrinsipal layanan AWSdiPanduan Pengguna IAM

Peran Layanan

Fitur ini memungkinkan layanan untuk menerima peran layanan atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti administrator IAM dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

Amazon Textract mendukung peran layanan.

Contoh Kebijakan Berbasis Identitas Amazon Textract

Secara default, pengguna IAM dan role tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon Textract Texact. Mereka juga tidak dapat melakukan tugas menggunakan API Konsol Manajemen AWS, AWS CLI, or AWS. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber

daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat Kebijakan pada Tab JSON dalam Panduan Pengguna IAM.

Topik

- Praktik Terbaik Kebijakan
- Izinkan Pengguna untuk Melihat Izin Mereka Sendiri
- Memberikan Akses ke Operasi Sinkron di Amazon Textract
- Memberikan Akses ke Operasi Asynchronous di Amazon Textract

Praktik Terbaik Kebijakan

Kebijakan berbasis identitas adalah pilihan yang sangat tepat. Kebijakan-kebijakan ini menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon Textract Texact di akun Anda. Tindakan ini membuat Akun AWS Anda terkena biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Memulai menggunakanAWSkebijakan yang dikelola— Untuk mulai menggunakan Amazon Textract dengan cepat, gunakanAWSkebijakan yang dikelola untuk memberi karyawan Anda izin yang mereka butuhkan. Kebijakan ini sudah tersedia di akun Anda dan dikelola, serta diperbarui oleh AWS. Untuk informasi selengkapnya, lihat Memulai menggunakan izin dengan kebijakan yang dikelola AWS dalam Panduan Pengguna IAM.
- Pemberian hak istimewa terendah – Ketika Anda membuat kebijakan kustom, berikan izin yang diperlukan saja untuk melakukan tugas. Mulai dengan satu set izin minimum dan berikan izin tambahan sesuai kebutuhan. Melakukan hal tersebut lebih aman daripada memulai dengan izin yang terlalu fleksibel, lalu mencoba memperketatnya nanti. Untuk informasi selengkapnya, lihat Pemberian hak istimewa terendah dalam Panduan Pengguna IAM.
- Aktifkan autentikasi multifaktor (MFA) untuk operasi sensitif – Untuk keamanan ekstra, mintalah pengguna IAM untuk menggunakan autentikasi multifaktor (MFA) guna mengakses sumber daya atau operasi API yang sensitif. Untuk informasi selengkapnya, lihat Menggunakan autentikasi multifaktor (MFA) dalam AWS dalam Panduan Pengguna IAM.
- Gunakan syarat kebijakan untuk keamanan tambahan – Selama bisa dilakukan, tentukan persyaratan agar kebijakan berbasis identitas Anda mengizinkan akses ke sumber daya. Misalnya, Anda dapat menulis persyaratan untuk menentukan jangkauan alamat IP yang diizinkan untuk

mengajukan permintaan. Anda juga dapat menulis ketentuan untuk mengizinkan permintaan hanya dalam rentang tanggal atau waktu tertentu, atau untuk mengharuskan penggunaan SSL atau MFA. Untuk informasi lebih lanjut, lihat Elemen kebijakan IAM JSON: Ketentuan dalam Panduan Pengguna IAM.

Izinkan Pengguna untuk Melihat Izin Mereka Sendiri

Contoh ini menunjukkan cara Anda dapat membuat kebijakan yang mengizinkan para pengguna IAM untuk melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau secara terprogram menggunakan API AWS CLI atau AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Memberikan Akses ke Operasi Sinkron di Amazon Textract

Contoh kebijakan ini memberikan akses ke tindakan sinkron di Amazon Textract Texact ke pengguna IAM pada AndaAWSakun.

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

Memberikan Akses ke Operasi Asynchronous di Amazon Textract

Contoh kebijakan berikut memberikan pengguna IAM pada AndaAWSakses akun ke semua operasi asinkron yang digunakan di Amazon Textract.

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

Pemecahan masalah identitas dan akses Amazon Textract

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan mengatasi masalah umum yang mungkin Anda temui saat bekerja menggunakan Amazon Textract dan IAM.

Topik

Saya tidak Berwenang untuk Melakukan Tindakan di Amazon Textract

Jika Konsol Manajemen AWS memberi tahu bahwa Anda tidak diotorisasi untuk melakukan tindakan, Anda harus menghubungi administrator untuk mendapatkan bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika`mateojackson`Pengguna IAM mencoba menjalankan`DetectDocumentText`pada gambar tes tetapi tidak memiliki`textract:DetectDocumentText`izin.

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

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya agar dia dapat mengakses sumber daya `textimage.png` menggunakan tindakan `textract:DetectDocumentText`.

Saya tidak Berwenang untuk Melakukan iam:PassRole

Jika Anda menerima kesalahan bahwa Anda tidak terotorisasi untuk melakukan tindakan `iam:PassRole`, Anda harus menghubungi administrator untuk mendapatkan bantuan. Administrator Anda adalah orang yang memberikan Anda nama pengguna dan kata sandi Anda. Meminta orang tersebut untuk memperbarui kebijakan Anda agar Anda dapat memberikan peran Amazon Textract.

Beberapa layanan AWS mengizinkan Anda untuk meneruskan peran yang sudah ada ke layanan tersebut, alih-alih membuat peran layanan atau peran yang tertaut layanan baru. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama`marymajor`mencoba menggunakan konsol untuk melakukan tindakan di Amazon Textract. Namun, tindakan tersebut mengharuskan layanan untuk memiliki izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut ke layanan.

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

Dalam kasus ini, Mary meminta administrator untuk memperbarui kebijakannya agar mengizinkannya untuk melakukan tindakan `iam:PassRole`.

Saya Ingin Melihat Access Key Saya

Setelah membuat access key pengguna IAM, Anda dapat melihat access key ID Anda setiap saat. Namun, Anda tidak dapat melihat secret access key Anda lagi. Jika Anda kehilangan secret key, Anda harus membuat pasangan access key baru.

Access key terdiri dari dua bagian: access key ID (misalnya, `AKIAIOSFODNN7EXAMPLE`) dan secret access key (misalnya, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Seperti nama pengguna dan kata sandi, Anda harus menggunakan access key ID dan secret access key sekaligus untuk mengautentikasi permintaan Anda. Kelola access key Anda seaman nama pengguna dan kata sandi Anda.

> **Important**
>
> Jangan memberikan access key Anda kepada pihak ke tiga, bahkan untuk membantu menemukan ID pengguna kanonis Anda. Anda mungkin memberi seseorang akses permanen ke akun Anda, dengan melakukan ini.

Saat Anda membuat pasangan access key, Anda diminta menyimpan access key ID dan secret access key di lokasi yang aman. secret access key hanya tersedia saat Anda membuatnya. Jika Anda kehilangan secret access key Anda, Anda harus menambahkan access key baru ke pengguna IAM Anda. Anda dapat memiliki maksimum dua access key. Jika Anda sudah memiliki dua, Anda

harus menghapus satu pasangan kunci sebelum membuat pasangan baru. Untuk melihat instruksi, lihat Mengelola access keys di Panduan Pengguna IAM.

Saya seorang Administrator dan Ingin Mengizinkan Orang Lain Mengakses Amazon Textract

Untuk mengizinkan orang lain mengakses Amazon Textract, Anda harus membuat entitas IAM (pengguna atau peran) untuk orang atau aplikasi yang memerlukan akses. Mereka akan menggunakan kredensial untuk entitas tersebut untuk mengakses AWS. Anda kemudian harus melampirkan kebijakan pada entitas yang memberi mereka izin yang benar di Amazon Textract.

Untuk segera mulai, lihat Membuat pengguna dan grup khusus IAM pertama Anda di Panduan Pengguna IAM.

Saya ingin mengizinkan orang di luar sayaAWSAkun untuk Mengakses Sumber Daya Amazon Textract Saya

Anda dapat membuat peran yang dapat digunakan para pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi akses pada orang ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mempelajari apakah Amazon Textract mendukung fitur ini, lihatCara Amazon Textract Bekerja dengan IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di akun Akun AWS lain yang Anda miliki dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke Akun AWS pihak ketiga, lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (gabungan identitas) dalam Panduan Pengguna IAM .
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan IAM role dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Pencatatan dan Pemantauan

Untuk memantau Amazon Textract, gunakan Amazon CloudWatch. Bagian ini memberikan informasi tentang cara mengatur pemantauan untuk Amazon Textract. Ini juga menyediakan konten referensi untuk metrik Amazon Textract Texact.

Topik

Pemantauan Amazon Textract

Dengan CloudWatch, Anda bisa mendapatkan metrik untuk operasi Amazon Textract individual atau metrik Amazon Textract untuk akun Anda. Anda dapat menggunakan metrik untuk melacak kondisi solusi berbasis Amazon Textract, dan menyiapkan alarm untuk memberi tahu ketika satu atau beberapa metrik berada di luar ambang batas yang ditentukan. Misalnya, Anda dapat melihat metrik untuk jumlah kesalahan server yang terjadi. Anda juga dapat melihat metrik untuk berapa kali operasi Amazon Textract tertentu telah berhasil. Untuk melihat metrik, Anda dapat menggunakanAmazon CloudWatch, yangAWS CLI, atauAPI CloudWatch.

Menggunakan Metrik CloudWatch untuk Amazon Textract

Untuk menggunakan metrik, Anda harus menentukan informasi berikut:

- Dimensi metrik atau tanpa dimensi. Dimensi adalah pasangan nama-nilai yang membantu Anda mengidentifikasi metrik secara unik. Amazon Textract memiliki satu dimensi, bernamaOperasi. Ini menyediakan metrik untuk operasi tertentu. Jika Anda tidak menentukan dimensi, metrik akan dicakup ke semua operasi Amazon Textract di akun Anda.
- Nama metrik, seperti `UserErrorCount`.

Anda bisa mendapatkan data pemantauan untuk Amazon Textract Texact dengan menggunakanKonsol Manajemen AWS, yangAWS CLI, atau API CloudWatch. Anda juga dapat menggunakan API CloudWatch melalui salah satu Kit Pengembangan Perangkat Lunak (SDK) Amazon AWS atau alat API CloudWatch. Konsol tersebut menampilkan serangkaian grafik berdasarkan data mentah dari API CloudWatch. Tergantung kebutuhan, Anda mungkin lebih memilih menggunakan grafik yang ditampilkan di konsol atau diterima dari API.

Daftar berikut menunjukkan beberapa penggunaan umum untuk metrik. Berikut ini adalah saran untuk memulai, bukan daftar komprehensif.

Bagaimana caranya?	Metrik Terkait
Bagaimana saya tahu jika aplikasi saya telah mencapai jumlah maksimum permintaan per detik?	Pantau statistik `Sum` metrik `ThrottledCount` .
Bagaimana saya dapat memantau kesalahan permintaan?	Gunakan statistik `Sum` metrik `UserErrorCount` .
Bagaimana saya dapat menemukan jumlah total permintaan?	Gunakan statistik `SampleCount` metrik `ResponseTime` . Termasuk setiap permintaan yang menghasilkan kesalahan. Jika Anda ingin melihat panggilan operasi yang berhasil saja, maka gunakan metrik `SuccessfulRequestCount` .
Bagaimana saya bisa memantau latensi panggilan operasi Amazon Textract?	Gunakan metrik `ResponseTime` .

Anda harus memiliki izin CloudWatch yang sesuai untuk memantau Amazon Textract dengan CloudWatch. Untuk informasi selengkapnya, lihat Autentikasi dan Kontrol Akses untuk Amazon CloudWatch.

Mengakses Metrik Amazon Textract

Contoh-contoh berikut menunjukkan cara mengakses metrik Amazon Textract Texact menggunakan konsol CloudWatch,AWS CLI, dan API CloudWatch.

Untuk melihat metrik (konsol)

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. PilihMetrik, pilihSemua Metriktab, dan kemudian pilihAmazon Textract.
3. PilihDengan operasi, dan kemudian pilih metrik.

Misalnya, pilihStartDocumentAnalysismetrik untuk mengukur berapa kali analisis dokumen asinkron telah dimulai.

4. Pilih nilai untuk rentang tanggal. Hitungan metrik ditampilkan dalam grafik.

Melihat metrik agar sukses**`StartDocumentAnalysis`**panggilan operasi yang telah dilakukan selama periode waktu tertentu (CLI)

- Buka AWS CLI dan masukkan perintah berikut:

```
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
```

  Contoh ini menunjukkan panggilan operasi `StartDocumentAnalysis` yang berhasil dilakukan selama periode waktu tertentu. Untuk informasi selengkapnya, lihat dapatkan-metrik-statistik.

Untuk mengakses metrik (API CloudWatch)

- Panggil `GetMetricStatistics`. Untuk informasi selengkapnya, lihat Referensi API Amazon CloudWatch.

Buat Alarm

Anda dapat membuat alarm CloudWatch yang mengirimkan pesan Amazon Simple Notification Service (Amazon SNS) ketika status alarm berubah. Alarm mengawasi satu metrik selama suatu periode waktu yang Anda tentukan. Alarm tersebut juga dapat melakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas selama sejumlah periode waktu. Tindakan ini adalah notifikasi yang dikirim ke topik Amazon SNS atau kebijakan Auto Scaling.

Alarm memicu tindakan hanya untuk perubahan status berkelanjutan. Alarm CloudWatch tidak memicu tindakan hanya karena alarm tersebut berada dalam status tertentu. Status harus diubah dan dipelihara selama jangka waktu tertentu.

