aws

開發人員指南

Amazon Comprehend

Amazon Comprehend: 開發人員指南

Table of Contents

什麼是 Amazon Comprehend？

Amazon Comprehend 使用自然語言處理 (NLP) 來擷取有關文件內容的見解。它可透過識別文件中的實體、關鍵片語、語言、情感和其他常見元素，以此形成洞見。使用 Amazon Comprehend 根據了解文件的結構來建立新產品。例如，使用 Amazon Comprehend，您可以搜尋社交網路摘要以取得產品的提及，或掃描整個文件儲存庫中的關鍵片語。

您可以使用亞馬遜主控台或使用 Amazon Comprehend API 存取 Amazon Comprehend 文件分析功能。您可以針對小型工作負載執行即時分析，也可以針對大型文件集啟動非同步分析工作。您可以使用 Amazon Comprehend 提供的預先訓練模型，也可以訓練自己的自訂模型以進行分類和實體辨識。

Amazon Comprehend 可能會儲存您的內容，以持續改善其預先訓練模型的品質。請參閱亞馬遜常見問答集以進一步了解。

Amazon Comprehend 的所有功能都接受 UTF-8 文本文檔作為輸入。此外，自訂分類和自訂實體辨識可接受影像檔案、PDF 檔案和 Word 檔案做為輸入。

Amazon Comprehend 可以檢查和分析各種語言的文件，具體取決於特定功能。如需詳細資訊，請參閱 Amazon Comprehend 支持的語言。Amazon Comprehend 的主要语言功能可以檢查文件並判斷更廣泛的語言選擇的主要語言。

主題

- Amazon Comprehend 洞察
- Amazon Comprehend 定制
- 飞轮
- 文件叢集 (主題建模)
- 範例
- 優勢
- Amazon Comprehend 定價
- 您是亞馬遜的首次使用者嗎？

Amazon Comprehend 洞察

Amazon Comprehend 使用預先訓練的模型來檢查和分析文件或一組文件，以收集有關該文件的見解。此模型會在大型文字上持續訓練，因此您不需要提供訓練資料。

亞馬遜分析了以下類型的見解：

- 實體 — 文件中包含之人員、地點、項目和位置名稱的參照。
- 關鍵片語 — 出現在文件中的片語。例如，關於籃球比賽的文件可能會傳回球隊名稱、場地名稱和最終得分。
- 個人身份信息（PII）— 可以識別個人的個人數據，例如地址，銀行帳戶號碼或電話號碼。
- 語言 — 文件的主要語言。
- 情緒 — 文件的主要情緒，可以是正面、中性、負面或混合。
- 目標情緒 — 與文件中特定實體相關聯的情緒。每個實體出現的情緒可以是正面、負面、中性或混合。
- 語法 — 文件中每個單字的語音部分。

如需詳細資訊，請參閱 深入分析。

Amazon Comprehend 定制

您可以根據自己的特定需求自訂 Amazon Comprehend，而不必具備建置機器學習型 NLP 解決方案所需的技能組合。Amazon Comprehend 自訂可使用自動機器學習或 AutoML，使用您現有的資料代表您建立自訂的 NLP 模型。

自訂分類 — 建立自訂分類模型 (分類器)，將文件組織成您自己的類別。

自訂實體辨識 — 建立自訂實體辨識模型 (辨識器)，以分析特定詞彙和以名詞為基礎的片語的文字。

如需詳細資訊，請參閱 Amazon Comprehend 定制。

飞轮

使用飛輪簡化隨著時間的推移訓練和管理自訂模型版本的程序。飛輪有助於協調與訓練和評估新模型版本相關的任務。Flywheels 支援用於自訂分類和自訂實體辨識的純文字自訂模型。如需詳細資訊，請參閱 飞轮。

文件叢集 (主題建模)

您也可以使用 Amazon Comprehend 來檢查文件的語料庫，以根據其中的類似關鍵字來組織文件。文件叢集 (主題模型) 有助於將大型文件語料庫組織成以字詞頻率為基礎的類似主題或叢集。如需詳細資訊，請參閱 主題建模。

範例

下列範例顯示如何在應用程式中使用 Amazon Comprehend 操作。

Example 1：尋找有關主旨的文件

使用亞馬遜主題建模來尋找有關特定主題的文件。掃描一組文件以決定討論的主題，並尋找與每個主題相關聯的文件。您可以指定 Amazon Comprehend 應該從文件集傳回的主題數目。

Example 2：了解客戶對您的產品的看法

如果您的公司發佈目錄，請讓 Amazon Comprehend 告訴您客戶對您產品的看法。將每個客戶評論發送到`DetectSentiment`操作中，它會告訴您客戶對產品的感覺是否正面，負面，中立或混合。

Example 3：發現對您的客戶至關重要的事情

使用 Amazon Comprehend 主題模型來探索客戶在論壇和留言板上討論的主題，然後使用實體偵測來判斷與主題相關聯的人員、地點和事物。使用情緒分析來判斷客戶對某個主題的看法。

優勢

使用 Amazon Comprehend 的好處包括：

- 將強大的自然語言處理功能整合到您的應用程式中 — Amazon Comprehend 透過簡單的 API 提供強大且精確的自然語言處理功能，從而消除在應用程式中建立文字分析功能的複雜性。您不需要文字分析專業知識，就能利用 Amazon Comprehend 產生的深入解析。
- 以深度學習為基礎的自然語言處理 — Amazon Comprehend 使用深度學習技術精確地分析文字。我們的模型經過不斷訓練，涵蓋多個領域的新資料，以提高準確性。
- 可擴展的自然語言處理 — Amazon Comprehend 可讓您分析數百萬份文件，以便探索其中包含的深入解析。
- 與其他 AWS 服務整合 — Amazon Comprehend 的設計可與 Amazon S3 和等其他 AWS 服務無縫協作。 AWS KMS AWS Lambda將您的文件存放在 Amazon S3 中，或使用 Firehose 分析即時資料。Support AWS Identity and Access Management (IAM) 可讓您輕鬆安全地控制對 Amazon Comprehend 操作的存取。您可以使用 IAM 建立和管理使用者和群組，將適當的存取權授予開發人員和使用者。
- 輸出結果和磁碟區資料的加密 — Amazon S3 已經讓您加密輸入文件，而 Amazon Comprehend 將此文件擴展得更遠。透過使用自己的 KMS 金鑰，您可以加密任務的輸出結果，以及連接至處理分析工作之運算執行個體的儲存磁碟區上的資料。結果顯著增強了安全性。

- 低成本 — 有了 Amazon Comprehend，就沒有最低費用或前期承諾。您需要支付您分析的文件以及訓練的自訂模型。

Amazon Comprehend 定價

使用 Amazon Comprehend，您只需為使用的資源付費。如果您是新 AWS 客戶，您可以免費開始使用 Amazon Comprehend。如需詳細資訊，請參閱AWS 免費用量方案。

執行即時或非同步分析工作需支付使用費。您需要付費訓練自訂模型，而且需要支付自訂模型管理費用。對於使用自訂模型的即時請求，您需要從啟動端點開始支付端點費用，直到刪除端點為止。使用飛輪不收取額外費用。但是，當您執行飛輪反覆運算時，會產生訓練新模型版本和儲存模型資料的標準費用。

如需費率和其他詳細資訊，請參 Amazon Comprehend 定價。

您是亞馬遜的首次使用者嗎？

如果您是 Amazon Comprehend 的第一次使用者，我們建議您依序閱讀以下各節：

1. 運作方式— 本節介紹 Amazon Comprehend 概念。
2. 設定— 在本節中，您可以創建一個帳戶並設置 AWS CLI.
3. 開始使用 Amazon Comprehend— 在本節中，您執行 Amazon Comprehend 分析任務。
4. 教學：使用亞馬遜分析客戶評論中的見解— 在本節中，您會執行情緒和實體分析，並視覺化結果。
5. Amazon Comprehend API 參考 — Amazon Comprehend 操作的參考文檔。

AWS 提供下列資源，以了解有關亞馬遜服務的資源：

- M AWS achine Learning 部落格包含有關 Amazon Comprehend 的實用文章。
- Amazon Comprehend 資源提供有關 Amazon Comprehend 的有用視頻和教程。

運作方式

Amazon Comprehend 使用預先訓練的模型來收集有關文件或一組文件的見解。此模型會在大型文字上持續進行訓練，因此您不需要提供訓練資料。

您可以使用 Amazon Comprehend 建立自己的自訂模型，以進行自訂分類和自訂實體辨識。您可以使用飞轮來協助管理自訂模型。

亞馬遜提供使用內建模型的主題建模。主題建模會檢查文件的語料庫，並根據其中的類似關鍵字來組織文件。

Amazon Comprehend 提供同步和非同步文件處理模式。使用同步模式處理一份文件或最多 25 份文件的批次。使用非同步工作來處理大量文件。

Amazon Comprehend 可與 AWS Key Management Service (AWS KMS) 搭配使用，為您的資料提供增強的加密功能。如需詳細資訊，請參閱 Amazon Comprehend 中的 KMS 加密。

重要概念

- 深入分析
- Amazon Comprehend 定制
- 主題建模
- 文件處理模式

深入分析

Amazon Comprehend 可以分析一份文件或一組文件，以收集有關文件的見解。Amazon Comprehend 針對文件開發的一些見解包括：

- 實體— Amazon Comprehend 會傳回文件中識別的實體清單，例如人員、地點和位置。
- 事件— Amazon Comprehend 可偵測特定類型的事件和相關詳細資料。
- 關鍵短語— Amazon Comprehend 提取出現在文檔中的關鍵短語。例如，關於籃球比賽的文件可能會傳回球隊名稱、場地名稱和最終得分。
- 個人身分識別資訊 (PII) — Amazon Comprehend 會分析文件以偵測可識別個人身分的個人資料，例如地址、銀行帳號或電話號碼。
- 主要语言— Amazon Comprehend 識別文檔中的主要語言。Amazon Comprehend 可以識別 100 種語言。

- 情緒 — Amazon Comprehend 決了文件的主要情緒。情緒可以是正面、中性、負面或混合。
- 目標情緒 — Amazon Comprehend 會決定文件中提及之特定實體的情緒。每次提及的情緒可以是正面、中性、負面或混合。
- 語法分析— Amazon Comprehend 會剖析文件中的每個單字，並決定單字的語音部分。例如，在句子「今天在西雅圖下雨」中，「它」被識別為代詞，「下雨」被識別為動詞，「西雅圖」被識別為專有名詞。

實體

實體是對真實世界物件 (例如人員、地點和商業項目) 的唯一名稱的文字參考，以及對測量 (例如日期和數量) 的精確參考。

例如，在文本「約翰搬到 1313 知更鳥巷 2012,」「約翰」可能被認為是PERSON,「1313 知更鳥車道」可能被認為是LOCATION, 和「2012」可能被認為是. DATE

每個實體也有一個分數，指出 Amazon Comprehend 正確偵測到實體類型的信賴程度。您可以篩選出分數較低的實體，以降低使用錯誤偵測的風險。

下表列出了實體類型。

Type	描述
商業項目	品牌產品
DATE	完整日期（例如，2017 年 11 月 25 日）、日（星期二）、月份（五月）或時間（上午 8 時 30 分）
EVENT	活動，例如節日，音樂會，選舉等
LOCATION	特定位置，例如國家，城市，湖泊，建築物等。
組織	大型組織，例如政府，公司，宗教，運動隊等。
OTHER	不適合任何其他實體類別的實體
人	個人，人群，暱稱，虛構人物
數量	量化金額，例如貨幣、百分比、數字、位元組等
標題	任何創作或創作作品（例如電影，書籍，歌曲等）的正式名稱。

偵測實體操作可以使用 Amazon Comprehend 支援的任何主要語言來執行。這僅包括預先定義的（非自訂）實體偵測。所有文件必須使用相同的語言。

您可以使用下列任何 API 作業來偵測文件或一組文件中的實體。

- DetectEntities
- BatchDetectEntities
- StartEntitiesDetectionJob

作業會傳回 API 實體物件清單，文件中的每個實體各一個。此BatchDetectEntities作業會傳回Entity物件清單，批次中每個文件都有一個清單。StartEntitiesDetectionJob作業會啟動非同步工作，該工作會產生一個檔案，其中包含工作中每個文件的Entity物件清單。

下列範例是來自DetectEntities作業的回應。

```
{
    "Entities": [
        {
            "Text": "today",
            "Score": 0.97,
            "Type": "DATE",
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
            "Score": 0.95,
            "Type": "LOCATION",
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ],
    "LanguageCode": "en"
}
```

事件

使用事件偵測來分析文字文件，找出特定類型的事件及其相關實體。Amazon Comprehend 使用非同步分析任務，支援跨大型文件集合的事件偵測。如需事件的詳細資訊 (包括事件分析工作範例)，請參閱宣布推出 Amazon Comprehend 事件

實體

Amazon Comprehend 會從輸入文字中擷取與偵測到的事件相關的實體清單。實體可以是真實世界的物件，例如人物、地點或位置；實體也可以是概念，例如度量、日期或數量。實體的每個出現次數都會以提及來識別，該提及是對輸入文字中實體的文字參照。對於每個唯一實體，所有提及都會分組到一個清單中。此清單提供輸入文字中實體出現的每個位置的詳細資訊。Amazon Comprehend 只會偵測與支援的事件類型相關聯的實體。

與支援的事件類型相關聯的每個實體都會傳回下列相關詳細資訊：

- 提及：輸入文字中每次出現相同實體的詳細資料。
  - BeginOffset: 輸入文字中的字元偏移量，顯示提及的開始位置 (第一個字元位於 0 的位置)。
  - EndOffset: 輸入文字中的字元偏移量，顯示提及結束位置。
  - 分數：Amazon Comprehend 對實體類型精確度的信心程度。
  - GroupScore：Amazon Comprehend 認為提及已正確分組與相同實體的其他提及項目的信心程度。
  - 文字：實體的文字。
  - 類型：實體的類型。如需所有支援的實體類型，請參閱圖元類型。

事件

Amazon Comprehend 會傳回在輸入文字中偵測到的事件 (受支援的事件類型) 清單。每個事件都會傳回下列相關詳細資訊：

- 類型：事件的類型。如需所有支援的事件類型，請參閱事件類型。
- 引數：與偵測到的事件相關的引數清單。引數由與偵測到的事件相關的實體所組成。參數的角色描述了這種關係，例如誰做了什麼，在何時何地。
  - EntityIndex：可從 Amazon Comprehend 針對此分析傳回的實體清單中識別實體的索引值。
  - 角色：引數類型，說明此引數的實體與事件的關係。如需所有支援的引數類型，請參閱引數類型。
  - 得分：Amazon Comprehend 對角色偵測準確性的信心程度。
- 觸發器：偵測到事件的觸發器清單。觸發器是指示事件發生的單個字或片語。
  - BeginOffset: 輸入文字中的字元偏移量，顯示觸發器的開始位置 (第一個字元位於 0 位置)。
  - EndOffset: 輸入文字中的字元偏移量，顯示觸發器結束位置。
  - 得分：Amazon Comprehend 對偵測準確性的信心程度。
  - 文字：觸發程式的文字。

- GroupScore：Amazon Comprehend 認為觸發器已正確地與相同事件的其他觸發器分組的信賴程度。
- 類型：此觸發程式所指示的事件類型。

偵測事件結果格式

事件偵測任務完成後，Amazon Comprehend 會將分析結果寫入您在開始任務時指定的 Amazon S3 輸出位置。

對於每個偵測到的事件，輸出會以下列格式提供詳細資訊：

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

```
          "GroupScore": number,
          "Type": "string"
        }, ...
      ]
    }, ...
  ]
}
```

實體、事件和引數的支援類型

圖元類型

Type	描述
DATE	對日期或時間的任何引用，無論是特定的還是一般。
設施	建築物、機場、公路、橋樑和其他永久性人造結構和房地產改善工程。
LOCATION	物理位置，例如街道，城市，州，國家，水體或地理坐標。
貨幣價值	美國或其他貨幣的價值。該值可以是特定的或近似值。
組織	公司和一個建立的組織結構定義的人的其他群體。
人	個人或虛構角色的姓名或暱稱。
人物標題	描述一個人的任何職稱，通常是僱傭類別（例如 CEO）或尊敬的（例如 Mr.）。
數量	一個數字或數值以及測量單位。
股票代碼	股票代碼，例如 AMZN、國際證券識別號碼 (ISIN)、統一證券識別程序委員會 (CUMIP) 或證券交易所每日正式名單 (SEDOL)。

事件類型

Type	描述
破產	涉及無法償還未償還債務的人士或公司的法律程序。
就業	當僱員被僱用，解僱，退休，或以其他方式改變就業狀態時發生。
企業收購	當一家公司獲得另一家公司的大部分或全部股份或實物資產的擁有以獲得該公司的控制權時發生。
一般投資	當個人或公司購買具有 future 收入或升值的前景的資產時發生。
企業合併	當兩個或兩個以上的公司團結創建一個新的法律實體時發生。
IPO	私人公司在新股發行中向公眾發行的股份的首次公開發行（IPO）。
權利問題	向現有股東提供一組權利，以按其現有持有的比例購買額外股票（稱為認購權證）。
次要提供項目	公司股東的要約證券。
擱置提供	一項證券交易委員會 (SEC) 條款，允許發行人在一段時間內註冊新發行的證券並出售部分發行品，而無需重新註冊證券或遭受罰款。也稱為貨架登記。
招標提供	購買公司部分或全部股東股份的要約。
股票 _ 分割	當公司的董事會通過向當前股東發行更多股份來增加已發行的股份數量時發生。此事件也適用於反向股票分割。

引數類型

破產的論據類型

引數類型	描述
文件管理員	申請破產的人士或公司。
DATE	破產的日期或時間。
地方	破產發生（或最接近的地方）的地點或設施。

雇用的引數類型

Type	描述
僱員	受僱於公司的人。
員工職稱	員工的職稱。
雇主	僱用該僱員的個人或公司。
START_DATE	僱用的開始日期或時間。
結束日期	僱用的結束日期或時間。

企業的論據類型 _ 收購、投資 _ 一般

Type	描述
量	與交易相關的貨幣價值。
被投資者	與投資有關的人士或公司。
投資者	投資資產的個人或公司。
DATE	收購或投資的日期或時間。
地方	收購或投資發生的地點（或最接近的地方）。

公司合併的引數類型

Type	描述
DATE	合併的日期或時間。
新公司	合併產生的新法人實體。
參與者	涉及合併的公司。

IPO、權利問題、次要提供項目、擱置提供項目、招標提供的引數類型

Type	描述
到期日	提供項目的到期日或時間。
投資者	投資資產的個人或公司。
受要約人	接受發售的人士或公司。
提供金額	與提供項目相關的貨幣價值。
提供日期	提供項目的日期或時間。
要約人	發起提供的人或公司。
要約者總價值	與提供項目相關的總貨幣價值。
記錄日期	提供項目的記錄日期或時間。
銷售代理	促進銷售該項要約的人士或公司。
股價	與股價相關的貨幣價值。
分享數量	與發行項目相關的股份數目。
承銷商	與發行的承銷有關的公司。

股票 _ 分割的引數類型

Type	描述
公司	該公司發行股票拆分的股份。
DATE	股票分割的日期或時間。
分割比	在股票分割前增加新的流通股數量與當前股份數量的比率。

關鍵短語

關鍵片語是包含描述特定事物的名詞短語的字符串。它通常由一個名詞和區分它的修飾符組成。例如，「日」是一個名詞；「美麗的一天」是一個名詞短語，其中包含文章（「a」）和形容詞（「美麗」）。每個關鍵片語都包含一個分數，表示 Amazon Comprehend 對該字串是名詞片語的信心程度。您可以使用分數來判斷偵測是否對您的應用程式具有足夠的信賴度。

偵測關鍵片語操作可以使用 Amazon Comprehend 支援的任何主要語言來執行。所有文件必須使用相同的語言。

您可以使用下列任何操作來偵測文件或文件集中的關鍵片語。

- DetectKeyPhrases
- BatchDetectKeyPhrases
- StartKeyPhrasesDetectionJob

作業會傳回KeyPhrase物件清單，文件中的每個關鍵片語都有一個物件清單。
此BatchDetectKeyPhrases作業會傳回KeyPhrase物件清單，批次中的每個文件都會傳回一份物件清單。StartKeyPhrasesDetectionJob作業會啟動非同步工作，該工作會產生一個檔案，其中包含工作中每個文件的KeyPhrase物件清單。

下列範例是來自DetectKeyPhrases作業的回應。

```
{
    "LanguageCode": "en",
    "KeyPhrases": [
        {
            "Text": "today",
```

```
            "Score": 0.89,
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
            "Score": 0.91,
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ]
}
```

主要语言

您可以使用 Amazon Comprehend 來檢查文字，以判斷主要語言。Amazon Comprehend 使用來自 RFC 5646 的識別碼來識別語言 — 如果有兩個字母的 ISO 639-1 識別碼，並在必要時具有區域子標籤，則會使用該識別碼。否則，它會使用 ISO 639-2 3 個字母的代碼。

如需 RFC 5646 的詳細資訊，請參閱 IETF 工具網站上用於識別語言的標籤。

回應包含一個分數，指出 Amazon Comprehend 具有特定語言是文件中主要語言的信賴等級。每個分數獨立於其他分數。分數並不表示某種語言構成了文檔的特定百分比。

如果長文件 (例如書冊) 包含多種語言，您可以將長文件分成較小的部分，然後在個別片段上執行`DetectDominantLanguage`作業。然後，您可以彙總結果，以決定較長文件中每種語言的百分比。

Amazon Comprehend 語言偵測有下列限制：

- 它不支持語音語言檢測。例如，它不會將「arigato」檢測為日語或「nihao」檢測為中文。
- 它可能具有不同區分的近距語言對，例如印度尼西亞語和馬來語；或波斯尼亞語，克羅地亞語和塞爾維亞語。
- 為獲得最佳結果，請至少提供 20 個字元的輸入文字。

亞馬遜偵測到下列語言。

代碼	語言
af	南非荷蘭文

代碼	語言
am	阿姆哈拉文
ar	Arabic
as	阿薩姆
az	亞塞拜然文
ba	巴什基爾文
be	白俄羅斯文
bn	孟加拉文
bs	波士尼亞文
bg	保加利亞文
ca	加泰隆尼亞文
ceb	宿霧
cs	捷克文
cv	楚瓦什
cy	威爾斯文
da	丹麥文
de	德文
el	Greek
en	英文
eo	世界語
et	Estonian

代碼	語言
eu	巴斯克文
fa	波斯文
fi	芬蘭文
fr	法文
gd	苏格兰盖尔语
ga	愛爾蘭人
gl	加利西亞文
gu	古吉拉特文
ht	海地人
he	Hebrew
ha	豪沙文
hi	北印度文
hr	克羅埃西亞文
hu	匈牙利文
hy	亞美尼亞文
ilo	伊洛克
id	印尼文
is	冰島文
it	義大利文
jv	爪哇語

代碼	語言
ja	日文
kn	坎那達文
ka	喬治亞文
kk	哈薩克文
km	中部高棉
ky	吉尔吉斯
ko	韓文
ku	庫爾德人
lo	老挝
la	拉丁語
lv	拉脫維亞文
lt	立陶宛文
lb	卢森堡语
ml	馬來亞拉姆文
mt	馬爾他文
mr	馬拉地文
mk	馬其頓文
mg	馬達加斯加的
mn	Mongolian
ms	馬來文

代碼	語言
my	緬甸語
ne	尼泊爾人
new	紐瓦里
nl	荷蘭文
no	挪威文
or	奧里亞
om	奧羅莫
pa	旁遮普文
pl	Polish
pt	葡萄牙文
ps	普什托
qu	克丘亞
ro	羅馬尼亞文
ru	俄文
sa	梵文
si	僧伽羅文
sk	斯洛伐克文
sl	斯洛維尼亞文
sd	信德
so	索馬利亞文

代碼	語言
es	西班牙文
sq	阿爾巴尼亞文
sr	塞爾維亞文
su	巽他文
sw	史瓦西里文
sv	瑞典文
ta	坦米爾文
tt	韃靼語
te	特拉古
tg	塔吉克人的
tl	他加祿文
th	Thai
tk	土庫曼
tr	Turkish
ug	维吾尔族
uk	烏克蘭文
ur	烏都文
uz	烏茲別克文
vi	越南文
yi	意第緒語

代碼	語言
yo	約魯巴
zh	簡體中文
zh-TW	繁體中文

您可以使用下列任何操作來偵測文件或一組文件中的主要語言。

- DetectDominantLanguage
- BatchDetectDominantLanguage
- StartDominantLanguageDetectionJob

該DetectDominantLanguage操作返回一個DominantLanguage對象。此BatchDetectDominantLanguage作業會傳回DominantLanguage物件清單，批次中的每個文件都會傳回一份物件清單。StartDominantLanguageDetectionJob作業會啟動非同步工作，該工作會產生一個包含DominantLanguage物件清單的檔案，該檔案用於工作中的每個文件。

下列範例是來自DetectDominantLanguage作業的回應。

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

情緒

使用 Amazon Comprehend 來判斷 UTF-8 編碼文字文件中內容的情緒。例如，您可以使用情緒分析來判斷部落格貼文上留言的情緒，以判斷您的讀者是否喜歡該貼文。

您可以使用 Amazon Comprehend 支援的任何主要語言來判斷文件的情緒。單一工作中的所有文件必須使用相同的語言。

情緒判定會傳回下列值：

- 正面 — 文字表達了整體的正面情緒。
- 負面 — 文字表示整體負面情緒。
- 混合-文本表達了積極和消極的情緒。
- 中性 — 文字不表達正面或負面的情緒。

您可以使用下列任何 API 作業來偵測文件或一組文件的情緒。

- DetectSentiment
- BatchDetectSentiment
- StartSentimentDetectionJob

作業會傳回文字最可能的情緒，以及每個情緒的分數。分數代表正確偵測到的情緒的可能性。例如，在下面的範例中，文字有Positive情緒的可能性為 95%。文字有情Negative緒的可能性不到 1%。您可以使用SentimentScore來判斷偵測的精確度是否符合您應用程式的需求。

此DetectSentiment作業會傳回包含偵測到的情緒和SentimentScore物件的物件。此BatchDetectSentiment作業會傳回情緒和SentimentScore物件的清單，批次中的每個文件都會傳回一份清單。此StartSentimentDetectionJob作業會啟動非同步工作，該工作會產生一個包含情緒和SentimentScore物件清單的檔案，該檔案用於工作中的每個文件。

下列範例是來自DetectSentiment作業的回應。

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

目標情緒

鎖定目標情緒可讓您精細瞭解輸入文件中與特定實體 (例如品牌或產品) 相關聯的情緒。

目標情緒和情緒之間的差異在於輸出資料中的粒度層級。情緒分析會決定每個輸入文件的主要情緒，但不提供資料以供進一步分析。目標情緒分析會決定每個輸入文件中特定實體的實體層級情緒。您可以分析輸出數據，以確定獲得正面或負面反饋的特定產品和服務。

例如，在一組餐廳評論中，顧客提供了以下評論：「炸玉米餅很美味，工作人員很友好。」 對此審查進行分析會產生以下結果：

- 情緒分析會決定每個餐廳評論的整體情緒為正面、負面、中立或混合。在此範例中，整體情緒是正面的。
- 目標情緒分析會決定顧客在評論中提及的餐廳實體和屬性的情緒。在這個例子中，客戶對「炸玉米餅」和「員工」做出了積極的評論。

目標情緒會為每個分析工作提供下列輸出：

- 在文件中提到的實體的身份。
- 每個實體提及的實體類型分類。
- 每個實體提及的情緒分數和情緒分數。
- 對應於單一實體的提及群組 (共同參照群組)。

您可以使用主控台或 API 執行目標情緒分析。主控台和 API 支援針對目標情緒的即時分析和非同步分析。

Amazon Comprehend 支援英文文件的目標情緒。

如需有關目標情緒的其他資訊 (包括教學課程)，請參閱AWS機器學習部落格中的 Amazon Comprehend 目標情緒以文字擷取精細情緒。

主題

- 圖元類型
- 共同參考群組
- 輸出文件組織
- 使用主控台進行即時分析
- 目標情緒輸出範例

圖元類型

目標情緒可識別下列實體類型。它分配實體類型 OTHER 如果實體不屬於任何其他類別。輸出檔案中的每個實體提及都包括實體類型，例如"Type": "PERSON"。

實體類型定義

實體類型	定義
人	例子包括個人，人群，暱稱，虛構人物和動物名稱。
LOCATION	地理位置，如國家, 城市, 州, 地址, 地質構造, 水體, 自然地標, 和天文位置.
組織	例子包括政府，公司，體育隊和宗教。
設施	建築物、機場、公路、橋樑和其他永久性人造結構和房地產改善工程。
品牌	特定商業項目或產品系列的組織、群組或製造者。
商業項目	任何非通用可購買或取得的物品，包括車輛，以及只生產一件物品的大型產品。
電影	電影或電視節目。實體可以是全名、暱稱或副標題。
音樂	全部或部分歌曲。此外，個別音樂創作的集合，例如專輯或選集。
書	一本書，專業出版或自行出版。
軟體	正式發行的軟體產品。
遊戲	遊戲，例如視頻遊戲，棋盤遊戲，常見遊戲或運動。
個人標題	官方頭銜和榮譽，例如總統，博士或博士。
EVENT	例子包括節日，音樂會，選舉，戰爭，會議和促銷活動。
DATE	對日期或時間的任何引用，無論是特定的還是一般的，無論是絕對還是相對的。
數量	所有測量及其單位（貨幣，百分比，數字，字節等）。

實體類型	定義
ATTRIBUTE	實體的屬性，特徵或特徵，例如產品的「質量」，電話的「價格」或 CPU 的「速度」。
OTHER	不屬於任何其他類別的實體。

共同參考群組

目標情緒可識別每個輸入文件中的共同參照群組。共同參照群組是文件中對應於一個真實實體的提及群組。

Example

在下列客戶檢閱範例中，「spa」是具有實體類型的實體FACILITY。該實體還有另外兩個提及作為代詞（「it」）。

輸出文件組織

目標情緒分析工作會建立 JSON 文字輸出檔案。該文件包含每個輸入文檔的一個 JSON 對象。每個 JSON 物件都包含下列欄位：

- 實體 — 在文件中找到的實體陣列。
- 檔案 — 輸入文件的檔案名稱。
- 行 — 如果輸入檔案是每行一個文件，則實體會包含檔案中文件的行號。

> **Note**
>
> 如果目標情緒未識別輸入文字中的任何實體，則會傳回空白陣列作為「實體」結果。

下列範例顯示具有三行輸入之輸入檔案的實體。輸入格式為 ONE_DOC_PER_LINE，因此每一行輸入都是一個文檔。

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

Entity 陣列中的實體包括文件中偵測到的實體提及的邏輯群組 (稱為共同參照群組)。每個實體具有以下整體結構：

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

實體包含下列欄位：

- 提及 — 文件中實體提及的陣列。陣列代表共同參照群組。如需範例，請參閱the section called “共同參考群組”。「提及」陣列中提及的順序就是它們在文件中的位置 (偏移) 順序。每個提及都包含該提及的情緒分數和群組分數。群組分數表示這些提及屬於相同實體的信賴等級。
- DescriptiveMentionIndex—「提及」陣列中的一或多個索引，可為實體群組提供最佳名稱。例如，一個實體可以有三個提及文字值「ABC 酒店」、「ABC 酒店」和「它」。最好的名字是「ABC 酒店」，其 DescriptiveMentionIndex 值為 [0,1]。

每個提及包括以下字段

- BeginOffset— 文件文字中提及開始處的偏移量。
- EndOffset— 文件文字中提及結束處的偏移量。
- GroupScore— 該組中提到的所有實體與相同實體相關的信心。
- 文字 — 文件中識別實體的文字。
- 類型 — 實體的類型。Amazon Comprehend 支援多種實體類型。
- 分數 — 實體相關的模型信賴度。值範圍是零到一，其中一個是最高可信度。
- MentionSentiment— 包含提及的情緒和情緒分數。
- 情緒 — 提及的情緒。值包括：正面、中性、負面和混合。
- SentimentScore— 為每個可能的情緒提供模型信心。值範圍是零到一，其中一個是最高可信度。

情緒值具有下列意義：

- 正面 — 實體提及表達了積極的情緒。
- 負面 — 實體提及表示負面情緒。
- 混合-實體提及表達了積極和消極的情緒。
- 中立-實體提及不表達正面或負面的情緒。

在下列範例中，實體在輸入文件中只有一個提及，因此 DescriptiveMentionIndex為零 (「提及」陣列中的第一個提及)。識別的實體是名稱為「I」的個人。 情緒分數是中立的。

```
{"Entities":[
  {
    "DescriptiveMentionIndex": [0],
    "Mentions": [
```

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

使用主控台進行即時分析

您可以使用 Amazon Comprehend 主控台即時執the section called “目標情緒”行。使用範例文字，或將自己的文字貼到輸入文字方塊中，然後選擇「分析」。

在「深入解析」面板中，主控台會顯示目標情緒分析的三種檢視：

- 分析文字 — 顯示分析的文字並為每個實體加底線。底線的顏色表示分析指派給實體的情緒值 (正面、中性、負面或混合)。控制台在分析文本框的右上角顯示顏色映射。如果您將游標停留在實體上，主控台會顯示包含實體分析值 (實體類型、情緒分數) 的快顯面板。
- 結果 — 顯示一個表格，其中包含文字中識別的每個實體提及的列。對於每個實體，此表格會顯示實體和實體分數。此列也包含主要情緒和每個情緒值的分數。如果同一個實體有多個提及 (稱為 athe section called “共同參考群組”)，表格會將這些提及顯示為與主實體相關聯的可摺疊資料列集。

  如果您將游標暫留在「結果」表格中的實體列上，則主控台會在「已分析」文字面板中反白顯示實體提及。

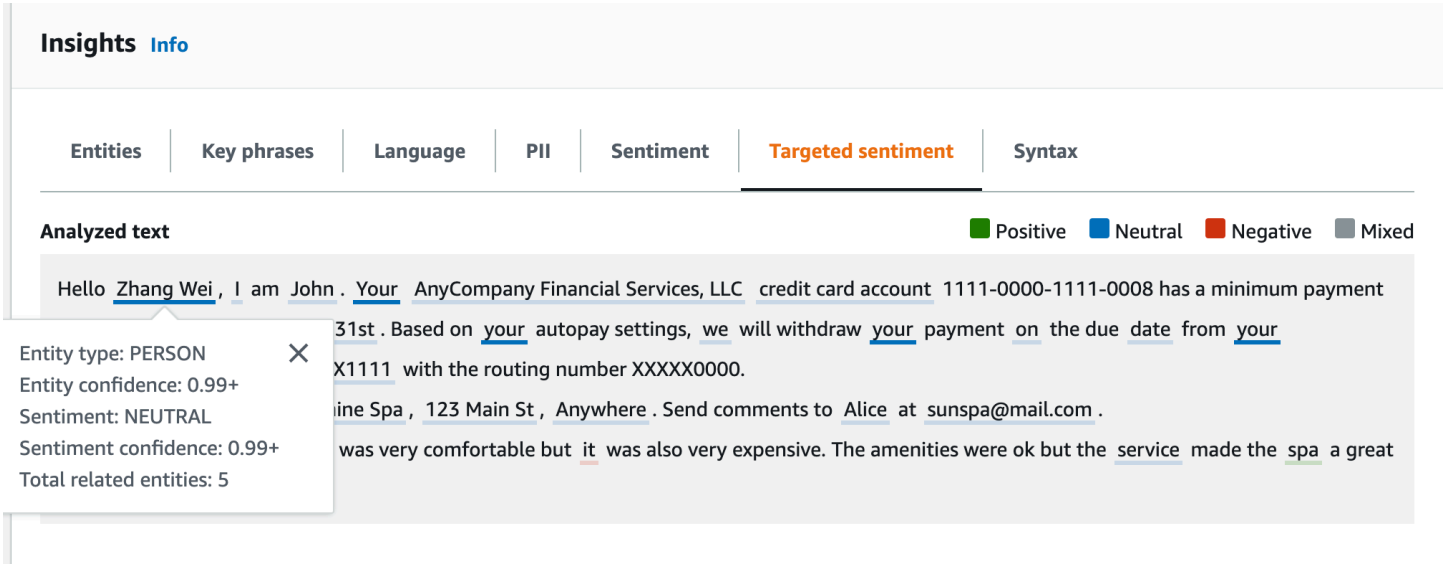
- 應用程式整合 — 顯示 API 請求的參數值，以及 API 回應中傳回的 JSON 物件結構。如需 JSON 物件中欄位的說明，請參閱the section called “輸出文件組織”。

主控台即時分析範例

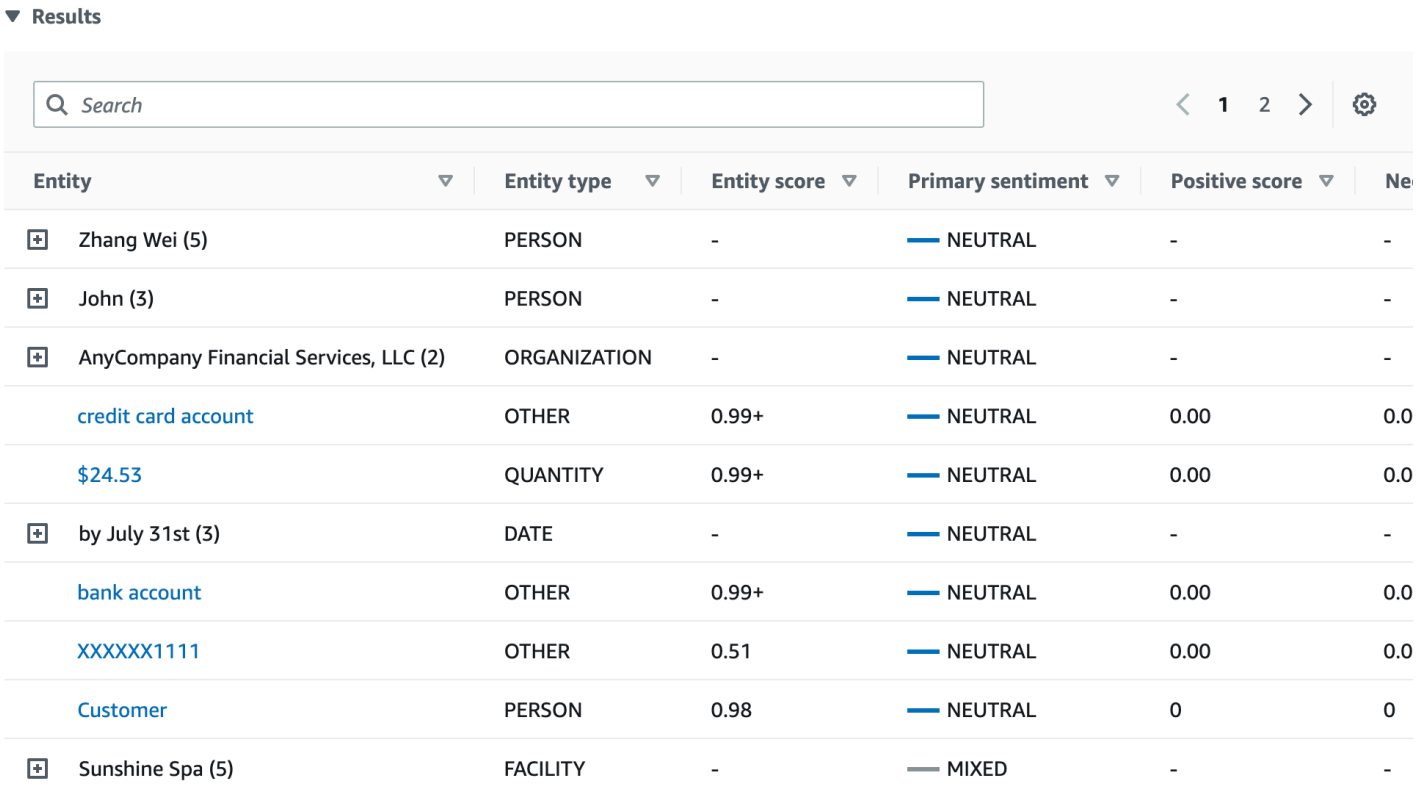
此範例使用下列文字做為 input，這是主控台提供的預設輸入文字。

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

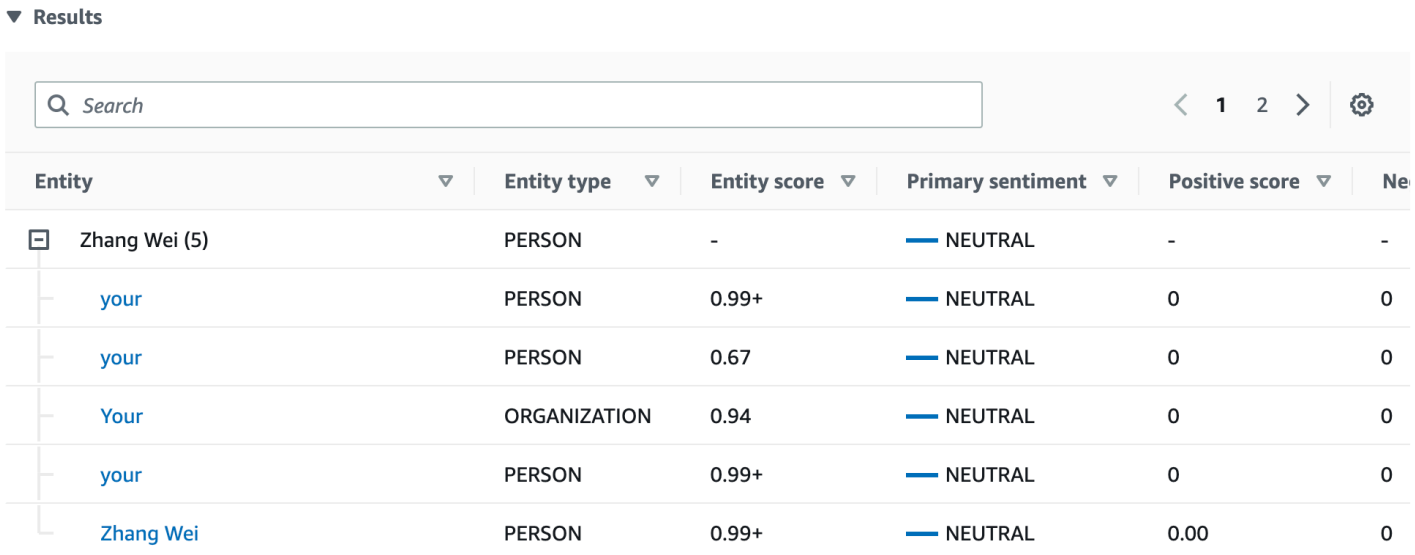
「已分析」文字面板會顯示此範例的下列輸出。將滑鼠移Zhang Wei至文字上方，即可檢視此實體的躍現式面板。



「結果」表格提供每個實體的其他詳細資訊，包括實體分數、主要情緒和每個情緒的分數。



在我們的範例中，目標情緒分析會識別輸入文字中的每一個提及，都是對個人實體張偉的參照。控制台將這些提及顯示為一組與主實體相關聯的可折疊行。



「應用程式整合」面板會顯示 DetectTargetedSentiment API 所產生的 JSON 物件。如需完整範例，請參閱下一節。

目標情緒輸出範例

下列範例顯示目標情緒分析工作的輸出檔案。輸入文件由三個簡單的文檔組成：

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

此輸入檔案的目標情緒分析會產生下列輸出。

```
  {"Entities":[
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 4,
          "EndOffset": 10,
          "Score": 0.999991,
          "GroupScore": 1,
          "Text": "burger",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 0,
              "Positive": 1
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 38,
```

```
          "EndOffset": 44,
          "Score": 1,
          "GroupScore": 1,
          "Text": "burger",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0.000005,
              "Negative": 0.000005,
              "Neutral": 0.999591,
              "Positive": 0.000398
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 45,
          "EndOffset": 48,
          "Score": 0.961575,
          "GroupScore": 1,
          "Text": "bun",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0.000327,
              "Negative": 0.000286,
              "Neutral": 0.050269,
              "Positive": 0.949118
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
```

```
      ],
      "Mentions": [
        {
          "BeginOffset": 73,
          "EndOffset": 89,
          "Score": 0.999988,
          "GroupScore": 1,
          "Text": "customer service",
          "Type": "ATTRIBUTE",
          "MentionSentiment": {
            "Sentiment": "NEGATIVE",
            "SentimentScore": {
              "Mixed": 0.000001,
              "Negative": 0.999976,
              "Neutral": 0.000017,
              "Positive": 0.000006
            }
          }
        }
      ]
    }
  ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 0
}
{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 0,
          "EndOffset": 2,
          "Score": 0.99995,
          "GroupScore": 1,
          "Text": "My",
          "Type": "PERSON",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
```

```
            "Neutral": 1,
            "Positive": 0
          }
        }
      }
    ]
  },
  {
    "DescriptiveMentionIndex": [
      0,
      2
    ],
    "Mentions": [
      {
        "BeginOffset": 3,
        "EndOffset": 9,
        "Score": 0.999999,
        "GroupScore": 1,
        "Text": "burger",
        "Type": "OTHER",
        "MentionSentiment": {
          "Sentiment": "POSITIVE",
          "SentimentScore": {
            "Mixed": 0.000002,
            "Negative": 0.000001,
            "Neutral": 0.000003,
            "Positive": 0.999994
          }
        }
      },
      {
        "BeginOffset": 24,
        "EndOffset": 26,
        "Score": 0.999756,
        "GroupScore": 0.999314,
        "Text": "it",
        "Type": "OTHER",
        "MentionSentiment": {
          "Sentiment": "POSITIVE",
          "SentimentScore": {
            "Mixed": 0,
            "Negative": 0.000003,
            "Neutral": 0.000006,
            "Positive": 0.999991
```

```
            }
          }
        },
        {
          "BeginOffset": 41,
          "EndOffset": 47,
          "Score": 1,
          "GroupScore": 0.531342,
          "Text": "burger",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0.000215,
              "Negative": 0.000094,
              "Neutral": 0.00008,
              "Positive": 0.999611
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 52,
          "EndOffset": 58,
          "Score": 0.965462,
          "GroupScore": 1,
          "Text": "plenty",
          "Type": "QUANTITY",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 1,
              "Positive": 0
            }
          }
        }
```

```
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 62,
          "EndOffset": 70,
          "Score": 0.998353,
          "GroupScore": 1,
          "Text": "toppings",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0,
              "Neutral": 0.999964,
              "Positive": 0.000036
            }
          }
        }
      ]
    }
  ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 1
}
{
  "Entities": [
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 4,
          "EndOffset": 10,
          "Score": 1,
          "GroupScore": 1,
          "Text": "burger",
          "Type": "OTHER",
```

```
          "MentionSentiment": {
            "Sentiment": "POSITIVE",
            "SentimentScore": {
              "Mixed": 0.001515,
              "Negative": 0.000822,
              "Neutral": 0.000243,
              "Positive": 0.99742
            }
          }
        },
        {
          "BeginOffset": 36,
          "EndOffset": 38,
          "Score": 0.999843,
          "GroupScore": 0.999661,
          "Text": "it",
          "Type": "OTHER",
          "MentionSentiment": {
            "Sentiment": "NEGATIVE",
            "SentimentScore": {
              "Mixed": 0,
              "Negative": 0.999996,
              "Neutral": 0.000004,
              "Positive": 0
            }
          }
        }
      ]
    },
    {
      "DescriptiveMentionIndex": [
        0
      ],
      "Mentions": [
        {
          "BeginOffset": 53,
          "EndOffset": 60,
          "Score": 1,
          "GroupScore": 1,
          "Text": "service",
          "Type": "ATTRIBUTE",
          "MentionSentiment": {
            "Sentiment": "NEUTRAL",
            "SentimentScore": {
```

```
              "Mixed": 0.000033,
              "Negative": 0.000089,
              "Neutral": 0.993325,
              "Positive": 0.006553
            }
          }
        }
      ]
    }
  ],
  "File": "TargetSentimentInputDocs.txt",
  "Line": 2
}
  }
```

語法分析

使用語法分析來剖析文件中的單字，並針對文件中的每個單字傳回語音或語法函數的一部分。您可以識別文檔中的名詞，動詞，形容詞等。使用此資訊可以更深入地瞭解文件內容，並瞭解文件中字詞的關係。

例如，您可以在文件中尋找名詞，然後尋找與這些名詞相關的動詞。在「我祖母移動了沙發」之類的句子中，您可以看到名詞，「祖母」和「沙發」以及動詞「移動」。您可以使用此資訊來建置應用程式，以分析您感興趣的文字組合的文字。

為了開始分析，Amazon Comprehend 會剖析來源文字，以尋找文字中的個別單字。剖析文字之後，會為每個單字指派來源文字中的語音部分。

Amazon Comprehend 可以識別語音的以下部分。

權杖	部分講話
調整	形容詞 通常會修改名詞的字詞。
ADP	沉積 一個前言或後置短語的頭部。
冒險	副詞

權杖	部分講話
	通常修改動詞的單詞。他們也可以修改形容詞和其他副詞。
AUX	輔助 伴隨動詞短語動詞的功能詞。
CCONJ	协调结合 協調連詞會將句子中的單字、片語或子句連接起來，而不會將其中一個與另一個從屬關聯。
連詞	連接詞 連詞會連接句子中的單字、片語或子句。
DET	決定詞 指定特定名詞短語的文章和其他單詞。
INTJ	感言 用作感嘆號或感嘆號的一部分的單詞。
名詞	名詞 指定人物、地點、事物、動物或想法的字詞。
NUM	數字 表示數字的單詞，通常是決定詞，形容詞或代詞。

權杖	部分講話
O	其他 無法指派為語音類別一部分的單字。
PART	助詞 與另一個單詞或短語相關聯的功能詞來賦予意義。
普朗	代詞 替代名詞或名詞短語的單詞。
PROPN	正确名詞 一個名詞，是一個特定的個人，地點或對象的名字。
關鍵的	標點符號 分隔文字的非字母字元。
SCONJ	次级连结 將相依子句連接到句子的結合。下屬連詞的一個例子是「因為」。
SYM	符號 類似單字的實體，例如貨幣符號 ($) 或數學符號。
動詞	動詞 表示事件和動作的單詞。

如需有關語音部分的詳細資訊，請參閱通用相依性網站上的通用 POS 標籤。

操作返回標記，用於標識單詞在文本中表示的單詞和語音部分。每個標記都代表來源文字中的一個字。它提供單字在來源中的位置、單字在文字中使用的語音部分、Amazon Comprehend 確定語音部分已正確識別的信心，以及從來源文字剖析的單字。

以下是語法令牌列表的結構。文件中的每個字都會產生一個語法 Token。

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

每個令牌都提供以下信息：

- `BeginOffset`和 `EndOffset` — 提供單字在輸入文字中的位置。
- `PartOfSpeech`提供兩項資訊，用於識別語音部分，以及表`Score`示 Amazon Comprehend 語法對於語音部分已正確識別的信心。`Tag`
- `Text`提供已識別的字詞。
- `TokenId`提供權杖的識別碼。標識符是令牌在令牌列表中的位置。

Amazon Comprehend 定制

您可以根據自己的特定需求自訂 Amazon Comprehend，而不必具備建置機器學習型 NLP 解決方案所需的技能組合。Comprehend 自訂使用自動機器學習或 AutoML，使用您提供的訓練資料代表您建立自訂的 NLP 模型。

輸入文件處理 — Amazon Comprehend 支援自訂分類和自訂實體辨識的單一步驟文件處理。例如，您可以混合輸入純文字文件和半結構化文件 (例如 PDF 文件、Microsoft Word 文件和影像) 至自訂分析工作。如需詳細資訊，請參閱 文件處理。

自訂分類 — 建立自訂分類模型 (分類器)，將文件組織成您自己的類別。針對每個分類標籤，提供一組最能代表該標籤的文件，並在其上訓練您的分類器。訓練完成後，可以在任意數量的未標籤文件集上使用分類器。您可以使用控制台獲得無代碼的體驗，也可以安裝最新的 AWS SDK。如需詳細資訊，請參閱 自訂分類。

自訂實體辨識 — 建立自訂實體辨識模型 (辨識器)，以分析特定詞彙和以名詞為基礎的片語的文字。您可以訓練辨識者擷取諸如政策編號之類的詞彙，或暗示客戶升級的詞組。若要訓練模型，您需要提供實體清單以及包含這些實體的一組文件。模型訓練完畢後，您可以針對該模型提交分析工作，以擷取其自訂實體。如需更多詳細資訊，請參閱 自訂實體辨識。

主題建模

您可以使用 Amazon Comprehend 來檢查文件集合的內容，以判斷常見的主題。例如，您可以為 Amazon Comprehend 提供新聞文章的集合，它將確定主題，例如體育，政治或娛樂。文檔中的文本不需要註釋。

Amazon Comprehend 使用潛在的狄利克雷分配為基礎的學習模型來判斷一組文件中的主題。它檢查每個文檔以確定單詞的上下文和含義。在整個文件設定中，經常屬於相同前後關聯的字組成了一個主題。

根據該主題在文件中的普遍程度，以及主題與單字的親和程度，單字與文件中的主題相關聯。根據特定文檔中的主題分佈，同一個單詞可以與不同文檔中的不同主題相關聯。

例如，主要談論運動的文章中的「葡萄糖」一詞可以指定給「運動」主題，而有關「醫學」的文章中的相同單詞將被指定給「醫學」主題。

與主題相關聯的每個字都會有一個權重，指出該字詞有多少幫助定義主題。寬度是指出單字在整個文件集中出現在主題中與主題中其他字詞相比的次數。

為了獲得最準確的結果，您應該向 Amazon Comprehend 提供可以使用的最大可能語料庫。為了獲得最佳結果：

- 您應該在每個主題建模工作中使用至少 1,000 個文件。
- 每個文件應至少有 3 個句子長。
- 如果一個文檔主要由數字數據組成，你應該從語料庫中刪除它。

主題建模是一個非同步處理程序。您可以使用操作將文件清單從 Amazon S3 儲存貯體提交給 Amazon Comprehend。StartTopicsDetectionJob回應會傳送至 Amazon S3 儲存貯體。您可以配置輸入和輸

出存儲桶。取得您使用作業提交的主題模型工作清單，並檢視使用該ListTopicsDetectionJobs作業之工DescribeTopicsDetectionJob作的相關資訊。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

文件必須是 UTF-8 格式的文字檔案。您可以通過兩種方式提交文件。下表顯示選項。

格式	描述
每個檔案一個文件	每個檔案都包含一個輸入文件。這是最適合大型文檔的集合。
每行一個文件	輸入為單一檔案。檔案中的每一行都被視為一個文件。這最適合短文件，例如社交媒體張貼。 每行必須以換行符（LF，\n），回車符（CR，\r）或兩者結束（CRLF，\r\n）。無法使用 Unicode 行分隔符號 (u+2028) 來結束一行。

如需詳細資訊，請參閱資InputDataConfig料類型。

Amazon Comprehend 處理您的文件收集之後，會傳回一個包含兩個檔案的壓縮存檔，`topic-terms.csv`以及。`doc-topics.csv`若要取得有關輸出檔案的更多資訊，請參閱OutputDataConfig。

第一個輸出檔案是集合中主題的清單。`topic-terms.csv`根據預設，每個主題清單會根據主題的重量包含最上層的術語。例如，如果您為 Amazon Comprehend 提供一系列報紙文章，它可能會傳回下列內容來描述集合中的前兩個主題：

主題	術語	Weight
000	球隊	0.118533
000	game	0.106072
000	player	0.031625
000	季節	0.023633
000	播放	0.021118

主題	術語	Weight
000	碼	0.024454
000	教練	0.016012
000	遊戲	0.016191
000	足球	0.015049
000	組織指揮人	0.014239
001	杯	0.205236
001	食品	0.040686
001	分鐘	0.036062
001	add	0.029697
001	湯匙	0.028789
001	石油	0.021254
001	胡椒	0.022205
001	茶匙	0.020040
001	酒	0.016588
001	糖	0.015101

權重表示給定主題中單詞的概率分佈。由於 Amazon Comprehend 只返回每個主題的前 10 個單詞的權重不會總和為 1.0。在少數情況下，主題中少於 10 個單詞，權重將總和為 1.0。

這些單詞通過查看所有主題中的出現情況按其歧視力進行排序。通常，這與它們的重量相同，但在某些情況下，例如表中的單詞「玩」和「碼」，這會導致與重量不相同的順序。

您可以指定要傳回的主題數目。例如，如果您要求 Amazon Comprehend 返回 25 個主題，它會返回集合中 25 個最突出的主題。Amazon Comprehend 多可以檢測到一個集合中 100 個主題。根據您對域的了解選擇主題的數量。可能需要一些實驗才能到達正確的號碼。

第二個檔案會列出與主題相關聯的文件`doc-topics.csv`，以及與主題有關的文件比例。如果您指`ONE_DOC_PER_FILE`定了文檔由文件名標識。如果您指`ONE_DOC_PER_LINE`定了文檔，則由文件名和文件中的 0 索引行號標識。例如，針對以每個檔案一份文件提交的文件集合，Amazon Comprehend 可能會傳回以下內容：

文件	主題	比例
示例文檔 1	000	0.999330137
示例文檔 2	000	0.998532187
示例文檔 3	000	0.998384574
...		
示例文檔	000	3.57E-04

亞馬 Amazon Comprehend 利用由 MBM，這是根據開放數據庫許可證（ODBL）V1.0 在這裡提供的列表數據集的信息。

文件處理模式

Amazon Comprehend 支持三種文檔處理模式。您選擇的模式取決於您需要處理的文件數量以及您需要如何立即查看結果：

- 單一文件同步 — 您可以使用單一文件呼叫 Amazon Comprehend，並接收同步回應，並立即傳送到您的應用程式 (或主控台)。
- 多文件同步 — 您可以使用最多 25 個文件的集合來呼叫 Amazon Comprehend API，並收到同步回應。
- 非同步批次 — 對於大量文件集合，請將文件放入 Amazon S3 儲存貯體，然後啟動非同步任務 (使用主控台或 API 操作) 以分析文件。Amazon Comprehend 會將分析結果存放在您在請求中指定的 S3 儲存貯體/資料夾中。

主題

- 單一文件處理
- 多文檔同步處理

- 非同步批次處理

單一文件處理

單一文件作業是將文件分析結果直接傳回至您的應用程式的同步作業。當您建立一次處理一個文件的互動式應用程式時，請使用單一文件同步作業。

如需同步 API 作業的詳細資訊，請參閱 使用內建模型進行即時分析 (針對主控台) 和使用 API 進行即時分析.