Untuk mengatur alarm (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilihAlarm, dan pilihBuat Alarm. Ini akan membukaBuat Wisaya Alarm.
3. Pilih Pilih metrik.
4. DiSemua metriktab, pilihTextract.
5. PilihOperasi, dan kemudian pilih metrik.

   Misalnya, pilihStartDocumentAnalysisuntuk mengatur alarm untuk jumlah maksimum operasi analisis dokumen asinkron.
6. Pilih tab Metrik bergrafik.
7. Untuk Statistik pilih Jumlah.
8. Pilih Pilih metrik.
9. Isi Nama dan Deskripsi. Untuk Kapan pun, pilih >=, dan masukkan nilai maksimum pilihan Anda.
10. Jika Anda ingin CloudWatch mengirimkan email jika status alarm sudah tercapai, untuk Kapan pun alarm ini:, pilih Status ALARM. Untuk mengirim alarm ke topik Amazon SNS yang sudah ada, untuk Kirim notifikasi ke:, pilih topik SNS yang sudah ada. Untuk mengatur nama dan alamat email untuk daftar langganan email baru, pilihDaftar baru. CloudWatch, simpan daftar dan tampilkan di bidang sehingga Anda dapat menggunakannya untuk mengatur alarm di masa mendatang.

    > **Note**
    >
    > Jika Anda menggunakanDaftar baruuntuk membuat topik Amazon SNS baru, alamat email harus diverifikasi sebelum penerima yang dituju menerima notifikasi. Amazon SNS hanya mengirim email ketika alarm memasuki status alarm. Jika perubahan status alarm ini terjadi sebelum alamat email diverifikasi, maka penerima yang dituju tidak akan menerima notifikasi.
11. Pilih Buat Alarm.

Untuk mengatur alarm (AWS CLI)

- Buka AWS CLI dan masukkan perintah berikut. Mengubah nilai`alarm-actions`parameter untuk mereferensikan topik Amazon SNS yang sebelumnya Anda buat.

```
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
```

Contoh ini menunjukkan cara membuat alarm ketika lebih dari 10 kesalahan pengguna terjadi dalam waktu 5 menit untuk panggilan`StartDocumentAnalysis`. Untuk informasi selengkapnya, lihat masukkan-metrik-alarm.

Untuk mengatur alarm (API CloudWatch)

- Panggil `PutMetricAlarm`. Untuk informasi selengkapnya, lihat Referensi API Amazon CloudWatch.

Metrik CloudWatch untuk Amazon Textract

Bagian ini memuat informasi tentang metrik Amazon CloudWatch danOperasidimensi yang tersedia untuk Amazon Textract.

Anda juga dapat melihat tampilan agregat metrik Amazon Textract Texact dari konsol Amazon Textract Texact.

Metrik CloudWatch untuk Amazon Textract

Tabel berikut merangkum metrik Amazon Textract.

Metrik	Deskripsi
SuccessfulRequestCount	Jumlah permintaan yang berhasil. Kisaran kode respons untuk permintaan yang berhasil adalah 200 hingga 299. Unit: Count Statistik valid: `Sum,Average`
ThrottledCount	Jumlah permintaan yang di-throttling. Amazon Texact men-throttling permintaan ketika menerima lebih banyak permintaan dari kuota transaksi per set detik untuk akun Anda. Jika batas yang ditetapkan untuk akun Anda sering terlampaui, Anda dapat meminta penambahan kuota. Untuk meminta penambahan, lihat Kuota Layanan AWS. Unit: Count Statistik valid: `Sum,Average`
ResponseTime	Waktu dalam milidetik untuk Amazon Textract untuk menghitung responsnya. Unit: 1. Jumlah untuk statistik `Data Samples` 2. Milidetik untuk statistik `Average` Statistik valid: `Data Samples,Average` **Note** ParameterResponseTime metrik tidak disertakan dalam panel metrik Amazon Textract Texact.
ServerErrorCount	Jumlah kesalahan server. Kisaran kode respons untuk kesalahan server adalah 500 hingga 599. Unit: Count

Metrik	Deskripsi
	Statistik valid: `Sum,Average`
UserErrorCount	Jumlah kesalahan pengguna (parameter tidak valid, citra tidak valid, tidak ada izin, dan sebagainya). Kisaran kode respons untuk kesalahan pengguna adalah 400 hingga 499. Unit: Count Statistik valid: `Sum,Average`

Dimensi CloudWatch untuk Amazon Textract

Untuk mengambil metrik khusus operasi, gunakan namespace `AWS/Textract` dan tentukan dimensi operasi. Untuk informasi selengkapnya tentang dimensi, lihat Dimensi di Panduan Pengguna Amazon CloudWatch.

Mencatat Panggilan Amazon Textract denganAWS CloudTrail

Amazon Textract terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atauAWSlayanan di Amazon Textract. CloudTrail menangkap semua panggilan API untuk Amazon Textract sebagai peristiwa. Panggilan yang direkam mencakup panggilan dari konsol Amazon Textract dan panggilan kode ke operasi API Amazon Textract.

Jika membuat jejak, Anda dapat mengaktifkan pengiriman kejadian CloudTrail berkelanjutan ke bucket Amazon S3, termasuk kejadian untuk Amazon Textract. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru dalam konsol CloudTrail di Riwayat peristiwa. Menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon Textract, alamat IP asal permintaan tersebut dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail lainnya.

Untuk mempelajari selengkapnya tentang CloudTrail, lihat Panduan Pengguna AWS CloudTrail.

Amazon Textract Informasi di CloudTrail

CloudTrail diaktifkan pada akun AWS Anda saat Anda membuat akun tersebut. Saat aktivitas terjadi di Amazon Textract, aktivitas tersebut dicatat di kejadian CloudTrail bersama aktivitas

lainnyaAWSacara layanan diRiwayat peristiwa. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk catatan kejadian berkelanjutan diAWSakun, termasuk peristiwa untuk Amazon Textract, membuat jejak. Sebuah Jejak mengaktifkan CloudTrail untuk mengirim berkas log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak tersebut diterapkan ke semua Wilayah AWS. Jejak mencatat kejadian dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengkonfigurasi lainnyaAWSlayanan untuk menganalisis lebih lanjut dan bertindak berdasarkan data peristiwa yang dikumpulkan di log CloudTrail. Untuk informasi selengkapnya, lihat yang berikut:

- Ikhtisar untuk Membuat Jejak
- Layanan yang Didukung dan Integrasi CloudTrail
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima Berkas Log CloudTrail dari Berbagai Wilayah dan Menerima Berkas Log CloudTrail dari Berbagai Akun

Semua operasi Amazon Textract Texact dicatat oleh CloudTrail dan didokumentasikan diReferensi API. Misalnya, panggilan untuk tindakan `DetectDocumentText` , `AnalyzeDocument`, dan `GetDocumentText` menghasilkan entri dengan berkas log CloudTrail.

Setiap entri peristiwa atau catatan berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut:

- Bahwa permintaan dibuat dengan kredensial pengguna root atau pengguna AWS Identity and Access Management (IAM).
- Bahwa permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Apakah permintaan dibuat oleh layanan AWS lain.

Untuk informasi lebih lanjut, lihat Elemen userIdentity CloudTrail.

Parameter Permintaan dan Bidang Respon yang Tidak Masuk

Untuk tujuan privasi, parameter permintaan tertentu dan bidang respons tidak di-loging—misalnya, meminta byte gambar atau informasi kotak pembatas respons. Nama bucket dan nama file Amazon

S3 yang disediakan dalam parameter permintaan disediakan dalam entri log CloudTrail. Tidak ada informasi tentang byte gambar yang diteruskan dalam permintaan disediakan dalam log CloudTrail. Tabel berikut menunjukkan parameter input dan parameter respons yang tidak dicatat untuk setiap operasi Amazon Textract.

Operasi	Parameter Permintaan	Bidang Respons
AnalyzeDocument	`Bytes`	Semua
DetectDocumentText	`Bytes`	Semua
StartDocumentAnalysis	Tidak ada	Tidak ada
GetDocumentAnalysis	Tidak ada	Semua
StartDocumentTextDetection	Tidak ada	Tidak ada
GetDocumentTextDetection	Tidak ada	Semua

Memahami Entri Berkas Log Amazon Textract

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai berkas log ke bucket Amazon S3 yang telah Anda tentukan. File log CloudTrail berisi satu atau beberapa entri log. Acara mewakili satu permintaan dari sumber apa pun dan mencakup informasi tentang operasi yang diminta, tanggal dan waktu operasi, parameter permintaan, dan sebagainya. File log CloudTrail bukan jejak tumpukan terurut dari panggilan API publik, sehingga berkas tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri log CloudTrail yang menunjukkan operasi `AnalyzeDocument`. Gambar byte untuk input`document`dan hasil analisis (`responseElements`) tidak login.

```
{
    "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"
}
```

Contoh berikut menunjukkan entri log CloudTrail untuk`StartDocumentAnalysis`operasi. Entri log menyertakan nama bucket Amazon S3 dan nama file citra di`documentLocation`. Log juga mencakup respon operasi.

```
{
    "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"
        }

    ]
}
```


Validasi Kepatuhan untuk Amazon Textract

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Textract sebagai bagian dari beberapaAWSprogram kepatuhan. Ini mencakup HIPAA, SOC, ISO, dan PCI.

> **Note**
>
> Jika Anda memproses data melalui layanan Texact yang tunduk pada kepatuhan PCI DSS maka Anda harus memilih keluar akun Anda dengan menghubungi AWS Support dan mengikuti proses yang diberikan kepada Anda.

Untuk daftar layanan AWS dalam cakupan program kepatuhan tertentu, lihat Layanan AWS dalam Cakupan Program Kepatuhan. Untuk informasi umum, lihat Program Kepatuhan AWS.

Anda bisa mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Pengunduhan Laporan dalam AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Amazon Textract ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta peraturan perundangan yang berlaku.AWSmenyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Quick Start Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah untuk deployment lingkungan dasar yang fokus pada keamanan dan kepatuhan di AWS.
- Merancang Laporan Resmi Keamanan dan Kepatuhan HIPAA – Laporan resmi ini menjelaskan cara perusahaan dapat menggunakan AWS untuk membuat aplikasi yang patuh-HIPAA.
- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan Aturan di Panduan Developer AWS Config – Layanan AWS Config menilai seberapa baik konfigurasi sumber daya Anda dalam mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub CSPM— IniAWSmenyediakan tampilan yang komprehensif tentang status keamanan Anda diAWS. Hub keamanan membantu Anda memeriksa kepatuhan terhadap standar industri dan praktik terbaik untuk keamanan.

Ketahanan di Amazon Textract

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Availability Zone. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan jaringan berlatensi rendah, throughput yang tinggi, dan sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Availability Zone lebih tersedia, memiliki toleransi kesalahan, dan dapat diskalakan dibandingkan dengan satu atau beberapa infrastruktur pusat data tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat AWS Infrastruktur Global.

> **Note**
>
> Transfer data lintas wilayah tidak diizinkan karena Peraturan Perlindungan Data Umum (GDPR).

Keamanan Infrastruktur di Amazon Textract

Sebagai layanan terkelola, Amazon Textract Texact dilindungi olehAWSprosedur keamanan jaringan global yang dijelaskan dalamAmazon Web Services: Whitepaper Ikhtisar Proses Keamanan.

Anda menggunakanAWSpanggilan API yang dipublikasikan untuk mengakses Amazon Textract melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.0 atau versi yang lebih baru. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru. Klien juga harus mendukung suite cipher dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan principal IAM. Atau Anda bisa menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Analisis Konfigurasi dan Kerentanan di Amazon Textract

Konfigurasi dan kontrol IT merupakan tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi selengkapnya, lihat model tanggung jawab bersama AWS.

Amazon Textract dan VPC endpoint antarmuka (AWS PrivateLink)

Anda dapat membangun koneksi privat antara VPC Anda dan Amazon Textract Texact dengan membuatTitik akhir antarmuka VPC. Endpoint antarmuka didukung olehAWS PrivateLink, teknologi yang memungkinkan Anda mengakses API Amazon Textract secara privat tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi AWS Direct Connect. Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan API Amazon Textract Texact. Lalu lintas antara VPC Anda dan Amazon Textract Texact tidak meninggalkan jaringan AWS.

Setiap titik akhir antarmuka diwakili oleh satu atau lebih Antarmuka Jaringan Elastis dalam subnet Anda.

Untuk informasi lebih lanjut, lihat VPC endpoint Antarmuka (AWS PrivateLink) dalam Panduan Pengguna Amazon VPC.

Pertimbangan untuk VPC endpoint Amazon Textract

Sebelum Anda menyiapkan VPC endpoint antarmuka untuk Amazon Textract, pastikan bahwa Anda meninjauProperti endpoint antarmuka dan keterbatasandiPanduan Pengguna Amazon VPC.

Amazon Textract Texact mendukung panggilan ke semua tindakan API-nya dari VPC Anda.

Membuat VPC endpoint antarmuka untuk Amazon Textract

Anda dapat membuat VPC endpoint untuk layanan Amazon Textract menggunakan konsol Amazon VPC atauAWS Command Line Interface(AWS CLI). Untuk informasi lebih lanjut, lihat Membuat titik akhir antarmuka di Panduan Pengguna Amazon VPC.

Buat VPC endpoint untuk Amazon Textract Texact menggunakan nama layanan berikut:

- com.amazonaws.*daerah*.textract - Untuk membuat endpoint untuk sebagian besar operasi Amazon Textract.
- com.amazonaws.*daerah*.textract-fips - Untuk membuat titik akhir untuk Amazon Textract yang sesuai dengan standar pemerintah AS Publikasi Federal Information Processing Standard (FIPS) 140-2.

Jika mengaktifkan DNS privat untuk titik akhir, Anda dapat mengajukan permintaan API ke Amazon Textract menggunakan nama DNS default untuk Wilayah, misalnya,`textract.us-east-1.amazonaws.com`.

Untuk informasi lebih lanjut, lihat Mengakses layanan melalui titik akhir antarmuka di Panduan Pengguna Amazon VPC.

Membuat kebijakan VPC endpoint untuk Amazon Textract

Anda dapat melampirkan kebijakan titik akhir ke VPC endpoint yang mengontrol akses ke Amazon Textract. Kebijakan menentukan informasi berikut ini:

- Prinsip-prinsip yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang dapat digunakan untuk mengambil tindakan.

Untuk informasi selengkapnya, lihat Mengendalikan akses ke layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Contoh: Kebijakan VPC endpoint untuk tindakan Amazon Textract

Berikut ini adalah contoh kebijakan titik akhir untuk Amazon Textract. Jika dilampirkan ke titik akhir, kebijakan ini memberikan akses ke tindakan Amazon Textract terdaftar untuk semua prinsip di semua sumber daya.

Contoh kebijakan ini memungkinkan akses ke hanya operasi`DetectDocumentText`dan`AnalyzeDocument`. Pengguna masih dapat memanggil operasi Amazon Textract Texact dari luar VPC Endpoint.

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

Referensi API

Bagian ini menyediakan dokumentasi untuk operasi API Amazon Textract Texact.

Topik

Tindakan

Tindakan berikut didukung:

AnalyzeDocument

Menganalisis dokumen masukan untuk hubungan antara item yang terdeteksi.

Jenis informasi yang dikembalikan adalah sebagai berikut:

- Data formulir (pasangan nilai kunci). Informasi terkait dikembalikan dalam duaBlockobjek, masing-masing jenisKEY_VALUE_SET: KUNCIBlockobjek dan NILAIBlockobjek. Misalnya,Nama: Ana Silvaberisi kunci dan nilai. Nama:adalah kuncinya. Ana Silvaadalah nilai.
- Tabel dan data sel tabel. TABELBlockobjek berisi informasi tentang tabel terdeteksi. SELBlockobjek dikembalikan untuk setiap sel dalam tabel.
- Garis dan kata-kata teks. GARISBlockobjek berisi satu atau lebih WORDBlockobjek. Semua baris dan kata-kata yang terdeteksi dalam dokumen dikembalikan (termasuk teks yang tidak memiliki hubungan dengan nilaiFeatureTypes).

Elemen seleksi seperti kotak centang dan tombol opsi (tombol radio) dapat dideteksi dalam data formulir dan dalam tabel. SELECTION_ELEMENTBlockobjek berisi informasi tentang elemen seleksi, termasuk status seleksi.

Anda dapat memilih jenis analisis yang akan dilakukan dengan menentukanFeatureTypesdaftar.

Output dikembalikan dalam daftarBlockobjek.

AnalyzeDocument adalah operasi tersinkron. Untuk menganalisis dokumen secara asinkron, gunakanStartDocumentAnalysis.

Untuk informasi selengkapnya, lihatAnalisis Teks Dokumen.

Sintaksis Permintaan

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

```
    "FeatureTypes": [ "string" ],
    "HumanLoopConfig": {
       "DataAttributes": {
          "ContentClassifiers": [ "string" ]
       },
       "FlowDefinitionArn": "string",
       "HumanLoopName": "string"
    }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

Document

Dokumen input sebagai byte yang dikodekan base64 atau objek Amazon S3. Jika Anda menggunakan AWS CLI untuk memanggil operasi Amazon Textract, Anda tidak dapat meneruskan byte citra. Dokumen harus berupa gambar dalam format JPEG, PNG, PDF, atau TIFF.

Jika Anda menggunakan AWS SDK untuk memanggil Amazon Textract, Anda mungkin tidak perlu byte citra yang dikodekan base64 yang diteruskan menggunakan`Bytes`bidang.

Tipe: Objek Document

Diperlukan: Ya

FeatureTypes

Daftar jenis analisis untuk melakukan. Tambahkan TABEL ke daftar untuk mengembalikan informasi tentang tabel yang terdeteksi dalam dokumen input. Tambahkan FORMS untuk mengembalikan data formulir yang terdeteksi. Untuk melakukan kedua jenis analisis, tambahkan TABEL dan FORM ke`FeatureTypes`. Semua baris dan kata yang terdeteksi dalam dokumen disertakan dalam respons (termasuk teks yang tidak terkait dengan nilai`FeatureTypes`).

Jenis: Array string

Nilai Valid: `TABLES | FORMS`

Diperlukan: Ya

HumanLoopConfig

Menetapkan konfigurasi untuk manusia dalam alur kerja loop untuk menganalisis dokumen.

Tipe: Objek HumanLoopConfig

Diperlukan: Tidak

Sintaksis Respons

```
{
   "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"
      }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

AnalyzeDocumentModelVersion

Versi model yang digunakan untuk menganalisis dokumen.

Jenis: String

Blocks

Item yang terdeteksi dan dianalisis oleh`AnalyzeDocument`.

Jenis: ArrayBlockobjek

DocumentMetadata

Metadata tentang dokumen yang dianalisis. Contohnya adalah jumlah halaman.

Tipe: Objek DocumentMetadata

HumanLoopActivationOutput

Menunjukkan hasil yang menunjukkan manusia dalam evaluasi loop.

Tipe: Objek HumanLoopActivationOutput

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

BadDocumentException

Amazon Textract tidak dapat membaca dokumen. Untuk informasi selengkapnya tentang batas dokumen di Amazon Textract, lihatBatas Keras di Amazon Textract.

Kode Status HTTP: 400

DocumentTooLargeException

Dokumen tidak dapat diproses karena terlalu besar. Ukuran dokumen maksimum untuk operasi sinkron 10 MB. Ukuran dokumen maksimum untuk operasi asinkron adalah 500 MB untuk file PDF.

Kode Status HTTP: 400

HumanLoopQuotaExceededException

Menunjukkan bahwa Anda telah melebihi jumlah maksimum manusia aktif dalam alur kerja loop yang tersedia

Kode Status HTTP: 400

InternalServerError

Amazon Textract Texact mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron, sebuah`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Konfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract Texact untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

UnsupportedDocumentException

Format dokumen input tidak didukung. Dokumen untuk operasi dapat dalam format PNG, JPEG, PDF, atau TIFF.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khususAWSSDK, lihat yang berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3

- AWSSDK for Python
- AWSSDK for Ruby V3

AnalyzeExpense

`AnalyzeExpense`serentak menganalisis dokumen masukan untuk hubungan finansial terkait antara teks.

Informasi dikembalikan sebagai`ExpenseDocuments`dan terpisah sebagai berikut.

- `LineItemGroups`- Sebuah set data yang berisi`LineItems`yang menyimpan informasi tentang baris teks, seperti barang yang dibeli dan harganya pada tanda terima.
- `SummaryFields`- Berisi semua informasi lain tanda terima, seperti informasi header atau nama vendor.

Sintaksis Permintaan

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

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

Document

Dokumen input, baik sebagai byte atau sebagai objek S3.

Anda meneruskan bit citra ke operasi API Amazon Textract dengan menggunakan bit`Bytes`properti. Misalnya, Anda akan menggunakan`Bytes`properti untuk lulus dokumen yang dimuat dari sistem file lokal. Gambar byte dilewatkan dengan menggunakan`Bytes`properti harus dikodekan dengan base64. Kode Anda mungkin tidak perlu mengodekan byte file dokumen jika Anda menggunakan AWS SDK untuk memanggil operasi API Amazon Textract.

Anda meneruskan citra yang disimpan dalam bucket S3 ke operasi API Amazon Textract dengan menggunakan bucket S3 dengan menggunakan bucket S3 ke operasi API Amazon TS30bjectproperti. Dokumen yang disimpan dalam bucket S3 tidak perlu dikodekan dengan base64.

Wilayah AWS untuk bucket S3 yang berisi objek S3 harus sesuai dengan Wilayah AWS yang Anda gunakan untuk operasi Amazon Textract.

Jika Anda menggunakan AWS CLI untuk memanggil operasi Amazon Textract, meneruskan bit citra menggunakan properti Bit tidak didukung. Anda harus mengunggah dokumen terlebih dahulu ke bucket Amazon S3, lalu memanggil operasi menggunakan properti S3Object.

Agar Amazon Textract memproses objek S3, pengguna harus memiliki izin untuk mengakses objek S3.

Tipe: Objek Document

Diperlukan: Ya

Sintaksis Respons

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

```
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

DocumentMetadata

Informasi tentang dokumen masukan.

Tipe: Objek DocumentMetadata

ExpenseDocuments

Biaya terdeteksi oleh Amazon Textract.

Jenis: ArrayExpenseDocumentobjek

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

BadDocumentException

Amazon Textract tidak dapat membaca dokumen. Untuk informasi selengkapnya tentang batas dokumen di Amazon Textract, lihatBatas Keras di Amazon Textract.

Kode Status HTTP: 400

DocumentTooLargeException

Dokumen tidak dapat diproses karena terlalu besar. Ukuran dokumen maksimum untuk operasi sinkron 10 MB. Ukuran dokumen maksimum untuk operasi asinkron adalah 500 MB untuk file PDF.

Kode Status HTTP: 400

InternalServerError

Amazon Textract mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron, sebuah`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Mengonfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

UnsupportedDocumentException

Format dokumen input tidak didukung. Dokumen untuk operasi dapat dalam format PNG, JPEG, PDF, atau TIFF.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu bahasa yang spesifikAWSSDK, lihat berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3
- AWSSDK for Python
- AWSSDK for Ruby V3

AnalyzeID

Menganalisis dokumen identitas untuk informasi yang relevan. Informasi ini diekstraksi dan dikembalikan sebagai`IdentityDocumentFields`, yang mencatat bidang normal dan nilai teks yang diekstraksi.Tidak seperti operasi Amazon Textract Texact lainnya,`AnalyzeID`tidak mengembalikan data Geometry.

Sintaksis Permintaan

```
{
   "DocumentPages": [
      {
         "Bytes": blob,
         "S3Object": {
            "Bucket": "string",
            "Name": "string",
            "Version": "string"
         }
      }
   ]
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

DocumentPages

Dokumen yang diteruskan ke AnalyzeID.

Jenis: ArrayDocumentobjek

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 2 item.

Diperlukan: Ya

Sintaksis Respons

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

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

AnalyzeIDModelVersion

Versi AnalyzeIdentity API yang digunakan untuk memproses dokumen.

Jenis: String

DocumentMetadata

Informasi tentang dokumen masukan.

Tipe: Objek DocumentMetadata

IdentityDocuments

Daftar dokumen yang diproses oleh AnalyzeID. Termasuk nomor yang menunjukkan tempat mereka dalam daftar dan struktur respon untuk dokumen.

Jenis: ArrayIdentityDocumentobjek

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

BadDocumentException

Amazon Textract tidak dapat membaca dokumen. Untuk informasi selengkapnya tentang batas dokumen di Amazon Textract, lihatBatas Keras di Amazon Textract.

Kode Status HTTP: 400

DocumentTooLargeException

Dokumen tidak dapat diproses karena terlalu besar. Ukuran dokumen maksimum untuk operasi sinkron 10 MB. Ukuran dokumen maksimum untuk operasi asinkron adalah 500 MB untuk file PDF.

Kode Status HTTP: 400

InternalServerError

Amazon Textract Texact mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron, sebuah`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Mengonfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

UnsupportedDocumentException

Format dokumen input tidak didukung. Dokumen untuk operasi dapat dalam format PNG, JPEG, PDF, atau TIFF.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3

- AWSSDK for Python
- AWSSDK for Ruby V3

DetectDocumentText

Mendeteksi teks dalam dokumen input. Amazon Textract dapat mendeteksi baris teks dan kata-kata yang membentuk baris teks. Dokumen input harus berupa gambar dalam format JPEG, PNG, PDF, atau TIFF.`DetectDocumentText`mengembalikan teks terdeteksi dalam arrayBlockobjek.

Setiap halaman dokumen memiliki sebagai terkait`Block`jenis HALAMAN. Setiap HALAMAN`Block`objek adalah induk dari LINE`Block`objek yang mewakili baris teks terdeteksi pada halaman. GARIS`Block`objek adalah orang tua untuk setiap kata yang membentuk baris. Kata-kata diwakili oleh`Block`objek tipe WORD.

`DetectDocumentText` adalah operasi tersinkron. Untuk menganalisis dokumen secara asinkron, gunakanStartDocumentTextDetection.

Untuk informasi selengkapnya, lihatPendeteksi Teks Dokumen.

Sintaksis Permintaan

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

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

Document

Dokumen input sebagai byte yang dikodekan base64 atau objek Amazon S3. Jika Anda menggunakan AWS CLI untuk memanggil operasi Amazon Textract, Anda tidak dapat meneruskan byte citra. Dokumen harus berupa gambar dalam format JPEG atau PNG.

Jika Anda menggunakan AWS SDK untuk memanggil Amazon Textract, Anda mungkin tidak perlu byte citra yang dikodekan base64 yang diteruskan menggunakan`Bytes`Bidang.