多文檔同步處理

當您有多個要處理的文件時，您可以使用 Batch* API 操作一次將多個文件傳送到 Amazon Comprehend。您最多可以在每個請求中發送 25 份文件。Amazon Comprehend 會傳回一份回應清單，請求中的每個文件都有一份回應清單。使用這些操作發出的請求是同步的。您的應用程式會呼叫作業，然後等待來自服務的回應。

使用Batch*操作與為請求中的每個文檔調用單個文檔 API 相同。使用這些 API 可以為您的應用程式帶來更好的效能。

每個 API 的輸入都是 JSON 結構，其中包含要處理的文件。對於除以外的所有操作BatchDetectDominantLanguage，您必須設置輸入語言。您只能為每個請求設置一種輸入語言。例如，以下是作BatchDetectEntities業的輸入。它包含兩個文件，是英文的。

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

Batch*作業的回應包含兩個清單，即ResultList和ErrorList。針對每個已成功處理的文件，ResultList包含一筆記錄。要求中每個文件的結果與您在文件上執行單一文件作業時所得到的結果相同。系統會根據輸入檔案中文件的順序，為每個文件的結果指派索引。來自BatchDetectEntities操作的響應是：

```
{
   "ResultList"  : [
      {
```

```
         "Index": 0,
         "Entities": [
            {
               "Text": "Seattle",
               "Score": 0.95,
               "Type": "LOCATION",
               "BeginOffset": 22,
               "EndOffset": 29
            },
            {
               "Text": "almost 4 years",
               "Score": 0.89,
               "Type": "QUANTITY",
               "BeginOffset": 34,
               "EndOffset": 48
            }
         ]
      },
      {
         "Index": 1,
         "Entities": [
            {
              "Text": "today",
              "Score": 0.87,
              "Type": "DATE",
              "BeginOffset": 14,
              "EndOffset": 19
            },
            {
               "Text": "Seattle",
               "Score": 0.96,
               "Type": "LOCATION",
               "BeginOffset": 23,
               "EndOffset": 30
            }
         ]
      }
   ],
   "ErrorList": []
}
```

當請求中發生錯誤時，響應包ErrorList含標識包含錯誤的文檔。該文檔由其在輸入列表中的索引來標識。例如，下列對BatchDetectLanguage作業的輸入包含無法處理的文件：

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

來自 Amazon Comprehend 的回應包含一個錯誤清單，用來識別包含錯誤的文件：

```
{
    "ResultList": [
        {
          "Index": 0,
          "Languages":[
            {
              "LanguageCode":"en",
              "Score": 0.99
            }
          ]
        },
        {
          "Index": 2
          "Languages":[
            {
              "LanguageCode":"es",
              "Score": 0.82
            }
          ]
        }
    ],
    "ErrorList": [
      {
        "Index": 1,
        "ErrorCode": "InternalServerException",
        "ErrorMessage": "Unexpected Server Error. Please try again."
      }
    ]
}
```

如需同步批次 API 作業的詳細資訊，請參閱即時批次處理 API。

非同步批次處理

若要分析大型文件和大型文件集合，請使用 Amazon Comprehend 非同步操作。

若要分析文件集合，您通常需要執行下列步驟：

1. 將文件存放在 Amazon S3 儲存貯體中。
2. 啟動一或多個分析工作以分析文件。
3. 監視分析工作的進度。
4. 任務完成時，從 S3 儲存貯體擷取分析結果。

如需使用非同步 API 作業的詳細資訊，請參閱 使用主控台執行分析工作 (主控台) 和使用 API 的非同步分析工作.

Amazon Comprehend 支持的語言

Amazon Comprehend 支援多種語言的各種功能。支援的語言以及支援這些語言的功能可在下表中看到。

主題

支援的語言

Amazon Comprehend (偵測主要語言功能除外) 針對一或多個功能支援下列語言。

代碼	語言
de	德文
en	英文
es	西班牙文
it	義大利文
pt	葡萄牙文
fr	法文
ja	日文
ko	韓文
hi	北印度文
ar	Arabic
zh	中文 (簡體)
zh-TW	中文 (繁體)

> **Note**
>
> Amazon Comprehend 使用來自 RFC 5646 的識別碼來識別語言 — 如果有一個兩個字母的 ISO 639-1 識別碼，並帶有區域子標籤。/如果必要，它會使用該識別碼。否則，它會使用 ISO 639-2 3 個字母的代碼。
> 如需 RFC 5646 的詳細資訊，請參閱 IETF 工具網站上用於識別語言的標籤。

亞馬遜功能支援的語言

功能	支援的語言
主要语言	請參閱 主要语言。
實體	所有支援的語言。
關鍵短語	所有支援的語言。
偵測 PII 實體	英文和西班牙文。
為 PII 實體加上標籤	英文和西班牙文。
情緒	所有支援的語言。
目標情緒	英語。
語法分析	德文 (de)、英文 (en)、西班牙文 (es)、法文 (fr)、義大利文 (it) 和葡萄牙文 (pt)。
主題建模	不依賴於使用的語言。不支援以字元為基礎的語言，例如中文、日文和韓文。
自訂分類	純文字模型支援下列語言：德文 (de)、英文 (en)、西班牙文 (es)、法文 (fr)、義大利文 (it) 和葡萄牙文 (pt)。 原生文件模型僅支援英文文件。

功能	支援的語言
自訂實體辨識	德文 (de)、英文 (en)、西班牙文 (es)、法文 (fr)、義大利文 (it) 和葡萄牙文 (pt)。 PDF 和 Word 的自訂實體辨識僅支援英文文件。

設定

第一次使用 Amazon Comprehend 之前，請先完成下列任務。

設定工作

- 註冊 AWS 帳戶
- 建立管理使用者
- 設置AWS Command Line Interface（AWS CLI）
- 授予程式設計存取權

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

設置AWS Command Line Interface（AWS CLI）

您不需要 AWS CLI 執行入門練習中的步驟。不過，本指南中的一些其他練習則需要它。如果您願意，可以跳過此步驟開始使用 Amazon Comprehend，然後轉到AWS CLI稍後設置。

安裝及設定 AWS CLI

1. 安裝AWS CLI。如需指示，請參閱使AWS Command Line Interface用者指南中的下列主題：

   安裝或更新最新版本的 AWS Command Line Interface

2. 設定 AWS CLI。如需指示，請參閱使AWS Command Line Interface用者指南中的下列主題：

   設定 AWS Command Line Interface

授予程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考

哪個使用者需要程式設計存取權？	到	By
		指南 中的 使用長期憑證進行驗證。 • 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

開始使用 Amazon Comprehend

以下練習使用 Amazon Comprehend 主控台建立和執行非同步實體偵測任務。本練習假設您熟悉亞馬遜簡單儲存服務 (Amazon S3)。如需更簡單的範例，請參閱使用內建模型進行即時分析。

建立實體偵測工作

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [分析工作]，然後選擇 [建立工作]。
3. 在 [Job 設定] 下，為工作命名。該名稱在「地區」和帳戶中必須是唯一的。
4. 在「分析類型」中選擇「實體」。
5. 在「語言」中，選擇輸入文件的語言。
6. 在輸入資料下，對於資料來源，選擇範例文件。主控台會將 S3 位置設定為包含公用範例的資料夾。
7. 在「輸出資料」下的 S3 位置，在 Amazon S3 中貼上輸出檔案的 URL 或資料夾位置。
8. 在 [存取權限] 區段下，選取 [建立 IAM 角色]。主控台會建立具有適當許可的新 IAM 角色，讓 Amazon Comprehend 存取輸入和輸出值區。
9. 填寫完表格後，請選擇 [建立工作] 以建立並啟動主題偵測工作。

   新工作會顯示在工作清單中，而狀態欄位顯示工作的狀態。此欄位可`IN_PROGRESS`用於正在處理的工`COMPLETED`作、已成功完成的工作，以及`FAILED`發生錯誤的工作。
10. 選擇要開啟「J ob 詳細資料」面板的工作。
11. 在「輸出」下的「輸出資料位置」中，選擇要開啟 Amazon S3 主控台的連結。
12. 在 Amazon S3 主控台中，選擇 [下載並儲存`output.tar.gz`檔案]。
13. 解壓縮檔案並將其儲存為 Json 檔案。
14. 如the section called “實體”需實體類型和每個偵測到的實體欄位的說明，請參閱。

使用亞馬遜控制台進行分析

您可以使用 Amazon Comprehend 主控台來即時分析文件或執行非同步分析任務。

使用內建模型的即時分析，您可以辨識實體、擷取關鍵片語、偵測主要語言、偵測 PII、判斷情緒、分析目標情緒，以及分析語法。

您可以使用內建模型執行分析工作，以尋找實體、事件、片語、主要語言、情緒、目標情緒和個人識別資訊 (PII) 等見解。您也可以執行主題建模工作。

該控制台還支持使用自定義模型的實時和異步分析。如需更多詳細資訊，請參閱 自訂分類 及 自訂實體辨識。

主題

- 使用內建模型進行即時分析
- 使用主控台執行分析工作

使用內建模型進行即時分析

您可以使用 Amazon Comprehend 主控台對 UTF-8 編碼的文字文件執行即時分析。該文件可以是英文，也可以是 Amazon Comprehend 支援的其他語言之一。結果會顯示在主控台中，以便您可以檢閱分析。

若要開始分析文件，請登入AWS Management Console並開啟 Amazon Comprehend 主控台。

您可以使用自己的文字取代範例文字，然後選擇 [分析] 以分析文字。在要分析的文字下方，「結果」窗格會顯示有關文字的詳細資訊。

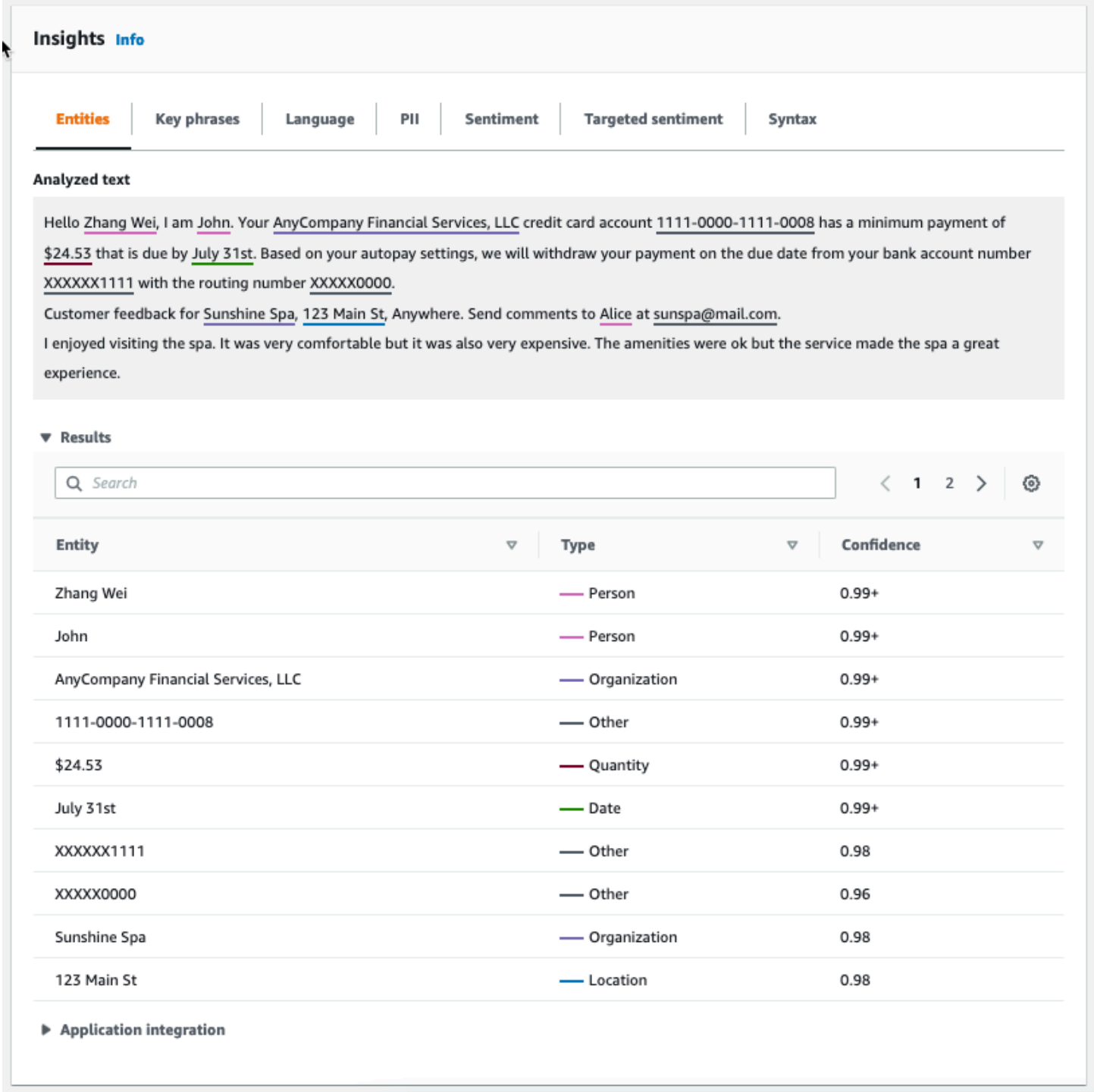
使用內建模型執行即時分析

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [即時分析]。
3. 在「輸入類型」下，選擇「內建」做為 「分析」
4. 輸入您要分析的文字。
5. 選擇「分析」。主控台會在「深入解析」面板中顯示文字分析結果。「深入解析」面板包含每個深入解析類型的索引標籤。下列各節說明分析類型的結果。

主題

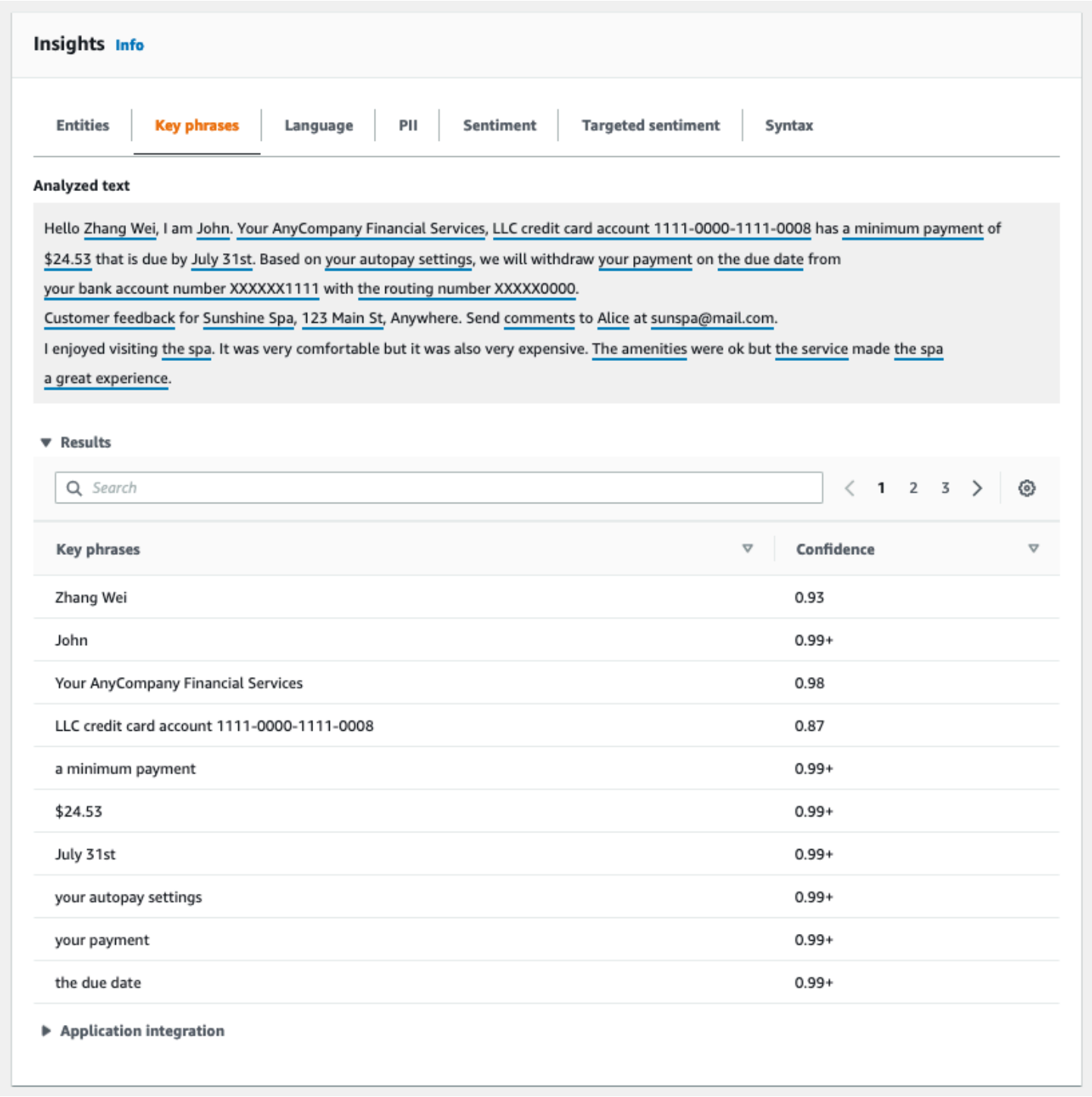
實體

「實體」索引標籤會列出每個實體、其類別，以及 Amazon Comprehend 在輸入文字中偵測到的可信度等級。結果會以顏色標示，以指出不同的實體類型，例如組織、地點、日期和人員。如需詳細資訊，請參閱 實體。



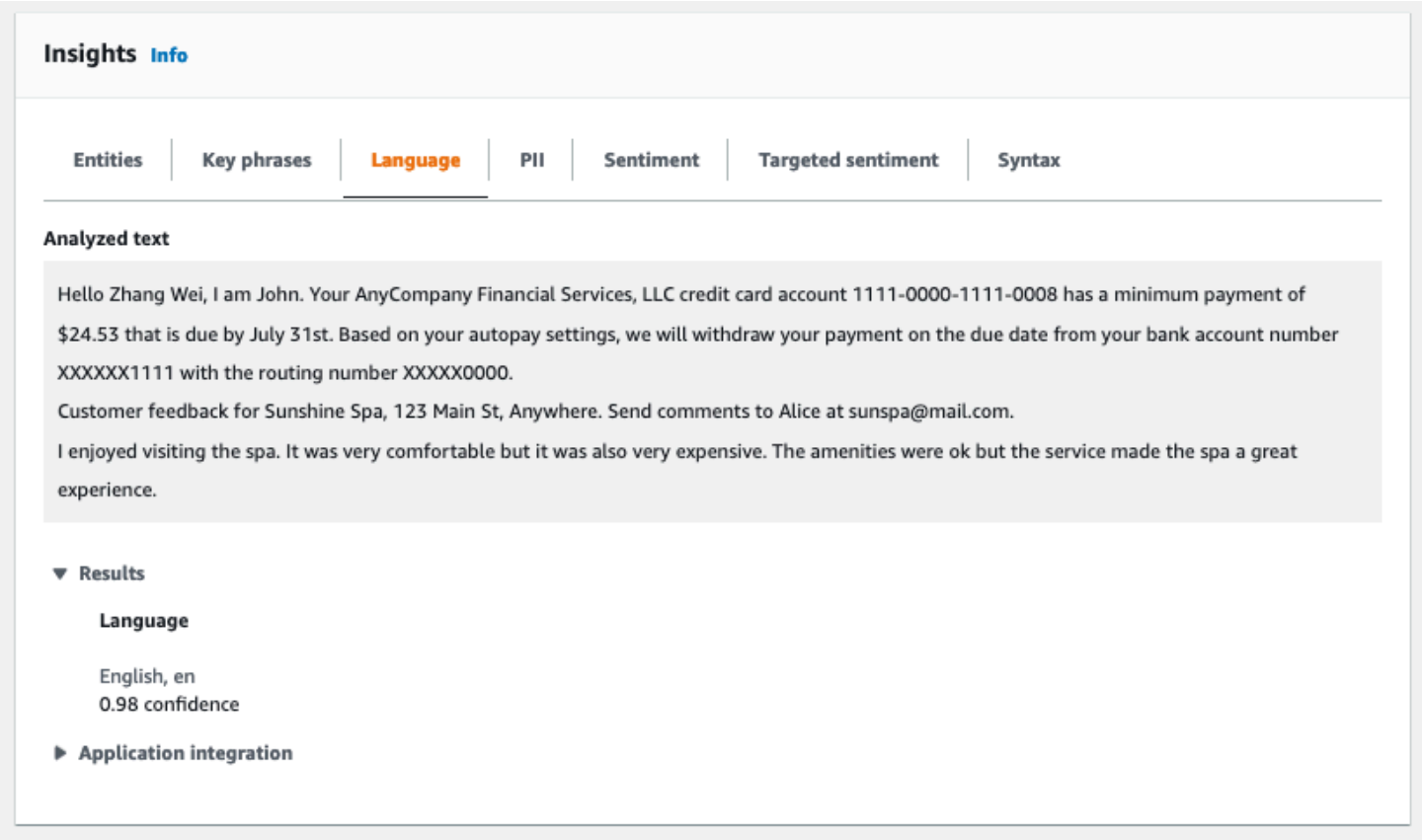
關鍵短語

[關鍵片語] 索引標籤會列出 Amazon Comprehend 在輸入文字中偵測到的關鍵名詞片語，以及相關聯的信賴等級。如需詳細資訊，請參閱 關鍵短語。



語言

[語言] 索引標籤會顯示文字的主要語言，以及 Amazon Comprehend 已正確偵測到主要語言的信心程度。Amazon Comprehend 可以識別 100 種語言。如需詳細資訊，請參閱 主要语言。



個人身分識別資訊 (PII)

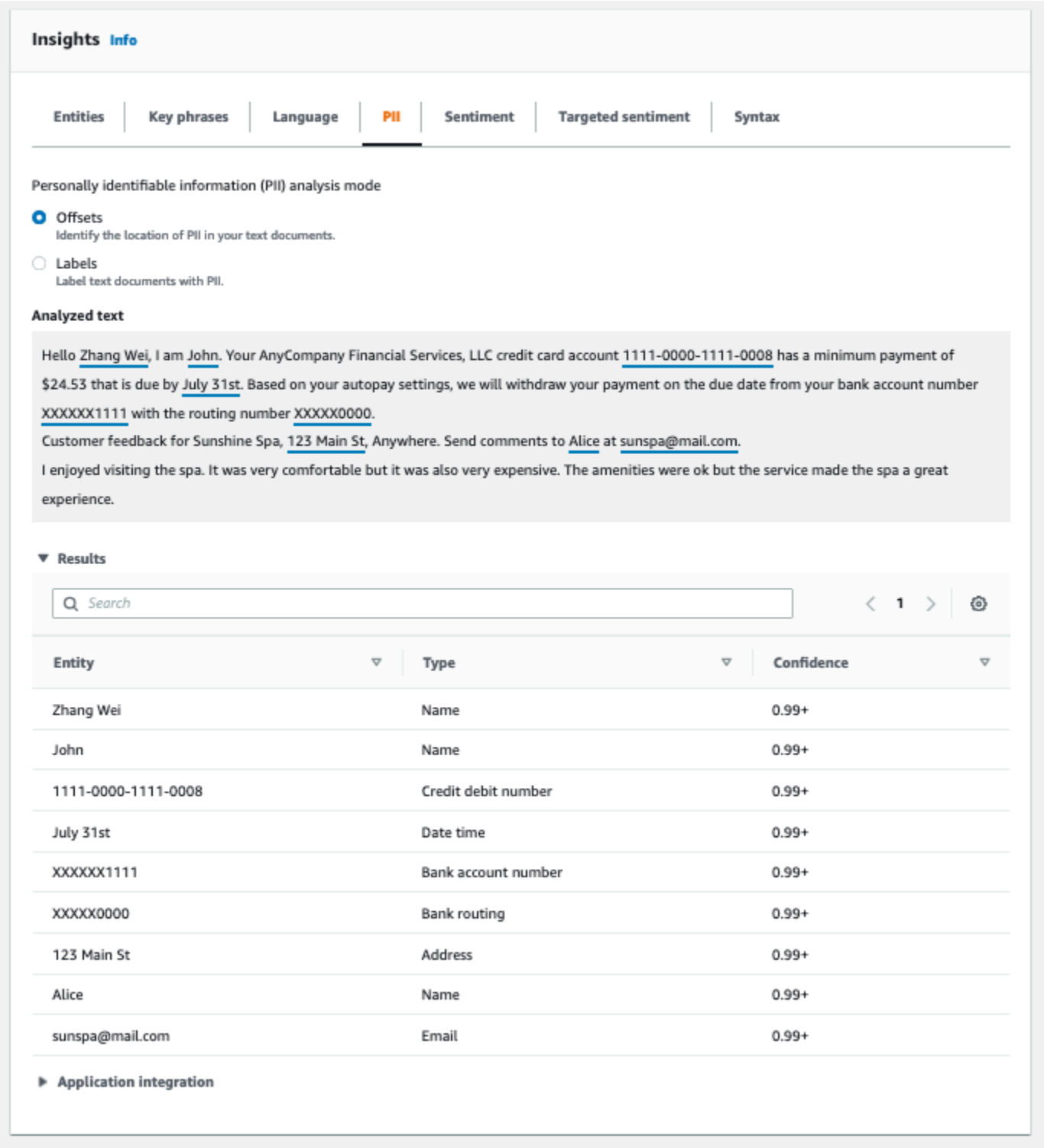
PII 索引標籤會列出輸入文字中包含個人識別資訊 (PII) 的實體。PII 實體是個人資料的文字參考，可用來識別個人身分，例如地址、銀行帳號或電話號碼。如需詳細資訊，請參閱 偵測 PII 實體。

P II 標籤提供兩種分析模式：

- 偏移
- 標籤

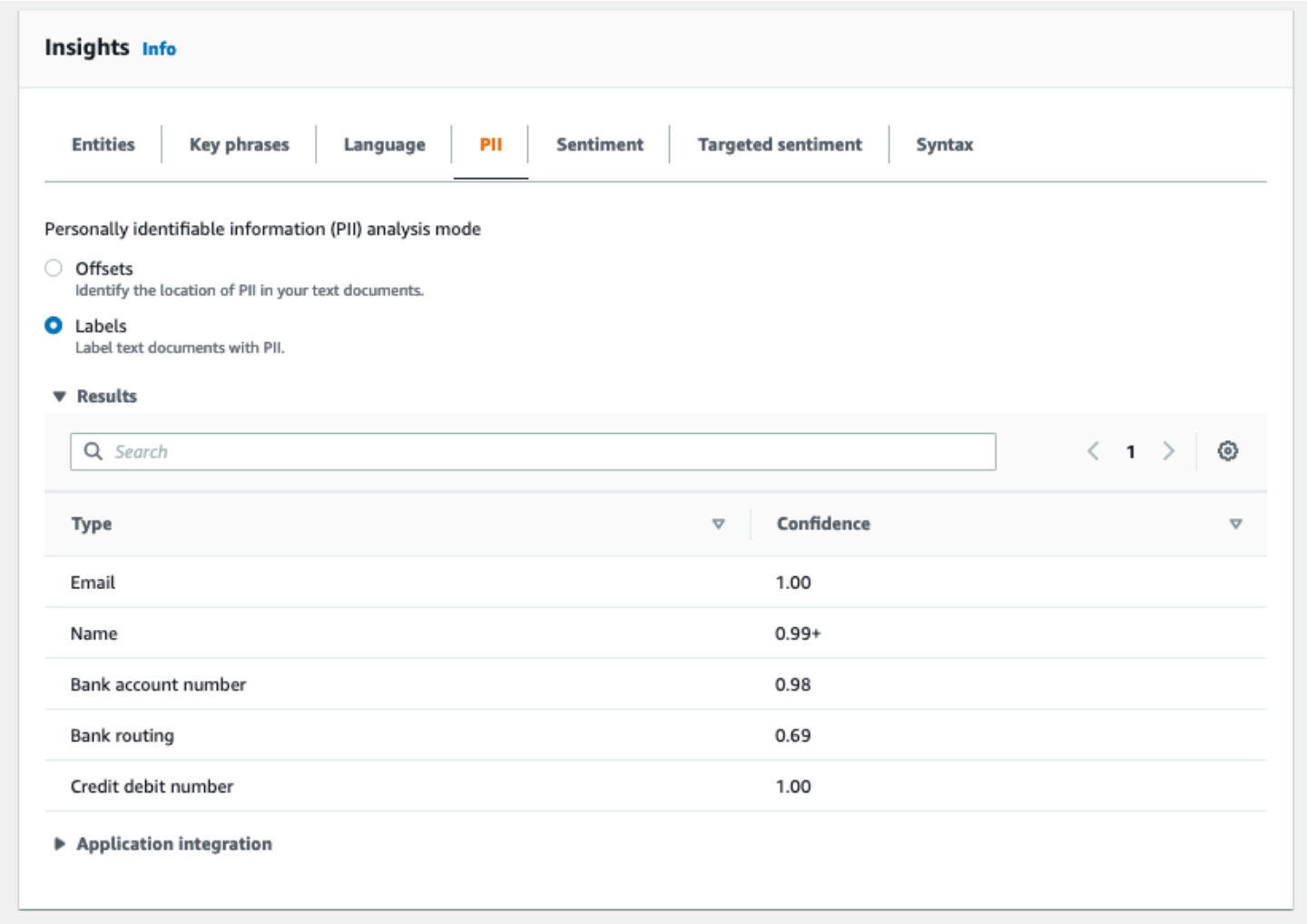
偏移

偏移分析模式可識別文字文件中 PII 的位置。如需詳細資訊，請參閱 尋找 PII 實體。



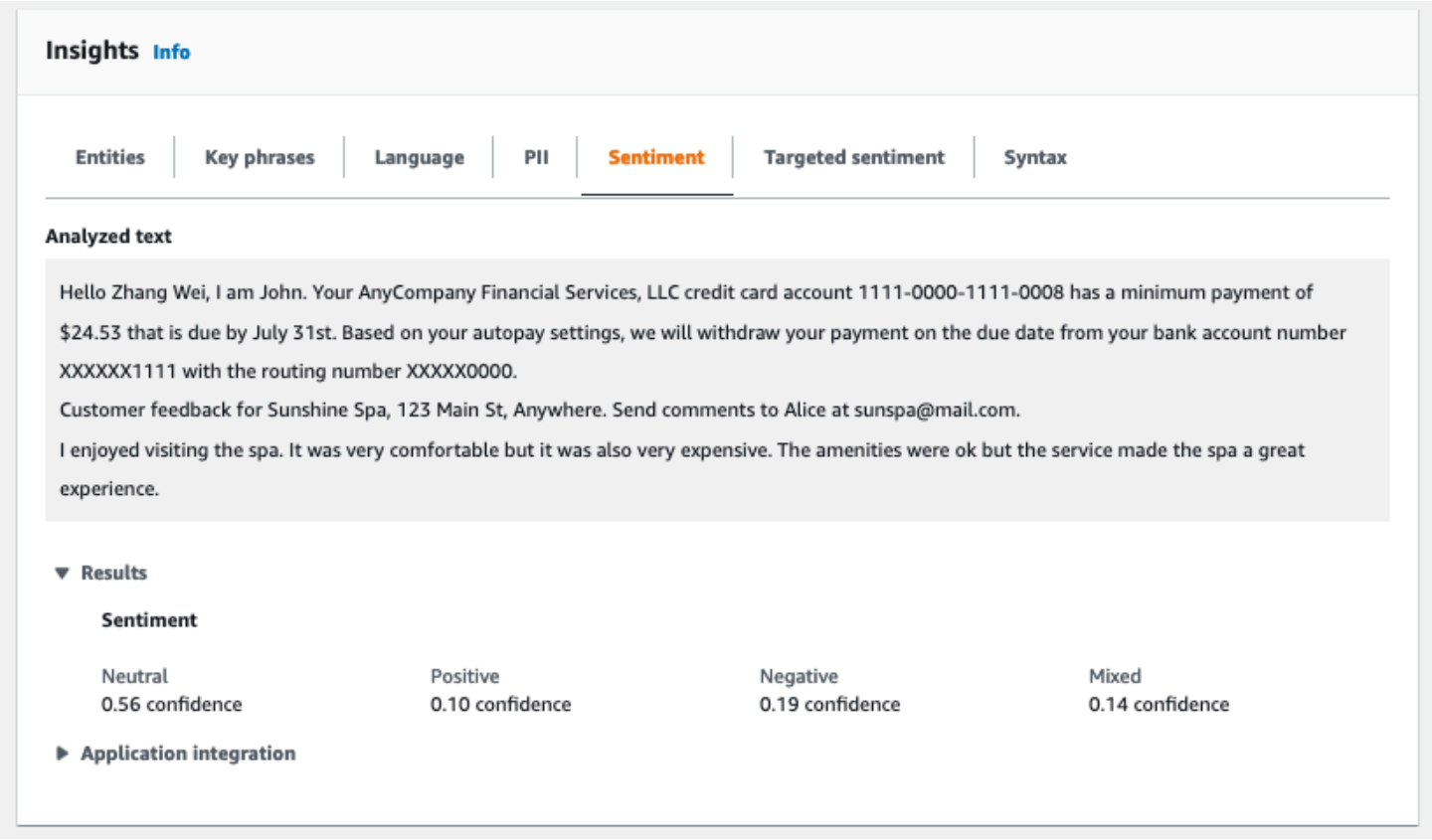
標籤

「標籤」分析模式會檢查文字文件中是否存在 PII，並傳回已識別之 PII 實體類型的標籤。如需詳細資訊，請參閱 為 PII 實體加上標籤。



情緒

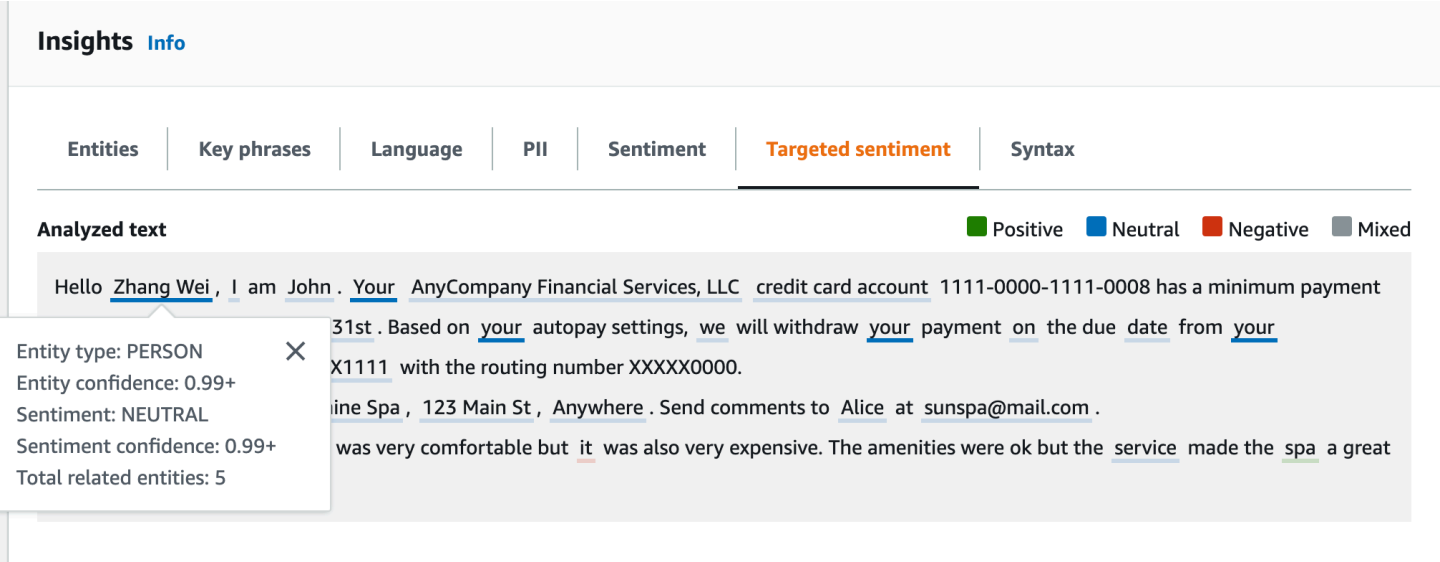
[情緒] 索引標籤會顯示文字的主要情緒。情緒可以被評為中性、正面、負面或混合。在這種情況下，每個情緒都有一個信心評級，提供 Amazon Comprehend 對該情緒佔主導地位的估計值。如需詳細資訊，請參閱 情緒。



目標情緒

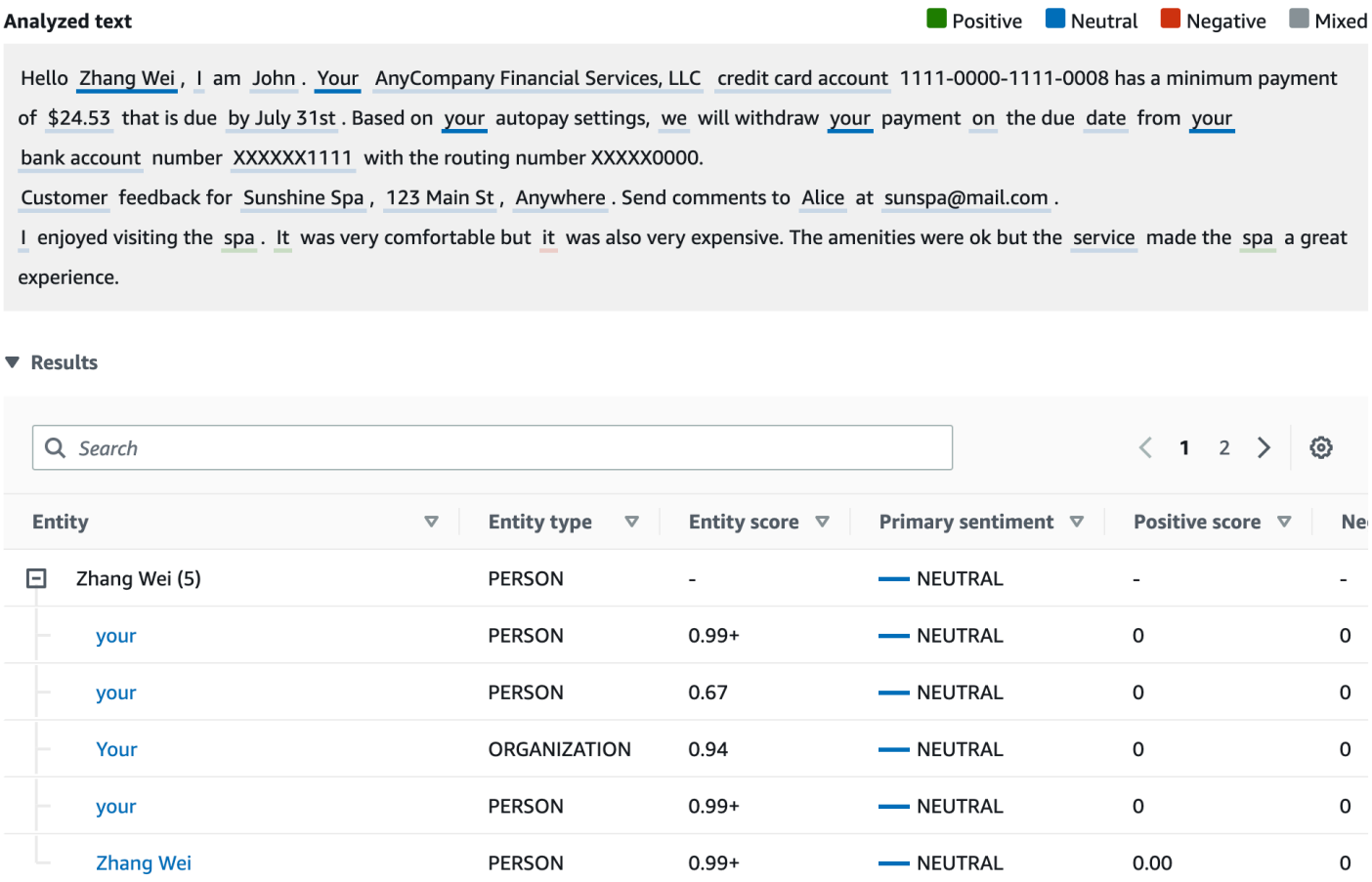
目標情緒分析可識別文字中所提到的實體所表達的情緒。Amazon Comprehend 會為實體的每個提及指派情緒分級，以及可信度評分和其他資訊。情緒分級可以是中性、正面、負面或混合。

在「已分析」文字面板中，主控台會為每個已分析的實體加上底線。加底線文字的顏色表示實體的整體情緒。如果您將游標停留在實體上，主控台會在快顯視窗中顯示其他資訊。



「結果」表格提供每個實體的其他詳細資訊。如果同一個實體有多個提及 (稱為共同參照群組)，表格會將這些提及顯示為與主實體相關聯的可摺疊資料列集。

在下面的例子中，實體是一個名為張偉的人。目標情緒分析會辨識出您的每一個提及都是對同一個人的參考。控制台將這些提及顯示為主實體的子條目。

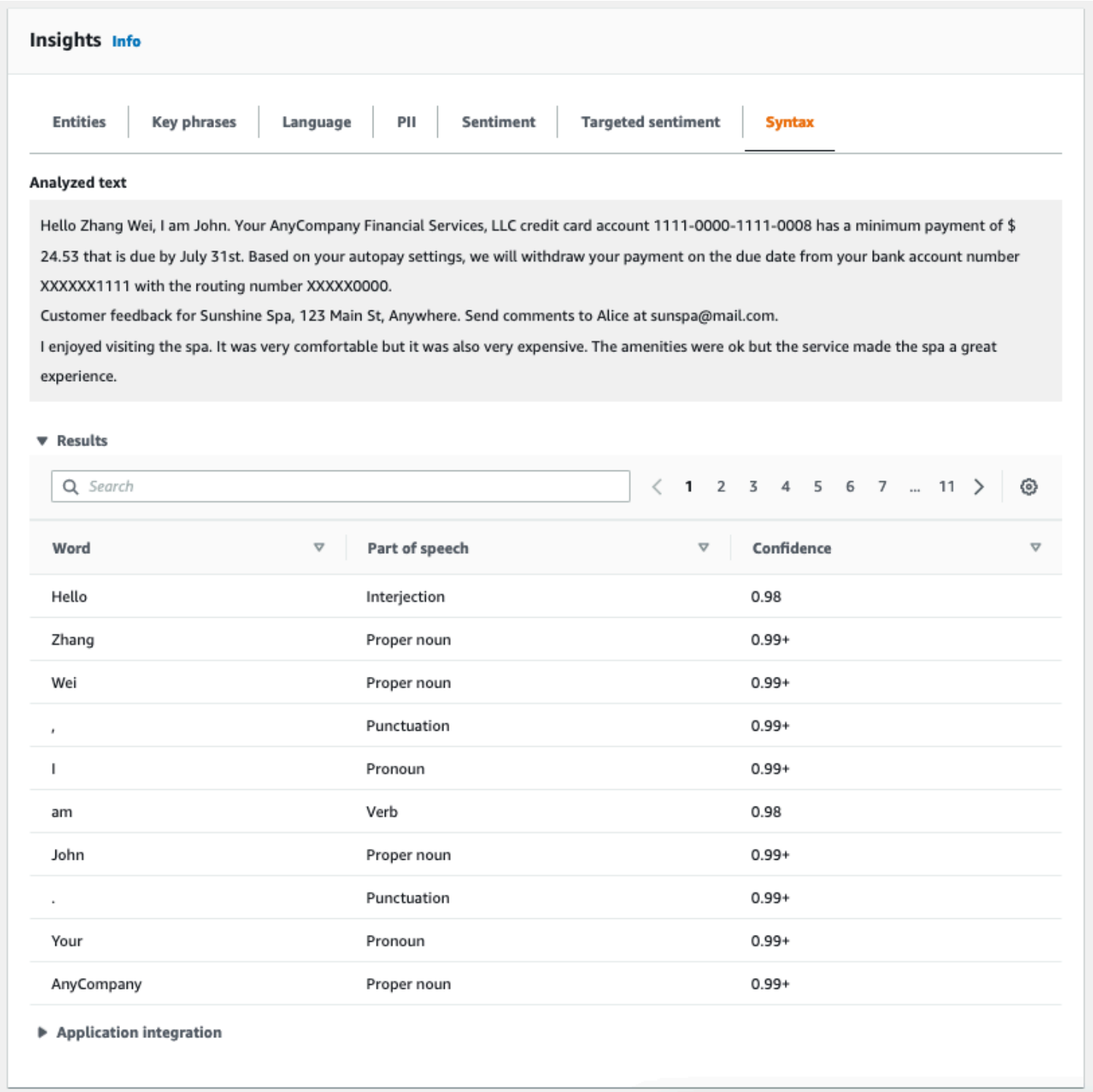


如果您正在分析的文字不包含任何目標情緒圖元類型，目標情緒分析會顯示空白的結果欄位。

如需如何使用主控台進行目標情緒即時分析的相關資訊，請參閱使用主控台進行即時分析。

語法

[語法] 索引標籤會顯示文字中每個元素的劃分，以及其語音部分和相關聯的置信度分數。如需詳細資訊，請參閱 語法分析。



使用主控台執行分析工作

您可以使用 Amazon Comprehend 主控台來建立和管理非同步分析任務。您的任務會分析存放在 Amazon S3 中的文件，以尋找實體，例如事件、片語、主要語言、情緒或個人識別資訊 (PII)。

建立分析工作

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [分析工作]，然後選擇 [建立工作]。
3. 在「Job 設定」 下，為分析工作指定一個唯一的名稱。
4. 針對「分析」類型，選擇其中一種內建分析類型。

   如果您選擇「主要語言」 或「主題模型」，則可以略過下一個步驟。
5. 根據您選擇的「分析」類型，主控台會顯示下列一或多個其他欄位：

   - 除了主要語言和主題模型以外，所有內建分析類型都需要語言。

     選擇輸入文件的語言。
   - 事件分析類型需要目標事件類型。

     選取要在輸入文件中偵測的事件類型。如需有關支援事件類型的詳細資訊，請參閱事件類型。
   - PII 分析類型需要 PII 偵測設定。

     選擇輸出模式。如需 PII 偵測設定的詳細資訊，請參閱偵測 PII 實體。
6. 在「輸入資料」下，指定輸入文件在 Amazon S3 中的位置：
   - 若要分析您自己的文件，請選擇 [我的文件]，然後選擇 [瀏覽 S3] 提供儲存貯體或包含檔案的資料夾路徑。
   - 若要分析 Amazon Comprehend 提供的範例，請選擇範例文件。在這種情況下，Amazon Comprehend 會使用由管理的儲存貯體 AWS，而您不會指定位置。
7. (選擇性) 對於「輸入格式」，請為輸入檔案指定下列其中一種格式：

   - 每個檔案一個文件 — 每個檔案包含一個輸入文件。這是最適合大型文檔的集合。
   - 每行一個文件 — 輸入為一個或多個檔案。檔案中的每一行都被視為一個文件。這最適合短文件，例如社交媒體張貼。每行必須以換行符（LF，\ n），回車符（CR，\ r）或兩者結束（CRLF，\ r\ n）。您不能使用 UTF-8 行分隔符（u+2028）來結束一行。
8. 在「輸出資料」下，選擇「瀏覽 S3」。選擇您希望 Amazon Comprehend 寫入分析產生的輸出資料的 Amazon S3 儲存貯體或資料夾。
9. (選擇性) 若要加密工作的輸出結果，請選擇加密。然後，選擇要使用與目前帳戶關聯的 KMS 金鑰，還是使用與其他帳戶相關聯的 KMS 金鑰：
   - 如果您使用與目前帳戶相關聯的金鑰，請為 KMS 金鑰 ID 選擇金鑰別名或 ID。

- 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ID 下輸入金鑰別名或 ID 的 ARN。

> **Note**
>
> 如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱金鑰管理服務 (KMS)。

10. 在「存取權限」下，提供 IAM 角色：

    - 授予對輸入文件之 Amazon S3 位置的讀取存取權限。
    - 授予對輸出文件之 Amazon S3 位置的寫入存取權。
    - 包含允許`comprehend.amazonaws.com`服務主體擔任角色並取得其權限的信任原則。

    如果您還沒有具有這些許可和適當信任政策的 IAM 角色，請選擇「建立 IAM 角色」來建立一個角色。
11. 填寫完表格後，請選擇 [建立工作] 以建立並啟動主題偵測工作。

新工作會顯示在工作清單中，而狀態欄位顯示工作的狀態。此欄位可`IN_PROGRESS`用於正在處理的工`COMPLETED`作、已成功完成的工作，以及`FAILED`發生錯誤的工作。您可以按一下工作以取得有關該工作的詳細資訊，包括任何錯誤訊息。

任務完成後，Amazon Comprehend 會將分析結果存放在您為該任務指定的輸出 Amazon S3 位置。如需每種見解類型之分析結果的說明，請參閱深入分析。

使用 Amazon Comprehend API

Amazon Comprehend API 支援執行即時 (同步) 分析和操作的操作，以啟動和管理非同步分析任務。

您可以直接使 Amazon Comprehend API 運算子，也可以使用 CLI 或其中一個開發套件。本章中的範例使用 CLI、Python 開發套件和 Java SDK。

若要執行AWS CLI和 Python 範例，您必須安裝AWS CLI. 如需詳細資訊，請參閱 設置AWS Command Line Interface（ AWS CLI ）。

若要執行 Java 範例，您必須安裝AWS SDK for Java. 如需有關安裝適用於 Java 的開發套件的指示，請參閱設定 AWS SDK for Java 發套件。

主題

使用 Amazon Comprehend 與 SDK AWS

AWS 軟體開發套件 (SDK) 適用於許多常用的程式設計語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	程式碼範例
AWS SDK for C++	AWS SDK for C++ 程式碼範例
AWS SDK for Go	AWS SDK for Go 程式碼範例
AWS SDK for Java	AWS SDK for Java 程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
AWS SDK for .NET	AWS SDK for .NET 程式碼範例
AWS SDK for PHP	AWS SDK for PHP 程式碼範例

SDK 文件	程式碼範例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 程式碼範例
AWS SDK for Ruby	AWS SDK for Ruby 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

可用性範例

找不到所需的內容嗎？ 請使用本頁面底部的提供意見回饋連結申請程式碼範例。

使用 API 進行即時分析

下列範例示範如何使用 Amazon Comprehend API 進行即時分析，使用和AWS開發套件 (適用於 .NET、Java 和 Python)。AWS CLI使用這些範例來了解 Amazon Comprehend 同步操作，以及做為您自己應用程式的建置區塊。

本節中的 .NET 範例使用 AWS SDK for .NET。您可以使用AWS Toolkit for Visual Studio來開發使用 .NET 的AWS應用程式。它包括實用的範本和AWS資源管理器，用於部署應用程式和管理服務。如需的 .NET 開發人員觀點AWS，請參閱 .NET 開發人員AWS指南。

主題

- 檢測主導語言
- 偵測具名實體
- 偵測關鍵片語
- 決定情緒
- 針對目標情緒的即時分析
- 偵測語法
- 即時批次處理 API

檢測主導語言

若要判斷文字中使用的主要語言，請使用此DetectDominantLanguage作業。若要在批次中偵測最多 25 個文件中的主要語言，請使用該BatchDetectDominantLanguage操作。如需詳細資訊，請參閱 即時批次處理 API。

主題

- 使用 AWS Command Line Interface
- 使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

使用 AWS Command Line Interface

下列範例示範如何使用`DetectDominantLanguage`作業與AWS CLI。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：

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

使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

如需如何判斷主要語言的 SDK 範例，請參閱使用軟體開發套件偵測文件中的主要語言 AWS。

偵測具名實體

若要決定文件中的具名實體，請使用此DetectEntities作業。若要在批次中偵測最多 25 個文件中的實體，請使用此BatchDetectEntities作業。如需詳細資訊，請參閱 即時批次處理 API。

主題

- 使用 AWS Command Line Interface
- 使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

使用 AWS Command Line Interface

下列範例示範如何使用`DetectEntities`操作AWS CLI。您必須指定輸入文字的語言。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：

```
{
    "Entities": [
        {
            "Text": "today",
            "Score": 0.97,
            "Type": "DATE",
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
            "Score": 0.95,
            "Type": "LOCATION",
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ],
    "LanguageCode": "en"
}
```

使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

如需如何判斷主要語言的 SDK 範例，請參閱使用開發套件偵測文件中的實體 AWS。

偵測關鍵片語

若要決定文字中使用的關鍵名詞片語，請使用此DetectKeyPhrases操作。若要在批次中偵測最多 25 個文件中的關鍵名詞片語，請使用該BatchDetectKeyPhrases操作。如需詳細資訊，請參閱 即時批次處理 API。

主題

- 使用 AWS Command Line Interface
- 使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

使用 AWS Command Line Interface

下列範例示範如何使用`DetectKeyPhrases`作業與AWS CLI。您必須指定輸入文字的語言。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：

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

```
}
```

使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

如需偵測關鍵片語的 SDK 範例，請參閱使用軟體開發套件偵測文件中的關鍵片語 AWS。

決定情緒

Amazon Comprehend 提供下列 API 操作來分析情緒：

- DetectSentiment— 決定文件的整體情緒。
- BatchDetectSentiment— 判斷一批最多 25 個文件中的整體情緒。如需更多資訊，請參閱 即時批次處理 API
- StartSentimentDetectionJob— 啟動文件集合的非同步情緒偵測工作。
- ListSentimentDetectionJobs— 傳回您已提交的情緒偵測工作清單。
- DescribeSentimentDetectionJob— 取得與指定情緒偵測工作相關聯的內容 (包括狀態)。
- StopSentimentDetectionJob— 停止指定的進行中情緒工作。

主題

- 使用 AWS Command Line Interface
- 使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

使用 AWS Command Line Interface

下列範例示範如何使用`DetectSentiment`作業與AWS CLI。這個例子指定了輸入文本的語言。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：

```
{
    "SentimentScore": {
```

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

使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

如需決定輸入文字情緒的 SDK 範例，請參閱使用開發套件使用 Amazon Comprehend 文件偵測情緒 AWS。

針對目標情緒的即時分析

Amazon Comprehend 提供下列 API 操作，以進行目標情緒即時分析：

- DetectTargetedSentiment— 分析文件中提到的實體的情緒。
- BatchDetectTargetedSentiment— 批次分析多達 25 份文件的目標情緒。如需更多資訊，請參閱 即時批次處理 API

如果您正在分析的文字不包含任何目標情緒圖元類型，API 會傳回空白的實體陣列。

使用 AWS Command Line Interface

下列範例示範如何使用`DetectTargetedSentiment`作業與AWS CLI。這個例子指定了輸入文本的語言。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：

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

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

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

偵測語法

若要剖析文字以擷取個別單字，並決定每個單字的語音部分，請使用此DetectSyntax作業。若要剖析批次中最多 25 個文件的語法，請使用此BatchDetectSyntax作業。如需詳細資訊，請參閱 即時批次處理 API。

主題

- 使用AWS Command Line Interface。
- 使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

使用AWS Command Line Interface。

下列範例示範如何使用`DetectSyntax`作業與AWS CLI。這個例子指定了輸入文本的語言。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：


```
{
    "SyntaxTokens": [
        {
            "Text": "It",
            "EndOffset": 2,
            "BeginOffset": 0,
            "PartOfSpeech": {
                "Tag": "PRON",
                "Score": 0.8389829397201538
            },
            "TokenId": 1
        },
        {
            "Text": "is",
            "EndOffset": 5,
            "BeginOffset": 3,
            "PartOfSpeech": {
                "Tag": "AUX",
                "Score": 0.9189288020133972
            },
            "TokenId": 2
        },
        {
            "Text": "raining",
            "EndOffset": 13,
            "BeginOffset": 6,
            "PartOfSpeech": {
                "Tag": "VERB",
                "Score": 0.9977611303329468
            },
            "TokenId": 3
        },
        {
            "Text": "today",
            "EndOffset": 19,
            "BeginOffset": 14,
```

```
            "PartOfSpeech": {
                "Tag": "NOUN",
                "Score": 0.9993606209754944
            },
            "TokenId": 4
        },
        {
            "Text": "in",
            "EndOffset": 22,
            "BeginOffset": 20,
            "PartOfSpeech": {
                "Tag": "ADP",
                "Score": 0.9999061822891235
            },
            "TokenId": 5
        },
        {
            "Text": "Seattle",
            "EndOffset": 30,
            "BeginOffset": 23,
            "PartOfSpeech": {
                "Tag": "PROPN",
                "Score": 0.9940338730812073
            },
            "TokenId": 6
        },
        {
            "Text": ".",
            "EndOffset": 31,
            "BeginOffset": 30,
            "PartOfSpeech": {
                "Tag": "PUNCT",
                "Score": 0.9999997615814209
            },
            "TokenId": 7
        }
    ]
}
```

使用AWS SDK for Java, 適用於 Python 的軟體開發套件，或 AWS SDK for .NET

如需偵測輸入文字語法的 SDK 範例，請參閱使用開發套件偵測使用 Amazon Comprehend 文件的語法元素 AWS。

即時批次處理 API

若要傳送最多 25 份文件的批次，您可以使用 Amazon Comprehend 即時批次操作。呼叫批次作業與呼叫要求中每個文件的單一文件 API 相同。使用批次 API 可以為您的應用程式帶來更好的效能。如需詳細資訊，請參閱 多文檔同步處理。

主題

- 使用 Batch 處理 AWS CLI
- 使用 Batch 處理 AWS SDK for .NET

使用 Batch 處理 AWS CLI

這些範例說明如何使用AWS Command Line Interface. 除了`BatchDetectDominantLanguage`使用下面的 JSON 文件稱為`process.json`輸入的所有操作。對於該操作，`LanguageCode`實體不包括在內。

JSON 檔案 ("$$$$$$$$") 中的第三個文件會在批次處理期間造成錯誤。它包括在內，以便操作將包括BatchItemError在響應中。

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

這些範例已針對 Unix、Linux 和 macOS 進行格式化。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

主題

- 使用 batch（AWS CLI）檢測主要語言
- 使用批次偵測實體 (AWS CLI)
- 使用批處理（AWS CLI）檢測關鍵短語
- 使用批次偵測情緒 (AWS CLI)

使用 batch（AWS CLI）檢測主要語言

此BatchDetectDominantLanguage作業會決定批次中每個文件的主要語言。如需 Amazon Comprehend 可偵測到的語言清單，請參閱。主要语言下列AWS CLI命令會呼叫BatchDetectDominantLanguage作業。

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

以下是來自BatchDetectDominantLanguage操作的響應：

```
{
    "ResultList": [
        {
          "Index": 0,
          "Languages":[
            {
              "LanguageCode":"en",
              "Score": 0.99
            }
          ]
        },
        {
          "Index": 1
          "Languages":[
            {
              "LanguageCode":"en",
              "Score": 0.82
            }
          ]
        }
    ],
    "ErrorList": [
      {
        "Index": 2,
        "ErrorCode": "InternalServerException",
        "ErrorMessage": "Unexpected Server Error. Please try again."
      }
    ]
}
```

使用批次偵測實體 (AWS CLI)

使用此BatchDetectEntities作業尋找存在於批次文件中的實體。如需實體的詳細資訊，請參閱實體。下列AWS CLI命令會呼叫BatchDetectEntities作業。

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

使用批處理（AWS CLI）檢測關鍵短語

該BatchDetectKeyPhrases操作返回一批文檔中的關鍵名詞短語。下列AWS CLI命令會呼叫BatchDetectKeyNounPhrases作業。

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

使用批次偵測情緒 (AWS CLI)

使用操作偵測批次文件的整體情BatchDetectSentiment緒。下列AWS CLI命令會呼叫BatchDetectSentiment作業。

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

使用 Batch 處理 AWS SDK for .NET

下列範例程式展示如何搭配使用此BatchDetectEntities作業AWS SDK for .NET。來自服務器的響應包含每個成功處理的文檔的BatchDetectEntitiesItemResult對象。如果處理文件時發生錯誤，回應中的錯誤清單中會有記錄。該示例獲取每個帶有錯誤的文檔並重新發送它們。

本節中的 .NET 範例使用 AWS SDK for .NET. 您可以使用AWS Toolkit for Visual Studio來開發使用 .NET 的AWS應用程式。它包括實用的範本和AWS資源管理器，用於部署應用程式和管理服務。如需的 .NET 開發人員觀點AWS，請參閱 .NET 開發人員AWS指南。

```
using System;
```

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

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

        static void Main(string[] args)
        {
            AmazonComprehendClient comprehendClient = new
 AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

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

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

            foreach (BatchDetectEntitiesItemResult item in
 batchDetectEntitiesResponse.ResultList)
            {
```

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

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

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

                batchDetectEntitiesResponse =
 comprehendClient.BatchDetectEntities(batchDetectEntitiesRequest);

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

使用 API 的非同步分析工作

下列範例使用 Amazon Comprehend 非同步 API 來建立和管理分析任務，並使用. AWS CLI

主題

- Amazon Comprehend 的非同步分析

- 針對目標情緒的非同步分析
- 事件偵測的非同步分析
- 主題建模的非同步分析

Amazon Comprehend 的非同步分析

以下各節使用 Amazon Comprehend API 執行非同步操作，以分析 Amazon Comprehend。

主題

- 必要條件
- 開始分析工作
- 監視分析工作
- 取得分析結果

必要條件

文件必須是 UTF-8 格式的文字檔案。您可以使用兩種格式提交文件。您使用的格式取決於您要分析的文件類型，如下表所述。

描述	格式
每個檔案都包含一個輸入文件。這是最適合大型文檔的集合。	每個檔案一個文件
輸入是一個或多個文件。檔案中的每一行都被視為一個文件。這最適合短文件，例如社交媒體張貼。 每行必須以換行符（LF，\ n），回車符（CR，\ r）或兩者結束（CRLF，\ r\ n）。您不能使用 UTF-8 行分隔符（u+2028）來結束一行。	每行一個文件

開始分析任務時，您可以指定輸入資料的 S3 位置。URI 必須與您呼叫的 API 端點位於相同的 AWS區域。URI 可以指向單個文件，也可以是數據文件集合的前綴。如需詳細資訊，請參閱資InputDataConfig料類型。

您必須將包含文件收集和輸出檔案的 Amazon S3 儲存貯體授與 Amazon Comprehend 存取權。如需詳細資訊，請參閱 非同步作業所需的角色型權限。

開始分析工作

若要提交分析任務，請使用 Amazon Comprehend 主控台或適當Start*的操作：

- StartDominantLanguageDetectionJob— 啟動工作以檢測集合中每個文檔中的主要語言。如需文件中主要語言的詳細資訊，請參閱主要语言。
- StartEntitiesDetectionJob— 啟動工作以偵測集合中每個文件中的實體。如需實體的詳細資訊，請參閱實體。
- StartKeyPhrasesDetectionJob— 啟動工作以檢測集合中每個文檔中的關鍵短語。如需關鍵片語的詳細資訊，請參閱關鍵短語。
- StartPiiEntitiesDetectionJob— 開始一項工作以檢測集合中的每個文檔中的個人身份信息（PII）。如需 PII 的詳細資訊，請參閱偵測 PII 實體。
- StartSentimentDetectionJob— 啟動工作以偵測集合中每個文件中的情緒。如需情緒的詳細資訊，請參閱情緒。

監視分析工作

作Start*業會傳回可用來監視工作進度的 ID。

若要使用 API 監視進度，請根據您要監視個別工作的進度還是多個工作的進度，使用兩種作業的其中一種。

若要監視個別分析工作的進度，請使用作Describe*業。您可以提供作業傳回的工Start*作 ID。Describe*作業的回應包含具有工作狀態的JobStatus欄位。

若要監視多個分析工作的進度，請使用這些作List*業。 List*作業會傳回您提交給亞馬遜的任務清單。回應包含每個工作的JobStatus欄位，告訴您工作的狀態。

如果狀態欄位設定為COMPLETED或FAILED，表示工作處理已完成。

若要取得個別工Describe*作的狀態，請使用您正在執行的分析作業。

- DescribeDominantLanguageDetectionJob
- DescribeEntitiesDetectionJob
- DescribeKeyPhrasesDetectionJob
- DescribePiiEntitiesDetectionJob

- DescribeSentimentDetectionJob

若要取得多個工List*作的狀態，請使用您正在執行的分析的作業。

- ListDominantLanguageDetectionJobs
- ListEntitiesDetectionJobs
- ListKeyPhrasesDetectionJobs
- ListPiiEntitiesDetectionJobs
- ListSentimentDetectionJobs

若要將結果限制為符合特定條件的工單，請使用作List*業的參數Filter。您可以篩選工作名稱、工作狀態以及提交工作的日期和時間。如需詳細資訊，請Filter參閱 Amazon Comprehend API 參考資料中每個List*作業的參數。

取得分析結果

分析工作完成後，使用Describe*作業取得結果的位置。如果任務狀態為COMPLETED，則回應會包含一個OutputDataConfig欄位，其中包含具有輸出檔案 Amazon S3 位置的欄位。檔案是output.tar.gz包含分析結果的壓縮歸檔。

如果工作的狀態為FAILED，則回應會包含描述分析工作未順利完成原因的Message欄位。

若要取得個別工作的狀態，請使用適當的Describe*作業：

- DescribeDominantLanguageDetectionJob
- DescribeEntitiesDetectionJob
- DescribeKeyPhrasesDetectionJob
- DescribeSentimentDetectionJob

結果會以單一檔案傳回，每個文件都有一個 JSON 結構。每個回應檔案也包含狀態欄位設定為的任何工作的錯誤訊息FAILED。

以下各節顯示兩種輸入格式的輸出範例。

取得主要語言偵測結果

以下是偵測到主要語言之分析的輸出檔案範例。輸入的格式是每行一個文檔。有關更多資訊，請參閱 DetectDominantLanguage 操作。

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

以下是分析輸出的範例，其中輸入的格式為每個檔案一個文件：

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

取得實體偵測結果

以下是分析中偵測到文件中圖元的輸出檔案範例。輸入的格式是每行一個文檔。有關更多資訊，請參閱 DetectEntities 操作。輸出包含兩個錯誤訊息，一個用於太長的文件，另一個用於不是 UTF-8 格式的文件。

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

以下是分析輸出的範例，其中輸入的格式為每個檔案一個文件。輸出包含兩個錯誤訊息，一個用於太長的文件，另一個用於不是 UTF-8 格式的文件。

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

取得關鍵片語偵測結果

以下是分析中偵測到文件中關鍵片語的輸出檔案範例。輸入的格式是每行一個文檔。有關更多資訊，請參閱 DetectKeyPhrases 操作。

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

以下是分析輸出的範例，其中輸入的格式為每個檔案一個文件。

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

取得個人識別資訊 (PII) 偵測結果

以下是分析工作中偵測到文件中 PII 實體的輸出檔案範例。輸入的格式是每行一個文檔。

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

以下是分析輸出的範例，其中輸入的格式為每個檔案一個文件。

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

取得情緒偵測結果

以下是來自分析的輸出檔案範例，此分析會偵測到文件中所表達的情緒。它包含錯誤訊息，因為一個文件太長。輸入的格式是每行一個文檔。有關更多資訊，請參閱 DetectSentiment 操作。

```
{"File": "50_docs", "Line": 0, "Sentiment": "NEUTRAL", "SentimentScore": {"Mixed":
 0.002734508365392685, "Negative": 0.008935936726629734, "Neutral": 0.9841893315315247,
 "Positive": 0.004140198230743408}}
```

```
{"File": "50_docs", "Line": 1, "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage":
 "Document size is exceeded maximum size limit 5120 bytes."}
{"File": "50_docs", "Line": 2, "Sentiment": "NEUTRAL", "SentimentScore":
 {"Mixed": 0.0023119584657251835, "Negative": 0.0029857370536774397, "Neutral":
 0.9866572022438049, "Positive": 0.008045154623687267}}
```

以下是分析輸出的範例，其中輸入的格式為每個檔案一個文件。

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

針對目標情緒的非同步分析

如需目標情緒的即時分析的相關資訊，請參閱the section called “針對目標情緒的即時分析”。

Amazon Comprehend 提供下列 API 操作來啟動和管理非同步目標情緒分析：

- StartTargetedSentimentDetectionJob— 啟動文件集合的非同步目標情緒偵測工作。
- ListTargetedSentimentDetectionJobs— 傳回您已提交的目標情緒偵測工作清單。
- DescribeTargetedSentimentDetectionJob— 取得與指定目標情緒偵測工作相關聯的內容 (包括狀態)。
- StopTargetedSentimentDetectionJob— 停止指定的進行中目標情緒工作。

主題

- 開始之前
- 使用分析目標情緒 AWS CLI

開始之前

在開始之前，請確保您擁有：

- 輸入和輸出儲存貯體 — 識別要用於輸入和輸出的 Amazon S3 儲存貯體。值區必須與您呼叫的 API 位於相同的區域。
- IAM 服務角色 — 您必須擁有具有存取輸入和輸出值區之權限的 IAM 服務角色。如需詳細資訊，請參閱 非同步作業所需的角色型權限。

使用分析目標情緒 AWS CLI

下列範例示範如何使用StartTargetedSentimentDetectionJob作業與AWS CLI。這個例子指定了輸入文本的語言。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

對於cli-input-json參數，您為包含請求資料的 JSON 檔案提供路徑，如下列範例所示。

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

如果啟動工作的要求成功，您會收到下列回應：

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

事件偵測的非同步分析

主題

- 使用描述事件 AWS CLI
- 取得事件偵測結果

若要偵測文件集中的事件，請使StartEventsDetectionJob用啟動非同步工作。

開始之前

在開始之前，請確保您擁有：

- 輸入和輸出儲存貯體 — 識別要用於輸入和輸出的 Amazon S3 儲存貯體。值區必須與您呼叫的 API 位於相同的區域。
- IAM 服務角色 — 您必須擁有具有存取輸入和輸出值區之權限的 IAM 服務角色。如需詳細資訊，請參閱 非同步作業所需的角色型權限。

使用偵測事件 AWS CLI

下面的實例演示了使用StartEventsDetectionJob操作與 AWS CLI

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

對於cli-input-json參數，您為包含請求資料的 JSON 檔案提供路徑，如下列範例所示。

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

```
      "EMPLOYMENT",
      "CORPORATE_ACQUISITION",
      "INVESTMENT_GENERAL",
      "CORPORATE_MERGER",
      "IPO",
      "RIGHTS_ISSUE",
      "SECONDARY_OFFERING",
      "SHELF_OFFERING",
      "TENDER_OFFERING",
      "STOCK_SPLIT"
  ]
}
```

如果啟動事件偵測工作的要求成功，您將會收到下列回應：

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

使用列出事件 AWS CLI

使用此ListEventsDetectionJobs作業可查看您已提交的事件偵測工作清單。此清單包含您使用的輸入和輸出位置以及每個偵測工作狀態的相關資訊。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

您將獲得類似於以下內容的 JSON 作為回應：

```
{
 "EventsDetectionJobPropertiesList": [
    {
       "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role",
       "EndTime": timestamp,
       "InputDataConfig": {
          "InputFormat": "ONE_DOC_PER_LINE",
          "S3Uri": "s3://input bucket/input path"
       },
       "JobId": "job ID",
       "JobName": "job name",
```

```
        "JobStatus": "COMPLETED",
        "LanguageCode": "en",
        "Message": "message",
        "OutputDataConfig": {
           "S3Uri": "s3://output bucket/ouput path"
        },
        "SubmitTime": timestamp,
        "TargetEventTypes": [
          "BANKRUPTCY",
          "EMPLOYMENT",
          "CORPORATE_ACQUISITION",
          "INVESTMENT_GENERAL",
          "CORPORATE_MERGER",
          "IPO",
          "RIGHTS_ISSUE",
          "SECONDARY_OFFERING",
          "SHELF_OFFERING",
          "TENDER_OFFERING",
          "STOCK_SPLIT"
   ]
     }
 ],
 "NextToken": "next token"
}
```

使用描述事件 AWS CLI

您可以使用此DescribeEventsDetectionJob作業來取得現有工作的狀態。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

您將獲得以下 JSON 作為響應：

```
{
 "EventsDetectionJobProperties": {
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role",
    "EndTime": timestamp,
```

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

取得事件偵測結果

以下是分析工作中偵測到文件中事件的輸出檔案範例。輸入的格式是每行一個文檔。

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

```
"CORPORATE_ACQUISITION"}, {"Arguments": [{"EntityIndex": 2, "Role": "PARTICIPANT",
"Score": 0.999283}], "Triggers": [{"BeginOffset": 115, "EndOffset": 123, "GroupScore":
1.0, "Score": 0.999967, "Text": "combined", "Type": "CORPORATE_MERGER"}], "Type":
"CORPORATE_MERGER"}], "File": "doc.txt", "Line": 0}
```

如需有關事件輸出檔案結構和支援的事件類型的詳細資訊，請參閱事件。

主題建模的非同步分析

若要確定文件集中的主題，請使StartTopicsDetectionJob用啟動非同步工作。您可以監控以英文或西班牙文撰寫的文件中的主題。

主題

- 開始之前
- 使用 AWS Command Line Interface
- 使用開發套件進行 Python 或 AWS SDK for .NET

開始之前

在開始之前，請確保您擁有：

- 輸入和輸出儲存貯體 — 識別要用於輸入和輸出的 Amazon S3 儲存貯體。值區必須與您呼叫的 API 位於相同的區域。
- IAM 服務角色 — 您必須擁有具有存取輸入和輸出值區之權限的 IAM 服務角色。如需詳細資訊，請參閱 非同步作業所需的角色型權限。

使用 AWS Command Line Interface

下面的實例演示了使用`StartTopicsDetectionJob`操作與 AWS CLI

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
aws comprehend start-topics-detection-job \
                --number-of-topics topics to return \
                --job-name "job name" \
                --region region \
```

```
    --cli-input-json file://path to JSON input file
```

對於cli-input-json參數，您為包含請求資料的 JSON 檔案提供路徑，如下列範例所示。

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

如果啟動主題偵測工作的要求成功，您將會收到下列回應：

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

使用此ListTopicsDetectionJobs作業可查看您已提交的主題偵測工作清單。此清單包含您使用的輸入和輸出位置以及每個偵測工作狀態的相關資訊。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

您將獲得類似於以下內容的 JSON 作為回應：

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

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

您可以使用此DescribeTopicsDetectionJob作業來取得現有工作的狀態。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

您將獲得以下 JSON 作為響應：

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

```
        "OutputDataConfig": {
            "S3Uri": "s3://output bucket/ouput path"
        },
        "EndTime": timestamp
    }
}
```

使用開發套件進行 Python 或 AWS SDK for .NET

如需如何開始主題建模工作的 SDK 範例，請參閱使用開發套件啟動 Amazon Comprehend 主題建模任務 AWS。

信任與安全

使用者可透過線上應用程式 (例如 peer-to-peer 聊天室和論壇討論)、網站上張貼的留言，以及生成 AI 應用程式 (生成 AI 模型的輸入提示和輸出)，產生大量的文字內容。Amazon Comprehend 信任與安全功能可協助您協調此內容，為您的使用者提供安全且包容的環境。

使用亞 Amazon Comprehend 信任和安全功能的好處包括：

- 更快的審核：快速準確地調節大量文本，以使您的在線平台免受不適當內容的侵害。
- 可自訂：自訂 API 回應中的協調閾值，以符合您的應用程式需求。
- 易於使用：透過 LangChain 整合或使用 AWS CLI 或 SDK 來設定信任和安全功能。

Amazon Comprehend 信任與安全解決了內容審核的下列各個方面：

- Toxicity detection— 檢測可能有害，令人反感或不適當的內容。例子包括仇恨言論，威脅或虐待。
- Intent classification— 檢測具有明確或隱含惡意意圖的內容。例子包括歧視性或非法內容，或表達或要求就醫療、法律、政治、爭議、個人或財務主體提供意見的內容。
- Privacy protection— 用戶可能無意中提供可能洩露個人身份信息（PII）的內容。Amazon Comprehend PII 提供偵測和編輯 PII 的功能。

主題

毒性檢測

Amazon Comprehend 毒性偵測可讓您即時偵測文字互動中的有毒內容。您可以使用毒性偵測來中度線上平台中的 peer-to-peer 對話，或監控生成 AI 輸入和輸出。

毒性檢測可檢測以下類別的冒犯性內容：

GRAPHIC

圖形語音使用視覺上描述性，詳細和令人不愉快的生動圖像。這種語言往往是冗長的，以放大對收件人的侮辱，不適或傷害。

騷擾或濫用

無論意圖如何，在演講者和聽者之間施加破壞性力量動態的語音都旨在影響收件人的心理健康，或者客觀化一個人。

仇恨演講

在身份的基礎上批評，侮辱，譴責或使某個人或團體不人道化的演講，無論是種族，種族，性別認同，宗教，性取向，能力，國籍還是其他身份組。

侮辱

包括貶低、羞辱、嘲笑、侮辱或貶低語言的言語。

褻瀆

包含不禮貌，粗俗或冒犯性的單詞，短語或首字母縮略詞的語音被視為褻瀆。

性

通過直接或間接引用身體部位或身體特徵或性別來表示性興趣，活動或喚醒的語音。

暴力或威脅

包括試圖對個人或團體造成痛苦，傷害或敵意的威脅的言論。

毒性

包含在上述任何類別中可能被認為是有毒的單詞，短語或首字母縮略詞的語音。

使用 API 偵測有毒內容

若要偵測文字中的有毒內容，請使用同步DetectToxicContent操作。此作業會對您提供做為輸入的文字字串清單執行分析。API 回應包含符合輸入清單大小的結果清單。

目前，有毒內容檢測僅支持英語。對於輸入文字，您最多可以提供 10 個文字字串的清單。每個字串的大小上限為 1KB。

有毒含量偵測會傳回分析結果清單，每個輸入字串都會在清單中傳回一個項目。項目包含文字字串中識別的有毒內容類型清單，以及每個內容類型的可信度分數。該項目還包括字符串的毒性分數。

下面的實例演示了如何使用 AWS CLI 和 Python 的`DetectToxicContent`操作。

AWS CLI

您可以在中使用以下命令偵測有毒內容 AWS CLI：

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

會以下列結果進行 AWS CLI 回應。文字區段在INSULT類別中獲得很高的可信度分數，因此產生的毒性分數很高：

```
{
   "ResultList": [
      {
         "Labels": [
                {
                    "Name": "PROFANITY",
                    "Score": 0.0006000000284984708
                },
                {
                    "Name": "HATE_SPEECH",
                    "Score": 0.00930000003427267
                },
                {
                    "Name": "INSULT",
                    "Score": 0.9204999804496765
                },
                {
                    "Name": "GRAPHIC",
                    "Score": 9.999999747378752e-05
                },
                {
                    "Name": "HARASSMENT_OR_ABUSE",
                    "Score": 0.0052999998442828655
                },
                {
                    "Name": "SEXUAL",
                    "Score": 0.01549999974668026
                },
                {
                    "Name": "VIOLENCE_OR_THREAT",
                    "Score": 0.007799999788403511
                }
            ],
            "Toxicity": 0.7192999720573425
      }
   ]
```

```
}
```

您最多可以輸入 10 個文字字串，使用下列text-segments參數格式：

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

會以下列結果進行 AWS CLI 回應：

```
{
   "ResultList": [
      {
         "Labels": [ (truncated) ],
            "Toxicity": 0.3192999720573425
      },
      {
         "Labels": [ (truncated) ],
            "Toxicity": 0.1192999720573425
      },
      {
         "Labels": [ (truncated) ],
            "Toxicity": 0.0192999720573425
      }
   ]
}
```

Python (Boto)

下面的例子演示了如何使用 Python 檢測有毒的內容：

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

response = client.detect_toxic_content(
    LanguageCode='en',
    TextSegments=[{'Text': 'You are so obtuse'}]
)
```

```
print("Response: %s\n" % response)
```

迅速安全分類

Amazon Comprehend 提供預先訓練的二進位分類器，可對大型語言模型 (LLM) 或其他生成 AI 模型的純文字輸入提示進行分類。

提示安全分類器分析輸入提示，並為提示是安全還是不安全指派可信度分數。

不安全提示是一種輸入提示，可表示惡意意圖，例如要求個人或私人資訊、產生攻擊性或非法內容，或要求醫療、法律、政治或金融主題的建議。

使用 API 迅速進行安全分類

若要針對文字字串執行提示安全分類，請使用同步ClassifyDocument作業。對於輸入，您需要提供英文純文字字串。字串的大小上限為 10 KB。

回應包括兩個類別 (SAFE 和不安全)，以及每個班級的可信度分數。分數的值範圍是零到一，其中一個是信賴度最高。

下列範例說明如何將提示的安全分類與 AWS CLI 和 Python 搭配使用。

AWS CLI

下面的例子演示了如何使用提示安全分類器與： AWS CLI

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

會以下列輸出 AWS CLI 回應：

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

```
            "Name": "SAFE_PROMPT"
        }
    ]
}
```

> **Note**
>
> 當您使用指`classify-document`令時，對於`--endpoint-arn`參數，您必須傳遞與您的 AWS CLI 組態相 AWS 區域 同的 ARN。若要配置 AWS CLI，請執行`aws configure`命令。在此範例中，端點 ARN 具有區域代碼`us-west-2`。您可以在以下任何區域中使用提示安全分類器：
>
> - us-east-1
> - us-west-2
> - eu-west-1
> - ap-southeast-2

Python (Boto)

下面的例子演示瞭如何使用提示安全分類器與 Python：

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

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

> **Note**
>
> 當您使用該`classify_document`方法時，對於`EndpointArn`引數，您必須傳遞一個 ARN，該 ARN 使用與您的 boto3 SDK 客戶端 AWS 區域 相同。在此範例中，用戶端和端點 ARN 都使用`us-west-2`。您可以在以下任何區域中使用提示安全分類器：
>
> - us-east-1

- us-west-2
- eu-west-1
- ap-southeast-2

PII 偵測和密文

您可以使用 Amazon Comprehend 主控台或 API 來偵測英文或西班牙文文字文字文件中的個人識別資訊 (PII)。PII 是個人資料的文字參考，可識別個人身分。PII 範例包括地址、銀行帳號和電話號碼。

您可以偵測或編輯文字中的 PII 實體。若要偵測 PII 實體，您可以使用即時分析或非同步批次工作。若要編輯 PII 實體，您必須使用非同步批次工作。

如需更多詳細資訊，請參閱 個人身分識別資訊 (PII) 。

個人身分識別資訊 (PII)

您可以使用 Amazon Comprehend 主控台或 API 來偵測英文或西班牙文文字文字文件中的個人識別資訊 (PII)。PII 是個人資料的文字參考，可用於識別個人身分。PII 範例包括地址、銀行帳號和電話號碼。

使用 PII 偵測，您可以選擇尋找 PII 實體或編輯文字中的 PII 實體。若要尋找 PII 實體，您可以使用即時分析或非同步批次工作。若要編輯 PII 實體，您必須使用非同步批次工作。

您可以使用 Amazon S3 物件 Lambda 存取點取得個人識別資訊 (PII)，以控制從 Amazon S3 儲存貯體擷取文件的作業。您可以控制存取包含 PII 的文件，並從文件中編輯個人識別資訊。如需詳細資訊，請參閱 使用 Amazon S3 物件 Lambda 存取點取得個人識別資訊 (PII)。

主題

偵測 PII 實體

您可以使用 Amazon Comprehend 來偵測英文或西班牙文文字文件中的 PII 實體。PII 實體是個人識別資訊 (PII) 的一種特定類型。使用 PII 偵測來尋找 PII 實體或編輯文字中的 PII 實體。

主題

尋找 PII 實體

若要在文字中尋找 PII 實體，您可以使用即時分析快速分析單一文件。您也可以在文件集合上啟動非同步批次工作。

您可以使用控制台或 API 對單個文檔進行實時分析。您的輸入文字最多可包含 100 KB 的 UTF-8 編碼字元。

例如，您可以提交下列輸入文字來尋找 PII 實體：

你好保羅·桑托斯 您的信用卡帳戶的最新月結單已郵寄至華盛頓州西雅圖市任意街 123 號。

輸出包括「保羅·桑托斯」具有類型的信息NAME，「1111-0000-1111-0000」具有該類型，並且「123 任何街道，西雅圖CREDIT_DEBIT_NUMBER，華盛頓州 98109」具有類型。ADDRESS

Amazon Comprehend 會傳回偵測到的 PII 實體清單，其中包含每個 PII 實體的下列資訊：

- 估計偵測到的文字範圍是偵測到的圖元類型之可能性的分數。
- PII 實體類型。
- PII 實體在文件中的位置，指定為實體開頭和結尾的字元位移。

例如，先前提到的輸入文字會產生下列回應：

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

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

編輯 PII 實體

若要編輯文字中的 PII 實體，您可以使用主控台或 API 啟動非同步批次工作。Amazon Comprehend 會傳回輸入文字的副本，其中包含每個 PII 實體的編輯。

例如，您可以提交下列輸入文字來編輯 PII 實體：

你好保羅·桑托斯 您的信用卡帳戶的最新月結單已郵寄至華盛頓州西雅圖市任意街 123 號。

輸出檔案包含下列文字：

你好 ***** ******。您信用卡戶口最新的月結單 ********** 已郵寄至 *** **********。

PII 通用實體類型

某些 PII 實體類型是通用的 (不特定於個別國家/地區)，例如電子郵件地址和信用卡號碼。亞馬遜偵測到下列類型的通用 PII 實體：

ADDRESS

物理地址，例如「美國任何城鎮 100 大街」或「套房 #12，123 號樓」。地址可以包括街道、建築物、位置、城市、州、國家、縣、郵遞區號、分區和鄰近地區等資訊。

AGE

個人的年齡，包括數量和時間單位。例如，在「我 40 歲」這句話中，Amazon Comprehend 承認「40 歲」是一個年齡。

AWS_ACCESS_KEY

與秘密存取金鑰相關聯的唯一識別碼；您可以使用存取金鑰 ID 和秘密存取金鑰，以密碼方式簽署程式設計 AWS 要求。

AWS_SECRET_KEY

與存取金鑰相關聯的唯一識別碼。您可以使用存取金鑰 ID 和秘密存取金鑰，以密碼方式簽署程式設計 AWS 要求。

信貸借記

VISA 上存在的三位數卡驗證碼（CVV） MasterCard，以及發現信用卡和借記卡。對於美國運通信用卡或借記卡，CVV 是一個四位數的數字代碼。

信貸借記到期

信用卡或簽帳卡到期日 此數字通常為四位數，通常格式為月/年或 MM/YY。Amazon Comprehend 到期日期，例如 2021 年 1 月 21 日，1 月和 2021 年 1 月。

借方信用卡號

信用卡或簽帳卡號碼。這些數字的長度可以從 13 到 16 位數不等。不過，當只有最後四位數字出現時，Amazon Comprehend 也會辨識信用卡或簽帳金融卡號碼。

日期時間

日期可以包括年、月、日、星期幾或一天中的時間。例如，Amazon Comprehend 認為「2020 年 1 月 19 日」或「上午 11 點」作為日期。Amazon Comprehend 將識別部分日期、日期範圍和日期間隔。它還將承認數十年，例如「1990 年代」。

驅動程式識別碼

分配給駕駛執照的號碼，這是一份正式文件，允許個人在公共道路上操作一輛或多輛機動車輛。駕照號碼由英數字元組成。

EMAIL

電子郵件地址，例如 marymajor@email.com。

國際銀行帳戶號碼

國際銀行帳戶號碼在每個國家/地區都有特定的格式。請參閱伊班網站結構。

IP_ADDRESS

一個 IPv4 位址，例如

牌照

車輛的車牌是由車輛註冊的州或國家簽發的。乘用車的格式通常為五到八位數字，由大寫字母和數字組成。格式取決於發行的州或國家的位置。

MAC_ADDRESS

媒體存取控制 (MAC) 位址是指派給網路介面控制器 (NIC) 的唯一識別碼。

NAME

個人的姓名。此實體類型不包括標題，例如博士、先生、太太或小姐。Amazon Comprehend 不會將此實體類型套用到屬於組織或地址的名稱。例如，Amazon Comprehend 將「約翰·多伊組織」識別為一個組織，並將「李四街」識別為地址。

PASSWORD

用作密碼的英數字串，例如「*very20 特殊 #pass *」。

PHONE

電話號碼。此實體類型還包括傳真和呼叫器號碼。

針腳

一個四位數的個人識別號碼（PIN），您可以使用它來訪問您的銀行帳戶。

迅捷代碼

SWIFT 代碼是銀行識別碼（BIC）的標準格式，用於指定特定的銀行或分行。銀行使用這些代碼進行匯款，例如國際電匯。

SWIFT 代碼由八個或 11 個字符組成。11 位數代碼是指特定分支機構，而 8 位數代碼（或以「XXX」結尾的 11 位數代碼）則指總部或主要辦公室。

URL

一個網址，例如：例如：

USERNAME

識別帳戶的使用者名稱，例如登入名稱、螢幕名稱、暱稱或帳號。

車輛識別號碼

車輛識別號碼 (VIN) 可唯一識別車輛。VIN 含量和格式在 ISO 3779 規範中定義。每個國家/地區都有 VIN 的特定代碼和格式。

特定國家的 PII 實體類型

某些 PII 實體類型是國家特定的，例如護照號碼和其他政府核發的身份證號碼。Amazon Comprehend 會偵測到下列國家/地區特定 PII 實體類型：

加健康號

加拿大 Health 服務號碼是一個 10 位數的唯一識別碼，個人需要獲得醫療保健福利。

社會保險號碼

加拿大社會保險號碼 (SIN) 是九位數的唯一識別碼，個人需要使用政府計劃和福利。

該罪被格式化為三組三個數字，如 123-456-789。可以通過稱為 Luhn 算法的簡單檢查數字過程來驗證 SIN。

IN_ 阿德哈爾

印度阿德哈爾（Aadhaar）是印度政府向印度居民發出的 12 位唯一識別號碼。Aadhaar 格式的第四個和第八位數字後面有一個空格或連字符。

因雷加

印度全國農村就業保證法（NREGA）編號由兩個字母組成，後跟 14 個數字。

無永久帳號 (_)

印度永久帳戶號碼是由所得稅部門核發的 10 位數字唯一的字母數字編號。

投票人數 (_)

印度選民身份證由三個字母組成，後跟七個數字。

UK_NATIONAL_HEALTH_SERVICE_NUMBER

英國國民 Health 服務號碼是一個 10-17 位數字的數字，例如：485 777 3456。目前的系統會將 10 位數字格式化，並在第三位和第六位數字之後加上空格。最後一個數字是偵測錯誤的總和檢查碼。

17 位數字格式在第 10 位和 13 位數字之後有空格。

UK_NATIONAL_INSURANCE_NUMBER

英國國民保險號碼（NINO）為個人提供國民保險（社會安全）福利的機會。它也用於英國稅收制度的某些目的。

數字長度為九位數，以兩個字母開頭，後跟六個數字和一個字母。NINO 可以在兩個字母之後以及第二個，第四和第六位數字之後使用空格或破折號進行格式化。

UK_UNIQUE_TAXPAYER_REFERENCE_NUMBER

英國唯一納稅人參考（UTR）是一個 10 位數字，用於識別納稅人或企業。

銀行帳戶號碼

美國銀行帳戶號碼，通常長度為 10 至 12 位數字。當只有最後四個數字出現時，Amazon Comprehend 也會識別銀行帳戶號碼。

銀行路由

美國銀行帳戶的分行代碼。這些通常是九位數的長度，但 Amazon Comprehend 也會在只有最後四個數字出現時識別路由號碼。

護照號碼

美國護照號碼。護照號碼範圍為六至九個英數字元。

使用個人稅務識別號碼

美國個人納稅識別號碼 (ITIN) 是一個九位數字，以「9」開頭，並以「7」或「8」作為第四位數字。個人納稅識別號碼可以在第三位和第四位數字之後使用空格或破折號進行格式化。

SSN

美國社會安全號碼 (SSN) 是發給美國公民、永久居民和臨時工作居民的九位數字。當只有最後四位數字出現時，Amazon Comprehend 也會辨識社會安全號碼。

為 PII 實體加上標籤

當您執行 PII 偵測時，Amazon Comprehend 會傳回已識別之 PII 實體類型的標籤。例如，如果您將下列輸入文字提交 Amazon Comprehend：

你好保羅·桑托斯 您的信用卡帳戶的最新月結單已郵寄至華盛頓州西雅圖市任意街 123 號。

輸出包括代表 PII 實體類型的標籤以及準確度的可信度分數。在這種情況下，文檔文本「保羅·桑托斯」，「1111-0000-1111-000」和「123，西雅圖，華盛頓州西雅圖的任何街道」生成標籤，並分別作為 PII 實體類型。NAME CREDIT_DEBIT_NUMBER ADDRESS如需支援實體類型的詳細資訊，請參閱PII 通用實體類型。

Amazon Comprehend 提供每個標籤的下列資訊：

- PII 實體類型的標籤名稱。
- 估計偵測到的文字標示為 PII 實體類型之可能性的分數。

上面的輸入文本示例導致以下 JSON 輸出。

```
{
    "Labels": [
        {
```

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

PII 即時分析 (主控台)

您可以使用主控台執行 PII 即時偵測文字文件。文字大小上限為 100 KB 的 UTF-8 編碼字元。控制台顯示結果，以便您可以查看分析。

使用內建模型執行 PII 偵測即時分析

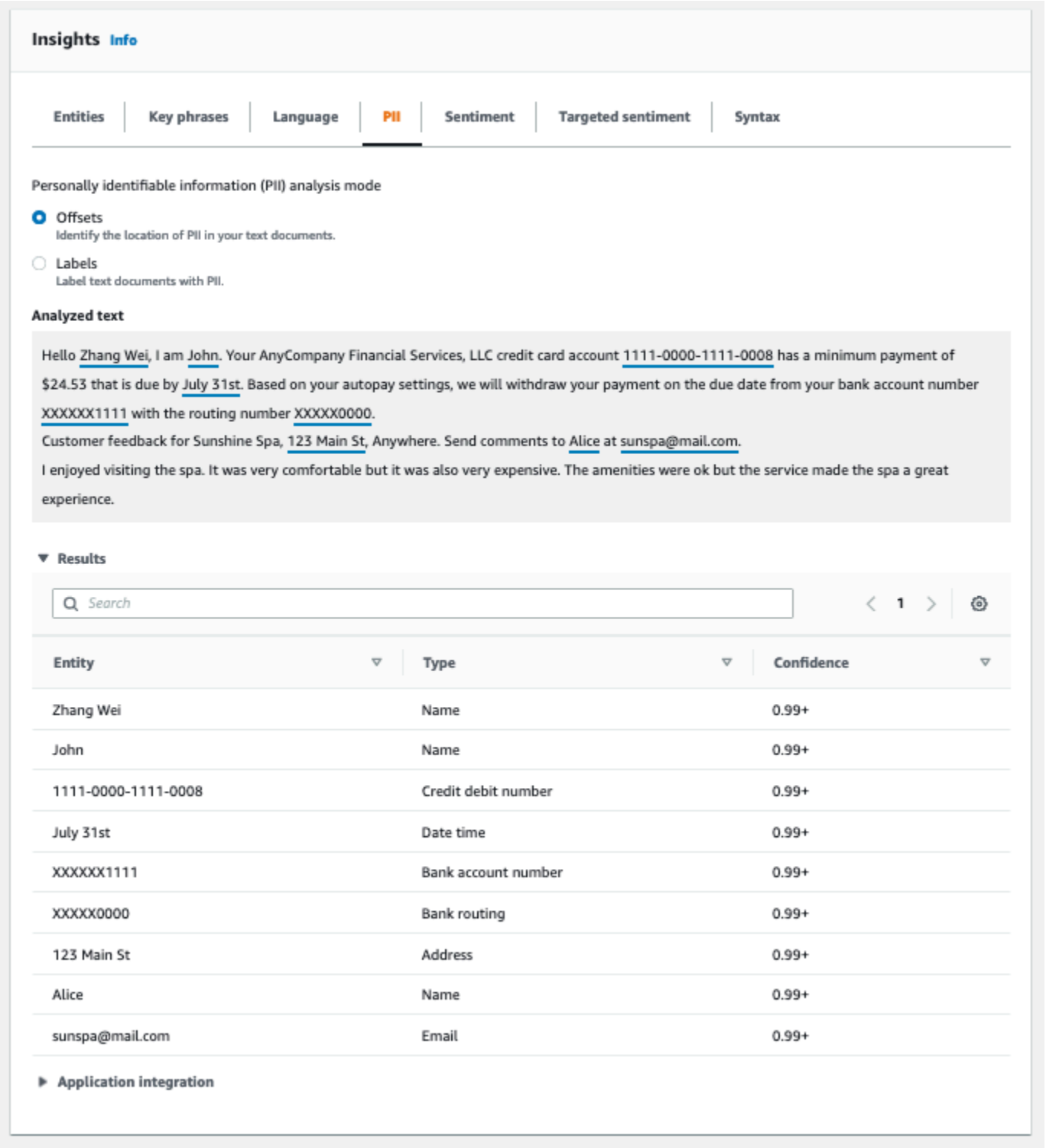
1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [即時分析]。
3. 在「輸入類型」下，選擇「內建」做為 「分析」
4. 輸入您要分析的文字。
5. 選擇「分析」。主控台會在「深入解析」面板中顯示文字分析結果。PII 索引標籤會列出在輸入文字中偵測到的 PII 實體。

在「深入解析」面板中，PII 索引標籤會顯示兩種分析模式的結果：

- 位移 — 識別 PII 在文字文件中的位置。
- 標籤 — 識別已識別 PII 實體類型的標籤。

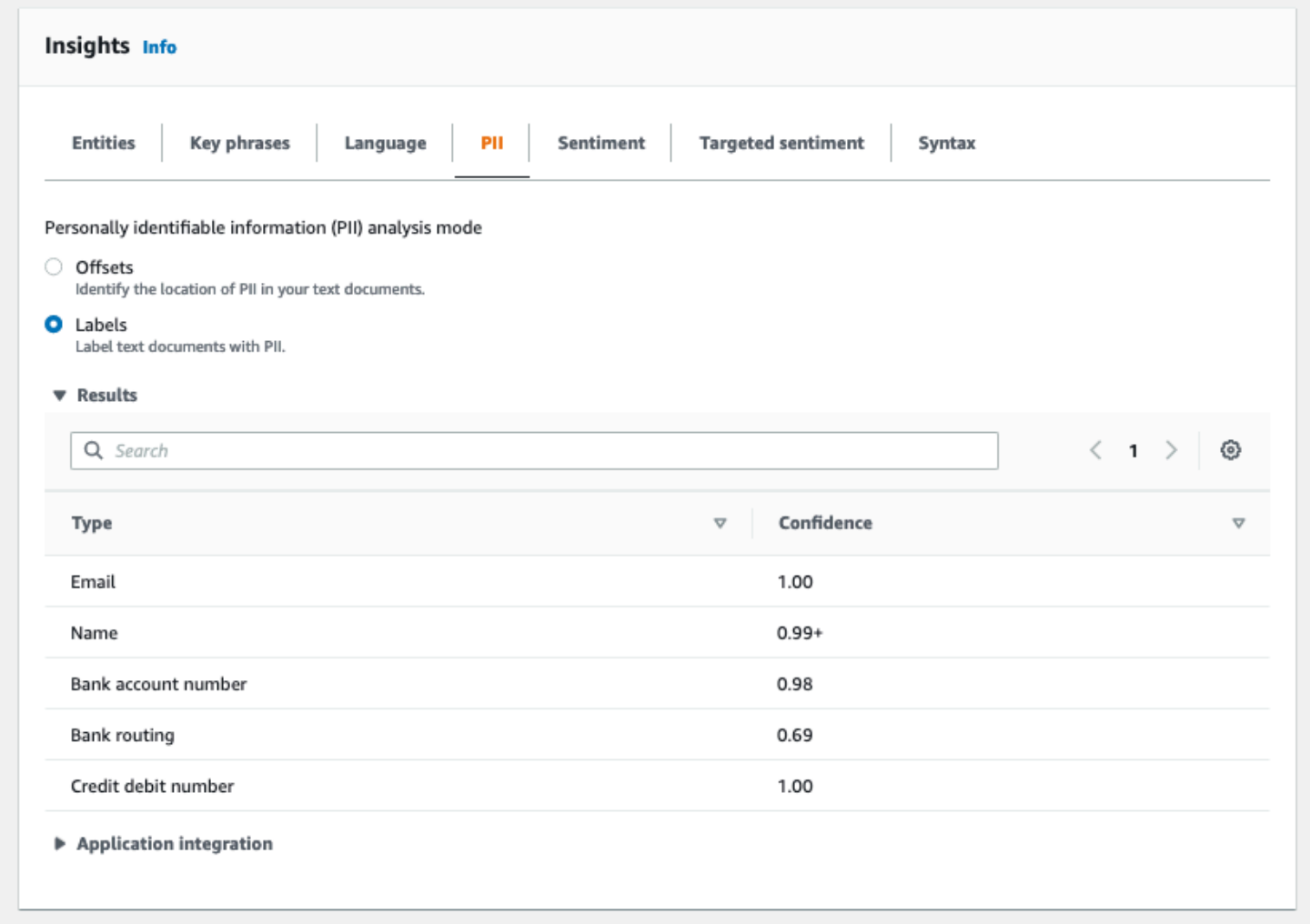
偏移

偏移分析模式可識別文字文件中 PII 的位置。如需詳細資訊，請參閱 尋找 PII 實體。



標籤

標籤分析模式會傳回已識別 PII 實體類型的標籤。如需詳細資訊，請參閱 為 PII 實體加上標籤。



PII 非同步分析工作 (主控台)

您可以使用主控台建立非同步分析工作，以偵測 PII 實體。如需 PII 實體類型的詳細資訊，請參閱偵測 PII 實體。

建立分析工作

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [分析工作]，然後選擇 [建立工作]。
3. 在「Job 設定」 下，為分析工作指定一個唯一的名稱。
4. 針對 [分析類型]，選擇 [個人識別資訊 (PII)]。
5. 在 [語言] 中，選擇其中一種支援的語言 (英文或西班牙文)。
6. 在「輸出」模式中，選取下列其中一個選項：

- 位移 — 工作輸出會傳回每個 PII 實體的位置。
- 密文 — 工作輸出會傳回每個 PII 項目編輯的輸入文字副本。

7. (選擇性) 如果您選擇「密文」作為輸出模式，則可以選取要編輯的 PII 實體類型。
8. 在「輸入資料」下，指定輸入文件在 Amazon S3 中的位置：
   - 若要分析您自己的文件，請選擇 [我的文件]，然後選擇 [瀏覽 S3] 提供儲存貯體或包含檔案的資料夾路徑。
   - 若要分析 Amazon Comprehend 提供的範例，請選擇範例文件。在這種情況下，Amazon Comprehend 會使用由管理的儲存貯體 AWS，而您不會指定位置。
9. (選擇性) 對於「輸入格式」，請為輸入檔案指定下列其中一種格式：
   - 每個檔案一個文件 — 每個檔案包含一個輸入文件。這是最適合大型文檔的集合。
   - 每行一個文件 — 輸入為一個或多個檔案。檔案中的每一行都被視為一個文件。這最適合短文件，例如社交媒體張貼。每行必須以換行符（LF，\n），回車符（CR，\r）或兩者結束（CRLF，\r\n）。您不能使用 UTF-8 行分隔符（u+2028）來結束一行。
10. 在「輸出資料」下，選擇「瀏覽 S3」。選擇您希望 Amazon Comprehend 寫入分析產生的輸出資料的 Amazon S3 儲存貯體或資料夾。
11. (選擇性) 若要加密工作的輸出結果，請選擇加密。然後，選擇要使用與目前帳戶關聯的 KMS 金鑰，還是使用與其他帳戶相關聯的 KMS 金鑰：
    - 如果您使用與目前帳戶相關聯的金鑰，請為 KMS 金鑰 ID 選擇金鑰別名或 ID。
    - 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ID 下輸入金鑰別名或 ID 的 ARN。

    > **Note**
    >
    > 如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱金鑰管理服務 (KMS)。

12. 在「存取權限」下，提供 IAM 角色，該角色可：
    - 授予對輸入文件之 Amazon S3 位置的讀取存取權限。
    - 授予對輸出文件之 Amazon S3 位置的寫入存取權。
    - 包含允許`comprehend.amazonaws.com`服務主體擔任角色並取得其權限的信任原則。

    如果您還沒有具有這些許可和適當信任政策的 IAM 角色，請選擇「建立 IAM 角色」來建立一個角色。
13. 填寫完表格後，請選擇 [建立工作] 以建立並啟動主題偵測工作。

新工作會顯示在工作清單中，而狀態欄位顯示工作的狀態。此欄位可IN_PROGRESS用於正在處理的工COMPLETED作、已成功完成的工作，以及FAILED發生錯誤的工作。您可以按一下工作以取得有關該工作的詳細資訊，包括任何錯誤訊息。

任務完成後，Amazon Comprehend 會將分析結果存放在您為該任務指定的輸出 Amazon S3 位置。如需分析結果的描述，請參閱偵測 PII 實體。

PII 即時分析 (API)

Amazon Comprehend 提供即時同步 API 作業，以分析文件中的個人識別資訊 (PII)。

主題

- 尋找 PII 即時實體 (API)
- 標記 PII 即時實體 (API)

尋找 PII 即時實體 (API)

若要在單一文件中尋找 PII，您可以使用 Amazon Comprehend DetectPiiEntities作業。您的輸入文字最多可包含 100 KB 的 UTF-8 編碼字元。支持的語言包括英語和西班牙語。

使用 (CLI) 尋找 PII

下列範例將作DetectPiiEntities業搭配使用 AWS CLI。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：

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

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

標記 PII 即時實體 (API)

您可以使用即時同步 API 作業來傳回已識別之 PII 實體類型的標籤。如需詳細資訊，請參閱 為 PII 實體加上標籤。

標示 PII 實體 (CLI)

下列範例將作ContainsPiiEntities業搭配使用 AWS CLI。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

Amazon Comprehend 回應與以下內容：

```
{
```

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

PII 非同步分析工作 (API)

PII 非同步分析 (API)

您可以使用非同步 API 作業來建立分析工作，以尋找或編輯 PII 實體。如需 PII 實體類型的詳細資訊，請參閱偵測 PII 實體。

主題

- 使用非同步工作 (API) 尋找 PII 實體
- 使用非同步工作 (API) 編輯 PII 實體

使用非同步工作 (API) 尋找 PII 實體

執行非同步批次工作，在文件集合中尋找 PII。若要執行任務，請將您的文件上傳到 Amazon S3，然後提交StartPiiEntitiesDetectionJob請求。

主題

- 開始之前
- 輸入參數
- 異步 Job 方法
- 輸出檔案格式

- 使用非同步分析 AWS Command Line Interface

開始之前

在開始之前，請確保您擁有：

- 輸入和輸出值區 — 識別要用於輸入檔案和輸出檔案的 Amazon S3 儲存貯體。值區必須與您呼叫的 API 位於相同的區域。
- IAM 服務角色 — 您必須擁有具有存取輸入和輸出值區之權限的 IAM 服務角色。如需詳細資訊，請參閱 非同步作業所需的角色型權限。

輸入參數

在您的請求中，包括以下必要參數：

- `InputDataConfig`— 提供請求的InputDataConfig定義，其中包括工作的輸入內容。對於`S3Uri`參數，請指定輸入文件的 Amazon S3 位置。
- `OutputDataConfig`— 提供請求的OutputDataConfig定義，其中包括工作的輸出內容。對於`S3Uri`參數，請指定 Amazon S3 位置寫入其分析結果的位置。
- `DataAccessRoleArn`— 提供 AWS Identity and Access Management 角色的 Amazon 資源名稱（ARN）。此角色必須授與 Amazon Comprehend 對您輸入資料的讀取存取權限，以及對您在 Amazon S3 中輸出位置的寫入存取權限。如需詳細資訊，請參閱 非同步作業所需的角色型權限。
- `Mode`— 將此參數設定為`ONLY_OFFSETS`。透過此設定，輸出會提供尋找輸入文字中每個 PII 實體的字元位移。輸出還包括可信度分數和 PII 實體類型。
- `LanguageCode`— 將此參數設定為`en`或`es`。Amazon Comprehend 支援英文或西班牙文文字的 PII 偵測。

異步 Job 方法

會`StartPiiEntitiesDetectionJob`傳回工作 ID，以便您可以監視工作進度，並在工作完成時擷取工作狀態。

若要監視分析工作的進度，請提供作業 ID 給DescribePiiEntitiesDetectionJob作業。來自的回應`DescribePiiEntitiesDetectionJob`包含具有工作目前狀態的`JobStatus`欄位。成功的工作會在下列狀態中轉換：

提交-> 進行中-> 完成。

分析工作完成後 (「JobStatus已完成」、「失敗」或「已停止」)，可用DescribePiiEntitiesDetectionJob來取得結果的位置。如果任務狀態為COMPLETED，則回應會包含一個OutputDataConfig欄位，其中包含具有輸出檔案 Amazon S3 位置的欄位。

如需 Amazon 理解非同步分析所要遵循的步驟的其他詳細資訊，請參閱。非同步批次處理

輸出檔案格式

輸出檔案使用輸入檔案的名稱，並在結尾附加 .out。它包含分析的結果。

以下是分析工作中偵測到文件中 PII 實體的輸出檔案範例。輸入的格式是每行一個文檔。