Tipe: Objek Document

Diperlukan: Ya

Sintaksis Respons

```
{
   "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
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Blocks

Susunan rangkaian`Block`objek yang berisi teks yang terdeteksi dalam dokumen.

Jenis: ArrayBlockobjek

DetectDocumentTextModelVersion

Jenis: String

DocumentMetadata

Metadata tentang dokumen. Ini berisi jumlah halaman yang terdeteksi dalam dokumen.

Tipe: Objek DocumentMetadata

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

BadDocumentException

Amazon Textract tidak dapat membaca dokumen. Untuk informasi selengkapnya tentang batas dokumen di Amazon Textract, lihatBatas Keras di Amazon Textract.

Kode Status HTTP: 400

DocumentTooLargeException

Dokumen tidak dapat diproses karena terlalu besar. Ukuran dokumen maksimum untuk operasi sinkron 10 MB. Ukuran dokumen maksimum untuk operasi asinkron adalah 500 MB untuk file PDF.

Kode Status HTTP: 400

InternalServerError

Amazon Textract Texact mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron,`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Konfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract Texact untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

UnsupportedDocumentException

Format dokumen input tidak didukung. Dokumen untuk operasi dapat dalam format PNG, JPEG, PDF, atau TIFF.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu spesifik bahasaAWSSDK, lihat yang berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3
- AWSSDK for Python
- AWSSDK for Ruby V3

GetDocumentAnalysis

Mendapat hasil untuk operasi asinkron Amazon Texact yang menganalisis teks dalam dokumen.

Anda memulai analisis teks asinkron dengan meneleponStartDocumentAnalysis, yang mengembalikan pengenal pekerjaan (`JobId`). Ketika operasi analisis teks selesai, Amazon Textract menerbitkan status selesai untuk topik Amazon Simple Notification Service (Amazon SNS) yang terdaftar dalam panggilan awal untuk`StartDocumentAnalysis`. Untuk mendapatkan hasil operasi deteksi teks, periksa terlebih dahulu bahwa nilai status yang diterbitkan ke topik Amazon SNS adalah`SUCCEEDED`. Jika ya, hubungi`GetDocumentAnalysis`, dan lulus pengenal pekerjaan (`JobId`) dari panggilan awal ke`StartDocumentAnalysis`.

`GetDocumentAnalysis`mengembalikan arrayBlockobjek. Jenis informasi berikut ini dikembalikan:

- Data formulir (pasangan nilai kunci). Informasi terkait dikembalikan duaBlockobjek, masing-masing jenis`KEY_VALUE_SET`: KUNCI`Block`objek dan NILAI`Block`objek. Misalnya,Nama: Ana Silvaberisi kunci dan nilai. Nama:adalah kuncinya. Ana Silvaadalah nilai.
- Tabel dan data sel tabel. TABEL`Block`objek berisi informasi tentang tabel terdeteksi. SEL`Block`objek dikembalikan untuk setiap sel dalam tabel.
- Garis dan kata-kata teks. GARIS`Block`objek berisi satu atau lebih WORD`Block`objek. Semua baris dan kata-kata yang terdeteksi dalam dokumen dikembalikan (termasuk teks yang tidak memiliki hubungan dengan nilai`StartDocumentAnalysis` `FeatureTypes`parameter masukan).

Elemen seleksi seperti kotak centang dan tombol opsi (tombol radio) dapat dideteksi dalam data formulir dan dalam tabel. SELECTION_ELEMENT`Block`objek berisi informasi tentang elemen seleksi, termasuk status seleksi.

Gunakan`MaxResults`parameter untuk membatasi jumlah blok yang dikembalikan. Jika hasil yang didapatkan lebih banyak daripada yang ditentukan dalam `MaxResults`, nilai `NextToken` dalam respons operasi berisi token pemberian nomor halaman untuk mendapatkan serangkaian hasil berikutnya. Untuk mendapatkan halaman hasil berikutnya, hubungi`GetDocumentAnalysis`, dan mengisi`NextToken`parameter permintaan dengan nilai token yang dikembalikan dari panggilan sebelumnya ke`GetDocumentAnalysis`.

Untuk informasi selengkapnya, lihatAnalisis Teks Dokumen.

Sintaksis Permintaan

```
{
```

```
    "JobId": "string",
    "MaxResults": number,
    "NextToken": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

JobId

Pengenal unik untuk tugas deteksi teks. `JobId` dikembalikan dari `StartDocumentAnalysis`. SEBUAH`JobId`Nilai hanya berlaku selama 7 hari.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Diperlukan: Ya

MaxResults

Jumlah hasil maksimum untuk mengembalikan per panggilan yang diberi nomor halaman. Nilai terbesar yang dapat Anda tentukan adalah 1.000. Jika Anda menentukan nilai yang lebih besar dari 1.000, maksimum 1.000 hasil dikembalikan. Nilai default adalah 1,000.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 1.

Diperlukan: Tidak

NextToken

Jika respons sebelumnya tidak lengkap (karena ada lebih banyak blok untuk diambil), Amazon Textract token pemberian nomor halaman sebagai responsnya. Anda dapat menggunakan token pemberian nomor halaman ini untuk mengambil set blok berikutnya.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum 255.

Pola: `.*\S.*`

Diperlukan: Tidak

Sintaksis Respons

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

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

AnalyzeDocumentModelVersion

Jenis: String

Blocks

Hasil operasi teks-analisis.

Jenis: ArrayBlockobjek

DocumentMetadata

Informasi tentang dokumen yang diproses Amazon Textract Texact.`DocumentMetadata`adikembalikan di setiap halaman respons pemberian nomor halaman dari operasi Amazon Textract video.

Tipe: Objek DocumentMetadata

JobStatus

Status tugas deteksi teks.

Jenis: Rangkaian

Nilai Valid: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

Jika respons terpotong, Amazon Textract mengembalikan token ini. Anda dapat menggunakan token ini dalam permintaan berikutnya untuk mengambil serangkaian hasil deteksi teks berikutnya.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum 255.

Pola: `.*\S.*`

StatusMessage

Pengembalian jika pekerjaan deteksi tidak dapat diselesaikan. Berisi penjelasan tentang kesalahan apa yang terjadi.

Jenis: String

Warnings

Daftar peringatan yang terjadi selama operasi dokumen-analisis.

Jenis: ArrayWarningobjek

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

InternalServerError

Amazon Textract Texact mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidJobIdException

Pengenal pekerjaan yang tidak valid diteruskan keGetDocumentAnalysisatau untukGetDocumentAnalysis.

Kode Status HTTP: 400

InvalidKMSKeyException

Menunjukkan bahwa Anda tidak memiliki izin mendekripsi dengan kunci KMS yang dimasukkan, atau kunci KMS dimasukkan secara tidak benar.

Kode Status HTTP: 400

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron,`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Mengkonfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract Texact untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu bahasa khususAWSSDK, lihat yang berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3
- AWSSDK for Python
- AWSSDK for Ruby V3

GetDocumentTextDetection

Mendapat hasil untuk operasi asinkron Amazon Texact yang mendeteksi teks dalam dokumen. Amazon Textract dapat mendeteksi baris teks dan kata-kata yang membentuk baris teks.

Anda memulai deteksi teks asinkron dengan meneleponStartDocumentTextDetection, yang mengembalikan pengenal pekerjaan (`JobId`). Ketika operasi deteksi teks selesai, Amazon Textract menerbitkan status penyelesaian ke topik Amazon Simple Notification Service (Amazon SNS) yang terdaftar dalam panggilan awal.`StartDocumentTextDetection`. Untuk mendapatkan hasil operasi deteksi teks, periksa terlebih dahulu bahwa nilai status yang diterbitkan ke topik Amazon SNS adalah`SUCCEEDED`. Jika ya, hubungi`GetDocumentTextDetection`, dan lulus pengenal pekerjaan (`JobId`) dari panggilan awal ke`StartDocumentTextDetection`.

`GetDocumentTextDetection`mengembalikan arrayBlockbenda.

Setiap halaman dokumen memiliki sebagai terkait`Block`jenis HALAMAN. Setiap HALAMAN`Block`objek adalah induk dari LINE`Block`objek yang mewakili baris teks terdeteksi pada halaman. GARIS`Block`objek adalah orang tua untuk setiap kata yang membentuk baris. Kata-kata diwakili oleh`Block`objek tipe WORD.

Gunakan parameter MaxResults untuk membatasi jumlah blok yang dikembalikan. Jika hasil yang didapatkan lebih banyak daripada yang ditentukan dalam `MaxResults`, nilai `NextToken` dalam respons operasi berisi token pemberian nomor halaman untuk mendapatkan serangkaian hasil berikutnya. Untuk mendapatkan halaman hasil berikutnya, hubungi`GetDocumentTextDetection`, dan mengisi`NextToken`parameter permintaan dengan nilai token yang dikembalikan dari panggilan sebelumnya ke`GetDocumentTextDetection`.

Untuk informasi selengkapnya, lihatPendeteksi Teks.

Sintaksis Permintaan

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

JobId

Pengidentifikasi unik untuk tugas deteksi teks. `JobId` dikembalikan dari `StartDocumentTextDetection`. SEBUAH`JobId`Nilai hanya berlaku selama 7 hari.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Diperlukan: Ya

MaxResults

Jumlah hasil maksimum untuk mengembalikan per panggilan yang diberi nomor halaman. Nilai terbesar yang dapat Anda tentukan adalah 1.000. Jika Anda menentukan nilai yang lebih besar dari 1.000, maksimum hasil dikembalikan adalah 1.000. Nilai default adalah 1,000.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 1.

Diperlukan: Tidak

NextToken

Jika respons sebelumnya tidak lengkap (karena ada lebih banyak blok untuk diambil), Amazon Textract mengembalikan token pemberian nomor halaman sebagai responsnya. Anda dapat menggunakan token pemberian nomor halaman ini untuk mengambil set blok berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 255.

Pola: `.*\S.*`

Diperlukan: Tidak

Sintaksis Respons

```
{
   "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 ]
```

```
        }
    ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

Blocks

Hasil operasi deteksi teks.

Jenis: ArrayBlockobjek

DetectDocumentTextModelVersion

Jenis: String

DocumentMetadata

Informasi tentang dokumen yang diproses Amazon Textract Texact.`DocumentMetadata`adikembalikan di setiap halaman respons yang diberi nomor halaman dari operasi video Amazon Textract.

Tipe: Objek DocumentMetadata

JobStatus

Status terkini tugas deteksi teks.

Jenis: Rangkaian

Nilai Valid: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

Jika respons terpotong, Amazon Textract mengembalikan token ini. Anda dapat menggunakan token ini dalam permintaan berikutnya untuk mengambil set hasil deteksi teks berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 255.

Pola: `.*\S.*`

StatusMessage

Pengembalian jika pekerjaan deteksi tidak dapat diselesaikan. Berisi penjelasan tentang kesalahan apa yang terjadi.

Jenis: String

Warnings

Daftar peringatan yang terjadi selama operasi deteksi teks untuk dokumen.

Jenis: ArrayWarningobjek

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

InternalServerError

Amazon Textract Texact mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidJobIdException

Pengenal pekerjaan yang tidak valid diteruskan keGetDocumentAnalysisatau untukGetDocumentAnalysis.

Kode Status HTTP: 400

InvalidKMSKeyException

Menunjukkan bahwa Anda tidak memiliki izin mendekripsi dengan kunci KMS yang dimasukkan, atau kunci KMS dimasukkan secara tidak benar.

Kode Status HTTP: 400

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron,`InvalidParameterException`pengecualian terjadi ketika salah

satuS3ObjectatauBytesnilai-nilai yang disediakan dalamDocumentparameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Konfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract Texact untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu spesifik bahasaAWSSDK, lihat yang berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3
- AWSSDK for Python

- AWSSDK for Ruby V3

GetExpenseAnalysis

Mendapat hasil untuk operasi asinkron Amazon Texact yang menganalisis faktur dan tanda terima. Amazon Textract menemukan informasi kontak, item yang dibeli, dan nama vendor, dari faktur masukan dan tanda terima.

Anda memulai analisis faktur asinkron/tanda terima dengan meneleponStartExpenseAnalysis, yang mengembalikan pengenal pekerjaan (`JobId`). Setelah menyelesaikan analisis faktur, Amazon Texact menerbitkan status penyelesaian ke topik Amazon Simple Notification Service (Amazon SNS). Topik ini harus didaftarkan dalam panggilan awal`StartExpenseAnalysis`. Untuk mendapatkan hasil operasi analisis faktur, pastikan terlebih dahulu bahwa nilai status yang diterbitkan ke topik Amazon SNS adalah`SUCCEEDED`. Jika ya, hubungi`GetExpenseAnalysis`, dan lulus pengenal pekerjaan (`JobId`) dari panggilan awal ke`StartExpenseAnalysis`.

Gunakan parameter MaxResults untuk membatasi jumlah blok yang dikembalikan. Jika hasil yang didapatkan lebih banyak daripada yang ditentukan dalam `MaxResults`, nilai `NextToken` dalam respons operasi berisi token pemberian nomor halaman untuk mendapatkan serangkaian hasil berikutnya. Untuk mendapatkan halaman hasil berikutnya, hubungi`GetExpenseAnalysis`, dan mengisi`NextToken`parameter permintaan dengan nilai token yang dikembalikan dari panggilan sebelumnya ke`GetExpenseAnalysis`.

Untuk informasi selengkapnya, lihatMenganalisis Faktur dan Penerimaan.

Sintaksis Permintaan

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

JobId

Penidentifikasi unik untuk tugas deteksi teks. `JobId` dikembalikan dari `StartExpenseAnalysis`. SEBUAH`JobId`Nilai hanya berlaku selama 7 hari.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Diperlukan: Ya

MaxResults

Jumlah hasil maksimum untuk mengembalikan per panggilan yang diberi nomor halaman. Nilai terbesar yang dapat Anda tentukan adalah 20. Jika Anda menentukan nilai yang lebih besar dari 20, maksimum hasil dikembalikan adalah 20. Nilai default adalah 20.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 1.

Diperlukan: Tidak

NextToken

Jika respons sebelumnya tidak lengkap (karena ada lebih banyak blok untuk diambil), Amazon Textract mengembalikan token pemberian nomor halaman sebagai responsnya. Anda dapat menggunakan token pemberian nomor halaman ini untuk mengambil set blok berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 255.

Pola: `.*\S.*`

Diperlukan: Tidak

Sintaksis Respons