```
{
  "Entities": [
    {
      "Type": "NAME",
      "BeginOffset": 40,
      "EndOffset": 69,
      "Score": 0.999995
    },
    {
      "Type": "ADDRESS",
      "BeginOffset": 247,
      "EndOffset": 253,
      "Score": 0.998828
    },
    {
      "Type": "BANK_ACCOUNT_NUMBER",
      "BeginOffset": 406,
      "EndOffset": 411,
      "Score": 0.693283
    }
  ],
  "File": "doc.txt",
  "Line": 0
},
{
  "Entities": [
    {
      "Type": "SSN",
      "BeginOffset": 1114,
      "EndOffset": 1124,
      "Score": 0.999999
```

```
    },
    {
      "Type": "EMAIL",
      "BeginOffset": 3742,
      "EndOffset": 3775,
      "Score": 0.999993
    },
    {
      "Type": "PIN",
      "BeginOffset": 4098,
      "EndOffset": 4102,
      "Score": 0.999995
    }
  ],
  "File": "doc.txt",
  "Line": 1
 }
```

以下是分析輸出的範例，其中輸入的格式為每個檔案一個文件。

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

使用非同步分析 AWS Command Line Interface

下列範例將作StartPiiEntitiesDetectionJob業搭配使用 AWS CLI。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

對於cli-input-json參數，您為包含請求資料的 JSON 檔案提供路徑，如下列範例所示。

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

如果啟動事件偵測工作的要求成功，您將會收到類似下列內容的回應：

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

您可以使用此DescribeEventsDetectionJob作業來取得現有工作的狀態。如果啟動事件偵測工作的要求成功，您將會收到類似下列內容的回應：

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

```
    --region region \
    --job-id job ID
```

當工作順利完成時，您會收到類似下列內容的回應：

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

使用非同步工作 (API) 編輯 PII 實體

若要編輯文字中的 PII 實體，請啟動非同步批次工作。若要執行任務，請將您的文件上傳到 Amazon S3，然後提交StartPiiEntitiesDetectionJob請求。

主題

- 開始之前
- 輸入參數
- 輸出檔案格式
- PII 編輯使用 AWS Command Line Interface

開始之前

在開始之前，請確保您擁有：

- 輸入和輸出值區 — 識別要用於輸入檔案和輸出檔案的 Amazon S3 儲存貯體。值區必須與您呼叫的 API 位於相同的區域。
- IAM 服務角色 — 您必須擁有具有存取輸入和輸出值區之權限的 IAM 服務角色。如需詳細資訊，請參閱 非同步作業所需的角色型權限。

輸入參數

在您的請求中，包括以下必要參數：

- `InputDataConfig`— 提供請求的InputDataConfig定義，其中包括工作的輸入內容。對於`S3Uri`參數，請指定輸入文件的 Amazon S3 位置。
- `OutputDataConfig`— 提供請求的OutputDataConfig定義，其中包括工作的輸出內容。對於`S3Uri`參數，請指定 Amazon S3 位置寫入其分析結果的位置。
- `DataAccessRoleArn`— 提供 AWS Identity and Access Management 角色的 Amazon 資源名稱（ARN）。此角色必須授與 Amazon Comprehend 對您輸入資料的讀取存取權限，以及對您在 Amazon S3 中輸出位置的寫入存取權限。如需詳細資訊，請參閱 非同步作業所需的角色型權限。
- `Mode`— 將此參數設定為`ONLY_REDACTION`。透過此設定，Amazon Comprehend 會將您的輸入文件複本寫入到 Amazon S3 中的輸出位置。在此副本中，每個 PII 實體都會編輯。
- `RedactionConfig`— 提供請求的RedactionConfig定義，其中包括密文的組態參數。指定要編輯的 PII 類型，並指定要將每個 PII 實體取代為其類型的名稱還是您選擇的字元：
  - 指定要在陣列中編輯的 PII 實體類型。`PiiEntityTypes`若要編輯所有實體類型，請`["ALL"]`將陣列值設定為。
  - 若要以其類型取代每個 PII 實體，請將`MaskMode`參數設定為`REPLACE_WITH_PII_ENTITY_TYPE`。例如，使用此設定時，PII 實體「多珍」會取代為「[NAME]」。
  - 若要以您選擇的字元取代每個 PII 實體中的字元，請將`MaskMode`參數設定為`MASK`，然後將`MaskCharacter`參數設定為取代字元。僅提供單一字元。有效字元為! 、#、$、%、&、* 和 @。例如，使用此設定時，PII 實體「多珍」可以取代為「**** ***」
- `LanguageCode`— 將此參數設定為`en`或`es`。Amazon Comprehend 支援英文或西班牙文文字的 PII 偵測。

輸出檔案格式

下列範例顯示編輯 PII 之分析工作的輸入和輸出檔案。輸入的格式是每行一個文檔。

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

編輯此輸入檔案的分析工作會產生下列輸出檔案。

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

PII 編輯使用 AWS Command Line Interface

下列範例將作StartPiiEntitiesDetectionJob業搭配使用 AWS CLI。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

對於cli-input-json參數，您為包含請求資料的 JSON 檔案提供路徑，如下列範例所示。

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

如果啟動事件偵測工作的要求成功，您將會收到類似下列內容的回應：

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

您可以使用此DescribeEventsDetectionJob作業來取得現有工作的狀態。

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

當工作順利完成時，您會收到類似下列內容的回應：

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

文件處理

Amazon Comprehend 支援自訂分類和自訂實體辨識的單一步驟文件處理。例如，您可以混合輸入純文字文件和半結構化文件 (例如 PDF 文件、Microsoft Word 文件和影像) 至自訂分析工作。

對於需要文字擷取的輸入檔案，Amazon Comprehend 會在執行分析之前自動執行文字擷取。為了擷取文字內容，Amazon Comprehend 會針對原生半結構化文件使用內部剖析器，並針對影像和掃描的文件使用 Amazon Textract API。

除了亞太區域 (東京) 和 AWS GovCloud (美國西部) 僅支援自訂分類的純文字模型外支援地區，每個 Amazon Comprehend 都提供 Amazon Comprehend 文件處理功能。

下列主題提供 Amazon Comprehend 支援用於自訂分析之輸入文件類型的詳細資訊。

主題

即時自訂分析的輸入

使用自定義模型進行實時分析需要單個文檔作為輸入。下列主題說明您可以使用的輸入文件類型。

主題

純文字文件

將輸入文件提供為 UTF-8 格式的文字。

半結構化文件

半結構化文件包括原生 PDF 文件和 Word 文件。

根據預設，即時自訂分析會使用 Amazon Comprehend 剖析器從 Word 檔案和數位 PDF 檔案擷取文字。對於 PDF 檔案，您可以覆寫此預設值，並使用 Amazon Textract 擷取文字。請參閱 設定文字擷取選項。

影像檔案和掃描的 PDF 檔案

支援的影像類型包括 JPEG、PNG 和 TIFF。

根據預設，自訂實體辨識會使用 Amazon Textract `DetectDocumentText` API 操作從影像檔案和掃描的 PDF 檔案擷取文字。您可以覆寫此預設值，改為使用 `AnalyzeDocument` API 作業。請參閱 設定文字擷取選項。

Amazon Textract 輸出

您可以提供來自 Amazon Textract `DetectDocumentText` API 或 API 的 JSON 輸出，做為即時 `AnalyzeDocument` API 操作的輸入，以進行自訂分類和自訂實體辨識。Amazon Comprehend 支援此輸入類型，用於即時 API 作業，但不支援主控台。

即時分析的最大文件大小

對於所有輸入文件類型，輸入檔案最多為一頁，不超過 10,000 個字元。

下表顯示輸入文件的最大檔案大小。

檔案類型	大小上限 (API)	最大尺寸（控制台）
UTF-8 文字文件	10 KB	10 KB
PDF 文件	10 MB	5 MB
文字文件	10 MB	1 MB
影像檔	10 MB	5 MB
文 Textract 輸出檔案	1 MB	N/A

半結構化文件中的錯誤

從半結構化文件ClassifyDocument或影像檔案擷取文字時，或 DetectEntitiesAPI 作業可能會遇到文件層級或頁面層級錯誤。

頁面層級錯誤

如果ClassifyDocument或 DetectEntitiesAPI 作業在處理輸入文件中的頁面時發生錯誤，則 API 回應會針對每個錯誤在「錯誤」清單中包含一個項目。

錯誤清單`ErrorCode`中的項目包含下列其中一個值：

- 文字檔案 — Amazon Textract 法讀取頁面。如需 Amazon Textract 中頁面限制的詳細資訊，請參閱 Amazon Textract 中的頁面配額。
- 已提供 _ 通過 _ 超過 — 請求數超出您的輸送量限制。如需 Amazon Textract 中輸送量配額的詳細資訊，請參閱 Amazon Textract 中的預設配額。
- 頁面字元超過 — 頁面上的文字字元太多 (最多 10,000 個字元)。
- 頁面大小超出 — 最大頁面大小為 10 MB。
- 內部伺服器錯誤 — 請求遇到服務問題。請再次嘗試 API 要求。

文件層級錯誤

如果ClassifyDocument或 DetectEntitiesAPI 作業在輸入文件中偵測到文件層級錯誤，API 會傳回`InvalidRequestException`錯誤回應。

在錯誤回應中，Reason欄位包含值`INVALID_DOCUMENT`。

此Detail欄位包含下列其中一個值：

- 文件大小超出 — 文件大小太大。檢查檔案大小，然後重新提交要求。
- 不支援的文件類型 — 不支援文件類型。檢查檔案類型並重新提交請求。
- 超出頁面限制 — 文件中頁面過多。檢查檔案中的頁數，然後重新提交要求。
- 文本訪問被拒絕 _ 異常 — 訪問被拒絕 Amazon Textract。確認您的帳戶具有使用 Amazon Textract DetectDocumentText和 AnalyzeDocumentAPI 操作的權限，並重新提交請求。

非同步自訂分析的輸入

您可以將多個文件輸入至自訂的非同步分析工作。下列主題說明您可以使用的輸入文件類型。最大檔案大小會根據輸入文件的類型而有所不同。

主題

純文字文件

將所有純文本輸入文檔提供為 UTF-8 格式的文本。下表列出檔案大小上限及其他準則。

> **Note**
>
> 當所有輸入文件都是純文本時，這些限制適用。

描述	名額 / 指引
每個檔案格式一個文件的最大檔案大小 (自訂分類)	1 個字節-10 MB
文件大小 (自訂實體辨識)	1 個字節-1 MB
檔案數目上限，每個檔案一個文件	1,000,000
最大行數，每行一個文檔（適用於請求中的所有文件）	1,000,000
文件語料庫大小 (所有文件皆以純文字組合)	1 個字節 —5 GB

半結構化文件

半結構化文件包括原生 PDF 文件和 Word 文件。

下表列出檔案大小上限及其他準則。

描述	名額 / 指引
文件大小	1 個字節-50 MB
文件大小 (DOCX)	1 個字節 —5 MB
檔案數目上限	500
PDF 文件或 DOCX 文件的最大頁數	100
文本提取後的文檔語料庫大小（純文本，所有文件合併）	1 個字節 —5 GB

根據預設，自訂分析會使用 Amazon Comprehend 剖析器從 Word 檔案和數位 PDF 檔案擷取文字。對於 PDF 檔案，您可以覆寫此預設值，並使用 Amazon Textract 擷取文字。請參閱 設定文字擷取選項。

影像檔案和掃描的 PDF 檔案

自訂分析支援 JPEG、PNG 和 TIFF 影像。

下表列出影像的最大檔案大小。掃描的 PDF 檔案的大小與原生 PDF 檔案的大小上限相同。

描述	名額 / 指引
影像大小 (JPG 或 PNG)	1 個字節-10 MB
影像尺寸	1 個字節-10 MB。最多一頁。

如需有關影像的其他資訊，請參閱影像的最佳做法。

根據預設，Amazon Comprehend 會使用 Amazon Textract 字提取 `DetectDocumentText` API 操作，從影像檔案和掃描的 PDF 檔案中擷取文字。您可以覆寫此預設值，改為使用 `AnalyzeDocument` API 作業。請參閱 設定文字擷取選項。

Amazon Textract 取輸出 JSON 文件

對於自訂實體辨識而非自訂分類，您可以提供 Amazon Textract `AnalyzeDocument` API 操作的輸出檔案作為分析任務的輸入。

設定文字擷取選項

根據預設，Amazon Comprehend 會根據輸入檔案類型，執行下列動作以從檔案擷取文字：

- Word 文件 — Amazon Comprehend 析器提取文本。
- 數字 PDF 文件 — Amazon Comprehend 析器提取文本。
- 影像檔案和掃描的 PDF 檔案 Amazon Comprehend 用 Amazon Textract 取 `DetectDocumentText` API 來擷取文字。

對於影像檔案和 PDF 檔案，您可以使用`DocumentReaderConfig`參數來取代這些預設擷取動作。當您使用 Amazon Comprehend 主控台或 API 進行即時或非同步自訂分析時，即可使用此參數。

`DocumentReaderConfig`參數包含三個欄位：

- DocumentReadMode— 設定`SERVICE_DEFAULT`為可讓 Amazon Comprehend 執行預設動作。

  設定為`FORCE_DOCUMENT_READ_ACTION`使用 Amazon Textract 解析數位 PDF 檔案。
- DocumentReadAction— 設置 Amazon Textract 提取使用亞馬遜文本提取時使用的 Amazon Textract 本提取 API（DetectDocumentText 或 AnalyzeDocument）。
- FeatureTypes— 如果您設DocumentReadAction定使用 AnalyzeDocument API 作業，您可以新增一或兩個 `FeatureTypes` (表格、表單)。這些功能提供有關文件中表格和表單的其他資訊。如需這些功能的詳細資訊，請參閱 Amazon Textract 文件分析回應物件。

下列範例顯示如何`DocumentReaderConfig`針對特定使用案例進行設定：

1. 使用 Amazon Textract 取所有 PDF 文件。
   a. DocumentReadMode – 設為 `FORCE_DOCUMENT_READ_ACTION`。
   b. DocumentReadAction – 設為 `TEXTRACT_DETECT_DOCUMENT_TEXT`。
   c. FeatureTypes— 不需要。
2. 針對所有 PDF 和影像檔案使用 Amazon Textract 取 `AnalyzeDocument` API。
   a. DocumentReadMode – 設為 `FORCE_DOCUMENT_READ_ACTION`。
   b. DocumentReadAction – 設為 `TEXTRACT_ANALYZE_DOCUMENT`。
   c. FeatureTypes— 設定為`TABLES`，`FORMS`或同時設定兩個功能。
3. 對於掃描的 PDF 檔案和所有影像檔案，請使用 Amazon Textract 取 `AnalyzeDocument` API。
   a. DocumentReadMode – 設為 `SERVICE_DEFAULT`。

b. DocumentReadAction – 設為 TEXTRACT_ANALYZE_DOCUMENT。

c. FeatureTypes— 設定為TABLES，FORMS或同時設定兩個功能。

如需 Amazon Textract 選項的詳細資訊，請參閱DocumentReaderConfig。

影像的最佳做法

當您使用影像檔案進行自訂分類或自訂圖元辨識時，請使用下列準則來獲得最佳結果：

- 提供高質量的圖像，理想情況下至少為 150 DPI。
- 如果影像檔案使用其中一種支援的格式 (TIFF、JPEG 或 PNG)，請勿在將檔案上傳到 Amazon S3 之前對其進行轉換或縮減取樣。

為了在從文件中的表格擷取文字時獲得最佳結果，請遵循下列作法：

- 文件中的表格會在視覺上與頁面上周圍的元素分開。例如，表格不會覆蓋到影像或複雜的圖樣上。
- 表格中的文字是直立的。例如，文字不會相對於頁面上的其他文字旋轉。

從表格擷取文字時，在下列情況下，您可能會看到不一致的結果：

- 合併的表格儲存格會跨越多個欄。
- 表格具有與同一表格的其他部分不同的儲存格、列或欄。

自訂分類

使用自訂分類將文件組織成您定義的類別 (類別)。自訂分類程序為兩個步驟。首先，您要訓練自訂分類模型 (也稱為分類器)，以辨識您感興趣的類別。然後，您可以使用模型來分類任意數量的文件集。

例如，您可以將支援要求的內容分類，以便將要求路由至適當的支援團隊。或者，您可以對從客戶收到的電子郵件進行分類，以根據客戶請求的類型提供指導。您可以將 Amazon Comprehend 與 Amazon Transcribe 相結合，將語音轉換為文字，然後將來自支援電話的請求進行分類。

您可以同步 (即時) 在單一文件上執行自訂分類，或啟動非同步工作來分類一組文件。您的帳戶中可以有多個自訂分類器，每個分類器都使用不同的資料進行訓練。自訂分類支援多種輸入文件類型，例如純文字、PDF、Word 和影像。

當您提交分類工作時，您可以根據需要分析的文件類型來選擇要使用的分類器模型。例如，若要分析純文字文件，您可以使用以純文字文件訓練的模型來獲得最準確的結果。若要分析半結構化文件 (例如 PDF、Word、影像、Amazon Textract 輸出或掃描檔案)，您可以使用以原生文件訓練的模型來獲得最準確的結果。

主題

準備分類器訓練資料

對於自訂分類，您可以在多重類別模式或多標籤模式下訓練模型。多類模式將單個類與每個文檔相關聯。多標籤模式將一個或多個類與每個文檔相關聯。每種模式的輸入檔案格式都不同，因此請先選擇要使用的模式，然後再建立訓練資料。

> **Note**
>
> 亞馬遜主控台將多類別模式稱為單一標籤模式。

自訂分類支援您使用純文字文件訓練的模型，以及使用原生文件 (例如 PDF、Word 或影像) 訓練的模型。如需有關分類器模型及其支援文件類型的詳細資訊，請參閱訓練分類模型。

若要準備資料以訓練自訂分類器模型：

1. 識別您要此分類器分析的類別。決定要使用的模式 (多類別或多標籤)。
2. 根據模型是用於分析純文字文件還是半結構化文件，決定分類器模型類型。
3. 收集每個類別的文件範例。如需最低訓練需求，請參閱文件分類的一般配額。
4. 對於純文字模型，請選擇要使用的訓練檔案格式 (CSV 檔案或增強資訊清單檔案)。若要訓練原生文件模型，您一律使用 CSV 檔案。

主題

分類器訓練檔案格式

對於純文字模型，您可以將分類器訓練資料提供為 CSV 檔案或使 SageMaker 用 Ground Truth 建立的增強資訊清單檔案。CSV 檔案或增強資訊清單檔案包含每個訓練文件的文字及其相關聯的標籤。

對於原生文件模型，您可以將分類器訓練資料提供為 CSV 檔案。CSV 檔案包括每個訓練文件的檔案名稱及其關聯的標籤。您可以在訓練任務的 Amazon S3 輸入資料夾中包含訓練文件。

CSV 檔案

您可以在 CSV 檔案中以 UTF-8 編碼的文字形式提供標籤訓練資料。請勿包含標題列。在檔案中新增標題列可能會導致執行階段錯誤。

對於 CSV 文件中的每一行，第一列包含一個或多個類標籤，類標籤可以是任何有效的 UTF-8 字符串。我們建議使用不重疊在意義上的清晰類名。該名稱可以包含空格，並且可以由多個用底線或連字符連接的單詞組成。

請勿在逗號之前或之後留下任何空格字元，以分隔列中的值。

CSV 檔案的確切內容取決於分類器模式和訓練資料的類型。如需詳細資訊，請參閱多類模式和中的各節多標籤模式。

增強清單文件

增強資訊清單檔案是您使用「 SageMaker Ground Truth」建立的標記資料集。Ground Truth 是一項資料標籤服務，可協助您 (或您雇用的員工) 為機器學習模型建置訓練資料集。

如需有關 Ground Truth 及其產生輸出的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的使用 SageMaker Ground Truth 來標記資料。

增強資訊清單檔案採用 JSON 行格式。在這些檔案中，每一行都是完整的 JSON 物件，其中包含訓練文件及其相關聯的標籤。每行的確切內容取決於分類器模式。如需詳細資訊，請參閱多類模式和中的各節多標籤模式。

當您將訓練資料提供給 Amazon Comprehend 時，您可以指定一個或多個標籤屬性名稱。您指定的屬性名稱數量取決於您的增強資訊清單檔案是單一標籤工作的輸出還是連結標籤工作的輸出。

如果您的檔案是單一標籤工作的輸出，請從「Ground Truth」工作中指定單一標籤屬性名稱。

如果您的檔案是鏈結標籤工作的輸出，請為鏈結中的一或多個工作指定標籤屬性名稱。每個標籤屬性名稱均提供個別工作的註釋。您最多可以為鏈結標籤工作中的增強資訊清單檔案指定其中 5 個屬性。

如需有關鏈結標籤任務的詳細資訊，以及它們產生的輸出範例，請參閱 Amazon SageMaker 開發人員指南中的鏈結標籤任務。

多類模式

在多類模式下，分類為每個文檔分配一個類。個別類別是相互排斥的。例如，您可以將電影分類為喜劇或科幻小說，但不能同時分類兩者。

> **Note**
>
> 亞馬遜主控台將多類別模式稱為單一標籤模式。

主題

- 純文字模型
- 原生文件模型

純文字模型

若要訓練純文字模型，您可以將標籤化的訓練資料提供為 CSV 檔案或 SageMaker Ground Truth 的增強資訊清單檔案。

CSV 檔案

如需將 CSV 檔案用於訓練分類器的一般資訊，請參閱CSV 檔案。

將訓練資料提供為兩欄 CSV 檔案。對於每一列，第一欄包含類別標籤值。第二欄包含該類別的範例文字文件。每一列必須以\ n 或\ r\ n 字元結尾。

下列範例顯示包含三個文件的 CSV 檔案。

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

下列範例顯示 CSV 檔案的其中一列，該資料列會訓練自訂分類器，以偵測電子郵件訊息是否為垃圾郵件：

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

增強清單文件

如需有關針對訓練分類器使用增強資訊清單檔案的一般資訊，請參閱增強清單文件。

對於純文字文件，增強資訊清單檔案的每一行都是完整的 JSON 物件，其中包含訓練文件、單一類別名稱，以及來自 Ground Truth 的其他中繼資料。下列範例是用於訓練自訂分類器辨識垃圾郵件訊息的增強資訊清單檔案：

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

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

下列範例顯示增強資訊清單檔案中的一個 JSON 物件，其格式化為可讀性：

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

在此範例中，`source`屬性提供訓練文件的文字，而`MultiClassJob`屬性會從分類清單指派類別的索引。`job-name`屬性是您在「Ground Truth」中為標籤工作定義的名稱。

當您在 Amazon Comprehend 中啟動分類器訓練任務時，您可以指定相同的標籤任務名稱。

原生文件模型

原生文件模型是您使用原生文件 (例如 PDF、DOCX 和影像) 進行訓練的模型。您可以將訓練資料提供為 CSV 檔案。

CSV 檔案

如需將 CSV 檔案用於訓練分類器的一般資訊，請參閱CSV 檔案。

將訓練資料提供為三欄 CSV 檔案。對於每一列，第一欄包含類別標籤值。第二列包含此類的示例文檔的文件名。第三列包含頁碼。如果範例文件是影像，則頁碼為選用。

下列範例顯示參考三個輸入文件的 CSV 檔案。

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

```
CLASS,input-doc-3.png
```

下列範例顯示 CSV 檔案的其中一列，該資料列會訓練自訂分類器，以偵測電子郵件訊息是否為垃圾郵件。PDF 檔案的第 2 頁包含垃圾郵件範例。

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

多標籤模式

在多標籤模式下，個別類別代表不相互排斥的不同類別。多標籤分類為每個文檔分配一個或多個類。例如，您可以將一部電影歸類為「紀錄片」，將另一部電影歸類為「科幻小說」、「動作」和「喜劇」。

對於培訓，多標籤模式支持多達 100 萬個示例，其中包含多達 100 個獨特的課程。

主題

- 純文字模型
- 原生文件模型

純文字模型

若要訓練純文字模型，您可以將標籤化的訓練資料提供為 CSV 檔案或 SageMaker Ground Truth 的增強資訊清單檔案。

CSV 檔案

如需將 CSV 檔案用於訓練分類器的一般資訊，請參閱CSV 檔案。

將訓練資料提供為兩欄 CSV 檔案。對於每一列，第一欄包含類別標籤值，第二欄包含這些類別的範例文字文件。要在第一列中輸入多個類別，請在每個類別之間使用分隔符號（如 | ）。

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

下列範例會顯示 CSV 檔案的其中一列，該資料列會訓練自訂分類器，以偵測電影摘要中的類型：

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

類別名稱之間的預設分隔符號為管線 (|)。但是，您可以使用不同的字元作為分隔符號。分隔符號必須與類別名稱中的所有字元不同。例如，如果您的類別是 CLASS_1、CLASS_2 和 CLASS_3，則底線 (_) 就是類別名稱的一部分。所以不要使用下劃線作為分隔類名的分隔符。

增強清單文件

如需有關針對訓練分類器使用增強資訊清單檔案的一般資訊，請參閱增強清單文件。

對於純文字文件，增強資訊清單檔案的每一行都是完整的 JSON 物件。它包含一個培訓文檔，類名和其他元數據 Ground Truth。下列範例是增強資訊清單檔案，用於訓練自訂分類器以偵測電影摘要中的類型：

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

下列範例顯示增強資訊清單檔案中的一個 JSON 物件，其格式化為可讀性：

```
{
      "source": "A band of misfit teens become unlikely detectives when
                  they discover troubling clues about their high school English
 teacher.
                    Could the strange Mrs. Doe be an alien from outer space?",
      "MultiLabelJob": [
          3,
          8,
          10,
          11
      ],
      "MultiLabelJob-metadata": {
          "job-name": "labeling-job/multilabeljob",
          "class-map": {
              "3": "comedy",
```

```
            "8": "mystery",
            "10": "science_fiction",
            "11": "teen"
        },
        "human-annotated": "yes",
        "creation-date": "2020-05-21T19:00:01.291202",
        "confidence-map": {
            "3": 0.95,
            "8": 0.77,
            "10": 0.83,
            "11": 0.92
        },
        "type": "groundtruth/text-classification-multilabel"
    }
}
```

在此範例中，source屬性提供訓練文件的文字，而MultiLabelJob屬性會從分類清單中指派數個類別的索引。中MultiLabelJob繼資料中的工作名稱是您在 Ground Truth 中為標籤工作定義的名稱。

原生文件模型

原生文件模型是您使用原生文件 (例如 PDF、DOCX 和影像檔案) 進行訓練的模型。您可以將標籤化的訓練資料提供為 CSV 檔案。

CSV 檔案

如需將 CSV 檔案用於訓練分類器的一般資訊，請參閱CSV 檔案。

將訓練資料提供為三欄 CSV 檔案。對於每一列，第一欄包含類別標籤值。第二欄包含這些類別的範例文件的檔案名稱。第三列包含頁碼。如果範例文件是影像，則頁碼為選用。

要在第一列中輸入多個類別，請在每個類別之間使用分隔符號（如 | ）。

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

下列範例會顯示 CSV 檔案中的一列，這些檔案會訓練自訂分類器，以偵測電影摘要中的類型。PDF 文件的第 2 頁包含喜劇/青少年電影的示例。

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

類別名稱之間的預設分隔符號為管線 (|)。但是，您可以使用不同的字元作為分隔符號。分隔符號必須與類別名稱中的所有字元不同。例如，如果您的類別是 CLASS_1、CLASS_2 和 CLASS_3，則底線 (_) 就是類別名稱的一部分。所以不要使用下劃線作為分隔類名的分隔符。

訓練分類模型

若要訓練自訂分類的模型，您可以定義類別並提供範例文件以訓練自訂模型。您可以在多重類別或多標籤模式下訓練模型。多類模式將單個類與每個文檔相關聯。多標籤模式將一個或多個類與每個文檔相關聯。

自訂分類支援兩種類型的分類器模型：純文字模型和原生文件模型。純文字模型會根據文件的文字內容來分類文件。原生文件模型也會根據文字內容對文件進行分類。原生文件模型也可以使用額外的訊號，例如從文件的版面配置。您可以使用原生文件來訓練模型的原生文件模型，以瞭解配置資訊。

純文字模型具有以下特性：

- 您可以使用 UTF-8 編碼的文字文件來訓練模型。
- 您可以使用下列其中一種語言的文件來訓練模型：英文、西班牙文、德文、義大利文、法文或葡萄牙文。
- 指定分類器的訓練文件必須全部使用相同的語言。
- 訓練文件為純文字，因此不需要額外的文字擷取費用。

原生文件模型具有下列特性：

- 您可以使用半結構化文件來訓練模型，其中包括下列文件類型：
  - 數字和掃描的 PDF 文檔。
  - 字文檔（DOCX）。
  - 圖片：JPG 文件，PNG 文件和單頁 TIFF 文件。
  - Textract 取 API 輸出 JSON 檔案。
- 您可以使用英文文件來訓練模型。
- 如果您的訓練文件包含掃描的文件檔案，您需要支付額外的文字擷取費用。如需詳細資訊，請參閱亞馬遜定價頁面。

您可以使用任一類型的模型來分類任何支援的文件類型。但是，為了獲得最準確的結果，我們建議使用純文字模型來分類純文字文件，使用原生文件模型來分類半結構化文件。

主題

訓練自訂分類器 (主控台)

您可以使用控制台創建和訓練自定義分類器，然後使用自定義分類器來分析您的文檔。

若要訓練自訂分類器，您需要一組訓練文件。您可以使用您希望文件分類器辨識的類別來標示這些文件。如需有關準備訓練文件的資訊，請參閱準備分類器訓練資料。

若要建立和訓練文件分類器模型

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇「自訂」，然後選擇「自訂分類」。
3. 選擇「建立新模型」。
4. 在模型設定下，輸入分類器的型號名稱。該名稱在您的帳戶和當前區域中必須是唯一的。

   (選擇性) 輸入版本名稱。該名稱在您的帳戶和當前區域中必須是唯一的。
5. 選取訓練文件的語言。若要查看分類器支援的語言，請參閱訓練分類模型。
6. (選擇性) 如果您想要在 Amazon Comprehend 處理訓練任務時加密儲存磁碟區中的資料，請選擇分類器加密。然後選擇要使用與您目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。

   - 如果您使用與目前帳戶相關聯的金鑰，請選擇 KMS 金鑰識別碼的金鑰識別碼。
   - 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ARN 下輸入金鑰識別碼的 AR N。

> **Note**
>
> 如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱 AWS Key Management Service (AWS KMS)。

7. 在 [資料規格] 下，選擇要使用的訓練模型類型。

   - 純文字文件：選擇此選項可建立純文字模型。使用純文字文件訓練模型。
   - 原生文件：選擇此選項可建立原生文件模型。使用原生文件 (PDF、Word、影像) 訓練模型。

8. 選擇訓練資料的資料格式。若要取得有關資料格式的資訊，請參閱分類器訓練檔案格式。

   - CSV 檔案：如果訓練資料使用 CSV 檔案格式，請選擇此選項。
   - 增強資訊清單：如果您使用 Ground Truth 為訓練資料建立增強資訊清單檔案，請選擇此選項。如果您選擇純文字文件做為訓練模型類型，則可使用此格式。

9. 選擇要使用的分類器模式。

   - 單一標籤模式：如果您指派給文件的類別是互斥的，而且您正在訓練分類器為每個文件指派一個標籤，請選擇此模式。在 Amazon Comprehend API，單標籤模式被稱為多類模式。
   - 多標籤模式：如果可以同時將多個類別套用至文件，而且您正在訓練分類器為每份文件指派一或多個標籤，請選擇此模式。

10. 如果選擇「多重標籤」模式，則可以為標籤選取「分隔符號」。當訓練文件有多個類別時，請使用此分隔符號字元來分隔標籤。預設分隔符號為直線字元。

11. (選擇性) 如果您選擇增強資訊清單做為資料格式，您最多可以輸入五個擴增資訊清單檔案。每個擴增資訊清單檔案都包含訓練資料集或測試資料集。您必須提供至少一個訓練資料集。測試數據集是可選的。使用下列步驟來設定增強資訊清單檔案：

    a. 在 [訓練和測試資料集] 下，展開 [輸入位置] 面板。

    b. 在資料集類型中，選擇訓練資料或測試資料。

    c. 對於 G SageMaker round Truth 增強資訊清單檔案 S3 位置，請輸入包含資訊清單檔案的 Amazon S3 儲存貯體的位置，或選擇瀏覽 S3 瀏覽至該儲存貯體。您用於訓練任務存取權限的 IAM 角色必須具有 S3 儲存貯體的讀取權限。

    d. 對於「屬性」名稱，請輸入包含註釋的屬性名稱。如果檔案包含來自多個鏈結標籤工作的註釋，請為每個工作新增一個屬性。

    e. 若要新增其他輸入位置，請選擇 [新增輸入位置]，然後設定下一個位置。

12. (選擇性) 如果您選擇 CSV 檔案做為資料格式，請使用下列步驟來設定訓練資料集和選擇性測試資料集：

    a. 在訓練資料集下，輸入包含訓練資料 CSV 檔案的 Amazon S3 儲存貯體的位置，或選擇瀏覽 S3 來瀏覽至該儲存貯體。您用於訓練任務存取權限的 IAM 角色必須具有 S3 儲存貯體的讀取權限。

       (選擇性) 如果您選擇原生文件做為訓練模型類型，您也會提供包含訓練範例檔案之 Amazon S3 資料夾的 URL。

    b. 在「測試資料集」下，選取是否要為 Amazon Comprehend 提供額外資料，以測試訓練過的模型。

       - 自動分割：自動分割會自動選取 10% 的訓練資料，以便保留做為測試資料使用。
       - (選擇性) 客戶提供：在 Amazon S3 中輸入測試資料 CSV 檔案的網址。您也可以導覽至 Amazon S3 中的位置，然後選擇選取資料夾。

         (選擇性) 如果您選擇原生文件做為訓練模型類型，您也會提供包含測試檔案之 Amazon S3 資料夾的 URL。

13. (選擇性) 對於「文件」讀取模式，您可以覆寫預設的文字擷取動作。純文字模型不需要此選項，因為它適用於掃描文件的文字擷取。如需詳細資訊，請參閱 設定文字擷取選項。

14. (純文字模型為選用) 對於輸出資料，請輸入 Amazon S3 儲存貯體的位置以儲存訓練輸出資料，例如混淆矩陣。如需詳細資訊，請參閱 混淆矩陣。

    (選擇性) 如果您選擇加密訓練工作的輸出結果，請選擇「加密」。然後選擇要使用與目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。

    - 如果您使用與目前帳戶相關聯的金鑰，請選擇 KMS 金鑰 ID 的金鑰別名。
    - 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ID 下輸入金鑰別名或 ID 的 ARN。

15. 對於 IAM 角色，請選擇選擇現有的 IAM 角色，然後為包含訓練文件的 S3 儲存貯體選擇具有讀取許可的現有 IAM 角色。角色必須具有開頭`comprehend.amazonaws.com`為有效的信任原則。

    如果您還沒有具有這些許可的 IAM 角色，請選擇「建立 IAM 角色」以建立 IAM 角色。選擇要授與此角色的存取權限，然後選擇名稱尾碼，以區分該角色與帳戶中的 IAM 角色。

> **Note**
>
> 對於加密的輸入文件，所使用的 IAM 角色也必須具有`kms:Decrypt`權限。如需詳細資訊，請參閱 使用 KMS 加密所需的權限。

16. (選擇性) 若要從 VPC 將資源啟動至 Amazon Comprehend，請在 VPC 下輸入虛擬私人雲端識別碼，或從下拉式清單中選擇識別碼。

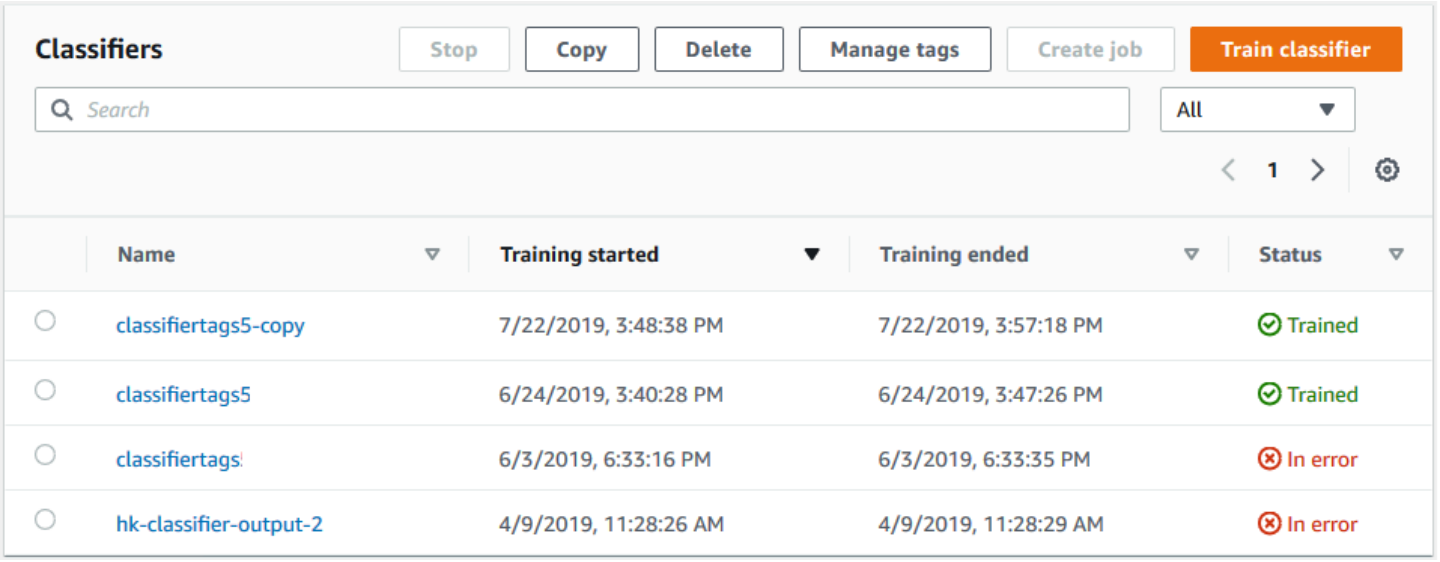
    1. 在「子網路」下選擇子網路。選取第一個子網路後，您可以選擇其他子網路。
    2. 在「安全性群組」下，選擇要使用的安全性群組 (如果您已指定安全性群組)。選取第一個安全性群組後，您可以選擇其他群組。

    > **Note**
    >
    > 當您將 VPC 與分類工作搭配使用時，`DataAccessRole`用於「建立」和「啟動」作業的 VPC 必須具有存取輸入文件和輸出值區的 VPC 權限。

17. (選擇性) 若要將標籤新增至自訂分類器，請在「標籤」下輸入鍵值配對。選擇 Add tag (新增標籤)。若要在建立分類器之前移除此配對，請選擇 [移除標籤]。如需詳細資訊，請參閱 標記您的 資源。
18. 選擇 Create (建立)。

主控台會顯示「分類器」頁面。新的分類器會顯示在表格中，顯示`Submitted`為其狀態。當分類器開始處理訓練文件時，狀態會變更為`Training`。當分類器準備好可供使用時，狀態會變更為`Trained`或`Trained with warnings`。如果狀態為`TRAINED_WITH_WARNINGS`，請檢閱中略過的檔案資料夾分類器訓練輸出。

如果 Amazon Comprehend 在建立或訓練期間遇到錯誤，狀態會變更為。`In error`您可以在表格中選擇分類器工作，以取得有關分類器的詳細資訊，包括任何錯誤訊息。



訓練自訂分類器 (API)

若要建立和訓練自訂分類器，請使用此CreateDocumentClassifier作業。

您可以使用DescribeDocumentClassifier操作監視請求的進度。`Status`欄位轉換為之後`TRAINED`，您可以使用分類器來分類文件。如果狀態為`TRAINED_WITH_WARNINGS`，請分類器訓練輸出從`CreateDocumentClassifier`作業中檢閱略過的檔案資料夾。

主題

- 使用訓練自訂分類 AWS Command Line Interface
- 使用 AWS SDK for Java 或開發套件

使用訓練自訂分類 AWS Command Line Interface

下列範例說明如何搭配使用`CreateDocumentClassifier`作`DescribeDocumentClassificationJob`業、作業和其他自訂分類器 AWS CLI API。

這些範例已針對 Unix、Linux 和 macOS 進行格式化。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

使用操作創建純文本自定義分類器。`create-document-classifier`

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

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

若要建立原生自訂分類器，請在create-document-classifier要求中提供下列其他參數。

1. DocumentType：將值設定為「半結構 _ 文件」。
2. 文件：訓練文件的 S3 位置 (以及可選的測試文件)。
3. OutputDataConfig：提供輸出文件的 S3 位置 (以及選用的 KMS 金鑰)。
4. DocumentReaderConfig: 文字擷取設定的選用欄位。

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

使用該操作使用文檔分類器 ARN 獲取有關自定義分類器的信息。DescribeDocumentClassifier

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

使用操作刪除自訂分類DeleteDocumentClassifier器。

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

使用ListDocumentClassifiers操作列出帳戶中的所有自訂分類器。

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

使用 AWS SDK for Java 或開發套件

如需如何建立和訓練自訂分類器的 SDK 範例，請參閱使用開發套件建立 Amazon Comprehend 文件分類器 AWS。

測試訓練資料

在訓練模型之後，Amazon Comprehend 會測試自訂分類器模型。如果您未提供測試資料集，Amazon Comprehend 會使用 90% 的訓練資料來訓練模型。它保留 10% 的培訓數據用於測試。如果您確實提供了測試資料集，則測試資料必須為訓練資料集中的每個唯一標籤包含至少一個範例。

測試模型會提供您可用來估算模型精確度的量度。主控台會在主控台「分類器詳細資訊」頁面的「分類器效能」段落中顯示測量結果。它們也會在DescribeDocumentClassifier作業傳回的`Metrics`欄位中傳回。

在下面的例子培訓數據, 有五個標籤, 紀錄片, 紀錄片, 科幻, 紀錄片, 浪漫喜劇. 有三個獨特的類別：紀錄片，科幻小說，浪漫喜劇。

第一欄	第二欄
紀錄片	文件文字 1
紀錄片	文件文字 2
科幻小說	文件文字 3
紀錄片	文件文字 4
浪漫喜劇	文件文字 5

對於 auto 拆分 (Amazon Comprehend 保留 10% 用於測試的訓練資料)，如果訓練資料包含特定標籤的有限範例，則測試資料集可能包含該標籤的零範例。例如，如果訓練資料集包含 1000 個紀錄片類的執行個體、900 個科學小說的執行個體，以及浪漫喜劇類別的單一執行個體，則測試資料集可能包含 100 個紀錄片和 90 個科學小說實例，但沒有浪漫喜劇實例，因為有一個可用的範例。

完成模型訓練後，訓練指標會提供資訊，讓您用來決定模型是否足以滿足您的需求。

分類器訓練輸出

Amazon Comprehend 完成自訂分類器模型訓練後，會在您在 CreateDocumentClassifierAPI 請求或對等主控台請求中指定的 Amazon S3 輸出位置建立輸出檔案。

當您訓練純文字模型或原生文件模型時，Amazon Comprehend 會建立混淆矩陣。當您訓練原生文件模型時，它可以建立額外的輸出檔案。

主題

混淆矩陣

訓練自訂分類器模型時，Amazon Comprehend 會建立混淆矩陣，提供有關模型在訓練中執行效能的指標。此矩陣會顯示模型預測的標籤矩陣，與實際的文件標籤相比。Amazon Comprehend 會使用部分訓練資料來建立混淆矩陣。

混淆矩陣會提供哪些類別可以使用更多資料來改善模型效能的指示。具有較高分數正確預測的類別沿著矩陣的對角線具有最多的結果。如果對角線上的數字是較低的數字，則該類具有正確預測的較低部分。您可以為此類別新增更多訓練範例，然後再次訓練模型。例如，如果 40% 的標籤 A 樣本被分類為標籤 D，則為標籤 A 和標籤 D 新增更多樣本可增強分類器的效能。

Amazon Comprehend 建立分類器模型之後，混淆矩陣就會出現在 S3 輸出位置的`confusion_matrix.json`檔案中。

混淆矩陣的格式會有所不同，具體取決於您是使用多類模式還是多標籤模式訓練分類器。

主題

多類模式的混淆矩陣

在多類別模式中，個別類別是互斥的，因此分類會為每個文件指派一個標籤。例如，動物可以是狗或貓，但不能同時是兩者。

請考慮以下多類訓練分類器的混淆矩陣示例：

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

在這種情況下，模型預測了以下情況：

- 準確地預測了一個「A」標籤，兩個「A」標籤被錯誤地預測為「B」標籤，四個「A」標籤被錯誤地預測為「Y」標籤。
- 準確地預測了三個「B」標籤，並錯誤地將一個「B」標籤預測為「Y」標籤。
- 準確地預測了一個「X」。
- 一個「Y」標籤被準確地預測，一個標籤被錯誤地預測為「A」標籤，一個被錯誤地預測為「B」標籤，並且一個被錯誤地預測為「X」標籤。

矩陣中的對角線（A：A，B：B，X：X 和 Y：Y）顯示準確的預測。預測誤差是對角線之外的值。在此情況下，矩陣會顯示下列預測錯誤率：

- 一個標籤：86%
- B 型標籤：25%
- X 個標籤：0%
- Y 標籤：75%

分類器返回混亂矩陣作為 JSON 格式的文件。下列 JSON 檔案代表前一個範例的矩陣。

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

```
}
```

多標籤模式的混淆矩陣

在多標籤模式下，分類可以將一個或多個類別指派給文件。請考慮下列多類別訓練分類器的混淆矩陣範例。

在此範例中，有三個可能的標籤：ComedyAction、和Drama。多標籤混淆矩陣會為每個標籤建立一個 2x2 矩陣。

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

 No  2   1               No  1   1               No  3   0

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

在此情況下，模型會針對Comedy標籤傳回下列內容：

- 準確地預測標Comedy籤存在的兩個執行個體。真正的陽性（TP）。
- 準確地預測標Comedy籤不存在的兩個執行個體。真正的負值（TN）。
- Comedy標籤未正確預測出現的零執行個體。假陽性（FP）。
- 不正確地預測Comedy標籤不存在的一個執行個體。假陰性（FN）。

與多類混淆矩陣一樣，每個矩陣中的對角線顯示準確的預測。

在這種情況下，該模型準確地預測Comedy標籤 80% 的時間（TP 加 TN），並不正確地預測他們 20% 的時間（FP 加 FN）。

分類器返回混亂矩陣作為 JSON 格式的文件。下列 JSON 檔案代表前一個範例的矩陣。

```
{
"type": "multi_label",
"confusion_matrix": [
 [[2, 1],
```

```
 [0, 2]],
 [[1, 1],
 [2, 1]],
 [[3, 0],
 [1, 1]]
],
"labels": ["Comedy", "Action", "Drama"]
"all_labels": ["Comedy", "Action", "Drama"]
}
```

原生文件模型的其他輸出

當您訓練原生文件模型時，Amazon Comprehend 可以建立額外的輸出檔案。

Amazon Textract 輸出

如果 Amazon Comprehend 呼叫 Amazon Textract 字提取 API 來擷取任何培訓文件的文字，它會將 Amazon Textract 輸出檔案儲存在 S3 輸出位置。它使用以下目錄結構：

- 培訓文件：

  `amazon-textract-output/train/<file_name>/<page_num>/textract_output.json`
- 測試文件：

  `amazon-textract-output/test/<file_name>/<page_num>/textract_output.json`

如果您在 API 請求中提供測試文件，Amazon Comprehend 就會填入測試資料夾。

文件註解失敗

如果有任何失敗的註釋，Amazon Comprehend 會在 Amazon S3 輸出位置 (在 skipped_Documents/ 資料夾中) 建立下列檔案：

- 失敗 _ 註釋 _ 火車。

  如果訓練資料中有任何註釋失敗，則檔案存在。
- 失敗 _ 註釋 _ 測試

  如果請求包含測試數據並且測試數據中的任何註釋失敗，則文件存在。

失敗的註釋檔案是具有下列格式的 JSONL 檔案：

```
{
     "File": "String", "Page": Number, "ErrorCode": "...", "ErrorMessage": "..."}
    {"File": "String", "Page": Number, "ErrorCode": "...", "ErrorMessage": "..."
  }
```

自訂分類器指標

Amazon Comprehend 提供的指標可協助您估算自訂分類器的效能。Amazon Comprehend 會使用分類器訓練任務的測試資料來計算指標。指標可準確地表示模型在訓練期間的效能，因此它們會對類似資料進行分類的模型效能近似。

使用 API 操作，例DescribeDocumentClassifier如檢索自定義分類器的指標。

> **Note**
>
> 如需瞭解基礎精確度、召回和 F1 評分量度，請參閱量度：精確度、召回和 fScore。這些測量結果是在類別層次定義。Amazon Comprehend 使用巨集平均法將這些指標合併到測試集 P、R 和 F1 中，如下所述。

主題

- 指標
- 提高自定義分類器的性能

指標

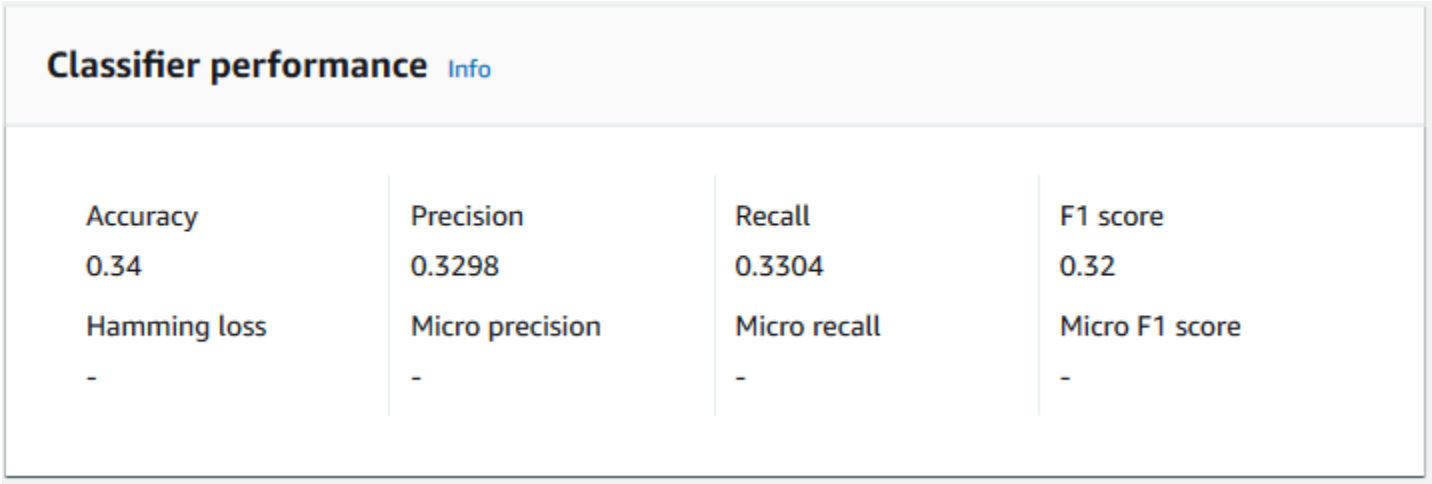
Amazon Comprehend 支援下列指標:

主題

- 準確性
- 精確度（宏觀精度）
- 召回（宏調用）
- F1 得分（宏 F1 得分）
- 汉明损失
- 微精密
- 微召回

- 微型 F1 比分

若要檢視「分類器」的測量結果，請在主控台中開啟「分類器詳細資訊」頁面。



準確性

準確度指示來自模型準確預測的測試資料中的標籤百分比。為了計算準確性，請將測試文件中準確預測標籤的數量除以測試文件中的標籤總數。

例如

實際標籤	預測標籤	準確/不正確
1	1	準確
0	1	不正確
2	3	不正確
3	3	準確
2	2	準確
1	1	準確
3	3	準確

準確度包括準確預測的數量除以整體測試樣本的數量 = 5/7 = 0.714 或 71.4%

精確度（宏觀精度）

精度是對測試數據中分類器結果的有用性的衡量標準。它被定義為正確分類的文檔數量，除以班級的分類總數。高精度意味著分類器返回的結果顯著比不相關的結果更加相關。

Precision量度也稱為「巨集精確度」。

下列範例顯示測試集的精確度結果。

標籤	樣本大小	標籤精度
商標貼紙 _1	400	0.75
標籤貼紙 _2	300	0.80
標籤貼紙 _3	30000	0.90 版
標籤貼紙 _4	20	0.50
標籤型 _5	10	0.40

因此，模型的精確度 (巨集精確度) 公制為：

```
Macro Precision = (0.75 + 0.80 + 0.90 + 0.50 + 0.40)/5 = 0.67
```

召回（宏調用）

這表示模型可以預測的文字中正確類別的百分比。此指標來自所有可用標籤的召回分數的平均值。召回是測試數據分類器結果的完整程度。

高召回意味著分類器返回了大部分相關結果。

Recall量度也稱為「巨集回復」。

下列範例顯示測試集的回收結果。

標籤	樣本大小	標籤召回
商標貼紙 _1	400	0.70
標籤貼紙 _2	300	0.70

標籤	樣本大小	標籤召回
標籤貼紙 _3	30000	0.98
標籤貼紙 _4	20	0.80
標籤型 _5	10	0.10

因此，模型的召回 (巨集回復) 量度為：

```
Macro Recall = (0.70 + 0.70 + 0.98 + 0.80 + 0.10)/5 = 0.656
```

F1 得分（宏 F1 得分）

F1 分數衍生自`Precision`和`Recall`值。它測量分類器的整體精度。最高分為 1，最低分數為 0。

Amazon Comprehend 計算宏 F1 得分。這是 F1 分數標籤的未加權平均值。使用以下測試集作為示例：

標籤	樣本大小	F1 得分标签
商標貼紙 _1	400	0.724
標籤貼紙 _2	300	0.824
標籤貼紙 _3	30000	0.94
標籤貼紙 _4	20	0.62
標籤型 _5	10	0.16

模型的 F1 分數 (巨集 F1 分數) 的計算方式如下：

```
Macro F1 Score = (0.724 + 0.824 + 0.94 + 0.62 + 0.16)/5 = 0.6536
```

汉明损失

不正確預測的標籤分數。與標籤總數相比，也被視為不正確標籤的分數。分數接近於零更好。

微精密

原件:

與精確度度量類似，不同之處在於微精確度是以所有精確度分數加在一起的整體得分為基礎。

微召回

與召回指標類似，不同之處在於微回收是基於所有召回分數加在一起的總分數。

微型 F1 比分

微型 F1 分數是「微型精確度」和「微型召回」指標的組合。

提高自定義分類器的性能

這些指標可讓您深入瞭解自訂分類器在分類工作期間的效能。如果指標很低，則分類模型可能對您的使用案例無效。您有幾個選項可以提高分類器性能：

1. 在訓練資料中，提供定義明確分隔品類的具體範例。例如，提供使用唯一字詞/句子來表示類別的文件。
2. 在訓練資料中為代表不足的標籤新增更多資料。
3. 嘗試減少類別中的歪斜。如果資料中最大的標籤在最小標籤中的文件是文件的 10 倍以上，請嘗試增加最小標籤的文件數量。確保在高度代表和最少代表的類之間將偏斜比減少到最多 10:1。您也可以嘗試從高度表示的類中刪除輸入文檔。

執行即時分析

訓練自訂分類器之後，您可以使用即時分析來分類文件。即時分析會將單一文件做為輸入，並同步傳回結果。自訂分類可接受各種文件類型做為即時分析的輸入。如需詳細資訊，請參閱 即時自訂分析的輸入。

如果您打算分析影像檔案或掃描的 PDF 文件，您的 IAM 政策必須授予許可，才能使用兩種 Amazon Textract API 方法 (DetectDocumentText 和 AnalyzeDocument)。Amazon Comprehend 文本提取過程中調用這些方法。如需政策範例，請參閱 執行文件分析動作所需的權限。

您必須建立端點，才能使用自訂分類模型執行即時分析。

主題

- 自訂分類的即時分析 (主控台)

- 自訂分類 (API) 的即時分析
- 即時分析的輸出

自訂分類的即時分析 (主控台)

您可以使用 Amazon Comprehend 主控台，使用自訂分類模型執行即時分析。

您可以建立端點來執行即時分析。端點包含受管資源，可讓您的自訂模型可用於即時推論。

如需佈建端點輸送量及相關成本的相關資訊，請參閱使用 Amazon Comprehend 端點。

主題

- 建立自訂分類的端點
- 執行即時自訂分類

建立自訂分類的端點

若要建立端點 (主控台)

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇「端點」，然後選擇「建立端點」按鈕。會開啟 「建立端點」畫面。
3. 為端點命名。名稱在目前的區域和帳戶中必須是唯一的。
4. 選擇您要附加新端點的自訂模型。從下拉菜單中，您可以按型號名稱進行搜索。

   > **Note**
   >
   > 您必須先建立模型，才能將端點貼附到模型上。如果您還沒有模型，請參閱訓練分類模型。

5. (選擇性) 若要將標籤新增至端點，請在「標籤」下輸入金鑰值配對，然後選擇「新增標籤」。若要在建立端點之前移除此配對，請選擇「移除標籤」
6. 輸入要指派給端點的推論單元 (IU) 數目。每個單位代表每秒 100 個字元的輸送量，每秒最多兩個文件。如需端點輸送量的相關資訊，請參閱使用 Amazon Comprehend 端點。
7. (選擇性) 如果要建立新端點，您可以選擇使用 IU 估算器。根據輸送量或每秒要分析的字元數，可能很難知道您需要多少個推論單位。這個選擇性步驟可協助您判斷要求的 IU 數目。

8. 在「購買」摘要中，檢閱您預估的每小時、每日和每月端點成本。
9. 如果您瞭解自己的帳戶從端點啟動到刪除為止，會對端點產生費用，請選取此核取方塊。
10. 選擇建立端點

執行即時自訂分類

建立端點後，您可以使用自訂模型執行即時分析。從主控台執行即時分析的方式有兩種。您可以輸入文字或上傳檔案，如下所示。

使用自訂模型 (主控台) 執行即時分析

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [即時分析]。
3. 在輸入類型下，選擇自訂為分析類型。
4. 在「自訂模型類型」下選擇「自訂分類」。
5. 針對「端點」，選擇您要使用的端點。此端點會連結至特定的自訂模型。
6. 要指定用於分析的輸入數據，您可以輸入文本或上傳文件。
   - 若要輸入文字：
     a. 選擇「輸入文字」。
     b. 輸入您要分析的文字。
   - 若要上傳檔案：
     a. 選擇「上傳檔案」，然後輸入要上傳的檔案名稱。
     b. (選擇性) 在「進階讀取動作」 下，您可以覆寫文字擷取的預設動作。如需詳細資訊，請參閱設定文字擷取選項

   為了獲得最佳結果，請將輸入類型與分類器模型類型相匹配。如果您將原生文件提交至純文字模型，或提交純文字至原生文件模型，主控台會顯示警告。如需詳細資訊，請參閱 訓練分類模型。
7. 選擇「分析」。Amazon Comprehend 使用您的自定義模型分析輸入的數據。Amazon Comprehend 會顯示探索到的類別，以及每個班級的信心評估。

自訂分類 (API) 的即時分析

您可以使用亞馬遜 API，透過自訂模型執行即時分類。首先，您要建立端點來執行即時分析。建立端點之後，即可執行即時分類。

本節中的範例使用了 Unix、Linux 和 macOS 的指令格式。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

如需佈建端點輸送量及相關成本的相關資訊，請參閱使用 Amazon Comprehend 端點。

主題

建立自訂分類的端點

下列範例顯示使用的 CreateEndpointAPI 作業 AWS CLI。

```
aws comprehend create-endpoint \
    --desired-inference-units number of inference units \
    --endpoint-name endpoint name \
    --model-arn arn:aws:comprehend:region:account-id:model/example \
    --tags Key=My1stTag,Value=Value1
```

Amazon Comprehend 回應與以下內容：

```
{
   "EndpointArn": "Arn"
}
```

執行即時自訂分類

為自訂分類模型建立端點後，您可以使用端點執行 ClassifyDocumentAPI 作業。您可以使用`text`或`bytes`參數提供文字輸入。使用參數輸入其他輸入類`bytes`型。

對於影像檔案和 PDF 檔案，您可以使用`DocumentReaderConfig`參數來取代預設文字萃取動作。如需詳細資訊，請參閱設定文字擷取選項

為了獲得最佳結果，請將輸入類型與分類器模型類型相匹配。如果您將原生文件提交至純文字模型，或提交純文字檔案至原生文件模型，則 API 回應會包含警告。如需詳細資訊，請參閱 訓練分類模型。

使用 AWS Command Line Interface

下面的實例演示如何使用分類文檔 CLI 命令。

使用將文字分類 AWS CLI

下列範例會針對文字區塊執行即時分類。

```
aws comprehend classify-document \
     --endpoint-arn arn:aws:comprehend:region:account-id:endpoint/endpoint name \
     --text 'From the Tuesday, April 16th, 1912 edition of The Guardian newspaper: The
 maiden voyage of the White Star liner Titanic,
     the largest ship ever launched ended in disaster. The Titanic started her trip
 from Southampton for New York on Wednesday. Late
     on Sunday night she struck an iceberg off the Grand Banks of Newfoundland. By
 wireless telegraphy she sent out signals of distress,
     and several liners were near enough to catch and respond to the call.'
```

Amazon Comprehend 回應與以下內容：

```
{
    "Classes": [
       {
          "Name": "string",
          "Score": 0.9793661236763
       }
    ]
 }
```

使用分類半結構化文件 AWS CLI

若要分析 PDF、Word 或影像檔案的自訂分類，請在bytes參數中使用輸入檔案執行classify-document指令。

下面的例子使用圖像作為輸入文件。它使用基於 64 對圖像文件字節進行編碼的fileb選項。若要取得更多資訊，請參閱《 AWS Command Line Interface 使用指南》中的二進位大型物件

此範例也會傳入名為的 JSON 檔案，config.json以設定文字擷取選項。

```
$ aws comprehend classify-document \
> --endpoint-arn arn \
> --language-code en \
```

```
> --bytes fileb://image1.jpg   \
> --document-reader-config file://config.json
```

config.json 檔案包含下列內容。

```
{
   "DocumentReadMode": "FORCE_DOCUMENT_READ_ACTION",
   "DocumentReadAction": "TEXTRACT_DETECT_DOCUMENT_TEXT"
}
```

Amazon Comprehend 回應與以下內容：

```
{
    "Classes": [
       {
          "Name": "string",
          "Score": 0.9793661236763
       }
    ]
}
```

如需詳細資訊，請參閱 Amazon Comprehend API 參考ClassifyDocument中的。

即時分析的輸出

文字輸入的輸出

對於文本輸入，輸出包括由分類器分析標識的類或標籤的列表。下面的例子顯示了兩個類的列表。

```
"Classes": [
  {
     "Name": "abc",
     "Score": 0.2757999897003174,
     "Page": 1
  },
  {
    "Name": "xyz",
    "Score": 0.2721000015735626,
```

```
    "Page": 1
  }
]
```

半結構化輸入的輸出

對於半結構化輸入文件或文字檔案，輸出可以包含下列其他欄位：

- DocumentMetadata — 擷取有關文件的資訊。元數據包括文檔中的頁面列表，其中包含從每個頁面中提取的字符數。如果請求包含Byte參數，則此字段存在於響應中。
- DocumentType — 輸入文件中每頁的文件類型。如果請求包含Byte參數，則此字段存在於響應中。
- Error — 系統在處理輸入文件時偵測到的頁面層級錯誤。如果系統沒有發生任何錯誤，則此欄位為空白。
- 警告 — 處理輸入文件時偵測到警告。如果輸入文件類型和與您指定的端點相關聯的模型類型不相符，則回應會包含警告。如果系統未產生任何警告，則此欄位為空白。

如需有關這些輸出欄位的詳細資訊，請參閱 Amazon Comprehend API 參考ClassifyDocument中的。

下列範例顯示單頁原生 PDF 輸入文件的輸出。

```
{
  "Classes": [
      {
          "Name": "123",
          "Score": 0.39570000767707825,
          "Page": 1
      },
      {
          "Name": "abc",
          "Score": 0.2757999897003174,
          "Page": 1
      },
      {
          "Name": "xyz",
          "Score": 0.2721000015735626,
          "Page": 1
      }
  ],
  "DocumentMetadata": {
      "Pages": 1,
```

```
      "ExtractedCharacters": [
          {
              "Page": 1,
              "Count": 2013
          }
      ]
  },
  "DocumentType": [
      {
          "Page": 1,
          "Type": "NATIVE_PDF"
      }
  ]
}
```

執行非同步工作

訓練自訂分類器之後，您可以使用非同步工作在一個批次中分析大型文件或多個文件。

自訂分類可接受各種輸入文件類型。如需詳細資訊，請參閱 非同步自訂分析的輸入。

如果您打算分析影像檔案或掃描的 PDF 文件，您的 IAM 政策必須授予許可，才能使用兩種 Amazon Textract API 方法 (DetectDocumentText 和 AnalyzeDocument)。Amazon Comprehend 文本提取過程中調用這些方法。如需政策範例，請參閱 執行文件分析動作所需的權限。

若要使用純文字模型分類半結構化文件 (影像、PDF 或 Docx 檔案)，請使用輸入格式。`one document per file`此外，請在StartDocumentClassificationJob請求中包含`DocumentReaderConfig`參數。

主題

- 用於非同步分析的檔案格式
- 自訂分類的分析工作 (主控台)
- 自訂分類 (API) 的分析工作
- 非同步分析工作的輸出

用於非同步分析的檔案格式

當您使用模型執行非同步分析時，您可以選擇輸入文件的格式：`One document per line`或`one document per file`. 您使用的格式取決於您要分析的文件類型，如下表所述。

描述	格式
輸入包含多個文件。每個檔案都包含一個輸入文件。此格式最適合收集大型文件，例如報紙文章或科學論文。 此外，對使用原生文件分類器的半結構化文件 (影像、PDF 或 Docx 檔案) 使用此格式。	每個檔案一個文件
輸入是一個或多個文件。檔案中的每一行都是個別的輸入文件。此格式最適合簡短文件，例如文字訊息或社交媒體貼文。	每行一個文件

每個檔案一個文件

使用`one document per file`格式，每個文件代表一個輸入文檔。

每行一個文件

使用該`One document per line`格式時，每個文檔都放置在單獨的行上，並且不使用標題。標籤不包含在每一行（因為您還不知道文檔的標籤）。文件的每一行（單個文檔的末尾）必須以換行符（LF，\ n），回車符（CR，\ r）或兩者結束（CRLF，\ r\ n）。請勿使用 UTF-8 行分隔符號 (u+2028) 來結束一行。

下面的例子顯示了輸入文件的格式。

```
Text of document 1 \n
Text of document 2 \n
Text of document 3 \n
Text of document 4 \n
```

對於任一格式，請對文字檔案使用 UTF-8 編碼。準備好檔案之後，請將它們放在用於輸入資料的 S3 儲存貯體中。

開始分類任務時，您可以為輸入資料指定此 Amazon S3 位置。URI 必須與您正在呼叫的 API 端點位於相同的區域中。URI 可以指向單個文件（如使用「每行一個文檔」方法時，也可以是數據文件集合的前綴。

例如，如果您使用 URIS3://bucketName/prefix，如果前置詞是單一檔案，Amazon Comprehend 會使用該檔案做為輸入。如果有多個檔案以前綴開頭，Amazon Comprehend 會使用所有這些檔案作為輸入。

授予 Amazon Comprehend 對包含文件收集和輸出檔案的 S3 儲存貯體的存取權。如需詳細資訊，請參閱 非同步作業所需的角色型權限。

自訂分類的分析工作 (主控台)

建立並訓練自訂文件分類器之後，您可以使用主控台執行模型的自訂分類工作。

建立自訂分類工作 (主控台)

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [分析工作]，然後選擇 [建立工作]。
3. 為分類工作命名。名稱在您的帳戶和目前的區域中必須是唯一的。
4. 在 [分析類型] 下選擇 [自訂分類]。
5. 從選取分類器中，選擇要使用的自訂分類器。
6. (選擇性) 如果您選擇加密 Amazon Comprehend 在處理任務時使用的資料，請選擇任務加密。然後選擇要使用與目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。
   - 如果您使用與目前帳戶相關聯的金鑰，請選擇 KMS 金鑰識別碼的金鑰識別碼。
   - 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ARN 下輸入金鑰識別碼的 AR N。

   > **Note**
   >
   > 如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱金鑰管理服務 (KMS)。

7. 在「輸入資料」下，輸入包含輸入文件的 Amazon S3 儲存貯體的位置，或選擇瀏覽 S3 瀏覽至該儲存貯體。此值區必須與您呼叫的 API 位於相同的區域。您用於分類工作存取權限的 IAM 角色必須具有 S3 儲存貯體的讀取權限。

   若要在訓練模型時達到最高準確度，請將輸入類型與分類器模型類型相符。如果您將原生文件提交至純文字模型，或將純文字文件提交至原生文件模型，分類器工作會傳回警告。如需詳細資訊，請參閱 訓練分類模型。

8. (選擇性) 對於輸入格式，您可以選擇輸入文件的格式。格式可以是每個檔案一個文件，也可以是單一檔案中每行一個文件。每行一個文件僅適用於文字文件。
9. (選擇性) 對於「文件」讀取模式，您可以覆寫預設的文字擷取動作。如需詳細資訊，請參閱 設定文字擷取選項。
10. 在「輸出資料」下，輸入 Amazon S3 儲存貯體的位置，Amazon Comprehend 應在其中寫入任務的輸出資料，或選擇瀏覽 S3 瀏覽至該儲存貯體。此值區必須與您呼叫的 API 位於相同的區域。您用於分類工作存取權限的 IAM 角色必須具有 S3 儲存貯體的寫入許可。
11. (選擇性) 如果您選擇加密工作的輸出結果，請選擇加密。然後選擇要使用與目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。
    - 如果您使用與目前帳戶相關聯的金鑰，請為 KMS 金鑰 ID 選擇金鑰別名或 ID。
    - 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ID 下輸入金鑰別名或 ID 的 ARN。
12. (選擇性) 若要從 VPC 將資源啟動至 Amazon Comprehend，請在 VPC 下輸入虛擬私人雲端識別碼，或從下拉式清單中選擇識別碼。
    1. 在 [子網路] 下選擇子網路。選取第一個子網路後，您可以選擇其他子網路。
    2. 在「安全性群組」下，選擇要使用的安全性群組 (如果您已指定安全性群組)。選取第一個安全性群組後，您可以選擇其他群組。

    > **Note**
    >
    > 當您將 VPC 與分類工作搭配使用時，`DataAccessRole`用於「建立」和「啟動」作業的必須將權限授與存取輸出值區的 VPC。

13. 選擇建立工作以建立文件分類工作。

自訂分類 (API) 的分析工作

建立並訓練自訂文件分類器之後，您可以使用分類器執行分析工作。

使用此作StartDocumentClassificationJob業開始分類未標籤的文件。您可以指定包含輸入文件的 S3 儲存貯體、輸出文件的 S3 儲存貯體，以及要使用的分類器。

若要在訓練模型時達到最高準確度，請將輸入類型與分類器模型類型相符。如果您將原生文件提交至純文字模型，或將純文字文件提交至原生文件模型，分類器工作會傳回警告。如需詳細資訊，請參閱 訓練分類模型。

StartDocumentClassificationJob是異步的。開始工作之後，請使用該DescribeDocumentClassificationJob作業來監視其進度。當回應中的Status欄位顯示時COMPLETED，您可以存取指定位置中的輸出。

主題

- 使用 AWS Command Line Interface
- 使用 AWS SDK for Java 或開發套件

使用 AWS Command Line Interface

下面的實例StartDocumentClassificationJob操作，以及其他自定義分類器 API 與. AWS CLI

下列範例會使用 Unix、Linux 和 macOS 的指令格式。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

使用作業執行自訂分類工StartDocumentClassificationJob作。

```
aws comprehend start-document-classification-job \
     --region region \
     --document-classifier-arn arn:aws:comprehend:region:account number:document-
classifier/testDelete \
     --input-data-config S3Uri=s3://S3Bucket/docclass/file
 name,InputFormat=ONE_DOC_PER_LINE \
     --output-data-config S3Uri=s3://S3Bucket/output \
     --data-access-role-arn arn:aws:iam::account number:role/resource name
```

使用該DescribeDocumentClassificationJob操作獲取有關具有作業 ID 的自定義分類器的信息。

```
aws comprehend describe-document-classification-job \
     --region region \
     --job-id job id
```

使用該操作列出您帳戶中的所有自訂分類工ListDocumentClassificationJobs作。

```
aws comprehend list-document-classification-jobs
     --region region
```

使用 AWS SDK for Java 或開發套件

如需如何啟動自訂分類器工作的 SDK 範例，請參閱使用開發套件啟動 Amazon Comprehend 文件分類任務 AWS。

非同步分析工作的輸出

分析任務完成後，會將結果儲存在您在請求中指定的 S3 儲存貯體中。

文字輸入的輸出

對於文字輸入文件的任一格式 (多類別或多標籤)，工作輸出都包含一個名為的單一檔案。`output.tar.gz`這是一個壓縮的歸檔文件，其中包含帶有輸出的文本文件。

多類輸出

當您使用在多類模式下訓練的分類器時，您的結果會顯示出來。`classes`這些都`classes`是用於在訓練分類器時創建類別集合的類。

如需有關這些輸出欄位的詳細資訊，請參閱 Amazon Comprehend API 參考ClassifyDocument中的。

下列範例會使用下列互斥類別。

```
DOCUMENTARY
SCIENCE_FICTION
ROMANTIC_COMEDY
SERIOUS_DRAMA
OTHER
```

如果您的輸入資料格式是每行一個文件，輸出檔案會針對輸入中的每一行包含一行。每一行都包含檔案名稱、從零開始的輸入行號，以及在文件中找到的一個或多個類別。最終結果是 Amazon Comprehend 確信個別執行個體已正確分類。

例如：

```
{"File": "file1.txt", "Line": "0", "Classes": [{"Name": "Documentary", "Score":
 0.8642}, {"Name": "Other", "Score": 0.0381}, {"Name": "Serious_Drama", "Score":
 0.0372}]}
{"File": "file1.txt", "Line": "1", "Classes": [{"Name": "Science_Fiction", "Score":
 0.5}, {"Name": "Science_Fiction", "Score": 0.0381}, {"Name": "Science_Fiction",
 "Score": 0.0372}]}
```

```
{"File": "file2.txt", "Line": "2", "Classes": [{"Name": "Documentary", "Score": 0.1},
 {"Name": "Documentary", "Score": 0.0381}, {"Name": "Documentary", "Score": 0.0372}]}
{"File": "file2.txt", "Line": "3", "Classes": [{"Name": "Serious_Drama", "Score":
 0.3141}, {"Name": "Other", "Score": 0.0381}, {"Name": "Other", "Score": 0.0372}]}
```

如果您的輸入資料格式是每個檔案一個文件，則輸出檔案會針對每個文件包含一行。每一行都有檔案的名稱，以及在文件中找到的一個或多個類別。最後，Amazon Comprehend 可以準確地將個別執行個體分類。

例如：

```
{"File": "file0.txt", "Classes": [{"Name": "Documentary", "Score": 0.8642}, {"Name":
 "Other", "Score": 0.0381}, {"Name": "Serious_Drama", "Score": 0.0372}]}
{"File": "file1.txt", "Classes": [{"Name": "Science_Fiction", "Score": 0.5}, {"Name":
 "Science_Fiction", "Score": 0.0381}, {"Name": "Science_Fiction", "Score": 0.0372}]}
{"File": "file2.txt", "Classes": [{"Name": "Documentary", "Score": 0.1}, {"Name":
 "Documentary", "Score": 0.0381}, {"Name": "Domentary", "Score": 0.0372}]}
{"File": "file3.txt", "Classes": [{"Name": "Serious_Drama", "Score": 0.3141}, {"Name":
 "Other", "Score": 0.0381}, {"Name": "Other", "Score": 0.0372}]}
```

多標籤輸出

當您使用在多標籤模式下訓練的分類器時，您的結果會顯示出來。`labels`這些都`labels`是用於在訓練分類器時創建類別集的標籤。

下列範例使用這些唯一的標籤。

```
SCIENCE_FICTION
ACTION
DRAMA
COMEDY
ROMANCE
```

如果您的輸入資料格式是每行一個文件，輸出檔案會針對輸入中的每一行包含一行。每一行都包含檔案名稱、從零開始的輸入行號，以及在文件中找到的一個或多個類別。最終結果是 Amazon Comprehend 確信個別執行個體已正確分類。

例如：

```
{"File": "file1.txt", "Line": "0", "Labels": [{"Name": "Action", "Score": 0.8642},
 {"Name": "Drama", "Score": 0.650}, {"Name": "Science Fiction", "Score": 0.0372}]}
```

```
{"File": "file1.txt", "Line": "1", "Labels": [{"Name": "Comedy", "Score": 0.5},
 {"Name": "Action", "Score": 0.0381}, {"Name": "Drama", "Score": 0.0372}]}
{"File": "file1.txt", "Line": "2", "Labels": [{"Name": "Action", "Score": 0.9934},
 {"Name": "Drama", "Score": 0.0381}, {"Name": "Action", "Score": 0.0372}]}
{"File": "file1.txt", "Line": "3", "Labels": [{"Name": "Romance", "Score": 0.9845},
 {"Name": "Comedy", "Score": 0.8756}, {"Name": "Drama", "Score": 0.7723}, {"Name":
 "Science_Fiction", "Score": 0.6157}]}
```

如果您的輸入資料格式是每個檔案一個文件，則輸出檔案會針對每個文件包含一行。每一行都有檔案的名稱，以及在文件中找到的一個或多個類別。最後，Amazon Comprehend 可以準確地將個別執行個體分類。

例如：

```
{"File": "file0.txt", "Labels": [{"Name": "Action", "Score": 0.8642}, {"Name": "Drama",
 "Score": 0.650}, {"Name": "Science Fiction", "Score": 0.0372}]}
{"File": "file1.txt", "Labels": [{"Name": "Comedy", "Score": 0.5}, {"Name": "Action",
 "Score": 0.0381}, {"Name": "Drama", "Score": 0.0372}]}
{"File": "file2.txt", "Labels": [{"Name": "Action", "Score": 0.9934}, {"Name": "Drama",
 "Score": 0.0381}, {"Name": "Action", "Score": 0.0372}]}
{"File": "file3.txt", "Labels": [{"Name": "Romance", "Score": 0.9845}, {"Name":
 "Comedy", "Score": 0.8756}, {"Name": "Drama", "Score": 0.7723}, {"Name":
 "Science_Fiction", "Score": 0.6157}]}
```

半結構化輸入文件的輸出

對於半結構化輸入文件，輸出可以包含下列其他欄位：

- DocumentMetadata — 擷取有關文件的資訊。元數據包括文檔中的頁面列表，其中包含從每個頁面中提取的字符數。如果請求包含`Byte`參數，則此字段存在於響應中。
- DocumentType — 輸入文件中每頁的文件類型。如果請求包含`Byte`參數，則此字段存在於響應中。
- Error — 系統在處理輸入文件時偵測到的頁面層級錯誤。如果系統沒有發生任何錯誤，則此欄位為空白。

如需有關這些輸出欄位的詳細資訊，請參閱 Amazon Comprehend API 參考ClassifyDocument中的。

下列範例顯示兩頁掃描 PDF 檔案的輸出。

```
[{ #First page output
```

```
    "Classes": [
        {
            "Name": "__label__2 ",
            "Score": 0.9993996620178223
        },
        {
            "Name": "__label__3 ",
            "Score": 0.0004330444789957255
        }
    ],
    "DocumentMetadata": {
        "PageNumber": 1,
        "Pages": 2
    },
    "DocumentType": "ScannedPDF",
    "File": "file.pdf",
    "Version": "VERSION_NUMBER"
},
#Second page output
{
    "Classes": [
        {
            "Name": "__label__2 ",
            "Score": 0.9993996620178223
        },
        {
            "Name": "__label__3 ",
            "Score": 0.0004330444789957255
        }
    ],
    "DocumentMetadata": {
        "PageNumber": 2,
        "Pages": 2
    },
    "DocumentType": "ScannedPDF",
    "File": "file.pdf",
    "Version": "VERSION_NUMBER"
}]
```

自訂實體辨識

自訂實體辨識功能可協助您識別不在預設泛型實體類型中的特定新實體類型，藉此擴展 Amazon Comprehend 的功能。這表示您可以分析文件並擷取符合您特定需求的產品代碼或業務特定實體等實體。

自行建置精確的自訂實體辨識器可能是一個複雜的程序，需要準備大量手動註解的訓練文件集，以及選取正確的演算法和參數來進行模型訓練。Amazon Comprehend 透過提供自動註釋和模型開發來建立自訂實體辨識模型，協助降低複雜性。

建立自訂實體辨識模型比使用字串比對或規則運算式從文件擷取實體更有效的方法。例如，若要擷取文件中的 ENGINEER 名稱，很難列舉所有可能的名稱。此外，如果沒有上下文，區分工程師名稱和分析師名稱也很難。自訂圖元辨識模型可以瞭解這些名稱可能出現的前後關聯。此外，字串比對不會偵測具有錯別字或遵循新命名慣例的實體，但這可以使用自訂模型。

建立自訂模型有兩個選項：

1. 註釋 — 提供包含已註解實體的資料集，以進行模型訓練。
2. 實體清單 (僅限純文字) — 提供實體及其類型標籤的清單 (例如，以`PRODUCT_CODES`及一組包含這些實體的未註釋文件，以進行模型訓練。

當您使用帶註解的 PDF 檔案建立自訂實體辨識器時，您可以使用具有多種輸入檔案格式的辨識器：純文字、影像檔案 (JPG、PNG、TIFF)、PDF 檔案和 Word 文件，無需預先處理或文件平面化。Amazon Comprehend 不支持圖像文件或 Word 文檔的註釋。

> **Note**
>
> 使用帶註釋 PDF 檔案的自訂實體辨識器僅支援英文文件。

您一次最多可以訓練 25 個自訂實體的模型。如需詳細資訊，請參閱準則與配額頁面。

模型訓練完畢後，您可以使用模型進行即時實體偵測和實體偵測工作。

主題

- 準備實體辨識器訓練資料
- 訓練自訂實體辨識器模型

- 執行即時自訂辨識器分析
- 執行自訂實體辨識的分析工作

準備實體辨識器訓練資料

若要訓練成功的自訂實體辨識模型，請務必提供高品質資料作為輸入的模型訓練員。如果沒有良好的數據，模型將無法學習如何正確識別實體。

您可以選擇以下兩種方式之一，將資料提供給 Amazon Comprehend，以訓練自訂實體辨識模型：

- 實體清單 — 列出特定實體，讓 Amazon Comprehend 能夠訓練識別您的自訂實體。注意：實體清單只能用於純文字文件。
- 註釋 — 在多個文件中提供實體的位置，以便 Amazon Comprehend 可以在實體及其內容上進行訓練。要創建用於分析圖像文件，PDF 或 Word 文檔的模型，您必須使用 PDF 註釋訓練識別器。

在這兩種情況下，Amazon Comprehend 都會學習文件的種類和實體出現的環境，並建立辨識器，以便在您分析文件時一般偵測新實體。

當您建立自訂模型 (或訓練新版本) 時，您可以提供測試資料集。如果您未提供測試資料，Amazon Comprehend 會保留 10% 的輸入文件來測試模型。Amazon Comprehend 訓練與其餘文件的模型。

如果您為註釋訓練集提供測試資料集，測試資料必須至少包含建立要求中所指定之每個實體類型的一個註解。

主題

- 何時使用註釋與實體清單
- 實體清單 (僅限純文字)
- 註釋

何時使用註釋與實體清單

建立註釋比建立實體清單需要更多的工作，但產生的模型可能會更精確。使用實體清單會更快速且工作耗用較少，但結果不太精確且不太準確。這是因為註釋為 Amazon Comprehend 提供了更多的上下文，以便在訓練模型時使用。如果沒有這種情況，Amazon Comprehend 會在嘗試識別實體時產生較多的誤報。

在某些情況下，它使更具商業意義，以避免使用註釋的更高費用和工作負載。例如，John Johnson 這個名字對您的搜尋很重要，但這個名稱是否確切的個人並不相關。或者，使用實體列表時的指標足以為您提供所需的識別器結果。在這種情況下，使用實體列表可以是更有效的選擇。

我們建議在下列情況下使用註釋模式：

- 如果您打算對影像檔案、PDF 或 Word 文件執行推論。在這個案例中，您會使用帶註解的 PDF 檔案來訓練模型，並使用模型來執行影像檔、PDF 和 Word 文件的推論工作。
- 當實體的含義可能是模棱兩可的和上下文相關的。例如，術語 Amazon 可以指巴西的河流，或在線零售商 Amazon.com。當您建立自訂實體辨識器來識別商業實體 (例如 Amazon) 時，您應該使用註解而不是實體清單，因為此方法能夠更好地使用內容來尋找實體。
- 當您舒適地設置一個獲取註釋的過程時，這可能需要一些努力。

我們建議在下列情況下使用實體清單：

- 當你已經有一個實體列表，或者當它是相對容易的組成一個完整的實體列表。如果您使用實體清單，清單應該是完整的，或至少涵蓋了您提供訓練之文件中可能出現的大多數有效實體。
- 對於首次使用的用戶，通常建議使用實體列表，因為這需要比構建註釋更小的努力。但是，請務必注意，訓練過的模型可能不像您使用註釋一樣精確。

實體清單 (僅限純文字)

若要使用實體清單來訓練模型，您需要提供兩個資訊：實體名稱清單及其對應的自訂實體類型，以及您希望在其中顯示實體的未註解文件集合。

當您提供實體清單時，Amazon Comprehend 會使用智慧型演算法偵測文件中實體的出現次數，以做為訓練自訂實體辨識器模型的基礎。

對於實體清單，請在實體清單中為每個實體類型提供至少 25 個實體相符項目。

自訂實體辨識的實體清單需要逗號分隔值 (CSV) 檔案，其中包含下列欄：

- 文字 — 項目範例的文字，與隨附文件語料庫中所顯示的完全相同。
- 類型-客戶定義的實體類型。實體類型必須為大寫、下劃線分隔的字串，例如管理員或 SIENOR_MANAGER。每個模型最多可訓練 25 個實體類型。

該文件`documents.txt`包含四行：

```
Jo Brown is an engineer in the high tech industry.
John Doe has been a engineer for 14 years.
Emilio Johnson is a judge on the Washington Supreme Court.
Our latest new employee, Jane Smith, has been a manager in the industry for 4 years.
```

具有實體清單的 CSV 檔案具有以下幾行：

```
Text, Type
Jo Brown, ENGINEER
John Doe, ENGINEER
Jane Smith, MANAGER
```

> **Note**
>
> 在實體清單中，埃米利奧·約翰遜的項目不存在，因為它不包含工程師或經理實體。

建立您的資料檔案

您的實體清單必須放在正確設定的 CSV 檔案中，因此您的實體清單檔案出現問題的機會很小。若要手動設定 CSV 檔案，必須符合下列條件：

- UTF-8 編碼必須明確指定，即使在大多數情況下將其用作默認編碼也是如此。
- 它必須包括列名：`Type`和`Text`。

我們強烈建議以程式設計方式產生 CSV 輸入檔案，以避免潛在的問題。

下列範例會使用 Python 為上述註解產生 CSV：

```
import csv
with open("./entitylist/entitylist.csv", "w", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow(["Text", "Type"])
    csv_writer.writerow(["Jo Brown", " ENGINEER"])
    csv_writer.writerow(["John Doe", " ENGINEER"])
    csv_writer.writerow(["Jane Smith", " MANAGER"])
```

最佳實務

在使用實體列表時，需要考慮許多事項以獲得最佳結果，包括：

- 清單中實體的順序對模型訓練沒有影響。
- 使用涵蓋未註釋文件語料庫中所提及之正面實體範例的 80%-100% 的實體清單項目。
- 移除常用字詞和片語，避免符合文件語料庫中非實體的實體範例。即使是少數不正確的相符項目，也會顯著影響產生的模型的準確性。例如，像實體清單中的字詞會產生大量相符項目，這些相符項目不太可能是您要尋找的實體，因此會極大地影響您的準確性。
- 輸入數據不應包含重複項。存在重複樣本可能會導致測試集污染，因此對訓練過程、模型指標和行為產生負面影響。
- 盡可能提供類似實際使用案例的文件。不要將玩具數據或合成數據用於生產系統。輸入數據應盡可能多樣化，以避免過度擬合並幫助底層模型更好地概括在實際示例中。
- 實體清單區分大小寫，目前不支援規則運算式。但是，即使實體與實體清單中提供的外殼不完全相符，訓練過的模型通常仍可辨識實體。
- 如果您的實體是另一個實體的子字串 (例如「Smith」和「Jane Smith」)，請在實體清單中同時提供這兩者。

可以在以下位置找到其他建議 改善自訂實體辨識器效能

註釋

註釋會將您的自訂實體類型與訓練文件中的位置相關聯，來標示前後關聯中的實體。

透過與文件一起提交註釋，您可以提高模型的準確性。有了 Annotations，您不只是提供您要尋找的實體的位置，而且還為您要尋找的自訂實體提供更準確的內容。

例如，如果您正在搜尋名稱 John Johnson，而實體類型為「判斷」，提供您的註解可能會協助模型瞭解您要尋找的人是法官。如果它能夠使用上下文，那麼 Amazon Comprehend 將找不到名為約翰·約翰遜的人是律師或證人。Amazon Comprehend 將在不提供註釋的情況下建立自己的註釋版本，但在僅包括評審方面並不會有效。提供您自己的註釋可能有助於獲得更好的結果，並在提取自定義實體時生成能夠更好地利用上下文的模型。

主題

- 註釋的最小數目
- 註解最佳作法

- 純文字註解檔
- PDF 註釋文件
- 為 PDF 檔案加上註解

註釋的最小數目

訓練模型所需的最小輸入文件和註釋數量取決於註釋的類型。

注釋

若要建立分析影像檔案、PDF 或 Word 文件的模型，請使用 PDF 註解訓練辨識器。對於 PDF 註釋，每個實體至少提供 250 個輸入文件和至少 100 個註釋。

如果您提供測試資料集，測試資料必須至少包含建立要求中指定的每個實體類型的一個註解。

純文字註解

若要建立用於分析文字文件的模型，您可以使用純文字註解來訓練辨識器。

對於純文字註釋，請至少提供三個帶註解的輸入文件，每個實體至少提供 25 個註釋。如果您提供的註解總數少於 50 個，Amazon Comprehend 會保留 10% 以上的輸入文件來測試模型 (除非您在訓練請求中提供了測試資料集)。不要忘記，最小文檔語料庫大小為 5 KB。

如果您的輸入只包含幾個訓練文件，您可能會遇到錯誤，即訓練輸入資料包含的文件太少，提及其中一個實體。使用提及實體的其他文件再次提交工作。

如果您提供測試資料集，測試資料必須至少包含建立要求中指定的每個實體類型的一個註解。

如需如何使用小型資料集對模型進行基準測試的範例，請參閱 Amazon Comprehend 宣布降低 AWS部落格網站上自訂實體辨識的註釋限制。

註解最佳作法

在使用註釋時，有許多事情需要考慮以獲得最佳結果，包括：

- 小心註釋您的數據，並驗證您是否對實體的每次提及進行註釋。不精確的註釋可能會導致結果不佳。
- 輸入數據不應包含重複項，例如要註釋的 PDF 的副本。存在重複樣本可能會導致測試集污染，並可能對訓練過程、模型指標和模型行為產生負面影響。
- 確保您的所有文檔都被註釋，並且沒有註釋的文檔是由於缺乏合法實體，而不是由於疏忽。例如，如果您有一份文件上寫著「J Doe 擔任工程師已有 14 年」，則您還應該為「J Doe」和「Doe」提供註

釋。如果不這樣做會使模型混淆，並可能導致模型無法將「J Doe」識別為「工程師」。這應該是相同的文檔和跨文檔一致的。

- 一般來說，更多的註釋會導致更好的結果。
- 您可以使用最少數量的文件和註釋來訓練模型，但是新增資料通常可以改善模型。我們建議將已註解資料的數量增加 10%，以提高模型的精確度。您可以在測試資料集上執行推論，該資料集保持不變，而且可由不同的模型版本進行測試。然後，您可以比較連續模型版本的量度。
- 盡可能提供與實際使用案例相似的文件。應避免具有重複模式的合成數據。輸入的數據應盡可能多樣化，以避免過度擬合，並幫助底層模型更好地概括在真實的例子。
- 重要的是，文檔在字數方面應該是多樣化的。例如，如果訓練資料中的所有文件都很短，則產生的模型可能難以預測較長文件中的實體。
- 嘗試為訓練提供與實際偵測自訂實體時預期使用的相同資料分佈 (推論時間)。例如，在推論時，如果您希望傳送給我們沒有實體的文件，這也應該是訓練文件集的一部分。

如需其他建議，請參閱改善自訂實體辨識器效能。

純文字註解檔

對於純文字註釋，您可以建立包含註釋清單的逗號分隔值 (CSV) 檔案。如果您的訓練檔案輸入格式是每行一個文件，則 CSV 檔案必須包含下列欄。

檔案	折線圖	開始偏移	終點偏移	Type
包含文件的檔案名稱。例如，如果其中一個文件檔案位於`s3://my-S3-bucket/test-files/documents.txt`，則`File`欄中的值將為`documents.txt`。您必須包含檔案副檔名 (在本例中為	包含圖元的行號。如果您的輸入格式是每個檔案一個文件，請省略此欄。	顯示實體開始位置的輸入文字中的字元偏移量 (相對於行的開頭)。第一個字元位於 0 的位置。	顯示實體結束位置的輸入文字中的字元偏移量。	客戶定義的實體類型。實體類型必須是大寫、下劃線分隔的字串。我們建議使用描述性實體類型`MANAGER`，例如`SENIOR_MANAGER`、或`PRODUCT_CODE`。每個模型最多可訓練 25 個實體類型。

檔案	折線圖	開始偏移	終點偏移	Type
'.txt') 做為檔案名稱的一部分。				

如果訓練檔案輸入格式是每個檔案一個文件，則省略行號欄，而「開始位移」(Begin Offset) 和「結束」(End) 位移值是實體從文件開頭算起的偏移量。

下列範例適用於每行一個文件。檔案documents.txt包含四行 (列 0、1、2 和 3)：

```
Diego Ramirez is an engineer in the high tech industry.
Emilio Johnson has been an engineer for 14 years.
J Doe is a judge on the Washington Supreme Court.
Our latest new employee, Mateo Jackson, has been a manager in the industry for 4 years.
```

包含註釋清單的 CSV 檔案如下所示：

```
File, Line, Begin Offset, End Offset, Type
documents.txt, 0, 0, 13, ENGINEER
documents.txt, 1, 0, 14, ENGINEER
documents.txt, 3, 25, 38, MANAGER
```

> **Note**
>
> 在註釋檔案中，包含圖元的行號以 0 行開始。在此範例中，CSV 檔案不包含第 2 行的項目，因為第 2 行中沒有實體documents.txt。

建立您的資料檔案

請務必將註解放在正確設定的 CSV 檔案中，以降低發生錯誤的風險。若要手動設定 CSV 檔案，必須符合下列條件：

- UTF-8 編碼必須明確指定，即使在大多數情況下將其用作默認編碼也是如此。
- 第一行包含欄標題：File, Line (選擇性)、Begin Offset、End Offset、Type。

我們強烈建議您以程式設計方式產生 CSV 輸入檔案，以避免潛在的問題。

下列範例會使用 Python 為先前顯示的註解產生 CSV：

```
import csv
with open("./annotations/annotations.csv", "w", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow(["File", "Line", "Begin Offset", "End Offset", "Type"])
    csv_writer.writerow(["documents.txt", 0, 0, 11, "ENGINEER"])
    csv_writer.writerow(["documents.txt", 1, 0, 5, "ENGINEER"])
    csv_writer.writerow(["documents.txt", 3, 25, 30, "MANAGER"])
```

PDF 註釋文件

對於 PDF 註釋，您可以使用 SageMaker Ground Truth 在增強資訊清單檔案中建立標籤資料集。Ground Truth 是一項資料標籤服務，可協助您 (或您雇用的員工) 為機器學習模型建置訓練資料集。Amazon Comprehend 接受擴增資訊清單檔案做為自訂模型的訓練資料。當您使用 Amazon Comprehend 主控台或 API 動作建立自訂實體辨識器時，可以提供這些檔案。CreateEntityRecognizer

您可以使用 Ground Truth 內建工作類型「命名實體辨識」來建立標籤工作，讓 Worker 識別文字中的實體。若要進一步了解，請參閱 Amazon SageMaker 開發人員指南中的具名實體辨識。要了解有關 Amazon SageMaker Ground Truth 的更多信息，請參閱使用 Amazon SageMaker Ground Truth 來標記數據。

> **Note**
>
> 使用「Ground Truth」，您可以定義重疊的標籤（與多個標籤相關聯的文字）。不過，Amazon Comprehend 實體辨識不支援重疊的標籤。

增強的資訊清單檔案採用 JSON 行格式。在這些檔案中，每一行都是完整的 JSON 物件，其中包含訓練文件及其相關聯的標籤。以下示例是一個增強的清單文件，它訓練實體識別器以檢測文本中提到的個人的職業：

```
{"source":"Diego Ramirez is an engineer in the high tech
 industry.","NamedEntityRecognitionDemo":{"annotations":{"entities":
[{"endOffset":13,"startOffset":0,"label":"ENGINEER"}],"labels":
[{"label":"ENGINEER"}]}},"NamedEntityRecognitionDemo-metadata":
{"entities":[{"confidence":0.92}],"job-name":"labeling-job/
namedentityrecognitiondemo","type":"groundtruth/text-span","creation-
date":"2020-05-14T21:45:27.175903","human-annotated":"yes"}}
{"source":"J Doe is a judge on the Washington Supreme
 Court.","NamedEntityRecognitionDemo":{"annotations":{"entities":
```

```
[{"endOffset":5,"startOffset":0,"label":"JUDGE"}],"labels":
[{"label":"JUDGE"}]}},"NamedEntityRecognitionDemo-metadata":
{"entities":[{"confidence":0.72}],"job-name":"labeling-job/
namedentityrecognitiondemo","type":"groundtruth/text-span","creation-
date":"2020-05-14T21:45:27.174910","human-annotated":"yes"}}
{"source":"Our latest new employee, Mateo Jackson, has been a manager in
 the industry for 4 years.","NamedEntityRecognitionDemo":{"annotations":
{"entities":[{"endOffset":38,"startOffset":26,"label":"MANAGER"}],"labels":
[{"label":"MANAGER"}]}},"NamedEntityRecognitionDemo-metadata":
{"entities":[{"confidence":0.91}],"job-name":"labeling-job/
namedentityrecognitiondemo","type":"groundtruth/text-span","creation-
date":"2020-05-14T21:45:27.174035","human-annotated":"yes"}}
```


這個 JSON 行文件中的每一行都是一個完整的 JSON 對象，其中的屬性包括文檔文本，註釋和其他元數據 Ground Truth。下列範例是增強資訊清單檔案中的單一 JSON 物件，但已格式化以提高可讀性：


```
{
  "source": "Diego Ramirez is an engineer in the high tech industry.",
  "NamedEntityRecognitionDemo": {
    "annotations": {
      "entities": [
        {
          "endOffset": 13,
          "startOffset": 0,
          "label": "ENGINEER"
        }
      ],
      "labels": [
        {
          "label": "ENGINEER"
        }
      ]
    }
  },
  "NamedEntityRecognitionDemo-metadata": {
    "entities": [
      {
        "confidence": 0.92
      }
    ],
    "job-name": "labeling-job/namedentityrecognitiondemo",
    "type": "groundtruth/text-span",
    "creation-date": "2020-05-14T21:45:27.175903",
```

```
    "human-annotated": "yes"
  }
}
```

在此範例中，source屬性提供訓練文件的文字，而NamedEntityRecognitionDemo屬性則提供文字中實體的註釋。NamedEntityRecognitionDemo屬性的名稱是任意的，當您在 Ground Truth 中定義標籤工作時，您可以提供您選擇的名稱。

在這個範例中，NamedEntityRecognitionDemo屬性是 label 屬性名稱，這是提供「Ground Truth」Worker 指派給訓練資料之標籤的屬性。當您將訓練資料提供給 Amazon Comprehend 時，您必須指定一個或多個標籤屬性名稱。您指定的屬性名稱數量取決於您的增強資訊清單檔案是單一標籤工作的輸出還是連結標籤工作的輸出。

如果您的檔案是單一標籤工作的輸出，請指定在 Ground Truth 中建立工作時使用的單一標籤屬性名稱。

如果您的檔案是鏈結標籤工作的輸出，請為鏈結中的一或多個工作指定標籤屬性名稱。每個標籤屬性名稱均提供個別工作的註釋。您最多可以為由鏈結標籤工作產生的增強資訊清單檔案指定其中 5 個屬性。

在增強資訊清單檔案中，label 屬性名稱通常在source索引鍵之後。如果檔案是鏈結工作的輸出，則會有多個標籤屬性名稱。當您將訓練資料提供給 Amazon Comprehend 時，請僅提供那些包含與模型相關註釋的屬性。請勿指定以「-metadata」結尾的屬性。

如需有關鏈結標籤任務的詳細資訊，以及它們產生的輸出範例，請參閱 Amazon SageMaker 開發人員指南中的鏈結標籤任務。

為 PDF 檔案加上註解

在「 SageMaker Ground Truth」中為訓練 PDF 加上註解之前，請先完成下列先決條件：

- 安裝蟒蛇 3.8.x
- 安裝 JQ
- 安裝 AWS CLI 面

  如果您使用的是 us-east-1 區域，則可以略過安裝 AWS CLI，因為它已經與您的 Python 環境一起安裝。在這種情況下，您可以創建一個虛擬環境以在 AWS Cloud9 中使用 Python 3.8。
- 設定您的AWS 認證
- 創建一個私有的 SageMaker Ground Truth 員工來支持註釋

在安裝期間使用時，請務必記錄您在新的私人員工中選擇的工作團隊名稱。

主題

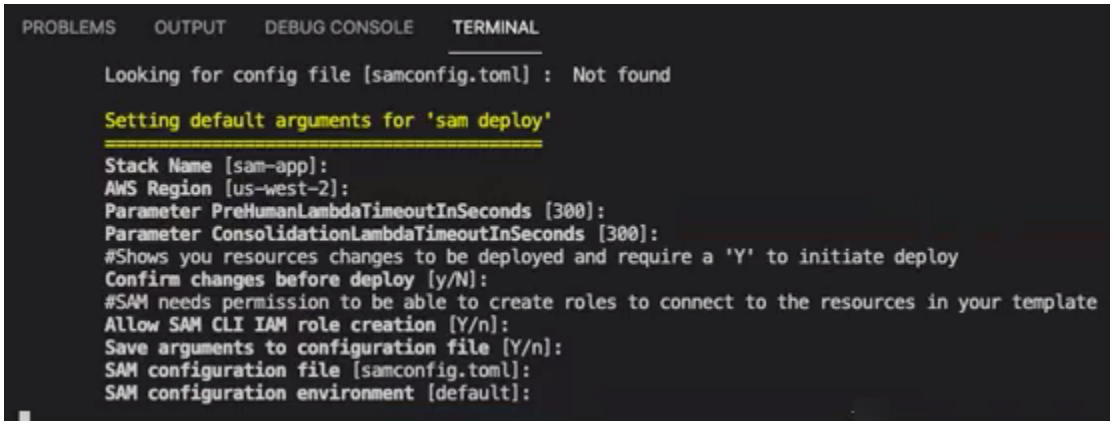
- 設定您的環境
- 將 PDF 上傳到 S3 儲存貯體
- 建立註釋工作
- 用 SageMaker Ground Truth 註釋

設定您的環境

1. 如果使用視窗，請安裝賽格溫；如果使用 Linux 或 Mac，請跳過此步驟。
2. 從中下載註釋人工因素 GitHub。解壓縮檔案。
3. 在終端機視窗中，瀏覽至解壓縮的資料夾 (amazon-comprehend-semi-structured-documents-annotation-tools-main)。
4. 此資料夾包含您執行`Makefiles`的選項，以安裝相依性、設定 Python 虛擬環境，以及部署必要的資源。檢閱讀我檔案以做出選擇。
5. 推薦的選項使用單個命令將所有依賴項安裝到 virtualenv 中，從模板構建 AWS CloudFormation 堆棧，並通過交互式指導將堆棧部署到您 AWS 帳戶 的堆棧中。執行以下命令：

   ```
   make ready-and-deploy-guided
   ```

   此指令會顯示一組組態選項。確保你的 AWS 區域 是正確的。對於所有其他欄位，您可以接受預設值或填入自訂值。如果您修改 AWS CloudFormation 堆棧名稱，請在接下來的步驟中根據需要將其寫下來。



該 CloudFormation 堆棧創建和管理註釋工具所需的 AWS lambda 表達式，AWS IAM 角色和 AWS S3 存儲桶。

您可以在 CloudFormation 主控台的堆疊詳細資料頁面中檢閱這些資源。

6. 指令會提示您開始部署。 CloudFormation 創建指定區域中的所有資源。

當 CloudFormation 堆棧狀態轉換為創建完成時，資源就可以使用了。

將 PDF 上傳到 S3 儲存貯體

在「設定」區段中，您部署了建立名為 comprehend-semi-structured-documents-$ {AWS::Region}-${AWS::AccountId} 的 S3 儲存貯體的 CloudFormation 堆疊。您現在可以將來源 PDF 文件上傳到此值區。

> **Note**
>
> 此儲存貯體包含標籤工作所需的資料。Lambda 執行角色政策授予 Lambda 函數存取此儲存貯體的權限。
> 您可以使用「SemiStructuredDocumentsS3Bucket」密鑰在CloudFormation 堆棧詳細信息中找到 S3 存儲桶名稱。

1. 在 S3 儲存貯體中建立新資料夾。將此新資料夾命名為「src」。
2. 將您的 PDF 源文件添加到「src」文件夾中。在稍後的步驟中，您可以註釋這些文件以訓練您的識別器。
3. (選擇性) 以下是 AWS CLI 範例，可用來將來源文件從本機目錄上傳至 S3 儲存貯體：

```
aws s3 cp --recursive local-path-to-your-source-docs s3://deploy-guided/src/
```

或者，使用您的地區和帳戶 ID：

```
aws s3 cp --recursive local-path-to-your-source-docs s3://deploy-guided-Region-AccountID/src/
```

4. 您現在擁有一個私有的 G SageMaker round Truth 工作人員，並已將來源檔案上傳到 S3 儲存貯體、部署指南 /src/；您已準備好開始註解。

建立註釋工作

bin目錄中的 comprehend-ssie-annotation-tool-cli.py 指令碼是簡單的包裝命令，可簡化 SageMaker Ground Truth 標籤工作的建立。python 腳本從 S3 存儲桶讀取源文檔，並創建一個對應的單頁清單文件，每行包含一個源文檔。然後，指令碼會建立標籤工作，這需要資訊清單檔案做為輸入。

python 指令碼使用您在 [設定] 區段中設定的 S3 儲存貯體和 CloudFormation 堆疊。指令碼的必要輸入參數包括：

- 輸入 S3 路徑：S3 Uri 到您上傳到 S3 儲存貯體的來源文件。例如：s3://deploy-guided/src/。您也可以將「地區」和「帳戶 ID」新增至此路徑。例如：s3://deploy-guided-Region-AccountID/src/。
- CF 名稱： CloudFormation 堆棧名稱。如果您使用堆棧名稱的默認值，則您的 cfn-name 是 sam-app。
- work-team-name: 您在《 SageMaker Ground Truth》中建立私人勞動力時所建立的勞動力名稱。
- job-name-prefix：「 SageMaker Ground Truth」標籤工作的前置詞。請注意，此欄位的字元限制為 29 個字元。時間戳記會附加至此值。例如：my-job-name-20210902T232116。
- 實體類型：您要在標籤工作期間使用的實體，以逗號分隔。此清單必須包含您要在訓練資料集中註解的所有實體。「Ground Truth」標籤工作僅顯示這些實體，供註解者標示 PDF 文件中的內容。

若要檢視指令碼支援的其他引數，請使用-h選項來顯示說明內容。

- 使用上一個清單中所述的輸入參數執行下列指令碼。

```
python bin/comprehend-ssie-annotation-tool-cli.py \
--input-s3-path s3://deploy-guided-Region-AccountID/src/ \
--cfn-name sam-app \
--work-team-name my-work-team-name \
--region us-east-1 \
--job-name-prefix my-job-name-20210902T232116 \
```

```
--entity-types "EntityA, EntityB, EntityC" \
--annotator-metadata "key=info,value=sample,key=Due Date,value=12/12/2021"
```

該腳本產生以下輸出：