```
{
   "AnalyzeExpenseModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
```

```
            "LineItems": [
               {
                  "LineItemExpenseFields": [
                     {
                        "LabelDetection": {
                           "Confidence": number,
                           "Geometry": {
                              "BoundingBox": {
                                 "Height": number,
                                 "Left": number,
                                 "Top": number,
                                 "Width": number
                              },
                              "Polygon": [
                                 {
                                    "X": number,
                                    "Y": number
                                 }
                              ]
                           },
                           "Text": "string"
                        },
                        "PageNumber": number,
                        "Type": {
                           "Confidence": number,
                           "Text": "string"
                        },
                        "ValueDetection": {
                           "Confidence": number,
                           "Geometry": {
                              "BoundingBox": {
                                 "Height": number,
                                 "Left": number,
                                 "Top": number,
                                 "Width": number
                              },
                              "Polygon": [
                                 {
                                    "X": number,
                                    "Y": number
                                 }
                              ]
                           },
                           "Text": "string"
```

```
                        }
                     }
                  ]
               }
            ]
         }
      ],
      "SummaryFields": [
         {
            "LabelDetection": {
               "Confidence": number,
               "Geometry": {
                  "BoundingBox": {
                     "Height": number,
                     "Left": number,
                     "Top": number,
                     "Width": number
                  },
                  "Polygon": [
                     {
                        "X": number,
                        "Y": number
                     }
                  ]
               },
               "Text": "string"
            },
            "PageNumber": number,
            "Type": {
               "Confidence": number,
               "Text": "string"
            },
            "ValueDetection": {
               "Confidence": number,
               "Geometry": {
                  "BoundingBox": {
                     "Height": number,
                     "Left": number,
                     "Top": number,
                     "Width": number
                  },
                  "Polygon": [
                     {
                        "X": number,
                        "Y": number,
```

```
                        "Y": number
                     }
                  ]
               },
               "Text": "string"
            }
         }
      ]
   }
   ],
   "JobStatus": "string",
   "NextToken": "string",
   "StatusMessage": "string",
   "Warnings": [
      {
         "ErrorCode": "string",
         "Pages": [ number ]
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

[AnalyzeExpenseModelVersion](#)

Versi model saat AnalyzeExpense.

Jenis: String

[DocumentMetadata](#)

Informasi tentang dokumen yang diproses Amazon Textract Texact.`DocumentMetadata`dikembalikan di setiap halaman respons pemberian nomor halaman dari operasi Amazon Textract.

Tipe: Objek [DocumentMetadata](#)

[ExpenseDocuments](#)

Biaya yang terdeteksi oleh Amazon Textract.

Jenis: ArrayExpenseDocumentobjek

JobStatus

Status terkini tugas deteksi teks.

Jenis: Rangkaian

Nilai Valid: `IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS`

NextToken

Jika respons terpotong, Amazon Textract mengembalikan token ini. Anda dapat menggunakan token ini dalam permintaan berikutnya untuk mengambil serangkaian hasil deteksi teks berikutnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 255.

Pola: `.*\S.*`

StatusMessage

Pengembalian jika pekerjaan deteksi tidak dapat diselesaikan. Berisi penjelasan tentang kesalahan apa yang terjadi.

Jenis: String

Warnings

Daftar peringatan yang terjadi selama operasi deteksi teks untuk dokumen.

Jenis: ArrayWarningobjek

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

InternalServerError

Amazon Textract Texact mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidJobIdException

Pengenal pekerjaan yang tidak valid diteruskan keGetDocumentAnalysisatau untukGetDocumentAnalysis.

Kode Status HTTP: 400

InvalidKMSKeyException

Menunjukkan bahwa Anda tidak memiliki izin mendekripsi dengan kunci KMS yang dimasukkan, atau kunci KMS dimasukkan secara tidak benar.

Kode Status HTTP: 400

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron,`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract Texact tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Konfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract Texact untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu spesifik bahasaAWSSDK, lihat yang berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3
- AWSSDK for Python
- AWSSDK for Ruby V3

StartDocumentAnalysis

Mulai analisis asinkron dari dokumen masukan untuk hubungan antara item terdeteksi seperti pasangan kunci-nilai, tabel, dan elemen seleksi.

`StartDocumentAnalysis`dapat menganalisis teks dalam dokumen yang ada dalam format JPEG, PNG, TIFF, dan PDF. Dokumen-dokumen tersebut disimpan di bucket Amazon S3. GunakanDocumentLocationuntuk menentukan nama bucket dan nama file dokumen.

`StartDocumentAnalysis`mengembalikan pengenal pekerjaan (`JobId`) yang Anda gunakan untuk mendapatkan hasil operasi. Ketika analisis teks selesai, Amazon Textract menerbitkan status selesai untuk topik Amazon Simple Notification Service (Amazon SNS) yang Anda tentukan di`NotificationChannel`. Untuk mendapatkan hasil operasi analisis teks, periksa terlebih dahulu bahwa nilai status yang diterbitkan ke topik Amazon SNS adalah`SUCCEEDED`. Jika ya, hubungiGetDocumentAnalysis, dan lulus pengenal pekerjaan (`JobId`) dari panggilan awal ke`StartDocumentAnalysis`.

Untuk informasi selengkapnya, lihatAnalisis Teks Dokumen.

Sintaksis Permintaan

```
{
   "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"
   }
```

```
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ClientRequestToken

Token idempotensi yang Anda gunakan untuk mengidentifikasi permintaan mulai. Jika Anda menggunakan token yang sama dengan beberapa permintaan `StartDocumentAnalysis`, `JobId` yang sama dikembalikan. Gunakan`ClientRequestToken`untuk mencegah agar tidak ada tugas yang sama yang dimulai secara tidak sengaja lebih dari sekali. Untuk informasi selengkapnya, lihatMemanggil Operasi Asinkron Amazon Texact.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Diperlukan: Tidak

DocumentLocation

Lokasi dokumen yang akan diproses.

Tipe: Objek DocumentLocation

Diperlukan: Ya

FeatureTypes

Daftar jenis analisis untuk melakukan. Tambahkan TABEL ke daftar untuk mengembalikan informasi tentang tabel yang terdeteksi dalam dokumen input. Tambahkan FORMS untuk mengembalikan data formulir yang terdeteksi. Untuk melakukan kedua jenis analisis, tambahkan TABEL dan BENTUK ke`FeatureTypes`. Semua baris dan kata yang terdeteksi dalam dokumen disertakan dalam respons (termasuk teks yang tidak terkait dengan nilai`FeatureTypes`).

Jenis: Array string

Nilai Valid: `TABLES | FORMS`

Diperlukan: Ya

JobTag

Pengenal yang Anda tetapkan yang disertakan dalam notifikasi penyelesaian yang dipublikasikan ke topik Amazon SNS. Misalnya, Anda dapat menggunakan`JobTag`untuk mengidentifikasi jenis dokumen yang sesuai dengan pemberitahuan penyelesaian (seperti formulir pajak atau tanda terima).

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9_.\-:]+`

Diperlukan: Tidak

KMSKeyId

Kunci KMS yang digunakan untuk mengenkripsi hasil kesimpulan. Hal ini dapat baik dalam Key ID atau Key Alias format. Ketika kunci KMS disediakan, kunci KMS akan digunakan untuk enkripsi sisi server dari objek dalam ember pelanggan. Ketika parameter ini tidak diaktifkan, hasilnya akan dienkripsi sisi server, menggunakan SSE-S3.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

Diperlukan: Tidak

NotificationChannel

ARN topik Amazon SNS yang Anda inginkan agar Amazon Textract mempublikasikan status penyelesaian operasi.

Tipe: Objek NotificationChannel

Diperlukan: Tidak

OutputConfig

Set jika output akan pergi ke ember pelanggan didefinisikan. Secara default, Amazon Textract akan menyimpan hasil secara internal untuk diakses oleh operasi GetDocumentAnalysis.

Tipe: Objek OutputConfig

Diperlukan: Tidak

Sintaksis Respons

```
{
   "JobId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

JobId

Pengidentifikasi untuk tugas deteksi teks dokumen. Gunakan `JobId` untuk mengidentifikasi tugas dalam panggilan berikutnya ke `GetDocumentAnalysis`. SEBUAH`JobId`Nilai hanya berlaku selama 7 hari.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

BadDocumentException

Amazon Textract tidak dapat membaca dokumen. Untuk informasi selengkapnya tentang batas dokumen di Amazon Textract, lihatBatas Keras di Amazon Textract.

Kode Status HTTP: 400

DocumentTooLargeException

Dokumen tidak dapat diproses karena terlalu besar. Ukuran dokumen maksimum untuk operasi sinkron 10 MB. Ukuran dokumen maksimum untuk operasi asinkron adalah 500 MB untuk file PDF.

Kode Status HTTP: 400

IdempotentParameterMismatchException

Parameter input `ClientRequestToken` digunakan kembali dengan suatu operasi, tapi setidaknya salah satu parameter input lainnya berbeda dari panggilan ke operasi sebelumnya.

Kode Status HTTP: 400

InternalServerError

Amazon Textract mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidKMSKeyException

Menunjukkan bahwa Anda tidak memiliki izin mendekripsi dengan kunci KMS yang dimasukkan, atau kunci KMS dimasukkan secara tidak benar.

Kode Status HTTP: 400

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron,`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Mengkonfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

LimitExceededException

Batas layanan Amazon Textract terlampaui. Misalnya, jika Anda memulai terlalu banyak pekerjaan asinkron secara bersamaan, panggilan untuk memulai operasi (`StartDocumentTextDetection`Misalnya) menaikkan pengecualian LimitExceededException (kode status HTTP: 400) hingga jumlah tugas yang berjalan bersamaan di bawah batas layanan Amazon Textract TExceededException.

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

UnsupportedDocumentException

Format dokumen input tidak didukung. Dokumen untuk operasi dapat dalam format PNG, JPEG, PDF, atau TIFF.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu bahasa yang spesifikAWSSDK, lihat yang berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript

- AWSSDK for PHP V3
- AWSSDK for Python
- AWSSDK for Ruby V3

StartDocumentTextDetection

Mulai deteksi tidak sinkron terhadap teks dalam dokumen. Amazon Textract dapat mendeteksi baris teks dan kata-kata yang membentuk baris teks.

`StartDocumentTextDetection`dapat menganalisis teks dalam dokumen yang ada dalam format JPEG, PNG, TIFF, dan PDF. Dokumen-dokumen tersebut disimpan di bucket Amazon S3. GunakanDocumentLocationuntuk menentukan nama bucket dan nama file dokumen.

`StartTextDetection`mengembalikan pengenal pekerjaan (`JobId`) yang Anda gunakan untuk mendapatkan hasil operasi. Ketika deteksi teks selesai, Amazon Textract menerbitkan status selesai untuk topik Amazon Simple Notification Service (Amazon SNS) yang Anda tentukan di`NotificationChannel`. Untuk mendapatkan hasil operasi deteksi teks, periksa terlebih dahulu bahwa nilai status yang diterbitkan ke topik Amazon SNS adalah`SUCCEEDED`. Jika ya, hubungiGetDocumentTextDetection, dan lulus pengenal pekerjaan (`JobId`) dari panggilan awal ke`StartDocumentTextDetection`.

Untuk informasi selengkapnya, lihatPendeteksi Teks Dokumen.

Sintaksis Permintaan

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

ClientRequestToken

Token idempotensi yang digunakan untuk mengidentifikasi permintaan mulai. Jika Anda menggunakan token yang sama dengan beberapa permintaan `StartDocumentTextDetection`, `JobId` yang sama dikembalikan. Gunakan`ClientRequestToken`untuk mencegah agar tidak ada tugas yang sama yang dimulai secara tidak sengaja lebih dari sekali. Untuk informasi selengkapnya, lihatMemanggil Operasi Asinkron Amazon Texact.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Diperlukan: Tidak

DocumentLocation

Lokasi dokumen yang akan diproses.

Tipe: Objek DocumentLocation

Diperlukan: Ya

JobTag

Pengenal yang Anda tetapkan yang disertakan dalam notifikasi penyelesaian yang dipublikasikan ke topik Amazon SNS. Misalnya, Anda dapat menggunakan`JobTag`untuk mengidentifikasi jenis dokumen yang sesuai dengan pemberitahuan penyelesaian (seperti formulir pajak atau tanda terima).