```
Downloaded files to temp local directory /tmp/a1dc0c47-0f8c-42eb-9033-74a988ccc5aa
Deleted downloaded temp files from /tmp/a1dc0c47-0f8c-42eb-9033-74a988ccc5aa
Uploaded input manifest file to s3://comprehend-semi-structured-documents-
us-west-2-123456789012/input-manifest/my-job-name-20220203-labeling-
job-20220203T183118.manifest
Uploaded schema file to s3://comprehend-semi-structured-documents-us-
west-2-123456789012/comprehend-semi-structured-docs-ui-template/my-job-
name-20220203-labeling-job-20220203T183118/ui-template/schema.json
Uploaded template UI to s3://comprehend-semi-structured-documents-us-
west-2-123456789012/comprehend-semi-structured-docs-ui-template/my-job-
name-20220203-labeling-job-20220203T183118/ui-template/template-2021-04-15.liquid
Sagemaker GroundTruth Labeling Job submitted: arn:aws:sagemaker:us-
west-2:123456789012:labeling-job/my-job-name-20220203-labeling-job-20220203t183118
(amazon-comprehend-semi-structured-documents-annotation-tools-main)
 user@3c063014d632 amazon-comprehend-semi-structured-documents-annotation-tools-
main %
```

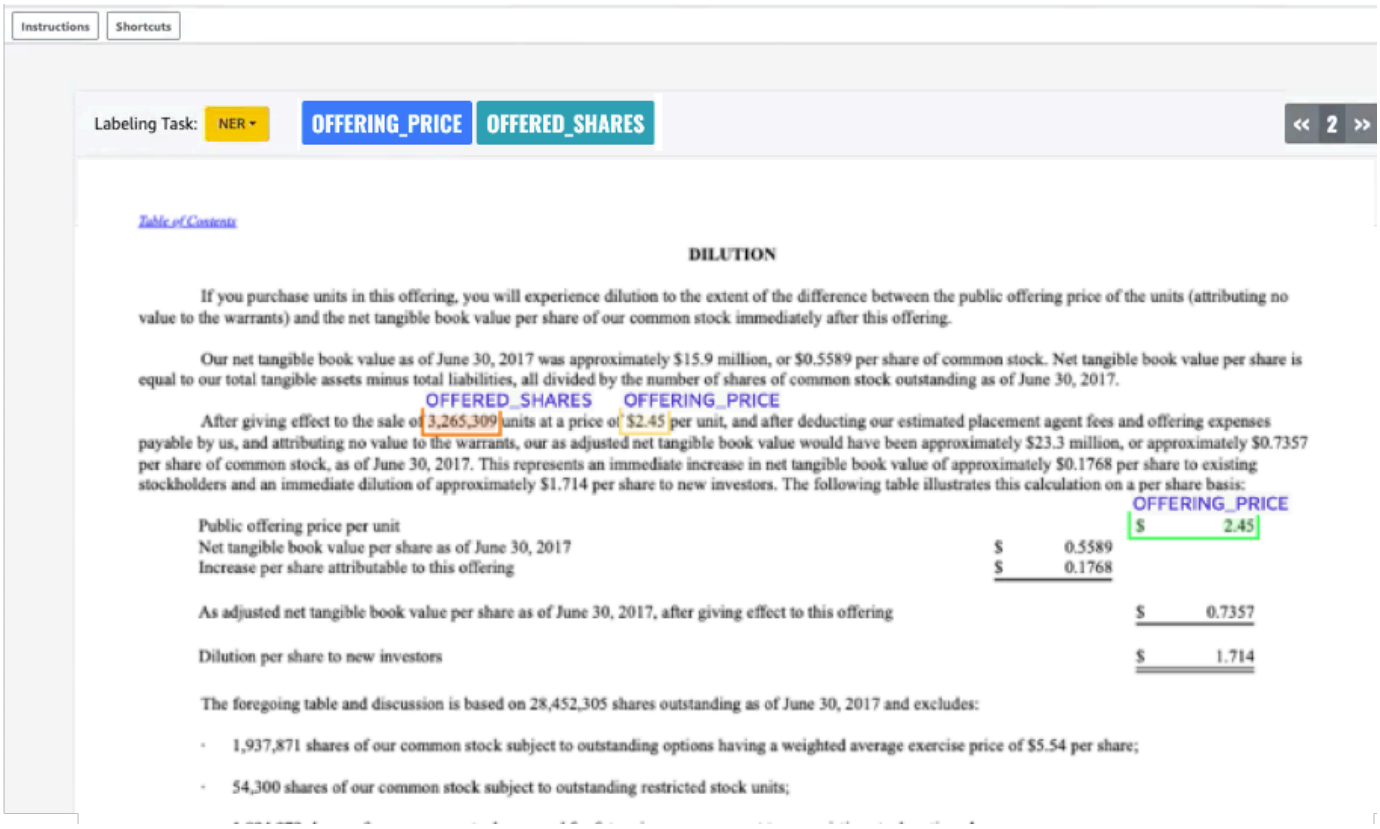
用 SageMaker Ground Truth 註釋

現在，您已經配置了所需的資源並創建了標籤工作，您可以登錄到標籤門戶並為 PDF 添加註釋。

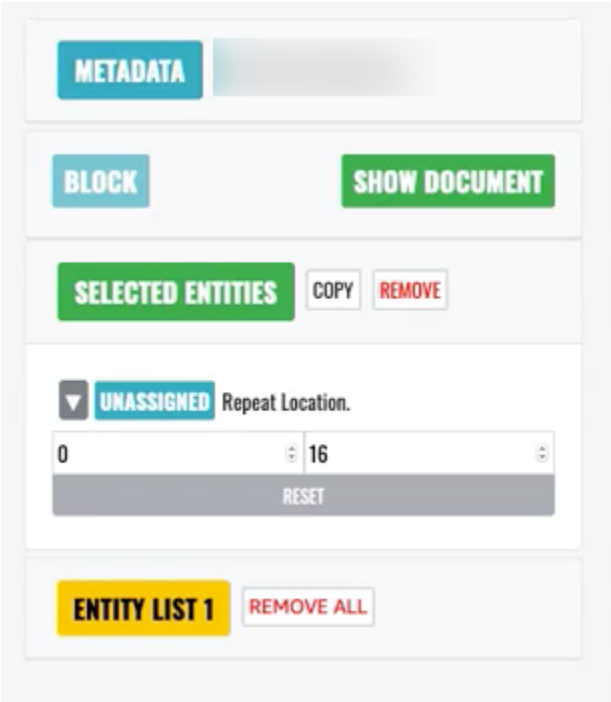
1. 使用 Chrome 瀏覽器或 Firefox 網頁瀏覽器登入SageMaker 主控台。
2. 選取標籤員工，然後選擇私人。
3. 在「私人員工摘要」下，選取您使用私人員工建立的標籤入口網站登入 URL。使用適當的認證登入。

   如果您沒有看到任何列出的工作，請不要擔心，這可能需要一段時間才能更新，具體取決於您上傳以進行註釋的檔案數量。
4. 選取您的工作，然後在右上角選擇 [開始工作] 以開啟註釋螢幕。

   您會在註釋螢幕中看到其中一個開啟的文件，以及您在設定期間提供的實體類型 (在其上方)。在實體類型的右側，有一個箭頭可用來瀏覽文件。



為開啟的文件加上註解。您也可以在每份文件上移除、還原或 auto 標記註解；這些選項可在註解工具的右側面板中找到。



若要使用 auto 動標籤，請為其中一個實體的例證加上註解；然後會自動以該圖元類型註解該特定單字的所有其他例證。

完成後，請選取右下角的「送出」，然後使用導覽箭頭移至下一個文件。重複此步驟，直到為所有 PDF 加上註解。

在註解所有訓練文件之後，您可以在 Amazon S3 儲存貯體的這個位置找到 JSON 格式的註釋：

```
/output/your labeling job name/annotations/
```

輸出資料夾也包含輸出資訊清單檔案，其中會列出訓練文件中的所有註釋。您可以在以下位置找到輸出清單文件。

```
/output/your labeling job name/manifests/
```

訓練自訂實體辨識器模型

自訂實體辨識器只會識別您訓練模型時包含的實體類型。它不會自動包括預設圖元類型。如果您也想要識別預設實體類型，例如「位置」、「日期」或「人員」，則需要為這些實體提供額外的訓練資料。

當您使用帶註解的 PDF 檔案建立自訂實體辨識器時，您可以將辨識器與多種輸入檔案格式搭配使用：純文字、影像檔案 (JPG、PNG、TIFF)、PDF 檔案和 Word 文件，無需預先處理或文件平面化。Amazon Comprehend 不支持圖像文件或 Word 文檔的註釋。

> **Note**
>
> 使用帶註釋 PDF 檔案的自訂實體辨識器僅支援英文文件。

建立自訂實體辨識器之後，您可以使用DescribeEntityRecognizer作業監視要求的進度。一旦該`Status`字段是`TRAINED`，識別器模型就可以用於自定義實體識別。

主題

- 訓練自訂辨識器 (主控台)
- 訓練自訂實體辨識器 (API)
- 自訂實體辨識器指標

訓練自訂辨識器 (主控台)

您可以使用 Amazon Comprehend 主控台建立自訂實體辨識器。本節說明如何建立和訓練自訂實體辨識器。

使用主控台建立自訂實體辨識器-CSV 格式

若要建立自訂實體辨識器，請先提供資料集來訓練模型。在此資料集中，包括下列其中一項：一組已註解的文件或實體清單及其類型標籤，以及包含這些實體的一組文件。如需更多資訊，請參閱 自訂實體辨識

使用 CSV 檔案訓練自訂實體辨識器

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [自訂]，然後選擇 [自訂實體辨識]。
3. 選擇「建立新模型」。
4. 給識別器一個名字。該名稱在「地區」和帳戶中必須是唯一的。
5. 選取語言。
6. 在 [自訂實體類型] 下，輸入您希望辨識器在資料集中尋找的自訂標籤。

   實體類型必須是大寫的，如果它由一個以上的單詞組成，則用底線分隔單詞。
7. 選擇 [新增類型]。
8. 如果您要新增其他實體類型，請輸入它，然後選擇 [新增類型]。如果您要移除其中一個已新增的實體類型，請選擇 [移除類型]，然後選擇要從清單中移除的實體類型。最多可列出 25 個圖元類型。
9. 若要加密訓練工作，請選擇辨識器加密，然後選擇要使用與目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。

   - 如果您使用與目前帳戶相關聯的金鑰，請針對 KMS 金鑰識別碼選擇金鑰識別碼。
   - 如果您使用與其他帳戶相關聯的金鑰，請針對 KMS 金鑰 ARN 輸入金鑰識別碼的 ARN。

> **Note**
>
> 如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱AWS Key Management Service。

10. 在「資料規格」下，選擇訓練文件的格式：

    - CSV 檔案 — 補充訓練文件的 CSV 檔案。CSV 檔案包含訓練過的模型將偵測到的自訂實體的相關資訊。所需的檔案格式取決於您是提供註釋還是實體清單。
    - 增強清單 — 由 Amazon SageMaker Ground Truth 生成的標籤數據集。這個文件是 JSON 行格式。每一行都是完整的 JSON 物件，其中包含訓練文件及其標籤。每個標籤都會在訓練文件中註解一個具名實體。您最多可以提供 5 個擴增資訊清單檔案。

    若要取得有關可用格式的更多資訊，以及範例，請參閱〈〉訓練自訂實體辨識器模型。

11. 在訓練類型下，選擇要使用的訓練類型：

    - 使用註釋和訓練文件
    - 使用實體清單和訓練文件

    如果選擇註釋，請在 Amazon S3 中輸入註釋檔案的網址。您也可以導覽至 Amazon S3 中註釋檔案所在的儲存貯體或資料夾，然後選擇瀏覽 S3。

    如果選擇實體清單，請在 Amazon S3 中輸入實體清單的網址。您也可以導覽至實體清單所在的 Amazon S3 儲存貯體或資料夾，然後選擇瀏覽 S3。

12. 輸入包含 Amazon S3 中訓練文件之輸入資料集的 URL。您也可以導覽至 Amazon S3 中訓練文件所在的儲存貯體或資料夾，然後選擇 [選取資料夾]。

13. 在「測試資料集」下，選取您要如何評估訓練模型的效能-您可以針對註釋和實體清單訓練類型執行此操作。

    - 自動拆分：自動拆分自動選擇您提供的培訓數據的 10% 用作測試數據
    - （可選）客戶提供：當您選擇提供的客戶時，您可以準確指定要使用的測試數據。

14. 如果您選取客戶提供的測試資料集，請在 Amazon S3 中輸入註釋檔案的 URL。您也可以導覽至 Amazon S3 中註釋檔案所在的儲存貯體或資料夾，然後選擇「選取資料夾」。

15. 在「選擇 IAM 角色」區段中，選取現有的 IAM 角色或建立新角色。

    - 選擇現有的 IAM 角色 — 如果您已經擁有可存取輸入和輸出 Amazon S3 儲存貯體的 IAM 角色，請選取此選項。
    - 建立新的 IAM 角色 — 當您想要建立具有適當許可的新 IAM 角色，讓 Amazon Comprehend 存取輸入和輸出值區時，請選取此選項。

> **Note**
>
> 如果輸入文件已加密，則使用的 IAM 角色必須具有`kms:Decrypt`權限。如需詳細資訊，請參閱 使用 KMS 加密所需的權限。

16. (選擇性) 若要從 VPC 將資源啟動至 Amazon Comprehend，請在 VPC 下輸入虛擬私人雲端識別碼，或從下拉式清單中選擇識別碼。

    1. 在 [子網路] 下選擇子網路。選取第一個子網路後，您可以選擇其他子網路。
    2. 在「安全性群組」下，選擇要使用的安全性群組 (如果您已指定安全性群組)。選取第一個安全性群組後，您可以選擇其他群組。

> **Note**
>
> 當您將 VPC 與自訂實體辨識工作搭配使用時，`DataAccessRole`用於「建立」和「開始」作業的 VPC 必須具有存取輸入文件和輸出值區的 VPC 權限。

17. (選擇性) 若要將標籤新增至自訂實體辨識器，請在「標籤」下輸入鍵值配對。選擇 Add tag (新增標籤)。若要在建立辨識器之前移除此配對，請選擇 [移除標籤]。
18. 選擇「火車」。

然後，新的辨識器會出現在清單中，並顯示其狀態。它將首先顯示為`Submitted`。然後，它將顯示`Training`正在處理訓練文檔的分類器，`Trained`準備使用的分類器以及`In error`發生錯誤的分類器。您可以按一下工作以取得有關辨識器的詳細資訊，包括任何錯誤訊息。

使用控制台創建自定義實體識別器-增強清單

若要使用純文字、PDF 或 Word 文件訓練自訂實體辨識器

1. 登入AWS Management Console並開啟 Amazon Comprehend 主控台。
2. 從左側功能表中選擇 [自訂]，然後選擇 [自訂實體辨識]。
3. 選擇火車辨識器。
4. 給識別器一個名字。該名稱在「地區」和帳戶中必須是唯一的。
5. 選取語言。注意：如果您正在訓練 PDF 或 Word 文件，則支援英文的語言為英文。
6. 在 [自訂實體類型] 下，輸入您希望辨識器在資料集中尋找的自訂標籤。

實體類型必須是大寫的，如果它由一個以上的單詞組成，則用底線分隔單詞。

7. 選擇 [新增類型]。
8. 如果您要新增其他實體類型，請輸入它，然後選擇 [新增類型]。如果您要移除其中一個已新增的實體類型，請選擇 [移除類型]，然後選擇要從清單中移除的實體類型。最多可列出 25 個圖元類型。
9. 若要加密訓練工作，請選擇辨識器加密，然後選擇要使用與目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。

   - 如果您使用與目前帳戶相關聯的金鑰，請針對 KMS 金鑰識別碼選擇金鑰識別碼。
   - 如果您使用與其他帳戶相關聯的金鑰，請針對 KMS 金鑰 ARN 輸入金鑰識別碼的 ARN。

   > **Note**
   >
   > 如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱AWS Key Management Service。

10. 在 [訓練資料] 下，選擇 [增強資訊清單] 做為資料格式

    - 增強清單-是由 Amazon SageMaker Ground Truth 生成的標籤數據集。這個文件是 JSON 行格式。檔案中的每一行都是完整的 JSON 物件，其中包含訓練文件及其標籤。每個標籤都會在訓練文件中註解一個具名實體。您最多可以提供 5 個擴增資訊清單檔案。如果您將 PDF 文件用於訓練資料，則必須選取「增強資訊清單」。您最多可以提供 5 個擴增資訊清單檔案。對於每個檔案，您最多可以命名 5 個屬性作為訓練資料。

    若要取得有關可用格式的更多資訊，以及範例，請參閱〈訓練自訂實體辨識器模型。

11. 選取訓練模型類型。

    如果您選取了純文字文件，請在「輸入位置」下輸入 Amazon SageMakerGround 真相增強資訊清單檔案的 Amazon S3URL。您也可以導覽至 Amazon S3 中增強資訊清單所在的儲存貯體或資料夾，然後選擇 [選取資料夾]。

12. 在「屬性名稱」下，輸入包含註釋的屬性名稱。如果檔案包含來自多個鏈結標籤工作的註釋，請為每個工作新增一個屬性。在這種情況下，每個屬性都包含來自標籤工作的一組註釋。附註：每個檔案最多可提供 5 個屬性名稱。
13. 選取新增。

14. 如果您在輸入位置下選擇 PDF，Word 文檔，請輸入 Amazon SageMaker Ground Truth 增強清單文件的 Amazon S3URL。您也可以導覽至 Amazon S3 中增強資訊清單所在的儲存貯體或資料夾，然後選擇 [選取資料夾]。
15. 輸入註釋資料檔案的 S3 前置詞。這些是您標記的 PDF 文件。
16. 輸入來源文件的 S3 前置詞。這些是您為標籤工作提供給 Ground Truth 的原始 PDF 文檔（數據對象）。
17. 輸入包含註釋的屬性名稱。附註：每個檔案最多可提供 5 個屬性名稱。系統會忽略檔案中未指定的任何屬性。
18. 在 IAM 角色區段中，選取現有的 IAM 角色或建立新角色。
    - 選擇現有的 IAM 角色 — 如果您已經擁有可存取輸入和輸出 Amazon S3 儲存貯體的 IAM 角色，請選取此選項。
    - 建立新的 IAM 角色 — 當您想要建立具有適當許可的新 IAM 角色，讓 Amazon Comprehend 存取輸入和輸出值區時，請選取此選項。

    > **Note**
    >
    > 如果輸入文件已加密，則使用的 IAM 角色必須具有`kms:Decrypt`權限。如需詳細資訊，請參閱 使用 KMS 加密所需的權限。

19. (選擇性) 若要從 VPC 將資源啟動至 Amazon Comprehend，請在 VPC 下輸入虛擬私人雲端識別碼，或從下拉式清單中選擇識別碼。
    1. 在 [子網路] 下選擇子網路。選取第一個子網路後，您可以選擇其他子網路。
    2. 在「安全性群組」下，選擇要使用的安全性群組 (如果您已指定安全性群組)。選取第一個安全性群組後，您可以選擇其他群組。

    > **Note**
    >
    > 當您將 VPC 與自訂實體辨識工作搭配使用時，`DataAccessRole`用於「建立」和「開始」作業的 VPC 必須具有存取輸入文件和輸出值區的 VPC 權限。

20. (選擇性) 若要將標籤新增至自訂實體辨識器，請在「標籤」下輸入鍵值配對。選擇 Add tag (新增標籤)。若要在建立辨識器之前移除此配對，請選擇 [移除標籤]。
21. 選擇「火車」。

然後，新的辨識器會出現在清單中，並顯示其狀態。它將首先顯示為Submitted。然後，它將顯示Training正在處理訓練文檔的分類器，Trained準備使用的分類器以及In error發生錯誤的分類器。您可以按一下工作以取得有關辨識器的詳細資訊，包括任何錯誤訊息。

訓練自訂實體辨識器 (API)

若要建立和訓練自訂實體辨識模型，請使 Amazon Comprehend CreateEntityRecognizerAPI 操作

主題

- 訓練自訂實體辨識器 AWS Command Line Interface
- 訓練自訂實體辨識器 AWS SDK for Java
- 使用 Python (肉毒桿菌 3) 訓練自訂實體辨識器

訓練自訂實體辨識器 AWS Command Line Interface

下列範例示範如何將CreateEntityRecognizer作業和其他相關聯的 API 與AWS CLI.

這些範例已針對 Unix、Linux 和 macOS 進行格式化。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

使用 create-entity-recognizer CLI 命令建立自訂實體辨識器。如需有關 input-data-config 參數的資訊，請參閱 Amazon Comprehend API 參考CreateEntityRecognizer中的。

```
aws comprehend create-entity-recognizer \
     --language-code en \
     --recognizer-name test-6 \
     --data-access-role-arn "arn:aws:iam::account number:role/service-role/
AmazonComprehendServiceRole-role" \
     --input-data-config "EntityTypes=[{Type=PERSON}],Documents={S3Uri=s3://Bucket
 Name/Bucket Path/documents},
                 Annotations={S3Uri=s3://Bucket Name/Bucket Path/annotations}" \
     --region region
```

使用 list-entity-recognizers CLI 命令列出區域中的所有實體識別器。

```
aws comprehend list-entity-recognizers \
     --region region
```

使用 describe-entity-recognizer CLI 命令檢查自訂實體辨識器的 Job 狀態。

```
aws comprehend describe-entity-recognizer \
     --entity-recognizer-arn arn:aws:comprehend:region:account number:entity-
recognizer/test-6 \
     --region region
```

訓練自訂實體辨識器 AWS SDK for Java

此範例會使用 Java 建立自訂實體辨識器並訓練模型

對於使用 Java Amazon Comprehend 的例子，請參閱 Amazon Comprehend Java 的例子。

使用 Python (肉毒桿菌 3) 訓練自訂實體辨識器

實例化博托 3 SDK：

```
import boto3
import uuid
comprehend = boto3.client("comprehend", region_name="region")
```

創建實體識別器：

```
response = comprehend.create_entity_recognizer(
    RecognizerName="Recognizer-Name-Goes-Here-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "EntityTypes": [
            {
                "Type": "ENTITY_TYPE"
            }
        ],
        "Documents": {
            "S3Uri": "s3://Bucket Name/Bucket Path/documents"
        },
        "Annotations": {
            "S3Uri": "s3://Bucket Name/Bucket Path/annotations"
        }
    }
)
recognizer_arn = response["EntityRecognizerArn"]
```

列出所有識別器：

```
response = comprehend.list_entity_recognizers()
```

等待辨識器達到訓練狀態：

```
while True:
    response = comprehend.describe_entity_recognizer(
        EntityRecognizerArn=recognizer_arn
    )

    status = response["EntityRecognizerProperties"]["Status"]
    if "IN_ERROR" == status:
        sys.exit(1)
    if "TRAINED" == status:
        break

    time.sleep(10)
```

自訂實體辨識器指標

Amazon Comprehend 為您提供指標，協助您估計實體辨識器在您的工作上應如何運作。它們是以訓練辨識器模型為基礎，因此雖然它們在訓練期間準確地表示模型的效能，但它們只是實體探索期間 API 效能的近似值。

只要傳回經過訓練的實體辨識器的中繼資料，就會傳回指標。

亞馬遜支援一次訓練多達 25 個實體的模型。從訓練過的實體辨識器傳回指標時，系統會針對辨識器整體 (全域量度) 和每個個別實體 (實體量度) 計算分數。

有三種量度可供使用，包括全域和實體量度：

- 精準

  這表示系統產生的圖元已正確識別並正確標示的部分。這顯示了模型的實體識別是真正良好的識別的次數。它是識別總數的百分比。

  換句話說，精度是基於真正陽性（TP）和誤報（fp），它被計算為精度 = TP/（TP + FP）。

  例如，如果一個模型預測實體的兩個例子存在於一個文件中，其中實際上只有一個，結果是一個真正的正值和一個假陽性。在這種情況下，精度 = 1/（1 + 1）。精確度為 50%，因為在模型所識別的兩個圖元中，有一個圖元是正確的。

- 召回

  這表示存在於由系統正確識別和標記的文件中實體的分數。在數學上，這是根據正確識別真正數（tp）和錯過標識虛假陰性（fn）的總數來定義。

  它被計算為召回 = TP/（TP + FN）。例如，如果模型正確地識別了一個實體，但遺漏了存在該實體的其他兩個實例，則結果為一個真正的正值和兩個假負數。在這種情況下，召回 = 1/（1 + 2）。召回的是 33.33%，因為一個實體在可能的三個例子中是正確的。

- F1 得分

  這是「精確度」和「召回」量度的組合，用於測量自訂實體辨識模型的整體精確度。F1 分數是「精確度」和「召回」量度的諧波平均值：F1 = 2 * 精確度 * 召回/(精確度 + 回收)。

  > **Note**
  >
  > 直覺上，諧波平均值比簡單的平均值或其他方式更加懲罰極端（例如：`precision`= 0，`recall`= 1 可以通過預測所有可能的跨度來輕鬆實現。 在這裡，簡單的平均值將是 0.5，但F1會將其懲罰為 0）。

  在上述範例中，`precision`= 50% 和`recall`等於 33.33%，因此 F1 = 2 * 0.5 * 0.3333/(0.5 + 0.3333)。一級方案的得分是 .3975 或 39.75%。

全域和個別實體量度

針對地點或個人的實體分析下列句子時，可以看到全域和個別實體量度之間的關係

```
John Washington and his friend Smith live in San Francisco, work in San Diego, and own
    a house in Seattle.
```

在我們的例子中，該模型進行了以下預測。

```
John Washington = Person
Smith = Place
San Francisco = Place
San Diego = Place
Seattle = Person
```

但是，預測應該是以下幾點。

```
John Washington = Person
Smith = Person
San Francisco = Place
San Diego = Place
Seattle = Place
```

此項目的個別實體指標為：

```
entity:  Person
  True positive (TP) = 1 (because John Washington is correctly predicted to be a
    Person).
  False positive (FP) = 1 (because Seattle is incorrectly predicted to be a Person,
    but is actually a Place).
  False negative (FN) = 1 (because Smith is incorrectly predicted to be a Place, but
    is actually a Person).
  Precision = 1 / (1 + 1) = 0.5 or 50%
  Recall = 1 / (1+1) = 0.5 or 50%
  F1 Score = 2 * 0.5 * 0.5 / (0.5 + 0.5) = 0.5 or 50%

entity:  Place
  TP = 2 (because San Francisco and San Diego are each correctly predicted to be a
    Place).
  FP = 1 (because Smith is incorrectly predicted to be a Place, but is actually a
    Person).
  FN = 1 (because Seattle is incorrectly predicted to be a Person, but is actually a
    Place).
  Precision = 2 / (2+1) = 0.6667 or 66.67%
  Recall = 2 / (2+1) = 0.6667 or 66.67%
  F1 Score = 2 * 0.6667 * 0.6667 / (0.6667 + 0.6667) = 0.6667 or  66.67%
```

這個全局指標將是：

全球：

```
Global:
  TP = 3 (because John Washington, San Francisco and San Diego are predicted
 correctly.
    This is also the sum of all individual entity TP).
  FP = 2 (because Seattle is predicted as Person and Smith is predicted as Place. This
    is the sum of all individual entity FP).
  FN = 2 (because Seattle is predicted as Person and Smith is predicted as Place. This
```

```
    is the sum of all individual FN).
  Global Precision = 3 / (3+2) = 0.6 or 60%
    (Global Precision = Global TP / (Global TP + Global FP))
  Global Recall = 3 / (3+2) = 0.6 or 60%
    (Global Recall = Global TP / (Global TP + Global FN))
  Global F1Score = 2 * 0.6 * 0.6 / (0.6 + 0.6) = 0.6 or 60%
    (Global F1Score = 2 * Global Precision *  Global Recall / (Global Precision +
    Global Recall))
```

改善自訂實體辨識器效能

這些指標可讓您深入瞭解訓練過的模型在您使用模型識別實體時執行的準確程度。如果指標低於預期，您可以使用以下幾個選項來改善指標：

1. 根據您是否使用註釋或實體清單 (僅限純文字)，請務必遵循相應文件中的準則，以改善資料品質。如果您在改善資料並重新訓練模型之後觀察到更好的指標，您可以持續反覆運算並改善資料品質，以達到更好的模型效能。
2. 如果您正在使用「實體清單」，請考慮改用註釋。手動註釋通常可以改善您的結果。
3. 如果您確定沒有數據質量問題，但指標仍然不合理地低，請提交支持請求。

執行即時自訂辨識器分析

即時分析對於在送達小型文件時處理小型文件的應用程式很有用。例如，您可以在社交媒體貼文、支援票證或客戶評論中偵測自訂實體。

> **開始之前**
>
> 您需要自訂實體辨識模型 (也稱為辨識器)，才能偵測自訂實體。如需這些模型的詳細資訊，請參閱the section called “訓練辨識器模型”。

使用純文字註解訓練的辨識器僅支援純文字文件的實體偵測。使用 PDF 文件註釋訓練的辨識器支援純文字文件、影像、PDF 檔案和 Word 文件的實體偵測。若要取得有關輸入檔案的資訊，請參閱即時自訂分析的輸入。

如果您打算分析影像檔案或掃描的 PDF 文件，您的 IAM 政策必須授予許可，才能使用兩種 Amazon Textract API 方法 (DetectDocumentText 和 AnalyzeDocument)。Amazon Comprehend 文本提取過程中調用這些方法。如需政策範例，請參閱 執行文件分析動作所需的權限。

主題

自訂實體辨識的即時分析 (主控台)

您可以使用 Amazon Comprehend 主控台，透過自訂模型執行即時分析。首先，您要建立端點來執行即時分析。建立端點之後，即可執行即時分析。

如需佈建端點輸送量及相關成本的相關資訊，請參閱使用 Amazon Comprehend 端點。

主題

建立自訂實體偵測的端點

建立端點（控制台）

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇「端點」，然後選擇「建立端點」按鈕。會開啟 「建立端點」畫面。
3. 為端點命名。名稱在目前的區域和帳戶中必須是唯一的。
4. 選擇您要附加新端點的自訂模型。從下拉菜單中，您可以按型號名稱進行搜索。

   > **Note**
   >
   > 您必須先建立模型，才能將端點貼附到模型上。如果您還沒有模型，請參閱訓練自訂實體辨識器模型。

5. (選擇性) 若要將標籤新增至端點，請在「標籤」下輸入金鑰值配對，然後選擇「新增標籤」。若要在建立端點之前移除此配對，請選擇「移除標籤」。
6. 輸入要指派給端點的推論單元 (IU) 數目。每個單位代表每秒 100 個字元的輸送量，每秒最多兩個文件。如需端點輸送量的詳細資訊，請參閱使用 Amazon Comprehend 端點。

7. (選擇性) 如果要建立新端點，您可以選擇使用 IU 估算器。估算器可以幫助您確定要請求的 IU 數量。推論單元的數目取決於輸送量或每秒要分析的字元數。
8. 在「購買」摘要中，檢閱您預估的每小時、每日和每月端點成本。
9. 如果您瞭解帳戶從端點啟動到您刪除為止，就會收取該端點的費用，請選取此核取方塊。
10. 選擇 建立端點。

執行即時自訂實體偵測

為自訂實體辨識器模型建立端點後，您可以執行即時分析來偵測個別文件中的實體。

完成下列步驟，以使用 Amazon Comprehend 主控台偵測文字中的自訂實體。

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [即時分析]。
3. 在「輸入文字」區段中，對於「分析類型」，選擇「自訂」。
4. 針對選取端點，選擇與您要使用的實體偵測模型相關聯的端點。
5. 要指定用於分析的輸入數據，您可以輸入文本或上傳文件。
   - 若要輸入文字：
     a. 選擇「輸入文字」。
     b. 輸入您要分析的文字。
   - 若要上傳檔案：
     a. 選擇上傳檔案，然後輸入要上傳的檔案名稱。
     b. (選擇性) 在「進階讀取動作」 下，您可以覆寫文字擷取的預設動作。如需詳細資訊，請參閱 設定文字擷取選項。
6. 選擇「分析」。控制台顯示分析的輸出以及可信度評估。

自訂實體辨識 (API) 的即時分析

您可以使用 Amazon Comprehend API 透過自訂模型執行即時分析。首先，您要建立端點來執行即時分析。建立端點之後，即可執行即時分析。

如需佈建端點輸送量及相關成本的相關資訊，請參閱使用 Amazon Comprehend 端點。

主題

- 建立用於自訂實體偵測的端點
- 執行即時自訂實體偵測

建立用於自訂實體偵測的端點

如需與端點相關聯之成本的資訊，請參閱使用 Amazon Comprehend 端點。

建立端點 AWS CLI

若要使用建立端點AWS CLI，請使用以下create-endpoint指令：

```
$ aws comprehend create-endpoint \
> --desired-inference-units number of inference units \
> --endpoint-name endpoint name \
> --model-arn arn:aws:comprehend:region:account-id:model/example \
> --tags Key=Key,Value=Value
```

如果您的命令成功，Amazon Comprehend 會使用端點 ARN 回應：

```
{
   "EndpointArn": "Arn"
}
```

若要取得有關此指令、其參數引數及其輸出的更多資訊，請參閱《AWS CLI指令參考》create-endpoint中的〈〉。

執行即時自訂實體偵測

為自訂實體辨識器模型建立端點後，您可以使用端點執行 DetectEntitiesAPI 作業。您可以使用text或bytes參數提供文字輸入。使用參數輸入其他輸入類bytes型。

對於影像檔案和 PDF 檔案，您可以使用DocumentReaderConfig參數來取代預設文字萃取動作。如需詳細資訊，請參閱 設定文字擷取選項。

使用偵測文字中的圖元 AWS CLI

若要偵測文字中的自訂實體，請使用text參數中的輸入文字執行detect-entities指令。

Example ：使用 CLI 偵測輸入文字中的實體

```
$ aws comprehend detect-entities \
> --endpoint-arn arn \
> --language-code en \
> --text  "Andy Jassy is the CEO of Amazon."
```

如果您的命令成功，Amazon Comprehend 會回應分析。對於 Amazon Comprehend 偵測到的每個實體，它都會提供實體類型、文字、位置和可信度分數。

偵測半結構化文件中的實體 AWS CLI

若要偵測 PDF、Word 或影像檔案中的自訂實體，請使用`bytes`參數中的輸入檔案執行`detect-entities`指令。

Example ：使用 CLI 偵測影像檔中的實體

這個例子演示了如何使用 base64 對圖像字節進行編碼的`fileb`選項傳遞圖像文件。若要取得更多資訊，請參閱《AWS Command Line Interface使用指南》中的二進位大型物件

此範例也會傳入名為的 JSON 檔案，`config.json`以設定文字擷取選項。

```
$ aws comprehend detect-entities \
> --endpoint-arn arn \
> --language-code en \
> --bytes fileb://image1.jpg   \
> --document-reader-config file://config.json
```

config.json 檔案包含下列內容。

```
{
   "DocumentReadMode": "FORCE_DOCUMENT_READ_ACTION",
   "DocumentReadAction": "TEXTRACT_DETECT_DOCUMENT_TEXT"
}
```

如需有關命令語法的詳細資訊，請參閱 Amazon Comprehend API 參考DetectEntities中的。

即時分析的輸出

文字輸入的輸出

如果您使用Text參數輸入文字，則輸出會由分析偵測到的圖元陣列組成。下列範例顯示偵測到兩個判斷實體的分析。

```
{
        "Entities":
        [
            {
                "BeginOffset": 0,
                "EndOffset": 22,
                "Score": 0.9763959646224976,
                "Text": "John Johnson",
                "Type": "JUDGE"
            },
            {
                "BeginOffset": 11,
                "EndOffset": 15,
                "Score": 0.9615424871444702,
                "Text": "Thomas Kincaid",
                "Type": "JUDGE"
            }
        ]
    }
```

半結構化輸入的輸出

對於半結構化輸入文件或文字檔案，輸出可以包含下列其他欄位：

- DocumentMetadata — 擷取有關文件的資訊。元數據包括文檔中的頁面列表，其中包含從每個頁面中提取的字符數。如果請求包含Byte參數，則此字段存在於響應中。
- DocumentType — 輸入文件中每頁的文件類型。此欄位會出現在包含Byte參數之要求的回應中。
- 區塊 — 輸入文件中每個文字區塊的相關資訊。圖塊是巢狀的。頁面區塊會針對每一行文字包含一個區塊，其中包含每個字詞的區塊。此欄位會出現在包含Byte參數之要求的回應中。
- BlockReferences — 此圖元每個圖塊的參考。此欄位會出現在包含Byte參數之要求的回應中。文字檔案不存在此欄位。
- Error — 系統在處理輸入文件時偵測到的頁面層級錯誤。如果系統沒有發生任何錯誤，則此欄位為空白。

如需這些輸出欄位的說明，請參閱 Amazon Comprehend API 參考DetectEntities中的。如需有關版面配置元素的詳細資訊，請參閱 Amazon Textract 開發人員指南中的 Amazon Textract 分析物件。

下列範例顯示單頁掃描 PDF 輸入文件的輸出。


```
{
    "Entities": [{
        "Score": 0.9984670877456665,
        "Type": "DATE-TIME",
        "Text": "September 4,",
        "BlockReferences": [{
            "BlockId": "42dcaaee-c484-4b5d-9e3f-ae0be928b3e1",
            "BeginOffset": 0,
            "EndOffset": 12,
            "ChildBlocks": [{
                    "ChildBlockId": "6e9cbb43-f8be-4da0-9a4b-ff9a6c350a14",
                    "BeginOffset": 0,
                    "EndOffset": 9
                },
                {
                    "ChildBlockId": "599e0d53-ae9f-491b-a762-459b22c79ff5",
                    "BeginOffset": 0,
                    "EndOffset": 2
                },
                {
                    "ChildBlockId": "599e0d53-ae9f-491b-a762-459b22c79ff5",
                    "BeginOffset": 0,
                    "EndOffset": 2
                }
            ]
        }]
    }],
    "DocumentMetadata": {
        "Pages": 1,
        "ExtractedCharacters": [{
            "Page": 1,
            "Count": 609
        }]
    },
    "DocumentType": [{
        "Page": 1,
        "Type": "SCANNED_PDF"
    }],
    "Blocks": [{
```

```
        "Id": "ee82edf3-28de-4d63-8883-40e2e4938ccb",
        "BlockType": "LINE",
        "Text": "Your Band",
        "Page": 1,
        "Geometry": {
            "BoundingBox": {
                "Height": 0.024125460535287857,
                "Left": 0.11745482683181763,
                "Top": 0.06821706146001816,
                "Width": 0.12074867635965347
            },
            "Polygon": [{
                    "X": 0.11745482683181763,
                    "Y": 0.06821706146001816
                },
                {
                    "X": 0.2382034957408905,
                    "Y": 0.06821706146001816
                },
                {
                    "X": 0.2382034957408905,
                    "Y": 0.09234252572059631
                },
                {
                    "X": 0.11745482683181763,
                    "Y": 0.09234252572059631
                }
            ]
        },
        "Relationships": [{
            "Ids": [
                "b105c561-c8d9-485a-a728-7a5b1a308935",
                "60ecb119-3173-4de2-8c5d-de182a5f86a5"
            ],
            "Type": "CHILD"
        }]
    }]
}
```

下列範例顯示原生 PDF 文件分析的輸出。

Example PDF 文件的自訂實體辨識分析輸出範例

```
{
```

```
        "Blocks":
        [
            {
                "BlockType": "LINE",
                "Geometry":
                {
                    "BoundingBox":
                    {
                        "Height": 0.012575757575757575,
                        "Left": 0.0,
                        "Top": 0.0015063131313131314,
                        "Width": 0.02262091503267974
                    },
                    "Polygon":
                    [
                        {
                            "X": 0.0,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.014082070707070706
                        },
                        {
                            "X": 0.0,
                            "Y": 0.014082070707070706
                        }
                    ]
                },
                "Id": "4330efed-6334-4fc4-ba48-e050afa95c8d",
                "Page": 1,
                "Relationships":
                [
                    {
                        "ids":
                        [
                            "f343ce48-583d-4abe-b84b-a232e266450f"
                        ],
                        "type": "CHILD"
                    }
```

```
                ],
                "Text": "S-3"
            },
            {
                "BlockType": "WORD",
                "Geometry":
                {
                    "BoundingBox":
                    {
                        "Height": 0.012575757575757575,
                        "Left": 0.0,
                        "Top": 0.0015063131313131314,
                        "Width": 0.02262091503267974
                    },
                    "Polygon":
                    [
                        {
                            "X": 0.0,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.014082070707070706
                        },
                        {
                            "X": 0.0,
                            "Y": 0.014082070707070706
                        }
                    ]
                },
                "Id": "f343ce48-583d-4abe-b84b-a232e266450f",
                "Page": 1,
                "Relationships":
                [],
                "Text": "S-3"
            }
        ],
        "DocumentMetadata":
        {
            "PageNumber": 1,
```

```
            "Pages": 1
        },
        "DocumentType": "NativePDF",
        "Entities":
        [
            {
                "BlockReferences":
                [
                    {
                        "BeginOffset": 25,
                        "BlockId": "4330efed-6334-4fc4-ba48-e050afa95c8d",
                        "ChildBlocks":
                        [
                            {
                                "BeginOffset": 1,
                                "ChildBlockId": "cbba5534-ac69-4bc4-beef-306c659f70a6",
                                "EndOffset": 6
                            }
                        ],
                        "EndOffset": 30
                    }
                ],
                "Score": 0.9998825926329088,
                "Text": "0.001",
                "Type": "OFFERING_PRICE"
            },
            {
                "BlockReferences":
                [
                    {
                        "BeginOffset": 41,
                        "BlockId": "f343ce48-583d-4abe-b84b-a232e266450f",
                        "ChildBlocks":
                        [
                            {
                                "BeginOffset": 0,
                                "ChildBlockId": "292a2e26-21f0-401b-a2bf-03aa4c47f787",
                                "EndOffset": 9
                            }
                        ],
                        "EndOffset": 50
                    }
                ],
                "Score": 0.9809727537330395,
```

```
                "Text": "6,097,560",
                "Type": "OFFERED_SHARES"
            }
        ],
        "File": "example.pdf",
        "Version": "2021-04-30"
    }
```

執行自訂實體辨識的分析工作

您可以執行非同步分析工作，以偵測一或多個文件集中的自訂實體。

> **開始之前**
>
> 您需要自訂實體辨識模型 (也稱為辨識器)，才能偵測自訂實體。如需這些模型的詳細資訊，請參閱the section called “訓練辨識器模型”。

使用純文字註解訓練的辨識器僅支援純文字文件的實體偵測。使用 PDF 文件註釋訓練的辨識器支援純文字文件、影像、PDF 檔案和 Word 文件的實體偵測。對於文字檔案以外的檔案，Amazon Comprehend 會在執行分析之前執行文字擷取。若要取得有關輸入檔案的資訊，請參閱非同步自訂分析的輸入。

如果您打算分析影像檔案或掃描的 PDF 文件，您的 IAM 政策必須授予許可，才能使用兩種 Amazon Textract API 方法 (DetectDocumentText 和 AnalyzeDocument)。Amazon Comprehend 文本提取過程中調用這些方法。如需政策範例，請參閱 執行文件分析動作所需的權限。

若要執行非同步分析工作，請執行下列整體步驟：

1. 將文件存放在 Amazon S3 儲存貯體中。
2. 使用 API 或主控台開始分析工作。
3. 監視分析工作的進度。
4. 任務執行完成後，從您開始工作時指定的 S3 儲存貯體擷取分析結果。

主題

啟動自訂實體偵測工作 (主控台)

您可以使用主控台啟動和監視非同步分析工作，以進行自訂實體辨識。

若要啟動非同步分析工作

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [分析工作]，然後選擇 [建立工作]。
3. 為分類工作命名。該名稱必須是您的帳戶和當前區域的唯一名稱。
4. 在分析類型之下，選擇自訂實體辨識。
5. 從辨識器模型中，選擇要使用的自訂實體辨識器。
6. 在版本中，選擇要使用的辨識器版本。
7. (選擇性) 如果您選擇加密 Amazon Comprehend 在處理任務時使用的資料，請選擇任務加密。然後選擇要使用與目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。
   - 如果您使用與目前帳戶相關聯的金鑰，請選擇 KMS 金鑰識別碼的金鑰識別碼。
   - 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ARN 下輸入金鑰識別碼的 AR N。

   > **Note**
   >
   > 如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱金鑰管理服務 (KMS)。

8. 在「輸入資料」下，輸入包含輸入文件的 Amazon S3 儲存貯體的位置，或選擇瀏覽 S3 瀏覽至該儲存貯體。此值區必須與您呼叫的 API 位於相同的區域。您用於分析任務存取權限的 IAM 角色必須具有 S3 儲存貯體的讀取權限。
9. (選擇性) 對於「輸入格式」，您可以選擇輸入文件的格式。格式可以是每個檔案一個文件，也可以是單一檔案中每行一個文件。每行一個文件僅適用於文字文件。
10. (選擇性) 對於「文件」讀取模式，您可以覆寫預設的文字擷取動作。如需詳細資訊，請參閱 設定文字擷取選項。
11. 在「輸出資料」下，輸入 Amazon S3 儲存貯體的位置，Amazon Comprehend 應在其中寫入任務的輸出資料，或選擇瀏覽 S3 瀏覽至該儲存貯體。此值區必須與您呼叫的 API 位於相同的區域。您用於分類任務存取權限的 IAM 角色必須具有 S3 儲存貯體的寫入許可。
12. (選擇性) 如果您選擇加密工作的輸出結果，請選擇加密。然後選擇要使用與目前帳戶相關聯的 KMS 金鑰，還是使用另一個帳戶的 KMS 金鑰。

- 如果您使用與目前帳戶相關聯的金鑰，請為 KMS 金鑰 ID 選擇金鑰別名或 ID。
- 如果您使用與其他帳戶相關聯的金鑰，請在 KMS 金鑰 ID 下輸入金鑰別名或 ID 的 ARN。

13. (選擇性) 若要從 VPC 將資源啟動至 Amazon Comprehend，請在 VPC 下輸入虛擬私人雲端識別碼，或從下拉式清單中選擇識別碼。

    1. 在 [子網路] 下選擇子網路。選取第一個子網路後，您可以選擇其他子網路。
    2. 在「安全性群組」下，選擇要使用的安全性群組 (如果您已指定安全性群組)。選取第一個安全性群組後，您可以選擇其他群組。

    > **Note**
    >
    > 當您將 VPC 與分析工作搭配使用時，`DataAccessRole`用於「建立」和「啟動」作業的使用者必須具有存取輸出值區之 VPC 的權限。

14. 選擇建立工作以建立實體辨識工作。

啟動自訂實體偵測工作 (API)

您可以使用 API 啟動和監視非同步分析工作，以進行自訂實體辨識。

若要透過StartEntitiesDetectionJob作業啟動自訂實體偵測任務 EntityRecognizerArn，請提供訓練模型的 Amazon 資源名稱 (ARN)。您可以在對CreateEntityRecognizer操作的響應中找到此 ARN。

主題

- 使用偵測自訂實體 AWS Command Line Interface
- 使用偵測自訂實體 AWS SDK for Java
- 使用偵測自訂實體 AWS SDK for Python (Boto3)
- 覆寫 PDF 檔案的 API 動作

使用偵測自訂實體 AWS Command Line Interface

以下範例適用於 Unix、Linux 和 macOS 環境。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。若要偵測文件集中的自訂實體，請使用下列要求語法：

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

```
     --entity-recognizer-arn "arn:aws:comprehend:region:account number:entity-
recognizer/test-6" \
     --job-name infer-1 \
     --data-access-role-arn "arn:aws:iam::account number:role/service-role/
AmazonComprehendServiceRole-role" \
     --language-code en \
     --input-data-config "S3Uri=s3://Bucket Name/Bucket Path" \
     --output-data-config "S3Uri=s3://Bucket Name/Bucket Path/" \
     --region region
```

Amazon Comprehend 會回應和，JobIDJobStatus並會傳回您在請求中指定之 S3 儲存貯體中任務的輸出。

使用偵測自訂實體 AWS SDK for Java

對於使用 Java Amazon Comprehend 的例子，請參閱 Amazon Comprehend Java 的例子。

使用偵測自訂實體 AWS SDK for Python (Boto3)

此範例會建立自訂實體辨識器、訓練模型，然後在實體辨識器工作中使用. AWS SDK for Python (Boto3)

為 Python 實例化開發套件。

```
import boto3
import uuid
comprehend = boto3.client("comprehend", region_name="region")
```

創建一個實體識別器：

```
response = comprehend.create_entity_recognizer(
    RecognizerName="Recognizer-Name-Goes-Here-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "EntityTypes": [
            {
                "Type": "ENTITY_TYPE"
            }
        ],
        "Documents": {
            "S3Uri": "s3://Bucket Name/Bucket Path/documents"
```

```
        },
        "Annotations": {
            "S3Uri": "s3://Bucket Name/Bucket Path/annotations"
        }
    }
)
recognizer_arn = response["EntityRecognizerArn"]
```

列出所有識別器：

```
response = comprehend.list_entity_recognizers()
```

等待實體辨識器達到訓練狀態：

```
while True:
    response = comprehend.describe_entity_recognizer(
        EntityRecognizerArn=recognizer_arn
    )

    status = response["EntityRecognizerProperties"]["Status"]
    if "IN_ERROR" == status:
        sys.exit(1)
    if "TRAINED" == status:
        break

    time.sleep(10)
```

啟動自訂實體偵測工作：

```
response = comprehend.start_entities_detection_job(
    EntityRecognizerArn=recognizer_arn,
    JobName="Detection-Job-Name-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "InputFormat": "ONE_DOC_PER_LINE",
        "S3Uri": "s3://Bucket Name/Bucket Path/documents"
    },
    OutputDataConfig={
        "S3Uri": "s3://Bucket Name/Bucket Path/output"
    }
)
```

覆寫 PDF 檔案的 API 動作

對於影像檔和 PDF 檔案，您可以使用中的DocumentReaderConfig參數來取代預設萃取動作InputDataConfig。

下列範例會定義名為 myInputData Config.json 的 JSON 檔案來設定值InputDataConfig。它設置DocumentReadConfig為使用 Amazon Textract 取 DetectDocumentText API 的所有 PDF 文件。

Example

```
"InputDataConfig": {
  "S3Uri": s3://Bucket Name/Bucket Path",
  "InputFormat": "ONE_DOC_PER_FILE",
  "DocumentReaderConfig": {
      "DocumentReadAction": "TEXTRACT_DETECT_DOCUMENT_TEXT",
      "DocumentReadMode": "FORCE_DOCUMENT_READ_ACTION"
  }
}
```

在StartEntitiesDetectionJob作業中，指定 myInputData組態 .json 檔案做為參數InputDataConfig：

```
  --input-data-config file://myInputDataConfig.json
```

如需DocumentReaderConfig參數的詳細資訊，請參閱設定文字擷取選項。

非同步分析工作的輸出

分析任務完成後，會將結果儲存在您在請求中指定的 S3 儲存貯體中。

文字輸入的輸出

對於文字輸入檔案，輸出由每個輸入文件的實體清單組成。

下列範例會使用每行格式一個文件50_docs，顯示名為的輸入檔案中兩個文件的輸出。

```
{
        "File": "50_docs",
        "Line": 0,
        "Entities":
```

```
        [
            {
                "BeginOffset": 0,
                "EndOffset": 22,
                "Score": 0.9763959646224976,
                "Text": "John Johnson",
                "Type": "JUDGE"
            }
        ]
    }
    {
        "File": "50_docs",
        "Line": 1,
        "Entities":
        [
            {
                "BeginOffset": 11,
                "EndOffset": 15,
                "Score": 0.9615424871444702,
                "Text": "Thomas Kincaid",
                "Type": "JUDGE"
            }
        ]
    }
```

半結構化輸入的輸出

對於半結構化輸入文件，輸出可以包含下列其他欄位：

- DocumentMetadata — 擷取有關文件的資訊。元數據包括文檔中的頁面列表，其中包含從每個頁面中提取的字符數。如果請求包含Byte參數，則此字段存在於響應中。
- DocumentType — 輸入文件中每頁的文件類型。此欄位會出現在包含Byte參數之要求的回應中。
- 區塊 — 輸入文件中每個文字區塊的相關資訊。圖塊可以在圖塊內巢狀。頁面區塊會針對每一行文字包含一個區塊，其中包含每個字詞的區塊。此欄位會出現在包含Byte參數之要求的回應中。
- BlockReferences — 此圖元每個圖塊的參考。此欄位會出現在包含Byte參數之要求的回應中。該字段不存在於文本文件。
- Error — 系統在處理輸入文件時偵測到的頁面層級錯誤。如果系統沒有發生任何錯誤，則此欄位為空白。

如需有關這些輸出欄位的詳細資訊，請參閱 Amazon Comprehend API 參考DetectEntities中的

下列範例顯示單頁原生 PDF 輸入文件的輸出。

Example PDF 文件的自訂實體辨識分析輸出範例


```
{
        "Blocks":
        [
            {
                "BlockType": "LINE",
                "Geometry":
                {
                    "BoundingBox":
                    {
                        "Height": 0.012575757575757575,
                        "Left": 0.0,
                        "Top": 0.0015063131313131314,
                        "Width": 0.02262091503267974
                    },
                    "Polygon":
                    [
                        {
                            "X": 0.0,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.0015063131313131314
                        },
                        {
                            "X": 0.02262091503267974,
                            "Y": 0.014082070707070706
                        },
                        {
                            "X": 0.0,
                            "Y": 0.014082070707070706
                        }
                    ]
                },
                "Id": "4330efed-6334-4fc4-ba48-e050afa95c8d",
                "Page": 1,
                "Relationships":
                [
                    {
                        "ids":
```

```
                    [
                        "f343ce48-583d-4abe-b84b-a232e266450f"
                    ],
                    "type": "CHILD"
                }
            ],
            "Text": "S-3"
        },
        {
            "BlockType": "WORD",
            "Geometry":
            {
                "BoundingBox":
                {
                    "Height": 0.012575757575757575,
                    "Left": 0.0,
                    "Top": 0.0015063131313131314,
                    "Width": 0.02262091503267974
                },
                "Polygon":
                [
                    {
                        "X": 0.0,
                        "Y": 0.0015063131313131314
                    },
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.0015063131313131314
                    },
                    {
                        "X": 0.02262091503267974,
                        "Y": 0.014082070707070706
                    },
                    {
                        "X": 0.0,
                        "Y": 0.014082070707070706
                    }
                ]
            },
            "Id": "f343ce48-583d-4abe-b84b-a232e266450f",
            "Page": 1,
            "Relationships":
            [],
            "Text": "S-3"
```

```
            }
        ],
        "DocumentMetadata":
        {
            "PageNumber": 1,
            "Pages": 1
        },
        "DocumentType": "NativePDF",
        "Entities":
        [
            {
                "BlockReferences":
                [
                    {
                        "BeginOffset": 25,
                        "BlockId": "4330efed-6334-4fc4-ba48-e050afa95c8d",
                        "ChildBlocks":
                        [
                            {
                                "BeginOffset": 1,
                                "ChildBlockId": "cbba5534-ac69-4bc4-beef-306c659f70a6",
                                "EndOffset": 6
                            }
                        ],
                        "EndOffset": 30
                    }
                ],
                "Score": 0.9998825926329088,
                "Text": "0.001",
                "Type": "OFFERING_PRICE"
            },
            {
                "BlockReferences":
                [
                    {
                        "BeginOffset": 41,
                        "BlockId": "f343ce48-583d-4abe-b84b-a232e266450f",
                        "ChildBlocks":
                        [
                            {
                                "BeginOffset": 0,
                                "ChildBlockId": "292a2e26-21f0-401b-a2bf-03aa4c47f787",
                                "EndOffset": 9
                            }
```