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9_.\-:]+`

Diperlukan: Tidak

KMSKeyId

Kunci KMS yang digunakan untuk mengenkripsi hasil inferensi. Hal ini dapat baik dalam Key ID atau Key Alias format. Ketika kunci KMS disediakan, kunci KMS akan digunakan untuk enkripsi sisi server dari objek dalam ember pelanggan. Ketika parameter ini tidak diaktifkan, hasilnya akan dienkripsi sisi server, menggunakan SSE-S3.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

Diperlukan: Tidak

NotificationChannel

ARN topik Amazon SNS yang Anda inginkan agar Amazon Textract mempublikasikan status selesai operasi.

Tipe: Objek NotificationChannel

Diperlukan: Tidak

OutputConfig

Set jika output akan pergi ke ember pelanggan didefinisikan. Secara default Amazon Textract akan menyimpan hasil secara internal untuk diakses dengan operasi GetDocumentTextDetection.

Tipe: Objek OutputConfig

Diperlukan: Tidak

Sintaksis Respons

```
{
   "JobId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

JobId

Pengidentifikasi tugas deteksi teks untuk dokumen tersebut. Gunakan `JobId` untuk mengidentifikasi tugas dalam panggilan berikutnya ke `GetDocumentTextDetection`. SEBUAH`JobId`Nilai hanya berlaku selama 7 hari.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

BadDocumentException

Amazon Textract tidak dapat membaca dokumen. Untuk informasi selengkapnya tentang batas dokumen di Amazon Textract, lihatBatas Keras di Amazon Textract.

Kode Status HTTP: 400

DocumentTooLargeException

Dokumen tidak dapat diproses karena terlalu besar. Ukuran dokumen maksimum untuk operasi sinkron 10 MB. Ukuran dokumen maksimum untuk operasi asinkron adalah 500 MB untuk file PDF.

Kode Status HTTP: 400

IdempotentParameterMismatchException

Parameter input `ClientRequestToken` digunakan kembali dengan suatu operasi, tapi setidaknya salah satu parameter input lainnya berbeda dari panggilan ke operasi sebelumnya.

Kode Status HTTP: 400

InternalServerError

Amazon Textract mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidKMSKeyException

Menunjukkan bahwa Anda tidak memiliki izin mendekripsi dengan kunci KMS yang dimasukkan, atau kunci KMS dimasukkan secara tidak benar.

Kode Status HTTP: 400

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron, sebuah`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Konfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

LimitExceededException

Batas layanan Amazon Textract terlampaui. Misalnya, jika Anda memulai terlalu banyak pekerjaan asinkron secara bersamaan, panggilan untuk memulai operasi (`StartDocumentTextDetection`, misalnya) menaikkan pengecualian LimitExceededException (kode status HTTP: 400) hingga jumlah tugas yang berjalan bersamaan di bawah batas layanan Amazon Textract TException.

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

UnsupportedDocumentException

Format dokumen input tidak didukung. Dokumen untuk operasi dapat dalam format PNG, JPEG, PDF, atau TIFF.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khususAWSSDK, lihat berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3
- AWSSDK for Python
- AWSSDK for Ruby V3

StartExpenseAnalysis

Memulai analisis asinkron faktur atau tanda terima untuk data seperti informasi kontak, item yang dibeli, dan nama vendor.

`StartExpenseAnalysis`dapat menganalisis teks dalam dokumen yang dalam format JPEG, PNG, dan PDF. Dokumen harus disimpan di bucket Amazon S3. GunakanDocumentLocationparameter untuk menentukan nama bucket S3 Anda dan nama dokumen dalam ember itu.

`StartExpenseAnalysis`mengembalikan pengenal pekerjaan (`JobId`) yang akan Anda berikan`GetExpenseAnalysis`untuk mengambil hasil operasi. Ketika analisis faktur input/tanda terima selesai, Amazon Textract menerbitkan status penyelesaian ke topik Amazon Simple Notification Service (Amazon SNS) yang Anda berikan kepada`NotificationChannel`. Untuk mendapatkan hasil operasi analisis faktur dan penerimaan, pastikan bahwa nilai status yang diterbitkan ke topik Amazon SNS adalah`SUCCEEDED`. Jika ya, hubungiGetExpenseAnalysis, dan lulus pengenal pekerjaan (`JobId`) yang dikembalikan oleh panggilan Anda ke`StartExpenseAnalysis`.

Untuk informasi selengkapnya, lihatMenganalisis Faktur dan Penerimaan.

Sintaksis Permintaan

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

Parameter Permintaan

Permintaan menerima data berikut dalam format JSON.

[ClientRequestToken](#)

Token idempotensi yang digunakan untuk mengidentifikasi permintaan mulai. Jika Anda menggunakan token yang sama dengan beberapa permintaan `StartDocumentTextDetection`, `JobId` yang sama dikembalikan. Gunakan`ClientRequestToken`untuk mencegah agar tidak ada tugas yang sama yang dimulai lebih dari sekali. Untuk informasi selengkapnya, lihat[Memanggil Operasi Asinkron Amazon Texact](#)

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Diperlukan: Tidak

[DocumentLocation](#)

Lokasi dokumen yang akan diproses.

Tipe: Objek [DocumentLocation](#)

Diperlukan: Ya

[JobTag](#)

Pengidentifikasi yang Anda tentukan yang disertakan dalam notifikasi penyelesaian yang diterbitkan ke topik Amazon SNS. Misalnya, Anda dapat menggunakan`JobTag`untuk mengidentifikasi jenis dokumen yang sesuai dengan pemberitahuan penyelesaian (seperti formulir pajak atau tanda terima).

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9_.\-:]+`

Diperlukan: Tidak

KMSKeyId

Kunci KMS yang digunakan untuk mengenkripsi hasil kesimpulan. Hal ini dapat baik dalam Key ID atau Key Alias format. Ketika kunci KMS disediakan, kunci KMS akan digunakan untuk enkripsi sisi server dari objek dalam ember pelanggan. Ketika parameter ini tidak diaktifkan, hasilnya akan dienkripsi sisi server, menggunakan SSE-S3.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum 2048.

Pola: `^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

Diperlukan: Tidak

NotificationChannel

ARN topik Amazon SNS yang Anda inginkan Amazon Textract mempublikasikan status penyelesaian operasi.

Tipe: Objek NotificationChannel

Diperlukan: Tidak

OutputConfig

Set jika output akan pergi ke ember pelanggan didefinisikan. Secara default, Amazon Textract akan menyimpan hasil secara internal untuk diakses oleh`GetExpenseAnalysis`operasi.

Tipe: Objek OutputConfig

Diperlukan: Tidak

Sintaksis Respons

```
{
   "JobId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

JobId

Pengidentifikasi unik untuk tugas deteksi teks. `JobId` dikembalikan dari `StartExpenseAnalysis`. SEBUAH`JobId`Nilai hanya berlaku selama 7 hari.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `^[a-zA-Z0-9-_]+$`

Kesalahan

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Kode Status HTTP: 400

BadDocumentException

Amazon Textract tidak dapat membaca dokumen. Untuk informasi selengkapnya tentang batas dokumen di Amazon Textract, lihatBatas Keras di Amazon Textract.

Kode Status HTTP: 400

DocumentTooLargeException

Dokumen tidak dapat diproses karena terlalu besar. Ukuran dokumen maksimum untuk operasi sinkron 10 MB. Ukuran dokumen maksimum untuk operasi asinkron adalah 500 MB untuk file PDF.

Kode Status HTTP: 400

IdempotentParameterMismatchException

Parameter input `ClientRequestToken` digunakan kembali dengan suatu operasi, tapi setidaknya salah satu parameter input lainnya berbeda dari panggilan ke operasi sebelumnya.

Kode Status HTTP: 400

InternalServerError

Amazon Textract mengalami masalah layanan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

InvalidKMSKeyException

Menunjukkan bahwa Anda tidak memiliki izin mendekripsi dengan kunci KMS yang dimasukkan, atau kunci KMS dimasukkan secara tidak benar.

Kode Status HTTP: 400

InvalidParameterException

Parameter input melanggar batasan. Misalnya, dalam operasi sinkron, sebuah`InvalidParameterException`pengecualian terjadi ketika salah satu`S3Object`atau`Bytes`nilai-nilai yang disediakan dalam`Document`parameter permintaan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Kode Status HTTP: 400

InvalidS3ObjectException

Amazon Textract tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. untuk informasi selengkapnya,Mengonfigurasi Akses ke Amazon S3Untuk informasi pemecahan masalah, lihatPemecahan Masalah Amazon S3

Kode Status HTTP: 400

LimitExceededException

Batas layanan Amazon Textract terlampaui. Misalnya, jika Anda memulai terlalu banyak pekerjaan asinkron secara bersamaan, panggilan untuk memulai operasi (`StartDocumentTextDetection`, misalnya) menaikkan pengecualian LimitExceededException (kode status HTTP: 400) hingga jumlah tugas yang berjalan bersamaan di bawah batas layanan Amazon Textract TExceededException.

Kode Status HTTP: 400

ProvisionedThroughputExceededException

Jumlah permintaan melebihi batas throughput Anda. Jika Anda ingin meningkatkan batas ini, hubungi Amazon Textract.

Kode Status HTTP: 400

ThrottlingException

Amazon Textract untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda.

Kode Status HTTP: 500

UnsupportedDocumentException

Format dokumen input tidak didukung. Dokumen untuk operasi dapat dalam format PNG, JPEG, PDF, atau TIFF.

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasaAWSSDK, lihat yang berikut ini:

- AWSAntarmuka Baris Perintah
- AWSSDK for .NET
- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for JavaScript
- AWSSDK for PHP V3
- AWSSDK for Python
- AWSSDK for Ruby V3

Tipe Data

tipe data berikut didukung:

- 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

Digunakan untuk berisi informasi yang terdeteksi oleh operasi AnalyzeID.

Isi

Confidence

Skor kepercayaan dari teks yang terdeteksi.

Jenis: Apung

Rentang yang Valid: Nilai minimum 0. Nilai maksimum 100.

Diperlukan: Tidak

NormalizedValue

Hanya dikembalikan untuk tanggal, mengembalikan jenis nilai yang terdeteksi dan tanggal ditulis dengan cara yang lebih mudah dibaca mesin.

Tipe: Objek NormalizedValue

Diperlukan: Tidak

Text

Teks baik bidang normal atau nilai yang terkait dengan itu.

Jenis: Tali

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Block

SEBUAH`Block`mewakili item yang diakui dalam dokumen dalam kelompok piksel dekat satu sama lain. Informasi yang dikembalikan dalam`Block`tergantung pada jenis operasi. Dalam deteksi teks untuk dokumen (misalnyaDetectDocumentText), Anda mendapatkan informasi tentang kata-kata dan baris teks yang terdeteksi. Dalam analisis teks (misalnyaAnalyzeDocument), Anda juga bisa mendapatkan informasi tentang bidang, tabel, dan elemen seleksi yang terdeteksi dalam dokumen.

Susunan rangkaian`Block`objek dikembalikan oleh kedua operasi sinkron dan asinkron. Dalam operasi sinkron, sepertiDetectDocumentText, larik`Block`objek adalah seluruh rangkaian hasil. Dalam operasi asinkron, sepertiGetDocumentAnalysis, array dikembalikan lebih dari satu atau lebih tanggapan.

Untuk informasi selengkapnya, lihatCara Amazon Textract.

Isi

BlockType

tipe item teks yang dikenali. Dalam operasi untuk deteksi teks, jenis berikut dikembalikan:

- HALAMAN- Berisi daftar LINE`Block`objek yang terdeteksi pada halaman dokumen.
- KATA- Sebuah kata terdeteksi pada halaman dokumen. Sebuah kata adalah satu atau lebih karakter skrip Latin dasar ISO yang tidak dipisahkan oleh spasi.
- LINI- Sebuah string dari tab-delimited, kata-kata bersebelahan yang terdeteksi pada halaman dokumen.

Dalam operasi analisis teks, jenis berikut dikembalikan:

- HALAMAN- Berisi daftar anak`Block`objek yang terdeteksi pada halaman dokumen.
- KEY_VALUE_SET- Menyimpan KUNCI dan NILAI`Block`objek untuk teks terkait yang terdeteksi pada halaman dokumen. Gunakan`EntityType`bidang untuk menentukan apakah objek KEY_VALUE_SET adalah KEY`Block`objek atau NILAI`Block`objek.
- KATA- Sebuah kata yang terdeteksi pada halaman dokumen. Sebuah kata adalah satu atau lebih karakter skrip Latin dasar ISO yang tidak dipisahkan oleh spasi.
- LINI- Sebuah string dari tab-delimited, kata-kata bersebelahan yang terdeteksi pada halaman dokumen.

- MEJA- Sebuah tabel yang terdeteksi pada halaman dokumen. Sebuah tabel adalah informasi berbasis grid dengan dua atau lebih baris atau kolom, dengan rentang sel satu baris dan satu kolom masing-masing.
- SEL- Sebuah sel dalam tabel terdeteksi. Sel adalah induk dari blok yang berisi teks dalam sel.
- SELECTION_ELEMENT- Elemen seleksi seperti tombol opsi (tombol radio) atau kotak centang yang terdeteksi pada halaman dokumen. Gunakan nilai`SelectionStatus`untuk menentukan status elemen seleksi.

Jenis: Rangkaian

Nilai Valid: `KEY_VALUE_SET | PAGE | LINE | WORD | TABLE | CELL | SELECTION_ELEMENT`

Diperlukan: Tidak

ColumnIndex

Kolom di mana sel tabel muncul. Posisi kolom pertama adalah 1.`ColumnIndex`tidak dikembalikan oleh`DetectDocumentText`dan`GetDocumentTextDetection`.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

ColumnSpan

Jumlah kolom yang rentang sel tabel. Saat ini nilai ini selalu 1, bahkan jika jumlah kolom membentang lebih besar dari 1.`ColumnSpan`tidak dikembalikan oleh`DetectDocumentText`dan`GetDocumentTextDetection`.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

Confidence

Skor kepercayaan yang dimiliki Amazon Textract dalam keakuratan teks yang dikenali dan keakuratan titik geometri di sekitar teks yang dikenali.

Jenis: Apung

Rentang yang Valid: Nilai minimum 0. Nilai maksimum 100.

Diperlukan: Tidak

EntityTypes

Jenis entitas. Berikut ini dapat dikembalikan:

- KUNCI- Sebuah pengenal untuk bidang pada dokumen.
- NILAI- Teks bidang.

`EntityTypes`tidak dikembalikan oleh`DetectDocumentText`dan`GetDocumentTextDetection`.

Jenis: Array string

Nilai Valid: `KEY | VALUE`

Diperlukan: Tidak

Geometry

Lokasi teks yang dikenali pada gambar. Ini termasuk kotak batas yang sejajar dengan sumbu yang mengelilingi teks, dan poligon untuk informasi spasial yang lebih akurat.

Tipe: Objek Geometry

Diperlukan: Tidak

Id

Pengidentifikasi untuk teks yang dikenali. Pengidentifikasi hanya unik untuk satu operasi.

Jenis: String

Pola: `.*\S.*`

Diperlukan: Tidak

Page

Halaman di mana blok terdeteksi.`Page`dikembalikan oleh operasi asinkron. Nilai halaman yang lebih besar dari 1 hanya dikembalikan untuk dokumen multipage yang dalam format PDF atau TIFF. Gambar yang dipindai (JPEG/PNG), bahkan jika berisi beberapa halaman

dokumen, dianggap sebagai dokumen satu halaman. Nilai dari`Page`selalu 1. Operasi sinkron tidak kembali`Page`karena setiap dokumen input dianggap sebagai dokumen satu halaman.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

Relationships

Daftar blok anak dari blok saat ini. Misalnya, objek LINE memiliki blok anak untuk setiap blok WORD yang merupakan bagian dari baris teks. Tidak ada objek Relationship dalam daftar untuk relasi yang tidak ada, seperti ketika blok saat ini tidak memiliki blok anak. Ukuran daftar dapat menjadi sebagai berikut:

- 0 - Blok tidak memiliki blok anak.
- 1 - Blok memiliki blok anak.

Jenis: Array[Relationship](#)objek

Diperlukan: Tidak

RowIndex

Baris di mana sel tabel berada. Posisi baris pertama adalah 1.`RowIndex`tidak dikembalikan oleh`DetectDocumentText`dan`GetDocumentTextDetection`.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

RowSpan

Jumlah baris yang mencakup sel tabel. Saat ini nilai ini selalu 1, bahkan jika jumlah baris membentang lebih besar dari 1.`RowSpan`tidak dikembalikan oleh`DetectDocumentText`dan`GetDocumentTextDetection`.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

SelectionStatus

Status pemilihan elemen seleksi, seperti tombol opsi atau kotak centang.

Jenis: Rangkaian

Nilai Valid: `SELECTED | NOT_SELECTED`

Diperlukan: Tidak

Text

Kata atau baris teks yang dikenali oleh Amazon Textract.

Jenis: Tali

Diperlukan: Tidak

TextType

Jenis teks yang Amazon Textract telah terdeteksi. Dapat memeriksa teks tulisan tangan dan teks cetak.

Jenis: Rangkaian

Nilai Valid: `HANDWRITING | PRINTED`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasaAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

BoundingBox

Kotak pembatas di sekitar halaman terdeteksi, teks, pasangan kunci-nilai, tabel, sel tabel, atau elemen seleksi pada halaman dokumen. Parameter`left`(koordinat x) dan`top`(koordinat y) adalah koordinat yang mewakili sisi atas dan kiri kotak batas. Perhatikan bahwa sudut kiri atas citra merupakan asal (0,0).

Parameter`top`dan`left`nilai yang dikembalikan adalah rasio dari ukuran halaman dokumen secara keseluruhan. Misalnya, jika citra input berukuran 700 x 200 piksel, dan koordinat kiri atas kotak batas adalah 350 x 50 piksel, maka API mengembalikan `a``left`nilai 0,5 (350/700) dan`top`nilai 0,25 (50/200).

Parameter`width`dan`height`nilai mewakili dimensi kotak pembatas sebagai rasio dimensi halaman dokumen secara keseluruhan. Misalnya, jika ukuran halaman dokumen berukuran 700 x 200 piksel, dan lebar kotak batas adalah 70 piksel, maka lebar yang dikembalikan adalah 0,1.

Isi

Height

Tinggi kotak batas sebagai rasio tinggi halaman dokumen secara keseluruhan.

Jenis: Apung

Diperlukan: Tidak

Left

Koordinat kiri kotak pembatas sebagai rasio lebar halaman dokumen secara keseluruhan.

Jenis: Apung

Diperlukan: Tidak

Top

Koordinat kiri kotak pembatas sebagai rasio tinggi halaman dokumen secara keseluruhan.

Jenis: Apung

Diperlukan: Tidak

Width

Lebar kotak batas sebagai rasio lebar halaman dokumen secara keseluruhan.

Jenis: Apung

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Document

Dokumen input, baik sebagai byte atau sebagai objek S3.

Anda meneruskan byte citra ke operasi API Amazon Textract dengan menggunakan`Bytes`properti. Misalnya, Anda akan menggunakan`Bytes`properti untuk lulus dokumen yang dimuat dari sistem file lokal. Gambar byte dilewatkan dengan menggunakan`Bytes`properti harus dikodekan base64. Kode Anda mungkin tidak perlu mengodekan byte file jika Anda menggunakan AWS SDK untuk memanggil operasi API Amazon Textract.

Anda meneruskan citra yang disimpan dalam bucket S3 ke operasi API Amazon Textract dengan menggunakan`S3Object`properti. Dokumen yang disimpan dalam bucket S3 tidak perlu dikodekan ke base64.

Wilayah AWS untuk bucket S3 yang berisi objek S3 harus sesuai dengan Wilayah AWS yang Anda gunakan untuk operasi Amazon Textract.

Jika Anda menggunakan AWS CLI untuk memanggil operasi Amazon Textract, meneruskan byte citra menggunakan properti Bytes tidak didukung. Anda harus mengunggah dokumen terlebih dahulu ke bucket Amazon S3, lalu memanggil operasi menggunakan properti S3Object.

Agar Amazon Textract Texact memproses objek S3, pengguna harus memiliki izin untuk mengakses objek S3.

Isi

Bytes

Sebuah gumpalan byte dokumen base64 dikodekan. Ukuran maksimum dokumen yang disediakan dalam gumpalan byte adalah 5 MB. Dokumen byte harus dalam format PNG atau JPEG.

Jika Anda menggunakan AWS SDK untuk memanggil Amazon Textract, Anda mungkin tidak perlu byte citra yang dikodekan base64 yang diteruskan menggunakan`Bytes`lapangan.

Jenis: Objek data biner berkode Base64

Batasan Panjang: Panjang minimum 1. Panjang maksimum 10485760.

Diperlukan: Tidak

S3Object

Mengidentifikasi objek S3 sebagai sumber dokumen. Ukuran maksimum dokumen yang disimpan dalam bucket S3 adalah 5 MB.

Tipe: Objek S3Object

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khususAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

DocumentLocation

Bucket Amazon S3 yang berisi dokumen yang akan diproses. Ini digunakan oleh operasi asinkron seperti[StartDocumentTextDetection](#).

Dokumen input dapat berupa file gambar dalam format JPEG atau PNG. Ini juga bisa menjadi file dalam format PDF.

Isi

S3Object

Bucket Amazon S3 yang berisi dokumen input.

Tipe: Objek [S3Object](#)

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa tertentuAWSSDK, lihat yang berikut ini:

- [AWSSDK for C++](#)
- [AWSSDK for Go](#)
- [AWSSDK for Java V2](#)
- [AWSSDK for Ruby V3](#)

DocumentMetadata

Informasi tentang dokumen masukan.

Isi

Pages

Jumlah halaman yang terdeteksi dalam dokumen.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

ExpenseDetection

Objek yang digunakan untuk menyimpan informasi tentang Nilai atau Label yang terdeteksi oleh Amazon Textract.

Isi

Confidence

Keyakinan dalam deteksi, sebagai persentase

Jenis: Apung

Rentang yang Valid: Nilai minimum 0. Nilai maksimum 100.

Diperlukan: Tidak

Geometry

Informasi tentang di mana item berikut terletak pada halaman dokumen: halaman terdeteksi, teks, pasangan kunci-nilai, tabel, sel tabel, dan elemen seleksi.

Tipe: Objek Geometry

Diperlukan: Tidak

Text

Kata atau baris teks yang dikenali oleh Amazon Textract

Jenis: Tali

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khususAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2

- AWSSDK for Ruby V3

ExpenseDocument

Struktur memegang semua informasi yang dikembalikan oleh AnalyzeExpense

Isi

ExpenseIndex

Menunjukkan faktur atau tanda terima mana dalam dokumen informasi berasal dari. Dokumen pertama akan 1, 2 kedua, dan seterusnya.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

LineItemGroups

Informasi terdeteksi pada setiap tabel dokumen, dipisahkan menjadi`LineItems`.

Jenis: ArrayLineItemGroupobjek

Diperlukan: Tidak

SummaryFields

Informasi apa pun yang ditemukan di luar tabel oleh Amazon Textract.

Jenis: ArrayExpenseFieldobjek

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

ExpenseField

Rincian informasi yang terdeteksi, dipisahkan ke dalam jenis katagori, LabelDetection, dan valueDetection

Isi

LabelDetection

Label secara eksplisit dinyatakan dari elemen terdeteksi.

Tipe: Objek ExpenseDetection

Diperlukan: Tidak

PageNumber

Nomor halaman nilai terdeteksi pada.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

Type

Label tersirat dari elemen terdeteksi. Hadir bersama labelDetection untuk elemen eksplisit.

Tipe: Objek ExpenseType

Diperlukan: Tidak

ValueDetection

Nilai elemen terdeteksi. Hadir dalam elemen eksplisit dan implisit.

Tipe: Objek ExpenseDetection

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

ExpenseType

Objek yang digunakan untuk menyimpan informasi tentang Tipe yang terdeteksi oleh Amazon Textract.

Isi

Confidence

Kepercayaan akurasi, sebagai persentase.

Jenis: Apung

Rentang yang Valid: Nilai minimum 0. Nilai maksimum 100.

Diperlukan: Tidak

Text

Kata atau baris teks yang terdeteksi oleh Amazon Textract.

Jenis: Tali

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Geometry

Informasi tentang di mana item berikut terletak pada halaman dokumen: halaman terdeteksi, teks, pasangan kunci-nilai, tabel, sel tabel, dan elemen seleksi.

Isi

BoundingBox

Sebuah representasi kasar sumbu selaras lokasi dari item yang diakui pada halaman dokumen.

Tipe: Objek BoundingBox

Diperlukan: Tidak

Polygon

Dalam kotak batas, detail poligon di sekitar item yang diakui.

Jenis: ArrayPointobjek

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

HumanLoopActivationOutput

Menunjukkan hasil yang menunjukkan manusia dalam evaluasi loop. Jika tidak ada HumanLoopArn, input tidak memicu peninjauan manual.

Isi

HumanLoopActivationConditionsEvaluationResults

Menunjukkan hasil evaluasi syarat, termasuk kondisi yang mengaktivasi peninjauan manual.

Jenis: String

Batasan Panjang: Panjang maksimum 10240.

Diperlukan: Tidak

HumanLoopActivationReasons

Menunjukkan apakah dan mengapa peninjauan manual dibutuhkan.

Jenis: Array string

Anggota Array: Jumlah minimum 1 item.

Diperlukan: Tidak

HumanLoopArn

Amazon Resource Name (ARN) HumanLoop berhasil dibuat.

Jenis: String

Batasan Panjang: Panjang maksimum 256.

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++

- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

HumanLoopConfig

Mengatur alur kerja peninjauan manusiawi tempat dokumen akan dikirim jika salah satu syarat terpenuhi. Anda juga dapat mengatur atribut tertentu citra sebelum meninjau.

Isi

DataAttributes

Mengatur atribut data input.

Tipe: Objek HumanLoopDataAttributes

Diperlukan: Tidak

FlowDefinitionArn

Amazon Resource Name (ARN) definisi aliran.

Jenis: String

Batasan: Panjang maksimum 256.

Diperlukan: Ya

HumanLoopName

Nama alur kerja manusia yang digunakan untuk citra ini. Nama ini harus tetap unik dalam suatu wilayah.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum 63.

Pola: `^[a-z0-9](-*[a-z0-9])*`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++

- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

HumanLoopDataAttributes

Memungkinkan Anda mengatur atribut citra. Saat ini, Anda dapat menyatakan citra terbebas dari informasi pengenal pribadi dan konten dewasa.

Isi

ContentClassifiers

Menetapkan apakah citra input terbebas dari informasi pribadi atau konten dewasa.

Jenis: Array string

Anggota Array: Jumlah maksimum 256 item.