```
                    ],
                    "EndOffset": 50
                }
            ],
            "Score": 0.9809727537330395,
            "Text": "6,097,560",
            "Type": "OFFERED_SHARES"
        }
    ],
    "File": "example.pdf",
    "Version": "2021-04-30"
}
```

建立和管理自訂模型

Amazon Comprehend 包含內建的 NLP (自然語言處理) 模型，可用於分析見解或主題建模。您也可以使用 Amazon Comprehend 建立用於實體辨識和文件分類的自訂模型。

您可以使用模型版本控制來追蹤模型的歷史記錄。當您建立和訓練新的模型版本時，您可以對訓練資料集進行變更。Amazon Comprehend 會在模型詳細資料頁面上顯示每個模型版本的詳細資料 (包括模型效能)。隨著時間的推移，您可以看到模型效能如何隨著您對訓練資料集進行變更。

您可以使用 Amazon Comprehend 主控台或 API 建立模型版本。作為替代方案，Amazon Comprehend 提供簡化與訓練相關的任務，飞轮以及評估新的自訂模型版本。

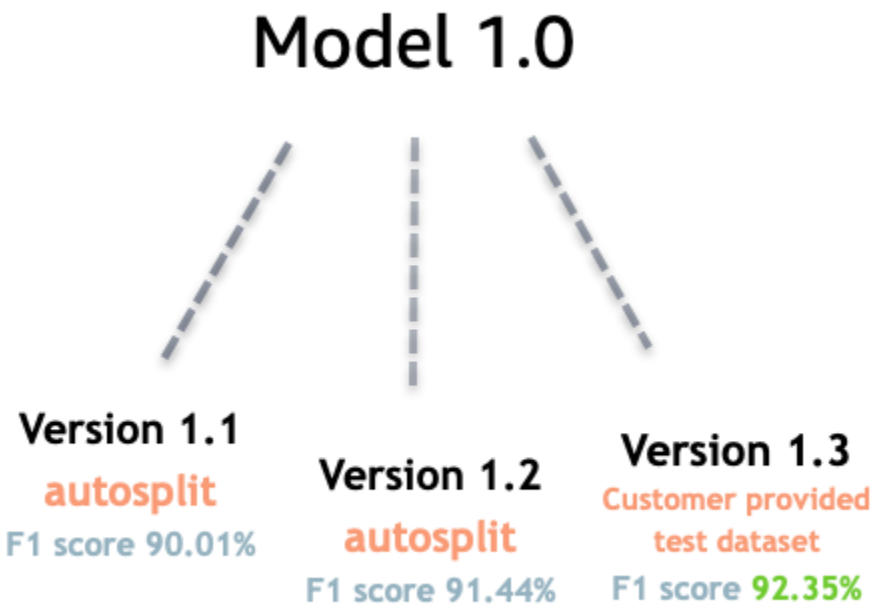
建立自訂模型後，您可以透過允許其他使用者匯入模型的副本，與其他AWS 帳戶使用者共用模型。

主題

使用 Amazon Comprehend 模型版本控制

人工智慧和機器學習 (AI/ML) 都是關於快速實驗。使用 Amazon Comprehend，您可以訓練和建置模型，以取得對資料的洞察力。使用模型版本控制，您可以在提供更多或不同的資料集時，追蹤模型歷史記錄和與模型執行結果相關聯的分數。您可以將版本控制與自訂分類模型或自訂實體辨識模型搭配使用。隨著時間的推移查看您的不同版本，您可以深入了解它們的成功程度，並深入了解您用來獲得成功狀態的參數。

當您訓練現有自訂分類器模型或實體辨識模型的新版本時，您需要做的就是從模型詳細資訊頁面建立新版本，並為您填入所有詳細資訊。新版本將與您之前的模型（我們稱之為 VersionID）具有相同的名稱，儘管您在創建過程中會為其提供唯一的版本名稱。當您將新版本新增至模型時，您可以從模型詳細資訊頁面在一個檢視中查看所有先前版本及其詳細資訊。使用版本控制，您可以在對訓練資料集進行變更時，查看模型效能如何變更。



建立新的自訂分類器版本 (主控台)

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [自訂]，然後選擇 [自訂分類]。
3. 從「分類器」清單中，選擇您要從中建立新版本的自訂模型名稱。此時會顯示自訂模型詳細資訊頁面。
4. 選取右上角的「建立新模型」。會開啟一個畫面，其中包含父自訂分類模型中預先填入的詳細資訊
5. 在「版本名稱」下，將唯一名稱添加到新版本中。
6. 在版本詳細資料下，您可以變更與新模型相關聯的語言和標籤數量。
7. 在「數據規格」部分下，配置如何將數據提供給新版本-確保提供完整數據，其中包括以前模型的文檔和新文檔。您可以變更分類器模式 (單一標籤或多標籤)、資料格式 (CSV 檔案、增強資訊清單)、訓練資料集和測試資料集 (自動分割或您的自訂測試資料組態)。
8. (選擇性) 更新輸出資料的 S3 位置
9. 在「存取權限」下，建立或使用現有的 IAM 角色。
10. (選擇性) 更新您的 VPC 設定

11. （可選）將標籤添加到新版本以幫助跟踪詳細信息。

    如需有關建立自訂分類器的詳細資訊，請參閱建立自訂分類器

建立新的自訂實體辨識器版本 (主控台)

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇 [自訂]，然後選擇 [自訂實體辨識]。
3. 從辨識器型號清單中，選擇您要從中建立新版本的辨識器名稱。此時會顯示詳細資訊頁面。
4. 選取右上角的 [訓練新版本]。螢幕會開啟，其中包含來自父實體辨識器的預先填入詳細資訊。
5. 在「版本名稱」下，將唯一名稱添加到新版本中。
6. 在 [自訂實體類型] 底下，新增您希望辨識器在資料集中識別的自訂標籤或標籤，然後選取 [新增類型]。從您提供的註釋或實體清單中選擇自訂實體類型。然後，辨識器將使用所有包含的實體類型，在執行工作時識別資料集中的實體。如果每個實體類型使用多個單字，每個實體類型都必須是大寫，並以及下劃線分隔。最多允許 25 種類型。
7. (選擇性) 選取辨識器加密，即可在處理工作時加密儲存磁碟區中的資料。
8. 在「訓練資料」區段下，指定「註釋」和「資料格式」詳細資訊 (CSV 檔案、增強資訊清單) 單一標籤或多標籤)、資料格式 (CSV、增強資訊清單)、訓練資料集和測試資料集 (自動分割或自訂測試資料組態)。
9. (選擇性) 更新輸出資料的 S3 位置
10. 在「存取權限」下，建立或使用現有的 IAM 角色。
11. (選擇性) 更新您的 VPC 設定
12. （可選）將標籤添加到新版本以幫助跟踪詳細信息。

要了解有關自定義實體識別器的更多信息，請參閱自定義實體識別和使用控制台創建自定義實體識別器。

在之間複製自訂模型 AWS 帳戶

Amazon Comprehend 使用者可以AWS 帳戶在兩個步驟中複製訓練過的自訂模型。首先，一個AWS 帳戶（帳戶 A）中的用戶共享其帳戶中的自定義模型。然後，另一個 AWS 帳戶 (帳戶 B) 中的使用者將模型匯入他們的帳戶。帳戶 B 使用者不需要訓練模型，也不需要複製 (或存取) 原始訓練資料或測試資料。

若要在帳戶 A 中共用自訂模型，使用者會將 AWS Identity and Access Management (IAM) 政策附加至模型版本。此政策授權帳戶 B 中的實體 (例如使用者或角色) 將模型版本匯入其中的 Amazon Comprehend。AWS 帳戶帳戶 B 使用者必須將模型匯入到與原始模型AWS 區域相同的模型中。

若要在帳戶 B 中匯入模型，此帳戶的使用者會向 Amazon Comprehend 提供必要的詳細資訊，例如模型的 Amazon 資源名稱 (ARN)。透過匯入模型，此使用者會在模型中建立新的自訂模型AWS 帳戶，以複製他們匯入的模型。此模型經過完整訓練，可供推論工作使用，例如文件分類或具名實體辨識。

在下列情況下，複製自訂模型很有用：

- 您屬於使用多個組織的組織AWS 帳戶。例如，您的組織可能會AWS 帳戶針對每個開發階段，例如建置、階段、測試和部署。或者，它AWS 帳戶對於商業功能（例如數據科學和工程）可能有所不同。
- 您的組織與另一個合作夥伴 (例如合作AWS夥伴) 合作，該合作夥伴會在 Amazon Comprehend 中訓練自訂模型，並將它們作為客戶提供給您。

在這種情況下，您可以快速將訓練有素的自訂實體辨識器或文件分類器從一個複製AWS 帳戶到另一個。以這種方式複製模型比替代方法更容易，您可以在其間複製訓練資料AWS 帳戶以訓練重複的模型。

主題

與另一個模型共用自訂模型 AWS 帳戶

使用 Amazon Comprehend，您可以與其他人共用您的自訂模型，以便他們可以將您的模型匯入他AWS們的帳戶。當使用者匯入您的其中一個自訂模型時，他們會在其帳戶中建立新的自訂模型。他們的新模型會複製您共用的模型。

若要共用自訂模型，請將原則附加至該模型，以授權其他人匯入該模型。然後，您可以為這些用戶提供他們所需的詳細信息。

> **Note**
>
> 當其他使用者匯入您已共用的自訂模型時，他們必須使用包含您模型AWS 區域的相同模型，例如美國東部 (維吉尼亞北部)。

主題

開始之前

在您可以共用模型之前，您必須在您的 Amazon Comprehend 中擁有訓練有素的自訂分類器或自訂實體辨識器。AWS 帳戶如需有關訓練自訂模型的詳細資訊，請參閱自訂分類或自訂實體辨識。

所需的許可

IAM 政策聲明

在將以資源為基礎的政策新增至自訂模型之前，您需要 AWS Identity and Access Management (IAM) 中的許可。您的使用者、群組或角色必須附加原則，以便您可以建立、取得和刪除模型原則，如下列範例所示。

Example 管理自訂模型資源型政策的 IAM 政策

```
{
  "Effect": "Allow",
  "Action": [
    "comprehend:PutResourcePolicy",
    "comprehend:DeleteResourcePolicy",
    "comprehend:DescribeResourcePolicy"
  ],
  "Resource": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/foo/
version/*"
}
```

如需建立 IAM 政策的相關資訊，請參閱 IAM 使用者指南中的建立 IAM 政策。如需附加 IAM 政策的相關資訊，請參閱 IAM 使用者指南中的新增和移除 IAM 身分許可。

AWS KMS關鍵政策聲明

如果您要共用加密模型，則可能需要新增的權限AWS KMS。此需求取決於您在 Amazon Comprehend 中用來加密模型的 KMS 金鑰類型。

A AWS 擁有的金鑰是由AWS服務擁有和管理。如果您使用AWS 擁有的金鑰，則不需要為其新增權限 AWS KMS，您可以略過本節。

客戶管理的金鑰是您在中建立、擁有和管理的金鑰AWS 帳戶。如果您使用客戶受管金鑰，則必須在 KMS 金鑰政策中新增陳述式。

政策陳述式會授權一或多個實體 (例如使用者或帳戶) 執行解密模型所需的AWS KMS作業。

您可以使用條件鍵來幫助防止混淆的副問題。如需詳細資訊，請參閱 the section called “預防跨服務混淆代理人”。

使用原則中的下列條件金鑰來驗證存取 KMS 金鑰的實體。當使用者匯入模型時，AWS KMS會檢查來源模型版本的 ARN 是否符合條件。如果您未在原則中包含條件，指定的主體可以使用您的 KMS 金鑰來解密任何模型版本：

- aws：SourceArn— 將此條件密鑰與`kms:GenerateDataKey`和`kms:Decrypt`操作一起使用。
- kms: EncryptionContext — 將此條件金鑰與`kms:GenerateDataKey`、`kms:Decrypt`、和`kms:CreateGrant`動作搭配使用。

在下列範例中，原則授權使AWS 帳戶`444455556666`用所擁有之指定分類器模型的版本 1。AWS 帳戶 `111122223333`

Example 存取特定分類器模型版本的 KMS 金鑰原則

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "AWS":
                "arn:aws:iam::444455556666:root"
      },
      "Action": [
          "kms:Decrypt",
          "kms:GenerateDataKey"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": {
              "aws:SourceArn":
```

```
                "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/classifierName/version/1"
           }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
          "AWS":  "arn:aws:iam::444455556666:root"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
           "kms:EncryptionContext:aws:comprehend:arn":
              "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/classifierName/version/1"
        }
      }
    }
  ]
}
```

下列範例政策會授權使用者ExampleUser 透過 Amazon Comprehend 服務存取此 KMS 金鑰。AWS 帳戶 444455556666 ExampleRoleAWS 帳戶123456789012

Example 允許存取 Amazon Comprehend 服務的 KMS 金鑰政策 (替代方案 1)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "arn:aws:iam::444455556666:user/ExampleUser",
                "arn:aws:iam::123456789012:role/ExampleRole"
              ]
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
        ],
```

```
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "aws:SourceArn": "arn:aws:comprehend:*"

            }
        }
    },
    {
      "Effect": "Allow",
      "Principal": {
         "AWS": [
                "arn:aws:iam::444455556666:user/ExampleUser",
                "arn:aws:iam::123456789012:role/ExampleRole"
              ]
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
          "StringLike": {
              "kms:EncryptionContext:aws:comprehend:arn": "arn:aws:comprehend:*"
          }
      }
    }
  ]
}
```

下列範例政策授權AWS 帳戶444455556666透過 Amazon Comprehend 服務存取此 KMS 金鑰，並使用上述範例的替代語法。

Example 允許存取 Amazon Comprehend 服務的 KMS 金鑰政策 (替代方案 2)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "AWS": "arn:aws:iam::444455556666:root"
      },
      "Action": [
          "kms:Decrypt",
          "kms:GenerateDataKey",
```

```
            "kms:CreateGrant"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "kms:EncryptionContext:aws:comprehend:arn": "arn:aws:comprehend:*"
            }
        }
    }
  ]
}
```

如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的在 AWS KMS 中使用金鑰政策。

自訂模型的資源型政策

在另一個 Amazon Comprehend 使用者AWS 帳戶可以從您的AWS帳戶匯入自訂模型之前，您必須先授權他們這樣做。若要授權它們，您可以將以資源為基礎的策略新增至您要共用的模型版本。以資源為基礎的政策是您附加至中AWS資源的 IAM 政策。

當您將資源策略附加到自訂模型版本時，該策略會授權使用者、群組或角色對模型版本執行`comprehend:ImportModel`動作。

Example 自訂模型版本的以資源為基礎的政策

此範例會指定`Principal`屬性中的授權實體。資源「*」是指您附加策略的特定模型版本。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "comprehend:ImportModel",
      "Resource": "*",
      "Principal": {
        "AWS": [
                "arn:aws:iam::111122223333:root",
                "arn:aws:iam::444455556666:user/ExampleUser",
                "arn:aws:iam::123456789012:role/ExampleRole"
         ]
      }
    }
```

```
  ]
}
```

對於您附加到自訂模型的政策，這`comprehend:ImportModel`是 Amazon Comprehend 支援的唯一動作。

如需有關以資源為基礎的政策的詳細資訊，請參閱 IAM 使用者指南中的身分型政策和以資源為基礎的政策。

步驟 1：將以資源為基礎的策略新增至自訂模型

您可以使用AWS Management Console、AWS CLI或 Amazon Comprehend API 來新增以資源為基礎的政策。

AWS Management Console

您可以使用 Amazon Comprehend 在. AWS Management Console

若要新增以資源為基礎的政策

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 在左側導覽功能表中的「自訂」下，選擇包含您自訂模型的頁面：
   a. 如果您要共用自訂文件分類器，請選擇 [自訂分類]。
   b. 如果您要共用自訂實體辨識器，請選擇 [自訂實體辨識]。
3. 在模型清單中，選擇型號名稱以開啟其詳細資訊頁面。
4. 在「版本」下，選擇您要共用的模型版本名稱。
5. 在版本詳細資料頁面上，選擇 [標籤]、[VPC 和原則] 索引標籤。
6. 在以資源為基礎的政策區段中，選擇編輯。
7. 在 [編輯以資源為基礎的政策] 頁面上，執行下列動作：
   a. 在「策略名稱」中，輸入可協助您在建立策略後辨識該策略的名稱。
   b. 在「授權」(Authorize) 下，指定下列一或多個實體，以授權它們匯入模型：

欄位	定義與範例
服務主體	可存取此模型版本之服務的服務主體識別碼。例如：

欄位	定義與範例
	理解. 亞馬遜
AWS 帳戶身份證	AWS 帳戶可以訪問此模型版本。授權屬於該帳戶的所有使用者。例如： 111122223333, 123456789012
IAM 實體	ARN 適用於可存取此模型版本的使用者或角色。例如： 阿姆:符號::: 11112222333: 使用者/, 阿姆:AWN:: IAM:: ExampleUser ExampleRole

8. 在「共用」(Share) 底下，您可以複製模型版本的 ARN，以協助您與將匯入模型的人員共用。當使用者從其他模型匯入自訂模型時AWS 帳戶，需要模型版本 ARN。
9. 選擇儲存。Amazon Comprehend 會建立以資源為基礎的政策，並將其附加到您的模型。

AWS CLI

若要使用將以資源為基礎的策略新增至自訂模型AWS CLI，請使用指PutResourcePolicy令。 命令接受下列參數：

- `resource-arn`— 自訂模型的 ARN，包括模型版本。
- `resource-policy`— 定義以資源為基礎的原則以附加至自訂模型的 JSON 檔案。

  您也可以將原則提供為內嵌 JSON 字串。若要為您的政策提供有效的 JSON，請使用雙引號括住屬性名稱和值。如果 JSON 主體也用雙引號括起來，您可以逸出原則內的雙引號。
- `policy-revision-id`— Amazon Comprehend 指派給您正在更新之政策的修訂識別碼。如果您要建立沒有先前版本的新原則，請勿使用此參數。Amazon Comprehend 為您創建修訂 ID。

Example 使用指**`put-resource-policy`**令將以資源為基礎的政策新增至自訂模型

此範例會在名為策略檔案 JSON 的 JSON 檔案中定義原則，並將原則與模型產生關聯。該模型是名為 my cf1 的分類器的第 v2 版本。

```
$ aws comprehend put-resource-policy \
```

```
> --resource-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/mycf1/
version/v2 \
> --resource-policy file://policyFile.json \
> --policy-revision-id revision-id
```

資源策略的 JSON 檔案包含下列內容：

- 動作 — 策略會授權要使用的具名主參與者。comprehend:ImportModel
- 資源 — 自訂模型的 ARN。資源「*」是指您在指put-resource-policy令中指定的模型版本。
- 主體 — 此原則會授權使用者以及jane來自 12 AWS 帳戶 3456789012 的所有使用者。AWS 帳戶

```
{
"Version":"2012-10-17",
 "Statement":[
    {"Sid":"ResourcePolicyForImportModel",
     "Effect":"Allow",
     "Action":["comprehend:ImportModel"],
     "Resource":"*",
     "Principal":
         {"AWS":
            ["arn:aws:iam::444455556666:user/jane",
             "123456789012"]
         }
   }
 ]
}
```

Amazon Comprehend API

若要使用 Amazon Comprehend API 將以資源為基礎的政策新增至自訂模型，請使用 API 操作PutResourcePolicy。

您也可以在建立模型的 API 要求中，將原則新增至自訂模型。若要這麼做，請在提交CreateDocumentClassifier或CreateEntityRecognizer要求時提供 ModelPolicy 參數的原則 JSON。

步驟 2：提供其他人需要導入的詳細信息

現在您已將以資源為基礎的政策新增至自訂模型，您已授權其他 Amazon Comprehend 使用者將您的模型匯入他們的模型。AWS 帳戶但是，在匯入之前，您必須提供下列詳細資訊：

- 模型版本的 Amazon 資源名稱（ARN）。
- 包AWS 區域含模型的。任何匯入模型的人都必須使用相同的模型AWS 區域。
- 模型是否已加密，如果是，您使用的AWS KMS金鑰類型：AWS 擁有的金鑰還是客戶管理的金鑰。
- 如果您的模型使用客戶管理的金鑰加密，則您必須提供 KMS 金鑰的 ARN。匯入模型的任何人都必須在其AWS 帳戶中的 IAM 服務角色中包含 ARN。此角色授權 Amazon Comprehend 匯入期間使用 KMS 金鑰來解密模型。

如需有關其他使用者如何匯入模型的詳細資訊，請參閱從其他模型匯入自訂模型 AWS 帳戶。

從其他模型匯入自訂模型 AWS 帳戶

在 Amazon Comprehend 中，您可以導入另一個自定義模型。AWS 帳戶匯入模型時，您會在帳戶中建立新的自訂模型。您的新自訂模型是您匯入模型的完整訓練複本。

主題

開始之前

在您可以從另一個模型匯入自訂模型之前AWS 帳戶，請確定與您共用模型的人員執行下列動作：

- 授權您執行匯入。此授權會在附加至模型版本的以資源為基礎的原則中授與。如需詳細資訊，請參閱自訂模型的資源型政策。
- 提供您下列資訊：
  - 模型版本的 Amazon 資源名稱（ARN）。
  - 包AWS 區域含模型的。匯入時必須使AWS 區域用相同的項目。
  - 模型是否使用AWS KMS金鑰加密，如果是，則是使用的金鑰類型。

如果模型已加密，您可能需要採取其他步驟，具體取決於所使用的 KMS 金鑰類型：

- AWS 擁有的金鑰— 此類型的 KMS 金鑰由擁有和管理AWS。如果使用加密模型AWS 擁有的金鑰，則不需要其他步驟。
- 客戶管理金鑰 — 此類型的 KMS 金鑰是由AWS客戶在其中建立、擁有和管理AWS 帳戶。如果使用客戶管理的金鑰加密模型，則共用模型的人員必須：

- 授權您解密模型。此授權會在客戶受管金鑰的 KMS 金鑰原則中授與。如需詳細資訊，請參閱 AWS KMS關鍵政策聲明。
- 提供客戶管理金鑰的 ARN。您可以在建立 IAM 服務角色時使用此 ARN。此角色授權 Amazon Comprehend 用 KMS 金鑰解密模型。

所需的許可

您或您的管理員必須在 AWS Identity and Access Management (IAM) 中授權所需動作，才能匯入自訂模型。身為 Amazon Comprehend 使用者，您必須獲得 IAM 政策聲明的授權才能匯入。如果在匯入期間需要AWS KMS加密或解密，則 Amazon Comprehend 必須獲得授權，才能使用必要的金鑰。

IAM 政策聲明

您的使用者、群組或角色必須附加允許`ImportModel`動作的原則，如下列範例所示。

Example 用於匯入自訂模型的 IAM 政策

```
{
  "Effect": "Allow",
  "Action": [
    "comprehend:ImportModel"
  ],
  "Resource": "arn:aws:comprehend:us-west-2:111122223333:document-classifier/foo/
version/*"
}
```

如需建立 IAM 政策的相關資訊，請參閱 IAM 使用者指南中的建立 IAM 政策。如需附加 IAM 政策的相關資訊，請參閱 IAM 使用者指南中的新增和移除 IAM 身分許可。

適用於AWS KMS加密的 IAM 服務角色

匯入自訂模型時，您必須授權 Amazon Comprehend 在下列任一情況下使用AWS KMS金鑰：

- 您正在匯入使用客戶管理金鑰加密的自訂模型AWS KMS。在這種情況下，Amazon Comprehend 需要存取 KMS 金鑰，以便在匯入期間解密模型。
- 您想要加密透過匯入建立的新自訂模型，而且想要使用客戶管理的金鑰。在此情況下，Amazon Comprehend 需要存取您的 KMS 金鑰，才能加密新模型。

若要授權 Amazon Comprehend 用這些AWS KMS金鑰，您必須建立 IAM 服務角色。這種 IAM 角色可讓AWS服務代表您存取其他服務中的資源。如需有關服務角色的詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將權限委派給AWS服務。

如果您使用 Amazon Comprehend 主控台匯入，您可以讓 Amazon Comprehend 為您建立服務角色。否則，您必須先在 IAM 中建立服務角色，然後再匯入。

IAM 服務角色必須具有許可政策和信任政策，如下列範例所示。

Example 許可政策

下列許可政策允許 Amazon Comprehend 用來加密和解密自訂模型的AWS KMS作業。它會授與兩個 KMS 金鑰的存取權：

- 包含要匯入的模型中AWS 帳戶有一個 KMS 金鑰。它被用來加密模型，而 Amazon Comprehend 使用它來在導入過程中解密模型。
- 其他 KMS 金鑰位於匯AWS 帳戶入模型的中。Amazon Comprehend 使用此金鑰來加密匯入所建立的新自訂模型。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:CreateGrant"
        ],
        "Resource": [
            "arn:aws:kms:us-west-2:111122223333:key/key-id",
            "arn:aws:kms:us-west-2:444455556666:key/key-id"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDatakey"
        ],
        "Resource": [
            "arn:aws:kms:us-west-2:111122223333:key/key-id",
            "arn:aws:kms:us-west-2:444455556666:key/key-id"
        ],
```

```
        "Condition": {
            "StringEquals": {
                "kms:ViaService": [
                    "s3.us-west-2.amazonaws.com"
                ]
            }
        }
    }
  ]
}
```

Example 信任政策

下列信任政策允許 Amazon Comprehend 擔任該角色並取得其許可。它可讓comprehend.amazonaws.com服務主體執行sts:AssumeRole作業。若要協助避免混淆的副手，您可以使用一或多個全域條件內容索引鍵來限制權限範圍。對於aws:SourceAccount，指定匯入模型之使用者的帳戶 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "comprehend.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "444455556666"
        }
      }
    }
  ]
}
```

匯入自訂模型

您可以使用AWS Management Console、AWS CLI或 Amazon Comprehend API 匯入自訂模型。

AWS Management Console

您可以使用 Amazon Comprehend 在. AWS Management Console

匯入自訂模型

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 在左側導覽功能表的「自訂」下，選擇您要匯入之模型類型的頁面：
   a. 如果您要匯入自訂文件分類器，請選擇 [自訂分類]。
   b. 如果您要匯入自訂實體辨識器，請選擇「自訂實體辨識」。
3. 選擇「匯入版本」。
4. 在 [匯入模型版本] 頁面上，輸入下列詳細資訊：
   - 模型版本 ARN — 要匯入之模型版本的 ARN。
   - 模型名稱-匯入所建立之新模型的自訂名稱。
   - 版本名稱-匯入所建立之新模型版本的自訂名稱。
5. 對於模型加密，請選擇用於加密您透過匯入建立的新自訂模型的 KMS 金鑰類型：
   - 使用AWS擁有的金鑰 — Amazon Comprehend 會使用代表您建立、管理和使用的金鑰 AWS Key Management Service (AWS KMS) 來加密您的模型。AWS
   - 選擇不同的AWS KMS金鑰 (進階) — Amazon Comprehend 會使用您管理的客戶受管金鑰來加密您的模型。AWS KMS

     如果您選擇此選項，請選取您的 KMS 金鑰AWS 帳戶，或選擇 [建立金鑰] 來建立新金AWS KMS鑰。
6. 在「服務存取權」區段中，授予 Amazon Comprehend 存取權，以執行下列作業所需的任何AWS KMS金鑰：
   - 解密您匯入的自訂模型。
   - 加密您使用匯入建立的新自訂模型。

   您可以透過 IAM 服務角色授予存取權，該角色允許 Amazon Comprehend 使用 KMS 金鑰。

   對於服務角色，請執行下列其中一個動作：
   - 如果您有要使用的現有服務角色，請選擇 [使用現有的 IAM 角色]。然後，在「角色名稱」下選擇它。
   - 如果您希望 Amazon Comprehend 為您建立角色，請選擇建立 IAM 角色。
7. 如果您選擇讓 Amazon Comprehend 為您建立角色，請執行下列動作：

a. 在 [角色名稱] 中，輸入角色名稱尾碼，以協助您稍後辨識角色。

b. 對於來源 KMS 金鑰 ARN，請輸入用來加密您要匯入之模型的 KMS 金鑰的 ARN。Amazon Comprehend 使用此密鑰在導入過程中解密模型。

8. (選擇性) 在「標籤」區段中，您可以將標籤新增至透過匯入建立的新自訂模型。如需標籤自訂模型的更多資訊，請參閱標記新資源。

9. 選擇確認。

AWS CLI

您可以使 Amazon Comprehend 運行命令與. AWS CLI

Example 匯入模型指令

若要匯入自訂模型，請使用以下`import-model`指令：

```
$ aws comprehend import-model \
> --source-model arn:aws:comprehend:us-west-2:111122223333:document-classifier/foo/
version/bar \
> --model-name importedDocumentClassifier \
> --version-name versionOne \
> --data-access-role-arn arn:aws:iam::444455556666:role/comprehendAccessRole \
> --model-kms-key-id kms-key-id
```

此範例使用下列參數：

- `source-model`— 要匯入的自訂模型的 ARN。
- `model-name`-匯入所建立之新模型的自訂名稱。
- `version-name`-匯入所建立之新模型版本的自訂名稱。
- `data-access-role-arn`— IAM 服務角色的 ARN，可讓 Amazon Comprehend 使用必要的AWS KMS金鑰來加密或解密自訂模型。
- `model-kms-key-id`— Amazon Comprehend 用來加密您透過此匯入建立的自訂模型的 KMS 金鑰的 ARN 或識別碼。這個金鑰必須AWS KMS在您的AWS 帳戶.

Amazon Comprehend API

若要使用 Amazon Comprehend API 匯入自訂模型，請使用 API 動ImportModel作。

飞轮

Amazon Comprehend 飛輪可簡化隨著時間的推移改善自訂模型的程序。您可以使用飛輪來協調與訓練相關的工作，以及評估新的自訂模型版本。Flywheels 支援用於自訂分類和自訂實體辨識的純文字自訂模型。

主題

飛輪概述

飛輪是一種 Amazon Comprehend 資源，可協調新版本自訂模型的訓練和評估。您可以建立飛輪以使用現有的訓練模型，或者 Amazon Comprehend 可以為飛輪建立和訓練新模型。將飛輪與純文字自訂模型搭配使用，以進行自訂分類或自訂圖元辨識。

您可以使用 Amazon Comprehend 主控台或 API 來設定和管理飛輪。您也可以使用AWS CloudFormation配置飛輪。

當您建立飛輪時，Amazon Comprehend 會在您的帳戶中建立資料湖。資料湖儲存和管理所有飛輪資料，例如所有模型版本的訓練資料和測試資料。

您可以將作用中模型版本設定為要用於推論任務或 Amazon Comprehend 端點的飛輪模型版本。最初，飛輪包含模型的一個版本。隨著時間的推移，當您訓練新模型版本時，您可以選取效能最佳的版本作為使用中模型版本。當使用者指定飛輪 ARN 來執行推論任務時，Amazon Comprehend 會使用飛輪的作用中模型版本來執行任務。

您會定期取得模型的新標籤資料 (訓練資料或測試資料)。您可以建立一個或多個資料集，讓新資料可供飛輪使用。資料集包含用於訓練或測試與飛輪關聯的自訂模型的輸入資料。Amazon Comprehend 將輸入資料上傳到飛輪的資料湖。

若要將新資料集合到您的自訂模型中，請建立並執行飛輪反覆項目。飛輪版序是使用新資料集來評估作用中模型版本並訓練新模型版本的工作流程。根據現有和新模型版本的量度，您可以決定是否將新模型版本升級為作用中版本。

您可以使用飛輪作用中模型版本來執行自訂分析 (即時或非同步作業)。若要使用飛輪模型進行即時分析，您必須建立飛輪的端點。

使用飛輪不收取額外費用。但是，當您執行飛輪反覆運算時，會產生訓練新模型版本和儲存模型資料的標準費用。如需詳細的定價資訊，請參 Amazon Comprehend 定價。

主題

- 飛輪資料集
- 創建飛輪
- 飛輪狀態
- 飛輪迭代

飛輪資料集

若要將新的標籤資料新增至飛輪，您需要建立資料集。您可以將每個資料集設定為訓練資料或測試資料。您可以將資料集與特定飛輪和自訂模型相關聯。

建立資料集之後，Amazon Comprehend 會將資料上傳到飛輪的資料湖。如需詳細資訊，請參閱 飛輪資料湖。

創建飛輪

建立飛輪時，您可以將飛輪與現有訓練模型相關聯，或者飛輪可以建立新模型。

當您使用現有模型建立飛輪時，您可以指定現行模型版本。Amazon Comprehend 會將模型的訓練資料和測試資料複製到飛輪的資料湖中。確保模型訓練和測試資料與建立模型時存在於相同的 Amazon S3 位置。

若要為新模型建立飛輪，請在建立飛輪時提供訓練資料的資料集 (以及測試資料的選用資料集)。當您執行飛輪以建立第一個飛輪迭代時，飛輪會訓練新模型。

訓練自訂模型時，您可以指定自訂標籤清單 (自訂分類) 或自訂圖元 (自訂圖元辨識) 以供模型辨識。請注意以下有關自定義標籤/實體的要點：

- 當您為新模型建立飛輪時，您在建立飛輪期間提供的標籤/圖元清單是飛輪的最終清單。

- 當您從現有模型建立飛輪時，與該模型相關聯的標籤/圖元清單會成為飛輪的最終清單。
- 如果您將新資料集與飛輪產生關聯，且該資料集包含其他標籤/實體，Amazon Comprehend 會忽略新的標籤/實體。
- 您可以使用 API 作業檢閱飛輪的標籤/實體清單。DescribeFlywheel

> **Note**
>
> 對於自訂分類，Amazon Comprehend 會在飛輪狀態變為「作用中」之後填入標籤清單。等到飛輪處於活動狀態，然後再調用 DescribeFlywheel API 操作。

飛輪狀態

飛輪在下列狀態之間轉換：

- 創建-Amazon Comprehend 正在創建飛輪資源。您可以在飛輪上執行讀取操作，例如`DescribeFlywheel`。
- 作用中-飛輪處於作用中狀態。您可以決定飛輪版序是否正在進行中，並檢視迭代的狀態。您可以在飛輪上執行讀取動作以及諸如和之類的動作。`DeleteFlywheel` `UpdateFlywheel`
- 更新-Amazon Comprehend 正在更新飛輪。您可以在飛輪上執行讀取操作。
- 刪除-Amazon Comprehend 正在刪除飛輪。您可以在飛輪上執行讀取操作。
- 失敗-飛輪建立作業失敗。

Amazon Comprehend 刪除飛輪之後，您可以保留飛輪資料湖中所有模型資料的存取權。Amazon Comprehend 會刪除管理飛輪資源所需的所有內部中繼資料。Amazon Comprehend 也會刪除與此飛輪相關聯的資料集 (模型資料會儲存在資料湖中)。

飛輪迭代

當您取得飛輪模型的新訓練或測試資料時，您會建立一或多個新資料集，以便將新資料上傳至飛輪的資料湖。

然後，您可以執行飛輪以建立新的飛輪迭代。飛輪迭代會使用新資料來評估目前的作用中模型版本，並將結果儲存在資料湖中。飛輪也會建立並訓練新的模型版本。

如果新模型的效能優於目前的現行模型版本，您可以將新模型版本晉級為現行模型版本。您可以使用控制台或 UpdateFlywheelAPI 操作來更新現行模型版本。

飛輪資料湖

當您建立飛輪時，Amazon Comprehend 會在您的帳戶中建立一個資料湖，以包含所有飛輪資料，例如模型版本所需的輸入和輸出資料。

Amazon Comprehend 會在您建立飛輪時指定的 Amazon S3 位置建立資料湖。您可以將該位置指定為 Amazon S3 儲存貯體或 Amazon S3 儲存貯體中的新資料夾。

資料湖資料夾結構

當 Amazon Comprehend 建立資料湖時，會在 Amazon S3 位置設定下列資料夾結構。

> **Warning**
>
> 亞馬遜管理資料湖資料夾的組織和內容。請務必使用 Amazon Comprehend API 操作來修改資料湖資料夾，否則您的飛輪可能無法正常運作。

```
Document Pool
Annotations Pool
Staging
Model Datasets
  (data for each version of the model)
  VersionID-1
    Training
    Test
    ModelStats
  VersionID-2
    Training
    Test
    ModelStats
```

若要檢視模型版本的訓練評量，請執行下列步驟：

1. 在資料湖的根層級開啟名為「模型資料集」的資料夾。此資料夾包含每個模型版本的子資料夾。
2. 開啟感興趣的模型版本的資料夾。
3. 開啟名為的資料夾ModelStats以檢視模型的統計資料。

資料湖管理

Amazon Comprehend 會執行下列任務，以代表您管理資料湖：

- 定義資料湖的資料夾結構，並將資料集內嵌到適當的資料夾中。
- 管理訓練模型所需的輸入文件 (例如文字檔案和註釋檔案)。
- 管理與每個模型版本相關聯的訓練和評估輸出資料。
- 管理儲存在資料湖中之檔案的加密。

Amazon Comprehend 會執行資料湖的所有資料建立和更新作業。您可以保留對資料湖中資料的完整存取權。例如：

- 您可以完整存取資料湖的內容。
- 刪除飛輪後，資料湖仍然可用。
- 您可以為包含資料湖的 Amazon S3 儲存貯體設定存取日誌。
- 您可以為資料提供加密金鑰。您可以在建立飛輪時指定這些項目。

建議遵循下列最佳實務：

- 請勿手動將自己的資料夾或檔案新增至資料湖。請勿修改或刪除資料湖中的任何檔案。
- 請務必使用 Amazon Comprehend 建立和更新操作來新增或修改資料湖中的資料。例如，用於`CreateDataset`提供訓練或測試資料，`StartFlywheelIteration`以及產生模型版本的評估資料。
- 資料湖結構可能會隨著時間的推移而發展。請勿建立明確依賴資料湖結構的下游指令碼或程式。
- 當您為飛輪提供資料湖位置時，我們建議您為與所有飛輪相關的資料建立一個共同字首，或為每個飛輪使用不同的字首。我們不建議使用一個飛輪的完整資料湖路徑作為另一個飛輪的前置詞。

IAM 政策和許可

您可以設定下列原則和權限以使用飛輪：

- the section called “設定 IAM 使用者許可”供用戶訪問飛輪操作。
- (選擇性) the section called “設定AWS KMS金鑰的權限” 用於資料湖。
- the section called “建立資料存取角色”授權 Amazon Comprehend 存取資料湖。

設定 IAM 使用者許可

若要使用飛輪功能，請將適當的許可政策新增至您的 AWS Identity and Access Management (IAM) 身分識別 (使用者、群組和角色)。

下列範例顯示建立資料集、建立和管理飛輪，以及執行飛輪的權限原則。

Example 管理飛輪的 IAM 政策

```
{
  "Effect": "Allow",
  "Action": [
    "comprehend:CreateFlywheel",
    "comprehend:DeleteFlywheel",
    "comprehend:UpdateFlywheel",
    "comprehend:ListFlywheels",
    "comprehend:DescribeFlywheel",
    "comprehend:CreateDataset",
    "comprehend:DescribeDataset",
    "comprehend:ListDatasets",
    "comprehend:StartFlywheelIteration",
    "comprehend:DescribeFlywheelIteration",
    "comprehend:ListFlywheelIterationHistory"
  ],
  "Resource": "*"
}
```

如需建立 IAM 政策的相關資訊 Amazon Comprehend 請參閱。Amazon Comprehend 如何與 IAM 合作

設定AWS KMS金鑰的權限

如果您在資料湖中為資料使用AWS KMS金鑰，請設定所需的權限。如需詳細資訊，請參閱使用 KMS 加密所需的權限。

建立資料存取角色

您可以在適用於 Amazon Comprehend 的 IAM 中建立資料存取角色，以存取資料湖中的飛輪資料。如果您使用主控台建立飛輪，系統可以選擇性地為此建立新角色。如需更多詳細資訊，請參閱 非同步作業所需的角色型權限。

使用控制台設定飛輪

您可以使用 Amazon Comprehend 主控台來建立、更新和刪除飛輪。

當您建立飛輪時，Amazon Comprehend 會建立一個資料湖來存放飛輪所需的所有資料，例如每個模型版本的訓練資料和測試資料。

刪除飛輪時，Amazon Comprehend 不會刪除資料湖或與飛輪關聯的模型。

在建立新飛輪創建飛輪之前，請先檢閱一節中的資訊。

主題

建立飛輪

建立飛輪時，必要的組態欄位取決於飛輪是用於現有的自訂模型還是新模型。

建立飛輪的步驟

1. 登入AWS Management Console並開啟 Amazon Comprehend 主控台。
2. 從左側選單中選擇飛輪。
3. 從飛輪表格中選擇「建立新飛輪」。
4. 在飛輪名稱下，輸入飛輪的名稱。
5. （可選）若要為現有模型建立飛輪，請在「現行模型版本」下配置欄位。
    a. 從「型號」下拉式清單中，選取模型
    b. 從版本下拉式清單中，選取模型版本。
6. (選擇性) 若要為飛輪建立新的分類器模型，請在「自訂模型類型」下選擇「自訂」分類，然後依照下列步驟配置參數。
    a. 在「語言」下，選取模型的語言。
    b. 在分類器模式下，選擇單標籤模式或多標籤模式。
    c. 在「自訂標籤」下，輸入一或多個用於訓練模型的自訂標籤。每個標籤都必須符合輸入訓練資料中的其中一個類別。

7. (選擇性) 若要為飛輪建立新的圖元辨識模型，請在自訂模型類型之下，選擇一個自訂圖元辨識，然後依照下列步驟配置參數。

   a. 在「語言」下，選取模型的語言。

   b. 在「自訂實體類型」下，輸入最多 25 個用於訓練模型的自訂實體。每個標籤都必須符合輸入訓練資料中的其中一個實體類型。

      若要建立多個標籤，請多次執行下列步驟。

      i. 輸入自訂標籤。標籤必須全部大寫。使用底線作為標籤中單詞之間的分隔符。

      ii. 選擇 [新增類型]。

      若要移除其中一個已新增的標籤，請選擇標籤名稱右邊的 [X]。

8. 設定磁碟區加密、模型加密和資料湖加密的選擇。針對這些項目，請選擇要使用AWS擁有的 KMS 金鑰還是您有權使用的金鑰。

   - 如果您使用AWS擁有的 KMS 金鑰，則沒有其他參數。
   - 如果您正在使用另一個現有金鑰，請針對 KMS 金鑰 ARN 輸入金鑰識別碼的 ARN。
   - 如果您要建立新金鑰，請選擇 [建立 AWS KMS 金鑰]。

   如需建立和使用 KMS 金鑰及相關加密的詳細資訊，請參閱AWS Key Management Service。

   a. 設定磁碟區加密金鑰。Amazon Comprehend 會在處理任務時使用此金鑰來加密儲存磁碟區中的資料。選擇要使用AWS擁有的 KMS 金鑰還是您有權使用的金鑰。

   b. 設定模型加密金鑰。Amazon Comprehend 使用此金鑰來加密此模型版本的模型資料。

9. 設定資料湖位置。如需詳細資訊，請參閱 資料湖管理。

10. (選擇性) 設定資料湖加密金鑰。Amazon Comprehend 使用此金鑰來加密資料湖中的所有檔案。

11. (選擇性) 設定 VPC 設定。在 VPC 下輸入 VPC ID，或從下拉列表中選擇 ID。

    1. 在「子網路」下選擇子網路。選取第一個子網路後，您可以選擇其他子網路。
    2. 在「安全性群組」下，選擇要使用的安全性群組 (如果您已指定安全性群組)。選取第一個安全性群組後，您可以選擇其他群組。

12. 設定服務存取權限。

    1. 如果您選取 [使用現有的 IAM 角色]，請在下拉式清單中選取角色名稱。

2. 如果您選取「建立 IAM 角色」，Amazon Comprehend 建立新角色。主控台會顯示 Amazon Comprehend 對該角色設定的許可。在「角色名稱」下，輸入角色的描述性名稱。

13. (選擇性) 設定標籤設定。若要新增標籤，請在「標籤」下輸入鍵值配對。選擇 Add tag (新增標籤)。若要在建立飛輪之前移除此配對，請選擇「移除標籤」。如需詳細資訊，請參閱 標記您的 資源。
14. 選擇 Create (建立)。

更新飛輪

您只能在建立飛輪時設定飛輪名稱、資料湖位置、模型類型和模型組態。

更新飛輪時，如果模型類型和組態選項與目前模型相同，您可以指定不同的模型。您可以配置新的現行模型版本。您也可以更新加密詳細資料、服務存取權限和 VPC 設定。

更新飛輪的步驟

1. 登入AWS Management Console並開啟 Amazon Comprehend 主控台。
2. 從左側選單中選擇飛輪。
3. 從飛輪表格中，選擇要更新的飛輪。
4. 在「現行模型版本」下，從「模型」下拉式清單中選擇模型，然後選擇模型版本。

   表單會填入模型類型和模型組態。
5. (選擇性) 設定磁碟區加密和模型加密設定。
6. (選擇性) 設定資料湖加密設定。
7. 設定服務存取權限。
8. (選擇性) 設定 VPC 設定。
9. (選擇性) 設定標籤設定。
10. 選擇儲存。

刪除飛輪

刪除飛輪的步驟

1. 登入AWS Management Console並開啟 Amazon Comprehend 主控台。
2. 從左側選單中選擇飛輪。

3. 從飛輪表格中，選擇要刪除的飛輪。
4. 選擇刪除。

使用 API 設定飛輪

您可以使用 Amazon Comprehend API 來建立、更新和刪除飛輪。

當您建立飛輪時，Amazon Comprehend 會建立一個資料湖來存放飛輪所需的所有資料，例如每個模型版本的訓練資料和測試資料。

刪除飛輪時，Amazon Comprehend 不會刪除資料湖或與飛輪關聯的模型。

如果飛輪正在運行迭代或創建數據集，飛輪刪除操作失敗。

在建立新飛輪創建飛輪之前，請先檢閱一節中的資訊。

為現有模型建立飛輪

使用此CreateFlywheel作業為現有模型建立飛輪。

Example

```
aws comprehend create-flywheel  \
    --flywheel-name "myFlywheel2"  \
    --active-model-arn  "modelArn"  \
    --data-access-role-arn   arn:aws::iam::111122223333:role/testFlywheelDataAccess \
    --data-lake-s3-uri": "https://s3-bucket-endpoint"   \
```

如果操作成功，則響應包括飛輪 ARN。

```
{
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

為新模型建立飛輪

使用此CreateFlywheel作業為新自訂分類模型建立飛輪。

Example

```
aws comprehend create-flywheel \
    --flywheel-name "myFlywheel2" \
    --data-access-role-arn  arn:aws::iam::111122223333:role/testFlywheelDataAccess \
    --model-type "DOCUMENT_CLASSIFIER" \
    --data-lake-s3-uri  "s3Uri"  \
    --task-config  file://taskConfig.json
```

任務配置 .json 文件包含以下內容。

```
{
    "LanguageCode": "en",
    "DocumentClassificationConfig": {
        "Mode": "MULTI_LABEL",
        "Labels": ["optimism", "anger"]
    }
}
```

API 回應主體包含下列內容。

```
{
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

描述一個飛輪

使用 Amazon Comprehend DescribeFlywheel操作來擷取有關飛輪的已設定資訊。

```
aws comprehend describe-flywheel \
    --flywheel-arn  "flywheelArn"
```

API 回應主體包含下列內容。

```
{
  "FlywheelProperties": {
      "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/
myTestFlywheel",
      "DataAccessRoleArn": "arn:aws::iam::111122223333:role/Admin",
```

```
      "TaskConfig": {
          "LanguageCode": "en",
          "DocumentClassificationConfig": {
              "Mode": "MULTI_LABEL"
          }
      },
      "DataLakeS3Uri": "s3://my-test-datalake/flywheelbasictest/myTestFlywheel/
schemaVersion=1/20220801T014326Z",
      "Status": "ACTIVE",
      "ModelType":  "DOCUMENT_CLASSIFIER",
      "CreationTime": 1659318206.102,
      "LastModifiedTime": 1659318249.05
  }
}
```

更新飛輪

使用此UpdateFlywheel作業更新飛輪的可修改組態值。

一些配置字段是帶有子字段的 JSON 結構。若要更新一個或多個子欄位，請提供所有子欄位的值 (Amazon Comprehend 會將要求中遺漏的任何分欄的值設定為空值)。

如果您在UpdateFlywheel請求中省略頂層參數，Amazon Comprehend 不會變更飛輪中的參數或其任何子欄位的值。

若要在飛輪上新增或移除標籤，請使用TagResource和UntagResource操作。

您可以透過設定ActiveModelArn參數來晉級模型版本，如下列範例所示。

```
aws comprehend update-flywheel \
    --region aws-region \
    --flywheel-arn  "flywheelArn" \
    --active-model-arn  "modelArn" \
```

API 回應主體包含下列內容。

```
{
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

刪除飛輪

使用 Amazon Comprehend DeleteFlywheel操作刪除飛輪。

```
aws comprehend delete-flywheel \
    --flywheel-arn  "flywheelArn"
```

成功的 API 響應包含一個空的響應消息正文

列出飛輪

使用 Amazon Comprehend ListFlywheels作業擷取目前區域中的飛輪清單。

```
aws comprehend list-flywheel \
    --region aws-region \
    --endpoint-url  "uri"
```

API 回應主體包含下列內容。

```
{
    "FlywheelSummaryList": [
        {
            "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/
myTestFlywheel",
            "DataLakeS3Uri": "s3://my-test-datalake/flywheelbasictest/myTestFlywheel/
schemaVersion=1/20220801T014326Z",
            "Status": "ACTIVE",
            ""ModelType":  "DOCUMENT_CLASSIFIER",
            "CreationTime": 1659318206.102,
            "LastModifiedTime": 1659318249.05
        }
    ]
}
```

設定資料集

若要將標籤化的訓練或測試資料新增至飛輪，請使用 Amazon Comprehend 主控台或 API 建立資料集。

您可以將每個資料集設定為訓練資料或測試資料。您可以將資料集與特定飛輪和自訂模型相關聯。當您建立資料集時，Amazon Comprehend 會將資料上傳到飛輪的資料湖。如需有關訓練資料檔案格式的詳細資訊，請參閱準備分類器訓練資料或準備實體辨識器訓練資料。

當您刪除飛輪時，Amazon Comprehend 會刪除這些資料集。上傳的資料仍然可以在資料湖中使用。

建立資料集 (主控台)

建立資料集

1. 登入AWS Management Console並開啟 Amazon Comprehend 主控台。
2. 從左側功能表中選擇飛輪，然後選擇您要新增資料的飛輪。
3. 選擇資料集索引標籤。
4. 在訓練資料集或測試資料集表格中，選擇建立資料集。
5. 在 [資料集詳細資料] 底下，輸入資料集的名稱和選用說明。
6. 在 [資料規格] 下，選擇 [資料格式] 和 [資料集類型] 設定欄位。
7. (選擇性) 在「輸入格式」下，選擇輸入文件的格式。
8. 在 S3 上的註釋位置下，輸入註釋檔案的 Amazon S3 位置。
9. 在 S3 上的訓練資料位置下，輸入文件檔案的 Amazon S3 位置。
10. 選擇建立。

建立資料集 (API)

您可以使用此CreateDataset作業建立資料集。

Example

```
aws comprehend create-dataset \
    --flywheel-arn "myFlywheel2" \
    --dataset-name "my-training-dataset"
    --dataset-type "TRAIN"
    --description "my training dataset"
    --cli-input-json file://inputConfig.json
}
```

inputConfig.json 檔案包含下列內容。

```
{
    "DataFormat": "COMPREHEND_CSV",
    "DocumentClassifierInputDataConfig": {
        "S3Uri": "s3://my-comprehend-datasets/multilabel_train.csv"
    }
}
```

若要在資料集上新增或移除標籤，請使用TagResource和UntagResource作業。

描述資料集

使用 Amazon Comprehend DescribeDataset操作來擷取有關飛輪的已設定資訊。

```
aws comprehend describe-dataset \
    --dataset-arn  "datasetARN"
```

響應包含以下內容。

```
{
   "DatasetProperties": {
      "DatasetArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/
myTestFlywheel/dataset/train-dataset",
      "DatasetName": "train-dataset",
      "DatasetType": "TRAIN",
      "DatasetS3Uri": "s3://my-test-datalake/flywheelbasictest/myTestFlywheel/
schemaVersion=1/20220801T014326Z/datasets/train-dataset/20220801T194844Z",
      "Description": "Good Dataset",
      "Status": "COMPLETED",
      "NumberOfDocuments": 90,
      "CreationTime": 1659383324.297
  }
}
```

飛輪迭代

使用飛輪反覆項目來協助您建立和管理新模型版本。

主題

- 版序工作流
- 管理版序 (主控台)

- 管理反覆項目 (API)

版序工作流

飛輪從訓練過的模型版本開始，或使用初始資料集來訓練模型版本。

隨著時間的推移，當您取得新的標籤資料時，您會訓練新的模型版本，以改善飛輪模型的效能。當您執行飛輪時，它會建立新版序來訓練和評估新模型版本。如果新模型版本的效能優於現有的現行模型版本，您可以升級新模型版本。

飛輪迭代工作流程包括下列步驟：

1. 您可以為新的標籤資料建立資料集。
2. 您可以執行飛輪來建立新的版序。迭代遵循以下步驟來訓練和評估新模型版本：
   a. 使用新資料評估現行模型版本。
   b. 使用新資料訓練新模型版本。
   c. 將評估和訓練結果儲存在資料湖中。
   d. 返回兩個模型的 F1 分數。
3. 迭代完成後，您可以比較現有活動模型和新模型的 F1 分數。
4. 如果新模型版本具有優異的效能，您可以將其提升為作用中模型版本。您可以使用主控台或 API 來推廣新模型版本。

管理版序 (主控台)

您可以使用主控台來啟動新的版序，並查詢進行中版序的狀態。您也可以檢視已完成版序的結果。

啟動飛輪迭代（控制台）

在開始新的迭代之前，請先建立一或多個新的訓練或測試資料集。請參閱 設定資料集

啟動飛輪迭代（控制台）

1. 登入AWS Management Console並開啟 Amazon Comprehend 主控台。
2. 從左側功能表中選擇飛輪。
3. 從飛輪表格中選擇飛輪。
4. 選擇運行飛輪。

分析迭代結果（控制台）

執行飛輪迭代之後，控制台會在 F lywheels 反覆項目表中顯示結果。

推廣新模型版本（控制台）

在主控台的模型詳細資訊頁面中，您可以將新的模型版本升級為現行模型版本。

將飛輪模型版本升級為現行模型版本（控制台）

1. 登入AWS Management Console並開啟 Amazon Comprehend 主控台。
2. 從左側功能表中選擇飛輪。
3. 從飛輪表格中選擇飛輪。
4. 從飛輪詳細資訊頁表格中，從 Fly wheels 版序表中選擇要推進的版本。
5. 選擇「建立現行模型」。

管理反覆項目 (API)

您可以使用 Amazon Comprehend API 開始新的迭代，並查詢進行中迭代的狀態。您也可以檢視已完成版序的結果。

開始飛輪迭代（API）

使用 Amazon Comprehend StartFlywheelIteration操作啟動飛輪迭代。

```
aws comprehend start-flywheel-iteration \
    --flywheel-arn  "flywheelArn"
```

響應包含以下內容。

```
{
  "FlywheelIterationArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name"
}
```

推廣新模型版本 (API)

使用此UpdateFlywheel作業將模型版本晉級為現行模型版本。

將ActiveModelArn參數設定的UpdateFlywheel請求傳送至新使用中模型版本的 ARN。

```
aws comprehend update-flywheel \
    --active-model-arn  "modelArn" \
```

響應包含以下內容。

```
{
  "FlywheelArn": "arn:aws::comprehend:aws-region:111122223333:flywheel/name",
  "ActiveModelArn": "modelArn"
}
```

描述飛輪迭代結果（API）

Amazon Comprehend DescribeFlywheelIteration作業會在執行到完成之後傳回迭代的相關資訊。

```
aws comprehend describe-flywheel-iteration \
 --flywheel-arn "flywheelArn" \
 --flywheel-iteration-id  "flywheelIterationId" \
 --region aws-region
```

響應包含以下內容。

```
{
    "FlywheelIterationProperties": {
        "FlywheelArn": "flywheelArn",
        "FlywheelIterationId": "iterationId",
        "CreationTime": <createdAt>,
        "EndTime": <endedAt>,
        "Status": <status>,
        "Message": <message>,
        "EvaluatedModelArn": "modelArn",
        "EvaluatedModelMetrics": {
            "AverageF1Score": <value>,
            "AveragePrecision": <value>,
            "AverageRecall": <value>,
            "AverageAccuracy": <value>
        },
        "TrainedModelArn": "modelArn",
        "TrainedModelMetrics": {
            "AverageF1Score": <value>,
            "AveragePrecision": <value>,
```

```
            "AverageRecall": <value>,
            "AverageAccuracy": <value>
        }
    }
}
```

獲取迭代歷史記錄（API）

使用此作ListFlywheelIterationHistory業可取得有關版序記錄的資訊。

```
aws comprehend list-flywheel-iteration-history \
 --flywheel-arn "flywheelArn"
```

響應包含以下內容。

```
{
    "FlywheelIterationPropertiesList": [
        {
            "FlywheelArn": "<flywheelArn>",
            "FlywheelIterationId": "20220907T214613Z",
            "CreationTime": 1662587173.224,
            "EndTime": 1662592043.02,
            "Status": "<status>",
            "Message": "<message>",
            "EvaluatedModelArn": "modelArn",
            "EvaluatedModelMetrics": {
                "AverageF1Score": 0.833333333333333,
                "AveragePrecision": 0.75,
                "AverageRecall": 0.9375,
                "AverageAccuracy": 0.8125
            },
            "TrainedModelArn": "modelArn",
            "TrainedModelMetrics": {
                "AverageF1Score": 0.865497076023392,
                "AveragePrecision": 0.7636363636363637,
                "AverageRecall": 1.0,
                "AverageAccuracy": 0.84375
            }
        }
    ]
}
```

使用飛輪進行分析

您可以使用飛輪的現行模型版本來執行自訂分類或圖元辨識的分析。現行模型版本是可配置的。您可以使用控制台或 UpdateFlywheelAPI 操作將模型的新版本設置為活動模型版本。

要使用飛輪，請在配置分析任務時指定飛輪 ARN 而不是自訂模型 ARN。Amazon Comprehend 使用飛輪的活動模型版本來執行分析。

即時分析

您可以使用端點執行即時分析。當您建立或更新端點時，您可以使用飛輪 ARN 來設定它，而不是模型 ARN。執行即時分析時，請選取與飛輪相關聯的端點。Amazon Comprehend 使用飛輪的活動模型版本運行分析。

當您使用UpdateFlywheel為飛輪設定新的現行模型版本時，端點會自動更新以開始使用新的現行模型版本。如果您不希望端點自動更新，請將端點 (使用 UpdateEndpoint) 設定為直接使用模型版本 ARN。如果飛輪活動模型版本發生變化，端點將繼續使用此模型版本。

對於自訂分類，請使用 ClassifyDocumentAPI 作業。如需自訂實體辨識，請使用 DetectEntitiesAPI 要求。在EndpointArn參數中提供飛輪的端點。

您也可以使用主控台執行即時分析，以進行自訂分類或自訂實體辨識。

非同步工作

對於自訂分類，請使用 StartDocumentClassificationJobAPI 要求來啟動非正常工作。提供FlywheelArn參數而不是DocumentClassifierArn.