Nilai Valid: `FreeOfPersonallyIdentifiableInformation | FreeOfAdultContent`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

IdentityDocument

Struktur yang mencantumkan setiap dokumen yang diproses dalam operasi AnalyzeID.

Isi

DocumentIndex

Menunjukkan penempatan dokumen dalam daftar IdentityDocument. Dokumen pertama ditandai 1, 2 kedua dan seterusnya.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

IdentityDocumentFields

Struktur yang digunakan untuk merekam informasi yang diekstrak dari dokumen identitas. Berisi baik bidang dinormalisasi dan nilai teks diekstraksi.

Jenis: ArrayIdentityDocumentFieldobjek

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

IdentityDocumentField

Struktur yang mengandung kedua jenis dinormalisasi informasi diekstraksi dan teks yang terkait dengan itu. Ini diekstraksi sebagai Jenis dan Nilai masing-masing.

Isi

Type

Digunakan untuk berisi informasi yang terdeteksi oleh operasi AnalyzeID.

Tipe: Objek AnalyzeIDDetections

Diperlukan: Tidak

ValueDetection

Digunakan untuk berisi informasi yang terdeteksi oleh operasi AnalyzeID.

Tipe: Objek AnalyzeIDDetections

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu dari bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

LineItemFields

Sebuah struktur yang menyimpan informasi tentang baris yang berbeda ditemukan dalam tabel dokumen.

Isi

LineItemExpenseFields

ExpenseFields digunakan untuk menampilkan informasi dari garis terdeteksi di atas meja.

Jenis: ArrayExpenseFieldobjek

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

LineItemGroup

Pengelompokan tabel yang berisi lineItems, dengan setiap tabel diidentifikasi oleh tabel`LineItemGroupIndex`.

Isi

LineItemGroupIndex

Jumlah yang digunakan untuk mengidentifikasi tabel tertentu dalam dokumen. Tabel pertama yang ditemui akan memiliki LineItemGroupIndex 1, 2 kedua, dll.

Jenis: Bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

LineItems

Rincian informasi pada baris tertentu dari tabel.

Jenis: ArrayLineItemFieldsobjek

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

NormalizedValue

Berisi informasi yang berkaitan dengan tanggal dalam dokumen, termasuk jenis nilai, dan nilai.

Isi

Value

Nilai tanggal, ditulis sebagai tahun-bulan-hari:menit: kedua.

Jenis: Tali

Diperlukan: Tidak

ValueType

Jenis dinormalisasi nilai terdeteksi. Dalam hal ini adalah, DATE.

Jenis: Rangkaian

Nilai Valid: `DATE`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu dari spesifik bahasaAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

NotificationChannel

Topik Amazon Simple Notification Service (Amazon SNS) tempat Amazon Textract menerbitkan status penyelesaian operasi dokumen asinkron, seperti StartDocumentTextDetection.

Isi

RoleArn

Amazon Resource Name (ARN) dari IAM role yang memberikan Amazon Textract izin penerbitan untuk topik Amazon SNS.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Diperlukan: Ya

SNSTopicArn

Topik Amazon SNS tempat Amazon Textract memposting status penyelesaian.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 1024.

Pola: `(^arn:([a-z\d-]+):sns:[a-zA-Z\d-]{1,20}:\w{12}:.+$)`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khususAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

OutputConfig

Menetapkan apakah output Anda akan pergi ke ember pengguna dibuat. Digunakan untuk mengatur nama bucket, dan awalan pada file output.

`OutputConfig`adalah parameter opsional yang memungkinkan Anda menyesuaikan di mana output Anda akan ditempatkan. Secara default, Amazon Textract akan menyimpan hasil secara internal dan hanya dapat diakses oleh operasi Get API. Dengan OutputConfig diaktifkan, Anda dapat mengatur nama bucket output akan dikirim ke dan awalan file hasil di mana Anda dapat men-download hasil Anda. Selain itu, Anda dapat mengatur`KMSKeyID`parameter kunci utama pelanggan (CMK) untuk mengenkripsi output Anda. Tanpa parameter ini, tetapkan Amazon Textract akan mengenkripsi sisi server menggunakan CMK terkelola AWS untuk Amazon S3.

Dekripsi Konten Pelanggan diperlukan untuk memproses dokumen oleh Amazon Textract. Jika akun Anda dipilih berdasarkan kebijakan penyisihan layanan AI, maka semua Konten Pelanggan yang tidak terenkripsi akan segera dihapus secara permanen setelah Konten Pelanggan diproses oleh layanan. Tidak ada salinan output yang disimpan oleh Amazon Textract. Untuk informasi tentang cara memilih keluar, lihat[Kebijakan berhenti berlangganan layanan AI.]

Untuk informasi selengkapnya tentang privasi data, lihat[FAQ Privasi Data].

Isi

S3Bucket

Nama bucket output Anda akan pergi ke.

Jenis: String

Batasan: Panjang minimum 3. Panjang maksimum 255.

Pola: `[0-9A-Za-z\.\-_]*`

Diperlukan: Ya

S3Prefix

Awalan kunci objek bahwa output akan disimpan ke. Bila tidak diaktifkan, awalan akan menjadi “textract_output”.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum 1024.

Pola: `.*\S.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu spesifik bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Point

Koordinat X dan Y titik pada halaman dokumen. Nilai X dan Y yang dikembalikan adalah rasio ukuran halaman dokumen secara keseluruhan. Misalnya, jika dokumen input 700 x 200 dan operasi mengembalikan X = 0,5 dan Y = 0,25, maka titiknya adalah pada koordinat (350,50) pixel pada halaman dokumen.

Susunan rangkaian`Point`objek,`Polygon`, dikembalikan sebagai bagian dariGeometryobjek yang dikembalikan dalamBlockobjek. SEBUAH`Polygon`objek merupakan poligon halus sekitar teks terdeteksi, pasangan kunci-nilai, tabel, sel tabel, atau elemen seleksi.

Isi

X

Nilai koordinat X untuk titik pada `Polygon`.

Jenis: Apung

Diperlukan: Tidak

Y

Nilai koordinat Y untuk titik pada `Polygon`.

Jenis: Apung

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu spesifik bahasaAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Relationship

Informasi tentang bagaimana blok terkait satu sama lain. SEBUAH`Block`objek berisi 0 atau lebih`Relation`objek dalam daftar,`Relationships`. Untuk informasi selengkapnya, lihat Block.

Parameter`Type`elemen menyediakan jenis hubungan untuk semua blok di`IDs`array.

Isi

Ids

Array ID untuk blok terkait. Anda bisa mendapatkan jenis hubungan dari`Type`elemen.

Jenis: String

Pola: `.*\S.*`

Diperlukan: Tidak

Type

Jenis hubungan yang blok dalam array ID memiliki dengan blok saat ini. Hubungan bisa`VALUE`atau`CHILD`. Hubungan tipe VALUE adalah daftar yang berisi ID blok VALUE yang terkait dengan KEY pasangan kunci-nilai. Hubungan tipe ANAK adalah daftar ID yang mengidentifikasi blok WORD dalam kasus baris blok sel dalam kasus Tabel, dan blok WORD dalam kasus Seleksi Elemen.

Jenis: Rangkaian

Nilai Valid: `VALUE | CHILD | COMPLEX_FEATURES`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khusus bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2

- AWSSDK for Ruby V3

S3Object

Nama bucket S3 dan nama file yang mengidentifikasi dokumen tersebut.

Wilayah AWS untuk bucket S3 yang berisi dokumen harus sesuai dengan Wilayah yang Anda gunakan untuk operasi Amazon Textract Texact.

Agar Amazon Textract Texact memproses file dalam bucket S3, pengguna harus memiliki izin untuk mengakses bucket S3 dan file.

Isi

Bucket

Nama bucket S3. Perhatikan bahwa karakter # tidak valid dalam nama file.

Jenis: String

Batasan: Panjang minimum 3. Panjang maksimum 255.

Pola: `[0-9A-Za-z\.\-_]*`

Diperlukan: Tidak

Name

Nama file dari dokumen input. Operasi sinkron dapat menggunakan file gambar yang dalam format JPEG atau PNG. Operasi asinkron juga mendukung file format PDF dan TIFF.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum 1024.

Pola: `.*\S.*`

Diperlukan: Tidak

Version

Jika bucket mendukung versioning, Anda dapat menentukan versi objek.

Jenis: String

Batasan: Panjang minimum 1. Panjang maksimum 1024.

Pola: `.*\S.*`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasa khususAWSSDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Warning

Peringatan tentang masalah yang terjadi selama analisis teks asinkron (StartDocumentAnalysis) atau deteksi teks dokumen asinkron (StartDocumentTextDetection).

Isi

ErrorCode

Kode kesalahan untuk peringatan.

Jenis: Tali

Diperlukan: Tidak

Pages

Daftar halaman yang peringatan berlaku untuk.

Jenis: Array bilangan bulat

Rentang yang Valid: Nilai minimum 0.

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu bahasaAWSSDK, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Batas Keras di Amazon Textract

Berikut ini adalah daftar batasan keras di Amazon Textract, yang tidak dapat diubah. Untuk informasi tentang batasan di lokasi dan batas Anda dapat mengubah lihatTitik Akhir dan Kuota Amazon Textract. Untuk informasi tentang batas yang dapat Anda ubah, lihatAWSService Limits. Untuk mengubah batas, lihatBuat kasus.

Amazon Textract

Kuota	Deskripsi
Format File	Operasi mendukung file JPEG, PNG, PDF, dan TIFF. (Gambar JPEG 2000-dikodekan dalam PDF didukung)..
Ukuran File dan Batas Hitungan Halaman	Untuk operasi sinkron, file JPEG, PNG, PDF, dan TIFF memiliki batas ukuran 10MB. File PDF dan TIFF juga memiliki batas 1 halaman. Untuk operasi asinkron, file JPEG dan PNG memiliki batas ukuran 10MB. File PDF dan TIFF memiliki batas 500MB. File PDF dan TIFF memiliki batas 3.000 halaman.
Batas Spesifik PDF	Tinggi dan lebar maksimum adalah 40 inci dan 2880 poin. PDF tidak dapat dilindungi kata sandi. PDF dapat berisi JPEG 2000 diformat gambar.
Rotasi Dokumen dan Ukuran Gambar	Amazon Textract mendukung semua rotasi dokumen dalam pesawat, misalnya rotasi dalam pesawat 45 derajat. Amazon Textract mendukung gambar dengan resolusi kurang dari atau sama dengan 10000 piksel di semua sisi.
Penyelarasan Teks	Teks dapat teks sejajar horizontal dalam dokumen. Amazon Textract tidak mendukung penyelarasan teks vertikal dalam dokumen.
BAHASA	Amazon Textract Texact mendukung deteksi teks bahasa Inggris, Prancis, Jerman, Italia, Portugis, dan Spanyol. Amazon

Kuota	Deskripsi
	Textract tidak akan mengembalikan bahasa yang terdeteksi dalam outputnya.
Ukuran karakter	Tinggi minimum untuk teks yang akan dideteksi adalah 15 piksel. Pada 150 DPI, ini akan sama dengan font 8 titik.
Tipe karakter	Amazon Textract mendukung pengenalan karakter tulisan tangan dan cetak.
Karakter	Amazon Textract Texact mendeteksi karakter berikut: • a-z • A-Z • 0-9 • ä ö Ö ü ü ç ç é É Â Â î ô ô dan à à è è è ù ë ï ï ü ü á á é é í í ó ó ú ú ü ñ ì ì ò ò ã • ! “# $% '& () * +, -./:; =? @ [\] ^ _ `{|} ~ > <° € £ ¥ # ß ß ¿¡€ £ ¥ # ø ø ©™ § ¹ ² ³ '
Batas Spesifik AnalyzeID	AnalyzeID hanya mendukung Paspor AS, dan Lisensi Pengemudi AS.

Riwayat Dokumen untuk Amazon Textract

Tabel berikut menjelaskan perubahan penting dalam setiap rilisPanduan Developer Amazon Textract. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

- Pembaruan dokumentasi terbaru: 29 Mei 2019

pembaruan-riwayat-perubahan	pembaruan-riwayat-deskripsi	pembaruan-riwayat-tanggal
Mengintegrasikan contoh kode dariAWSDokumen SDK Kode Contoh GitHub repo	Panduan Amazon Textract sekarang berisi contoh kode tambahan. Berganti nama contoh sebelumnya bagian untuk Tutorial.	30 Januari 2022
AnalyzeExpense Ditambahkan	Amazon Textract sekarang mendukung analisis faktur dan dokumen penerimaan menggunakan AnalyzeEx pense API. Fitur ini hanya tersedia di wilayah Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Kanada (Central), Eropa (Frankfur t), Eropa (Irlandia), Eropa (London), US East (N.Virgin ia), US East (Ohio), US West (N.California), dan US West (Oregon).	26 Juli 2021
Augmented AI	Amazon Textract sekarang mendukung Amazon Augmented AI untuk menerapkan tinjauan manusia.	3 Desember 2019

Layanan dan panduan baru	Amazon Textract sekarang tersedia untuk penggunaan umum.	29 Mei 2019
Support untuk elemen seleksi	Amazon Textract sekarang dapat mendeteksi elemen pilihan (tombol radio dan kotak centang).	24 April 2019
Rilis Amazon Textract	Ini adalah rilis pertama dari dokumentasi untuk Amazon Textract.	28 November, 2018

Daftar istilah AWS

Untuk terminologi AWS terbaru, lihat AWS daftar istilah di AWS Referensi Umum.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.