如需自訂實體辨識，請使用 StartEntitiesDetectionJobAPI 要求。提供FlywheelArn參數而不是EntityRecognizerArn.

您可以使用主控台執行非同步分析工作，以進行自訂分類或自訂實體辨識。建立工作時，請在辨識器模型或分類器模型欄位中輸入飛輪 ARN。

管理 Amazon Comprehend 端點

在 Amazon Comprehend 中，端點可讓您的自訂模型用於即時分類或實體偵測。建立端點後，您可以隨著業務需求的發展對其進行變更。例如，您可以監控端點使用率並套用自 auto 擴展來自動設定端點佈建以符合您的容量需求。您可以從單一檢視管理所有端點，當您不再需要端點時，可以將其刪除以節省成本。

您必須先建立一個端點，才能管理端點。如需詳細資訊，請參閱下列程序：

- 建立自訂分類的端點
- 建立自訂實體偵測的端點

主題

Amazon Comprehend 端點概觀

Amazon Comprehend 主控台的端點頁面可讓您全域檢視端點。在端點概觀頁面中，您可以在一個位置查看所有端點，以了解端點使用情況與實際資源使用情況。在端點頁面的右上角，您可以指定要檢視的端點 — 所有端點、自訂分類器端點或自訂實體端點。

您可以從此頁面建立、更新、監控和刪除端點。在端點概觀區段中，您可以檢視端點清單、端點託管的自訂模型、它們的建立時間、佈建的輸送量以及端點的狀態。當您從端點概觀表格中選取特定端點時，會顯示端點詳細資訊。

此外，如果您是AWS商業 Support 或AWS企業支 Support 客戶，您可以存取特定於您端點的 Trusted Advisor 檢查。如需進一步了解，請參閱 Trusted Advisor 與 Amazon Comprehend 使用。如需檢查和說明的完整清單，請參閱受信任的建議程式最佳作法。

如需有關管理端點的詳細資訊，請參閱下列主題。

- 使用 Amazon Comprehend 端點
- 監控 Amazon Comprehend 點
- 更新 Amazon Comprehend 端點
- Trusted Advisor 與 Amazon Comprehend 使用
- 刪除 Amazon Comprehend 端點

> **Important**
>
> 即時自訂分類的成本取決於您設定的輸送量和端點處於作用中狀態的時間長度。如果您不再使用該端點，或者長時間未使用該端點，則應設定 auto 擴展政策以降低成本。或者，如果您不再使用端點，則可以刪除端點以避免產生額外費用。如需更多詳細資訊，請參閱 使用端點自動調整。

使用 Amazon Comprehend 端點

您可以建立端點以使用自訂模型執行即時分析。端點包含受管資源，可讓您的自訂模型可用於即時推論。

Amazon Comprehend 使用推論單元 (IU) 將輸送量指派給端點。IU 代表每秒 100 個字元的資料輸送量。您最多可以使用 10 個推論單元佈建端點。您可以透過更新端點來擴展或縮減端點輸送量。

如果您的輸入文件包含半結構化文件或影像檔案，則每秒 100 個字元的輸送量是針對從輸入檔案中擷取的字元。您為端點佈建的 IU 數目取決於輸入文件的字元密度。

ClassifyDocument和 DetectEntitiesAPI 回應包含每個輸入頁面的字元計數。您可以使用此資訊來估算要佈建的推論單元數目，以達到所需的輸送量。

完成即時分析後，請刪除端點，因為只要端點處於作用中狀態，該端點的費用就會繼續進行。當您準備好執行進一步的即時分析時，您可以建立另一個端點。

如需有關端點成本的詳細資訊，請參閱 Amazon Comprehend 定價。

建立端點後，您可以使用 Amazon 對其進行監控 CloudWatch、更新以變更其推論單元，或在不再需要時將其刪除。如需更多詳細資訊，請參閱 監控 Amazon Comprehend 點。

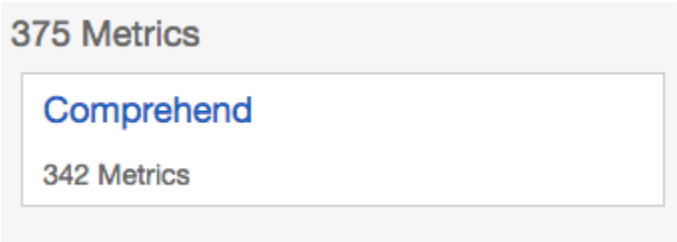
監控 Amazon Comprehend 點

您可以透過增加或減少推論單元 (IU) 的數目來調整端點的輸送量。如需更新端點的詳細資訊，請參閱the section called “更新端點”。

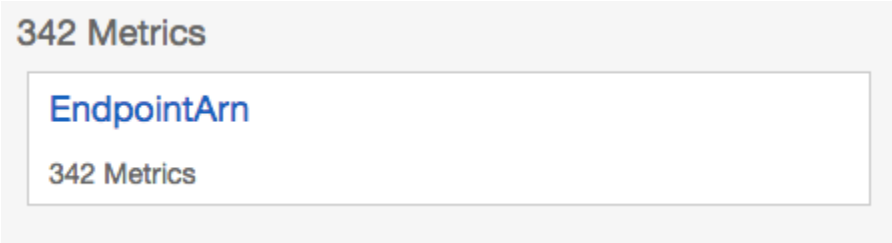
您可以透過 Amazon CloudWatch 主控台監控端點的使用情況，決定如何最佳調整端點的輸送量。

監控您的端點使用情況 CloudWatch

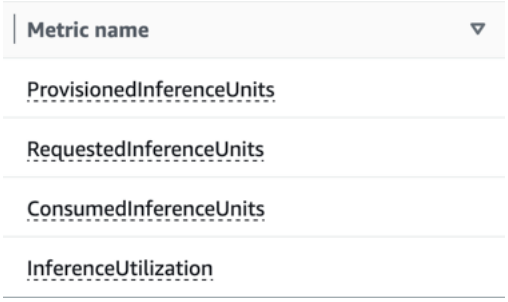
1. 登入 AWS Management Console 並開啟CloudWatch 主控台。
2. 在左側選擇「量度」，然後選取「所有量度」。
3. 在 [所有量度] 下，選擇 [Comprehend]。



4. 主 CloudWatch 控台會顯示 Comprehend 量度的維度。選擇EndpointArn尺寸。



主控台會ProvisionedInferenceUnitsInferenceUtilization針對每個端點顯示ConsumedInferenceUnits、、和。RequestedInferenceUnits

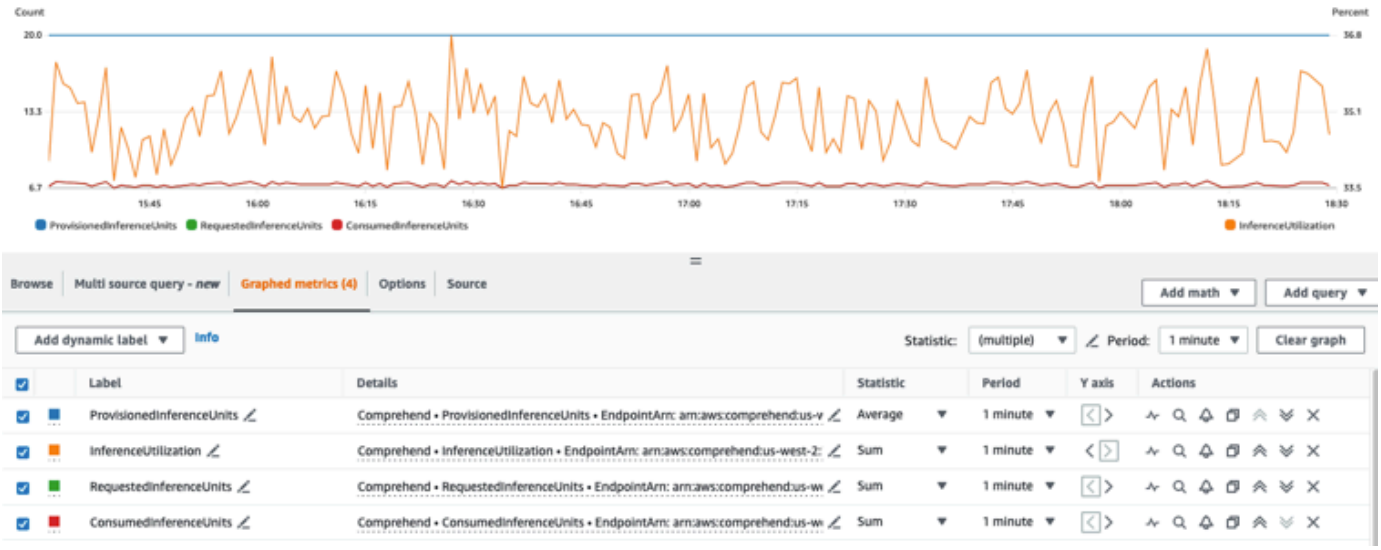


選取四個量度，然後瀏覽至「圖形化量度」標籤。

5. 將「統計資料」欄設定為「總RequestedInferenceUnits和 ConsumedInferenceUnits」。

6. 將的統計資料欄設定為 InferenceUtilization「總和」。
7. ProvisionedInferenceUnits將「統計值」欄設定為「平均」。
8. 將所有測量結果的「期間」欄變更為 1 分鐘。
9. 選取InferenceUtilization並選取箭頭，將其移至單獨的 Y 軸。

   您的圖表已準備好進行分析。



根據 CloudWatch 指標，您還可以設置自 auto 擴展以自動調整端點的輸送量。如需將 auto 調整與端點搭配使用的詳細資訊，請參閱使用端點自動調整。

- ProvisionedInferenceUnits-此量度代表提出要求時的平均佈建 IU 數目。
- RequestedInferenceUnits-這是基於提交給已發送要處理的服務的每個請求的使用情況。這對於將發送要處理的請求與實際處理的請求進行比較而不會調節（ ）ConsumedInferenceUnits會很有幫助。此量度的值是透過取得要處理的傳送字元數，除以 1 IU 可在一分鐘內處理的字元數目來計算。
- ConsumedInferenceUnits-這是根據提交至已成功處理 (未限制) 之服務的每個要求使用量而定。當您將使用的項目與佈建的 IU 進行比較時，這會很有幫助。此量度的值是以處理的字元數除以 1 IU 可在一分鐘內處理的字元數來計算。
- InferenceUtilization-這是每個請求發出。此值的計算方法是取得中定義的已耗用 IU，ConsumedInferenceUnits並將其除以，ProvisionedInferenceUnits然後轉換為 100 以外的百分比。

> **Note**
>
> 只有成功的要求才會發出所有量度。如果量度來自限制的要求或失敗，並出現內部伺服器錯誤或客戶錯誤，則不會顯示該量度。

更新 Amazon Comprehend 端點

建立端點後，您需要的輸送量等級通常會變更，或是您第一次估計需求時會發生變更。發生這種情況時，可能需要更新您的端點以調高或降低輸送量。輸送量是由您佈建端點的推論單元數量所控制。每個推論單元代表每秒 100 個字元的輸送量，每秒最多 2 個文件。您可能還想要更新與端點關聯的模型版本。編輯端點時，您可以為端點選擇不同的模型版本。

將標籤添加到端點以幫助保持組織也很有幫助。這也可以在更新端點時完成。如需端點的詳細資訊，請參閱 標記您的 資源

更新端點（控制台）

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇「端點」。
3. 從「分類器」清單中，選擇您要從中更新端點的自訂模型名稱，然後遵循連結。模型詳細資訊頁面隨即顯示。
4. 在模型詳細資訊頁面中，選取版本詳細資訊。端點清單隨即顯示。
5. 選取端點的端點核取方塊。選取端點表格右上角的「動作」圖示。
6. 選擇編輯。您可以更新已佈建的 IU 和編輯標籤。
7. 儲存您的變更。
8. 若要編輯用於佈建端點的推論單元數目，請選擇編輯。
9. 輸入要指派給端點的推論單元的更新數目。每個單位代表每秒 100 個字元的輸送量。每個端點最多可以指派 10 個推論單元。

   > **Note**
   >
   > 使用端點的成本取決於操作時間和輸送量（基於推論單元的數量）。因此，增加推論單元的數量會增加操作成本。如需詳細資訊，請參 Amazon Comprehend 定價。

10. 選擇編輯端點。接著顯示端點詳細資訊頁面。

11. 從頁面頂端的階層連結中選擇型號名稱，確認端點正在更新。在自訂模型詳細資料頁面上，導覽至「端點」清單，並確認其在端點旁邊顯示「正在更新」。更新完成後，它將顯示「就緒」。

    下列範例示範如何將作UpdateEndpoint業與 AWS CLI 搭配使用。

    此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

    ```
    aws comprehend update-endpoint \
        --desired-inference-units updated number of inference units \
        --desired-model-arn arn:aws:comprehend:region:account-id:model type/model name
     \
        --desired-data-access-role-arn arn:aws:iam:account id:role/role name
        --endpoint-arn arn:aws:comprehend:region:account id:endpoint/endpoint name
    ```

    如果動作成功，Amazon Comprehend HTTP 200 回應會使用空的 HTTP 主體回應。
12. 要編輯連接到端點的自定義模型，請從自定義模型詳細信息頁面導航到端點列表。
13. 選取要變更的端點，然後選取「編輯」。
14. 在端點設定頁面的 [選取分類器模型] 或 [根據您的端點選取辨識器型號] 下，您可以在下拉式清單中搜尋模型。選擇您想要的模型。
15. 在「選取版本」 下，您可以搜尋所需的模型版本。選取版本。
16. 選取編輯要儲存的端點。

Trusted Advisor 與 Amazon Comprehend 使用

AWS Trusted Advisor 是一種線上工具，可提供建議，協助您依照 AWS 最佳做法佈建資源。

如果您有基本或開發人員 Support 方案，則可以使用 Trusted Advisor 主控台存取「服務限制」類別中的所有檢查，並在「安全性」類別中進行六次檢查。如果您有商業或企業 Support 方案，則可以使用 Trusted Advisor 主控台和 AWS Support API 存取所有 Trusted Advisor 檢查。

Amazon Comprehend 支援下列 Trusted Advisor 檢查，藉由提供可行的建議，協助客戶最佳化其 Amazon Comprehend 端點的成本和安全性。

Amazon Comprehend 未充分利用的端點

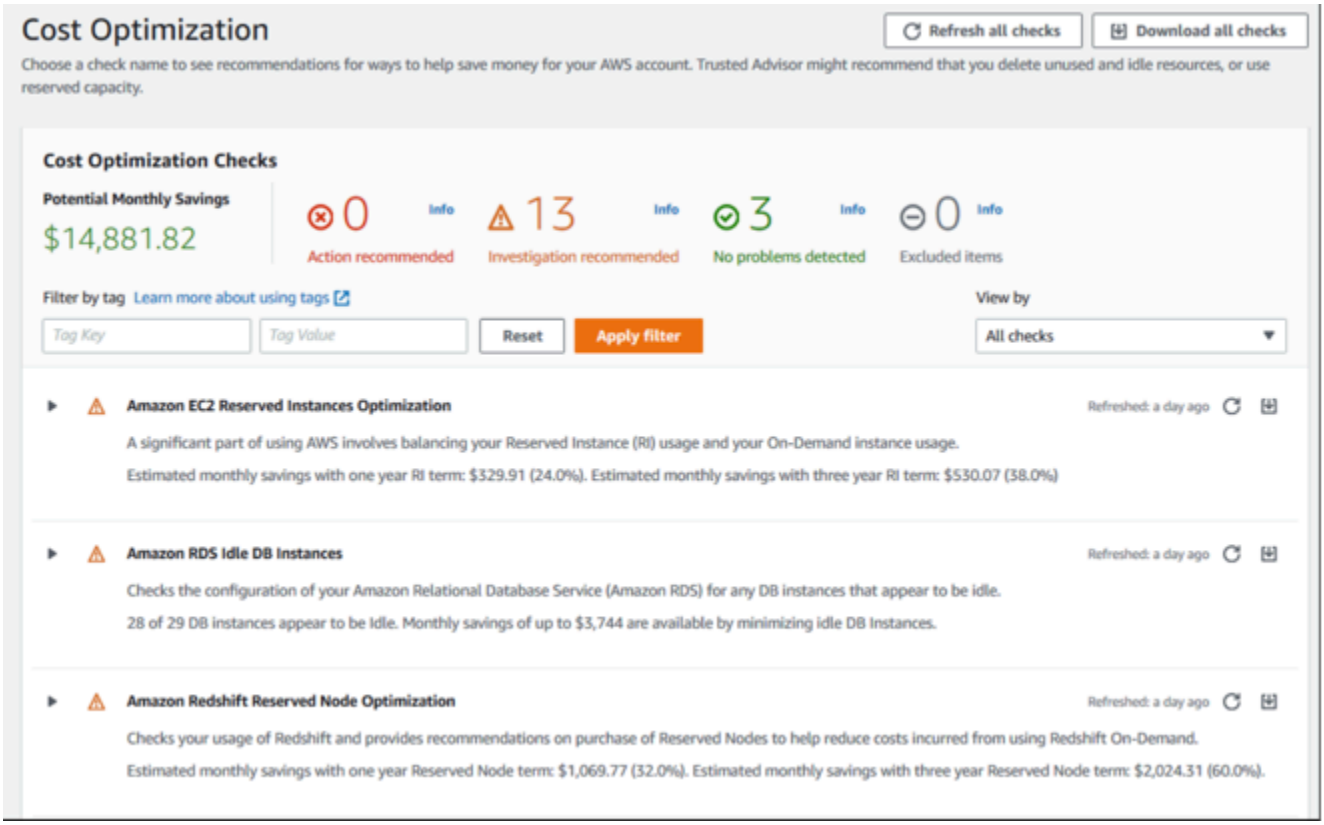
Amazon Comprehend 未充分利用的端點檢查會評估端點的輸送量組態。此檢查會在端點未主動用於即時推論請求時提醒您。未使用超過 15 天的端點會被視為未充分利用。所有端點都會根據設定的輸送

量和端點處於作用中的時間長度來計算費用。對於過去 15 天未使用的端點，建議您使用應用程式自動調度資源為資源定義資源調度資源調度政策。對於過去 30 天未使用過且確實定義了 auto 擴展政策的端點，我們建議您使用非同步推論或將其刪除。這些檢查結果會每天自動重新整理一次，並且可以在 Trusted Advisor 主控台的CostOptimization類別下檢視。

若要檢視所有端點的使用狀態和對應的建議

1. 登入 AWS Management Console 並開啟 Trusted Advisor 主控台。
2. 在導覽窗格中，選擇檢CostOptimization查類別。
3. 在類別頁面上，您可以檢視每個檢查類別的摘要：
   - 建議採取的動作 (紅色) — Trusted Advisor 建議檢查的動作。
   - Investigation recommended (建議進行調查) (黃色) - Trusted Advisor 偵測到可能的檢查問題。
   - 未偵測到問題 (綠色) — Trusted Advisor 未偵測到支票的問題。
   - 已排除項目 (灰色) — 已排除項目的檢查數目，例如您要檢查忽略的資源。
4. 選擇 Amazon Comprehend 未充分利用的端點檢查，以檢視檢查說明和下列詳細資訊：
   - Alert Criteria (提醒條件) - 說明檢查狀態變更的臨界值。
   - Recommended Action (建議動作) - 說明此檢查的建議動作。
   - 資源表：根據您的建議列出端點詳細資訊和每個端點的狀態的表格。
5. 在「資源」表格中，如果端點因為「過去 30 天未使用」警告而被標記為「建議調查」，您可以導覽至 Amazon Comprehend 主控台上的「端點詳細資訊」頁面。
   - 如果您不想再使用此端點，請選擇「刪除」。
   - 選擇 Delete (刪除) 以確認刪除。此時會顯示自訂模型詳細資訊頁面。確認您刪除的端點旁邊顯示已刪除。刪除後，端點會從「端點」清單中移除。
6. 在 Trusted Advisor 主控台的「資源」表格中，如果端點因為過去 15 天未使用過，而且已停用該端點，而且已 AutoScaling 停用，則您可以瀏覽至 Amazon Comprehend 主控台上的「端點詳細資訊」頁面以調整端點。
   - 如果要降低為此端點設定的輸送量，請按一下編輯。輸入要指派給端點的更新推論單元數目，然後選取核取方塊以確認，然後選擇「編輯端點」。更新完成時，狀態會顯示為「就緒」。
   - 如果您想要在端點上自動設定端點佈建，而不是手動調整輸送量組態，建議您使用「應用程式自動調度資源」。
7. 在 Trusted Advisor 主控台的「資源」(Resource) 表格中，如果端點因為「已使用主動」原因而被標記為「未偵測到問題」狀態，則表示端點正在主動使用執行即時推論要求，且不建議採取任何動作。

以下是在 Trusted Advisor 控制台上顯示 CostOptimization 類別視圖的示例：



Amazon Comprehend 端點訪問風險

Amazon Comprehend 端點存取風險檢查會針對使用客戶受管金鑰加密基礎模型的端點評估 AWS Key Management Service (AWS KMS) 金鑰許可。如果客戶受管金鑰已停用或金鑰政策變更為變更 Amazon Comprehend 允許的許可，端點可用性可能會受到影響。如果金鑰已停用，建議您將其啟用。如果金鑰原則已變更，而您希望繼續使用此端點，建議您更新金鑰原則。檢查結果會在一天中自動重新整理多次。您可以在 Trusted Advisor 主控台的「容錯」類別下檢視此檢查。

若要檢視您的亞馬遜端點的 AWS KMS 金鑰狀態

1. 登入 AWS Management Console 並開啟 Trusted Advisor 主控台。
2. 在導覽窗格中，選擇檢FaultTolerance查類別。
3. 在類別頁面上，您可以檢視每個檢查類別的摘要：

   - 建議採取的動作 (紅色) — Trusted Advisor 建議檢查的動作。
   - 建議調查 (黃色) — Trusted Advisor 偵測檢查可能的問題。
   - 未偵測到問題 (綠色) — Trusted Advisor 未偵測到支票的問題。
   - 排除的項目 (灰色) - 具有已排除項目的檢查數量，例如您想要檢查忽略的資源。

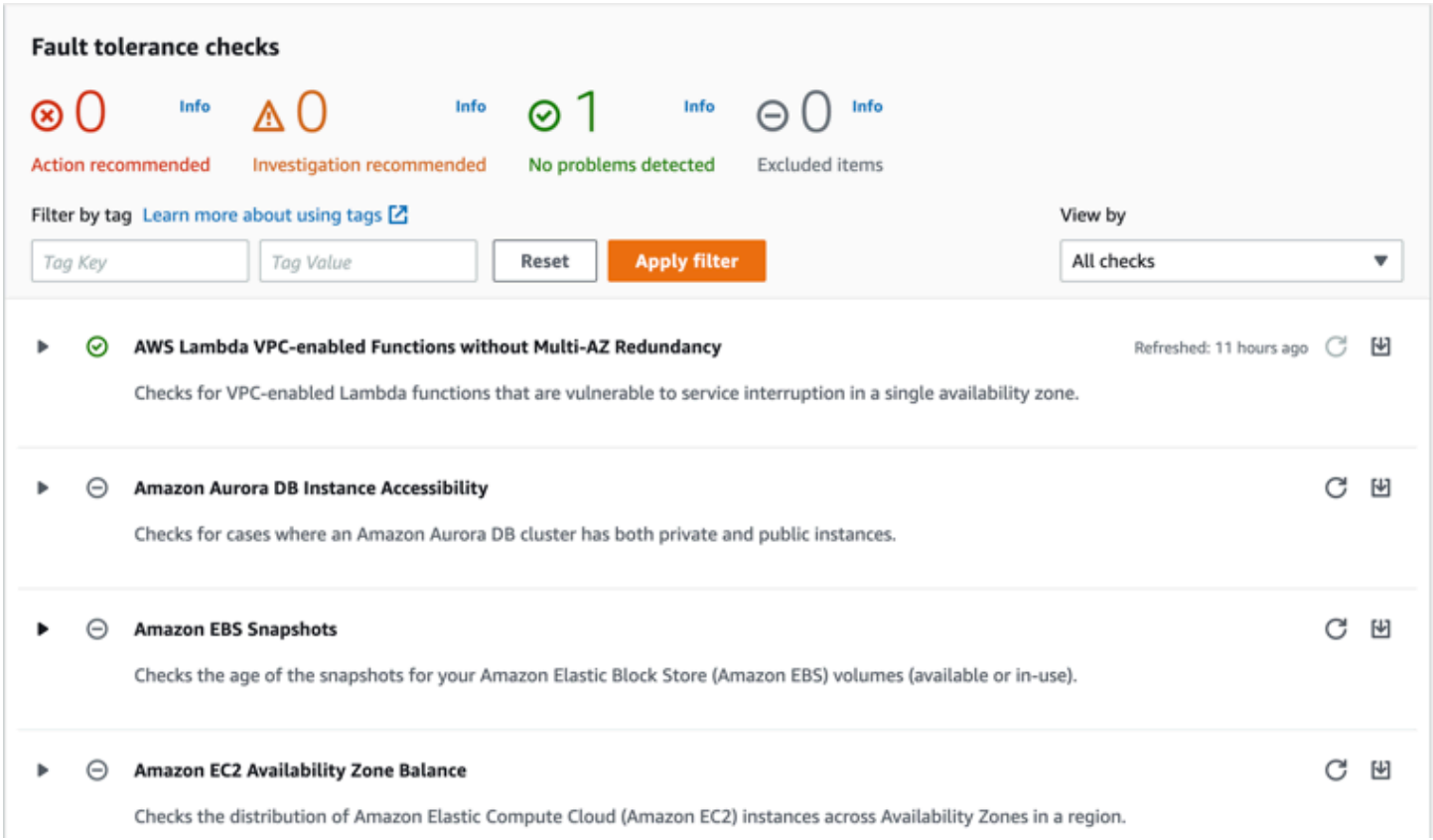
4. 選擇 Amazon Comprehend 端點存取風險檢查，您可以檢視檢查說明和下列詳細資訊：

   - 警示條件 — 說明檢查將變更狀態的臨界值。
   - Recommended Action (建議動作) - 說明此檢查的建議動作。
   - 資源表：根據是否有建議的動作，列出 KMS 加密端點詳細資訊和每個端點的狀態的表格。

5. 在「資源」表格中，如果端點已標示為「建議動作」狀態，請選取 KMS KeyId 欄中的連結，然後您將被重新導向至對應的 AWS KMS 索引鍵頁面。

   - 若要啟用已停用的 AWS KMS 按鍵，請選擇「按鍵動作」，然後選取「啟用」
   - 如果金鑰狀態列為「已啟用」，請在「金鑰原則」段落中選擇切換至原則檢視來更新金鑰制定原則。編輯金鑰政策文件以提供必要的權限給 Amazon Comprehend，然後選擇 [儲存變更]。

以下是 Trusted Advisor 控制台上 FaultTolerance 類別視圖的示例：



這些檢查及其結果也可以參考 AWS Support API 的 Trusted Advisor 章節來檢視。

若要深入瞭解如何使用設定鬧鐘 CloudWatch，請參閱 Trusted Advisor ：使用 CloudWatch. 如需完整的 Trusted Advisor 最佳作法檢查集，請參閱：AWS Trusted Advisor 最佳作法檢查清單。

刪除 Amazon Comprehend 端點

一旦您不再需要端點，您應該將其刪除，以免產生其成本。您可以在需要時從「端點」區段輕鬆建立另一個端點。

刪除端點 (主控台)

1. 登入AWS Management Console並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側功能表中選擇「端點」。
3. 從「端點」表格中找到您要刪除的端點。您可以搜尋或篩選所有端點，以尋找所需的端點。
4. 選取要刪除之端點的端點核取方塊。選取端點表格右上角的「動作」圖示。
5. 選擇刪除。
6. 選擇 Delete (刪除) 以確認刪除。接著顯示「端點」頁面。確認您刪除的端點旁邊顯示 [刪除]。刪除後，端點會從「端點」清單中移除。

刪除端點 (AWS CLI)

下列範例示範如何將作DeleteEndpoint業與 AWS CLI 搭配使用。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
aws comprehend delete-endpoint \
    --endpoint-arn arn:aws:comprehend:region:account-id endpoint/endpoint name
```

如果動作成功，Amazon Comprehend HTTP 200 回應會使用空的 HTTP 主體回應。

使用端點自動調整

除了手動調整為文件分類端點和實體辨識器端點佈建的推論單元數量，您可以使用 auto Scaling 自動設定端點佈建以符合您的容量需求。

有兩種方法可以使用 auto Scaling 來調整為端點佈建的推論單元數量：

- 目標追蹤：設定 auto 調整規模，根據使用情況調整端點佈建以符合容量需求。

- 排程擴展：設定 auto 調整規模以調整端點佈建，以符合指定排程的容量需求。

您只能使用 AWS Command Line Interface (AWS CLI) 設定 auto 縮放比例。如需有關 auto 縮放的詳細資訊，請參閱什麼是 Application Auto Scaling 放？

目標追蹤

透過目標追蹤，您可以根據使用情況調整端點佈建，以符合您的容量需求。推論單元的數目會自動調整，使用的容量在佈建容量的目標百分比內。您可以使用目標追蹤來處理文件分類端點和實體辨識器端點的暫時使用激增。如需詳細資訊，請參閱 Application Auto Scaling 的目標追蹤擴展政策。

> **Note**
>
> 下列範例會針對 Unix、Linux 和 macOS 進行格式化。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

設定目標追蹤

若要為端點設定目標追蹤，您可以使用 AWS CLI 命令註冊可擴展的目標，然後建立擴展政策。可擴展目標將推論單元定義為用於調整端點佈建的資源，而擴展政策則定義了控制佈建容量 auto 調整的指標。

若要設定目標追蹤

1. 登錄可擴展的目標。下列範例會註冊一個可擴充的目標，以調整端點佈建，其容量下限為 1 個推論單元，最大容量為 2 個推論單元。

   對於文件分類端點，請使用下列 AWS CLI 指令：

```
aws application-autoscaling register-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits \
    --min-capacity 1 \
    --max-capacity 2
```

對於實體識別器端點，請使用以下 AWS CLI 命令：

```
aws application-autoscaling register-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --min-capacity 1 \
    --max-capacity 2
```

2. 若要驗證可擴充目標的註冊，請使用下列 AWS CLI 命令：

```
aws application-autoscaling describe-scalable-targets \
    --service-namespace comprehend \
    --resource-id endpoint ARN
```

3. 為擴展政策建立目標追蹤組態，並將組態儲存在名為的檔案中config.json。以下是自動調整推論單元數目的目標追蹤組態範例，使用的容量永遠是佈建容量的 70%。

```
{
  "TargetValue": 70,
  "PredefinedMetricSpecification":
  {
  "PredefinedMetricType": "ComprehendInferenceUtilization"
  }
}
```

4. 建立擴展政策。下列範例會根據config.json檔案中定義的目標追蹤組態建立資源調整政策。

   對於文件分類端點，請使用下列 AWS CLI 指令：

```
aws application-autoscaling put-scaling-policy \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits \
    --policy-name TestPolicy \
    --policy-type TargetTrackingScaling \
```

```
    --target-tracking-scaling-policy-configuration file://config.json
```

對於實體識別器端點，請使用以下 AWS CLI 命令：

```
aws application-autoscaling put-scaling-policy \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --policy-name TestPolicy \
    --policy-type TargetTrackingScaling \
    --target-tracking-scaling-policy-configuration file://config.json
```

移除目標追蹤

若要移除端點的目標追蹤，您可以使用 AWS CLI 命令刪除擴展政策，然後取消註冊可擴展目標。

若要移除目標追蹤

1. 刪除資源調度政策。下列範例會刪除指定的資源調整政策。

   對於文件分類端點，請使用下列 AWS CLI 指令：

   ```
   aws application-autoscaling delete-scaling-policy \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:document-classifier-
   endpoint/name \
       --scalable-dimension comprehend:document-classifier-
   endpoint:DesiredInferenceUnits \
       --policy-name TestPolicy \
   ```

   對於實體識別器端點，請使用以下 AWS CLI 命令：

   ```
   aws application-autoscaling delete-scaling-policy \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
   endpoint/name \
       --scalable-dimension comprehend:entity-recognizer-
   endpoint:DesiredInferenceUnits \
   ```

```
    --policy-name TestPolicy
```

2. 取消註冊可擴展的目標。下列範例會取消註冊指定的可縮放目標。

   對於文件分類端點，請使用下列 AWS CLI 指令：

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits
```

   對於實體識別器端點，請使用以下 AWS CLI 命令：

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits
```

排程擴展

透過排程擴展，您可以根據指定的排程調整端點佈建，以符合您的容量需求。排程縮放會自動調整推論單元的數量，以適應特定時間的使用突波。您可以針對文件分類端點和實體辨識器端點使用排程縮放。如需排程調整的其他資訊，請參閱應用程式自動調整規模的排程調整。

> **Note**
>
> 下列範例會針對 Unix、Linux 和 macOS 進行格式化。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

設定排程擴展

若要為端點設定排程擴展，您可以使用 AWS CLI 命令註冊可擴展的目標，然後建立排程動作。可擴充的目標會將推論單元定義為用於調整端點佈建的資源，排程的動作則控制已佈建容量在特定時間的 auto 動調整規模。

若要設定排定的調整比例

1. 登錄可擴展的目標。下列範例會註冊一個可擴充的目標，以調整端點佈建，其容量下限為 1 個推論單元，最大容量為 2 個推論單元。

   對於文件分類端點，請使用下列 AWS CLI 指令：

   ```
   aws application-autoscaling register-scalable-target \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:document-classifier-
   endpoint/name \
       --scalable-dimension comprehend:document-classifier-
   endpoint:DesiredInferenceUnits \
       --min-capacity 1 \
       --max-capacity 2
   ```

   對於實體識別器端點，請使用以下 AWS CLI 命令：

   ```
   aws application-autoscaling register-scalable-target \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
   endpoint/name \
       --scalable-dimension comprehend:entity-recognizer-
   endpoint:DesiredInferenceUnits \
       --min-capacity 1 \
       --max-capacity 2
   ```

2. 建立排程動作。下列範例會建立排程動作，以在 UTC 每天 12:00 自動調整佈建的容量，最少有 2 個推論單元和最多 5 個推論單元。如需有關按時間順序的運算式和排程縮放的詳細資訊，請參閱排程運算

   對於文件分類端點，請使用下列 AWS CLI 指令：

   ```
   aws application-autoscaling put-scheduled-action \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:document-classifier-
   endpoint/name \
       --scalable-dimension comprehend:document-classifier-
   endpoint:DesiredInferenceUnits \
       --scheduled-action-name TestScheduledAction \
       --schedule "cron(0 12 * * ? *)" \
   ```

```
    --scalable-target-action MinCapacity=2,MaxCapacity=5
```

對於實體識別器端點，請使用以下 AWS CLI 命令：

```
aws application-autoscaling put-scheduled-action \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits \
    --scheduled-action-name TestScheduledAction \
    --schedule "cron(0 12 * * ? *)" \
    --scalable-target-action MinCapacity=2,MaxCapacity=5
```

移除排程的縮放

若要移除端點的排程擴展，您可以使用 AWS CLI 命令刪除排程的動作，然後取消註冊可擴展目標。

移除排定的縮放比例

1. 刪除排程的處理行動。下列範例會刪除指定的排程動作。

   對於文件分類端點，請使用下列 AWS CLI 指令：

   ```
   aws application-autoscaling delete-scheduled-action \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:document-classifier-
   endpoint/name \
       --scalable-dimension comprehend:document-classifier-
   endpoint:DesiredInferenceUnits \
       --scheduled-action-name TestScheduledAction
   ```

   對於實體識別器端點，請使用以下 AWS CLI 命令：

   ```
   aws application-autoscaling delete-scheduled-action \
       --service-namespace comprehend \
       --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
   endpoint/name \
       --scalable-dimension comprehend:entity-recognizer-
   endpoint:DesiredInferenceUnits \
   ```

```
    --scheduled-action-name TestScheduledAction
```

2. 取消註冊可擴展的目標。下列範例會取消註冊指定的可縮放目標。

   對於文件分類端點，請使用下列 AWS CLI 指令：

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
    --scalable-dimension comprehend:document-classifier-
endpoint:DesiredInferenceUnits
```

   對於實體識別器端點，請使用以下 AWS CLI 命令：

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace comprehend \
    --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-
endpoint/name \
    --scalable-dimension comprehend:entity-recognizer-
endpoint:DesiredInferenceUnits
```

標記您的 資源

標籤是鍵值配對，您可以將其作為中繼資料新增至 Amazon Comprehend 資源。您可以在分析工作、自訂分類模型、自訂實體辨識模型和端點上使用標籤。標籤有兩個主要功能：組織資源並提供基於標籤的訪問控制。

若要使用標籤來組織資源，您可以新增標籤金鑰「部門」和標記值「銷售」或「法律」。然後，您可以搜尋和篩選與您公司法律部門相關的資源。

若要提供以標籤為基礎的存取控制，請根據標籤建立具有許可的 IAM 政策。政策可以根據請求中提供的標籤（請求標籤）或與您正在調用的資源（資源標籤）關聯的標籤來允許或禁止操作。如需將標籤與 IAM 搭配使用的詳細資訊，請參閱 IAM 使用者指南中的使用標籤控制存取。

將標籤與 Amazon Comprehend 搭配使用的注意事項：

- 每個資源最多可以新增 50 個標籤，您可以在建立資源時新增標籤，也可以追溯新增標籤。
- 標籤鍵是必填欄位，但標籤值是選擇性的。
- 標籤在資源之間不一定是唯一的，但是給定的資源不能有重複的標籤鍵。
- 標籤鍵與值皆區分大小寫。
- 標籤關鍵字最多可包含 127 個字元；標籤值最多可包含 255 個字元。
- 'aws:' 前置詞會保留供AWS使用；您無法新增、編輯或刪除金鑰開頭為的標籤aws:。這些標籤不會計入您的 50 個 tags-per-resource 上限。

> **Note**
>
> 如果您打算在多個AWS服務和資源中使用標記結構描述，請記住，其他服務對於允許的字元可能有不同的需求。

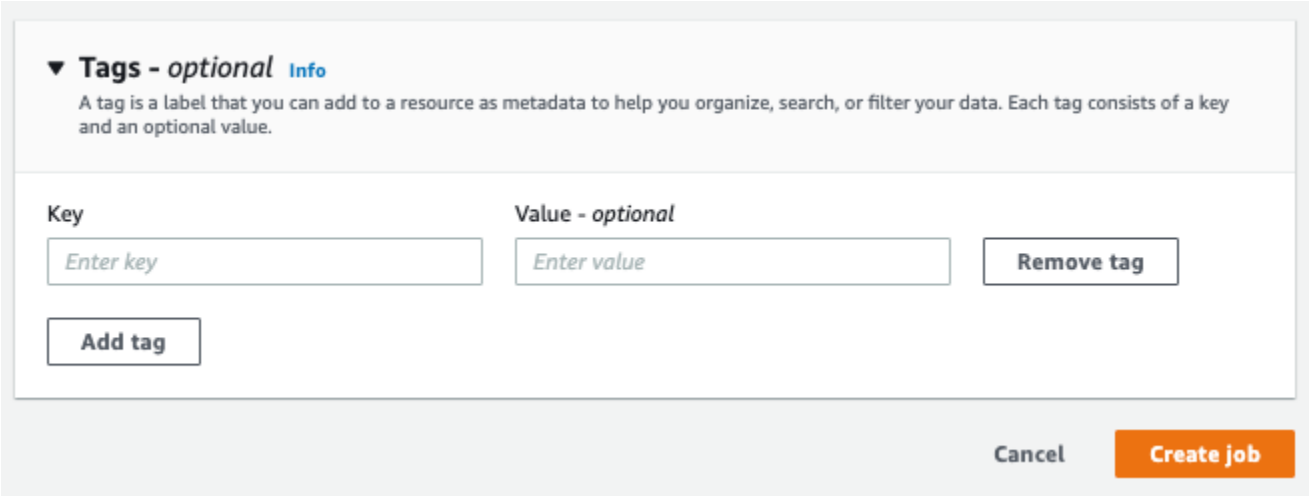
主題

- 標記新資源
- 檢視、編輯和刪除與資源相關聯的標籤

標記新資源

您可以將標籤新增至分析工作、自訂分類模型、自訂實體辨識模型或端點。

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/
2. 從左側導覽窗格中選取要建立的資源 (分析工作、自訂分類或自訂實體辨識)。
3. 按一下 [建立工作] (或 [建立新模型])。這將帶您進入資源的主要「創建」頁面。在此頁面的底部，您將看到一個「標籤- 可選」面板。



輸入標籤索引鍵和選用的標籤值。選擇 [新增標籤]，將其他標籤新增至資源。重複此程序，直到新增所有標籤為止。請注意，每個資源的標籤鍵必須是唯一的。

4. 選取 [建立工作] 或 [建立工作] 按鈕以繼續建立資源。

您也可以使用 AWS CLI 新增標籤。此範例顯示如何使用start-entities-detection-job指令新增標籤。

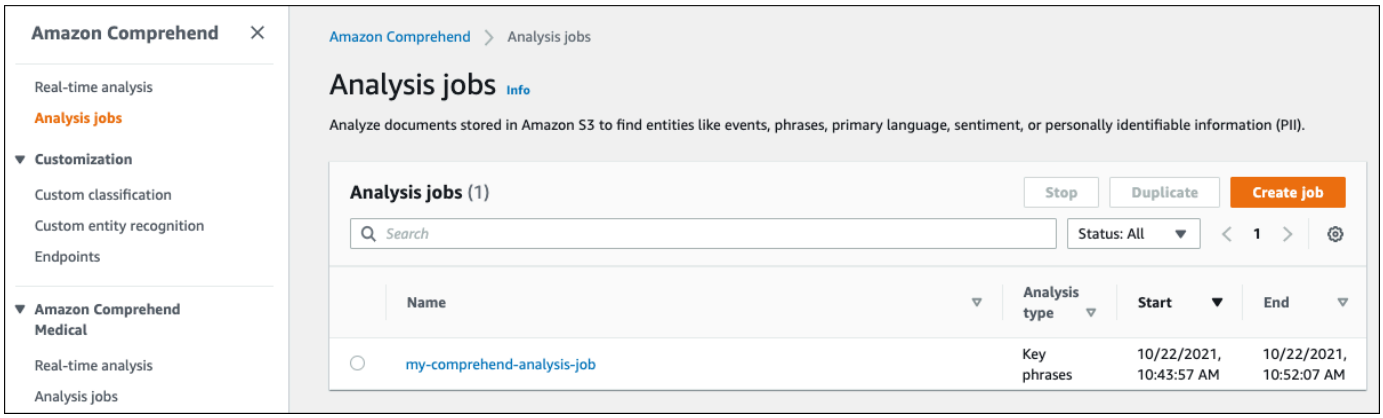
```
aws comprehend start-entities-detection-job \
--language-code "en" \
--input-data-config "{\"S3Uri\": \"s3://test-input/TEST.csv\"}" \
--output-data-config "{\"S3Uri\": \"s3://test-output\"}" \
--data-access-role-arn arn:aws:iam::123456789012:role/test \
--tags "[{\"Key\": \"color\",\"Value\": \"orange\"}]"
```

檢視、編輯和刪除與資源相關聯的標籤

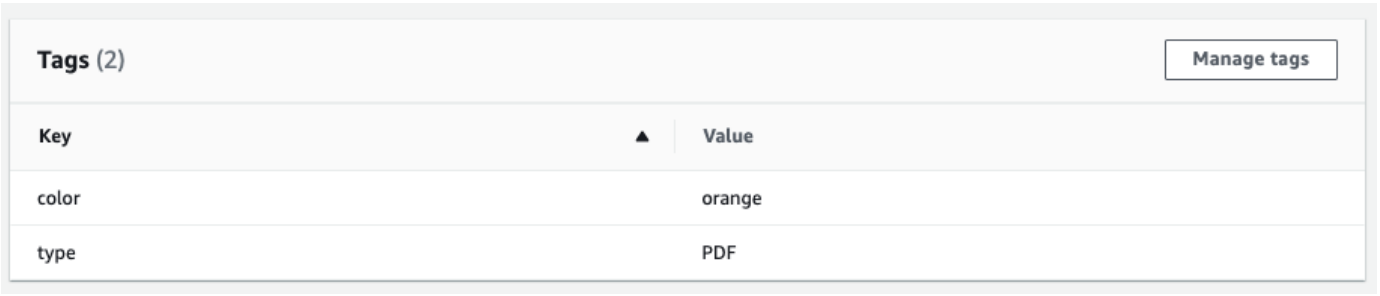
您可以檢視與分析工作、自訂分類模型或自訂實體辨識模型相關聯的標籤。

1. 登入 AWS Management Console 並開啟亞馬遜主控台，網址為 https://console.aws.amazon.com/comprehend/

2. 選取包含含有您要檢視、修改或刪除之標籤的檔案的資源 (分析工作、自訂分類或自訂實體辨識)。這會顯示所選資源的現有檔案清單。

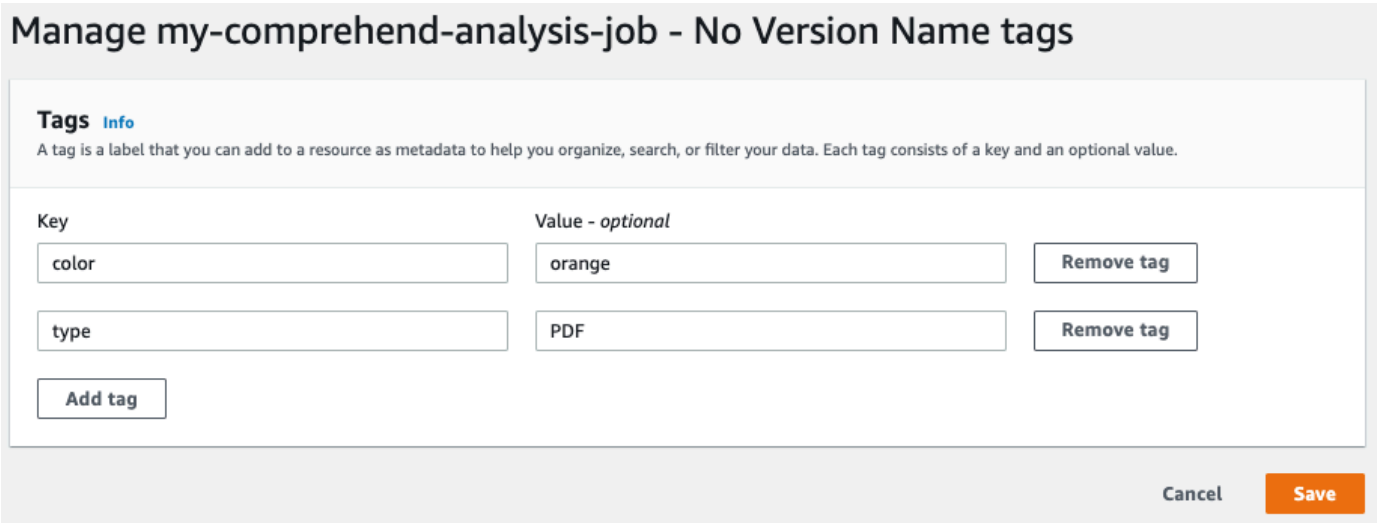


3. 按一下您要檢視、修改或刪除其標籤的檔案 (或模型) 名稱。這會帶您前往該檔案 (或模型) 的詳細資訊頁面。向下捲動，直到看到「標籤」方塊。在這裡，您可以看到與所選文件（或模型）關聯的所有標籤。



選取 [管理標籤] 以編輯或移除資源中的標籤。

4. 按一下您要修改的文字，然後編輯標籤。您也可以選取「移除標籤」來移除標籤。若要新增標籤，請選取「新增標籤」，然後在空白欄位中輸入所需的文字。



修改完標籤後，請選取 [儲存]。

使用 SDK 的 Amazon Comprehend 代碼示例 AWS

下列程式碼範例顯示如何搭配 AWS 軟體開發套件 (SDK) 使用 Amazon Comprehend。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境和跨服務範例中查看內容中的動作。

Scenarios (案例) 是向您展示如何呼叫相同服務中的多個函數來完成特定任務的程式碼範例。

Cross-service examples (跨服務範例) 是跨多個 AWS 服務執行的應用程式範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

程式碼範例

- 亞馬遜使用 SDK 理解的操作 AWS
  - 使用開發套件建立 Amazon Comprehend 文件分類器 AWS
  - 使用 SDK 刪除 Amazon Comprehend 文檔分類器 AWS
  - 使用開發套件描述 Amazon Comprehend 文件分類任務 AWS
  - 使用 SDK 描述 Amazon Comprehend 文檔分類器 AWS
  - 使用開發套件描述 Amazon Comprehend 主題建模任務 AWS
  - 使用開發套件偵測文件中的實體 AWS
  - 使用軟體開發套件偵測文件中的關鍵片語 AWS
  - 使用軟體開發套件偵測文件中的個人識別資訊 AWS
  - 使用開發套件偵測使用 Amazon Comprehend 文件的語法元素 AWS
  - 使用軟體開發套件偵測文件中的主要語言 AWS
  - 使用開發套件使用 Amazon Comprehend 文件偵測情緒 AWS
  - 使用開發套件列出 Amazon Comprehend 文件分類任務 AWS
  - 使用 SDK 列出 Amazon Comprehend 文檔分類器 AWS
  - 使用 SDK 列出 Amazon Comprehend 主題建模任務 AWS
  - 使用開發套件啟動 Amazon Comprehend 文件分類任務 AWS
  - 使用開發套件啟動 Amazon Comprehend 主題建模任務 AWS
- 使用 SDK 的 Amazon Comprehend 案例 AWS
  - 使用 Amazon Comprehend 和開發套件偵測文件元素 AWS

  - 使用開發套件在範例資料上執行 Amazon Comprehend 主題建模任務 AWS
  - 訓練自訂的亞馬遜分類器，並使用開發套件對文件進行分類 AWS
- 使用開發套件的 Amazon Comprehend 跨服務範例 AWS
  - 建置 Amazon Transcribe 串流應用程式
  - 建立 Amazon Lex 聊天機器人來吸引您的網站訪客
  - 使用 Amazon SQS 建立可傳送和擷取訊息的 Web 應用程式
  - 建立可分析客戶意見回饋並合成音訊的應用程式
  - 使用 AWS SDK 檢測從圖像中提取的文本中的實體

亞馬遜使用 SDK 理解的操作 AWS

下列程式碼範例示範如何使 AWS 用軟體開發套件執行個別的 Amazon Comprehend 動作。這些摘錄會呼叫 Amazon Comprehend API，是來自必須在內容中執行的大型程式碼摘錄。每個範例都包含一個連結 GitHub，您可以在其中找到設定和執行程式碼的指示。

下列範例僅包含最常使用的動作。如需完整清單，請參閱 Amazon Comprehend API 參考。

範例

- 使用開發套件建立 Amazon Comprehend 文件分類器 AWS
- 使用 SDK 刪除 Amazon Comprehend 文檔分類器 AWS
- 使用開發套件描述 Amazon Comprehend 文件分類任務 AWS
- 使用 SDK 描述 Amazon Comprehend 文檔分類器 AWS
- 使用開發套件描述 Amazon Comprehend 主題建模任務 AWS
- 使用開發套件偵測文件中的實體 AWS
- 使用軟體開發套件偵測文件中的關鍵片語 AWS
- 使用軟體開發套件偵測文件中的個人識別資訊 AWS
- 使用開發套件偵測使用 Amazon Comprehend 文件的語法元素 AWS
- 使用軟體開發套件偵測文件中的主要語言 AWS
- 使用開發套件使用 Amazon Comprehend 文件偵測情緒 AWS
- 使用開發套件列出 Amazon Comprehend 文件分類任務 AWS
- 使用 SDK 列出 Amazon Comprehend 文檔分類器 AWS
- 使用 SDK 列出 Amazon Comprehend 主題建模任務 AWS

- 使用開發套件啟動 Amazon Comprehend 文件分類任務 AWS
- 使用開發套件啟動 Amazon Comprehend 主題建模任務 AWS

使用開發套件建立 Amazon Comprehend 文件分類器 AWS

下列程式碼範例示範如何建立 Amazon Comprehend 文件分類器。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 訓練自訂分類器並對文件進行分類

CLI

AWS CLI

建立文件分類器以對文件進行分類的步驟

下列create-document-classifier範例會開始文件分類器模型的訓練程序。訓練資料檔案位於--input-data-config標籤上。training.csv training.csv是兩欄文件，其中標籤或分類會在第一欄中提供，而文件則在第二欄中提供。

```
aws comprehend create-document-classifier \
    --document-classifier-name example-classifier \
    --data-access-arn arn:aws:comprehend:us-west-2:111122223333:pii-entities-
detection-job/123456abcdeb0e11022f22a11EXAMPLE \
    --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \
    --language-code en
```

輸出：

```
{
    "DocumentClassifierArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/example-classifier"
}
```

如需詳細資訊，請參閱 Amazon 開發人員指南中的自訂分類。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateDocumentClassifier中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
 software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-
using-amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
                  dataAccessRoleArn - The ARN value of the role used for this
 operation.
```

```
                  s3Uri - The Amazon S3 bucket that contains the CSV file.
                  documentClassifierName - The name of the document classifier.
                """";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dataAccessRoleArn = args[0];
        String s3Uri = args[1];
        String documentClassifierName = args[2];

        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
        comClient.close();
    }

    public static void createDocumentClassifier(ComprehendClient comClient,
String dataAccessRoleArn, String s3Uri,
            String documentClassifierName) {
        try {
            DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
                    .s3Uri(s3Uri)
                    .build();

            CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
                    .documentClassifierName(documentClassifierName)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .languageCode("en")
                    .inputDataConfig(config)
                    .build();

            CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
                    .createDocumentClassifier(createDocumentClassifierRequest);
```

```
            String documentClassifierArn =
 createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " +
 documentClassifierArn);

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[CreateDocumentClassifier](#)中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
```

```
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
immediately
        starts training on the data found in the specified Amazon S3 bucket.
Training
        can take 30 minutes or longer. The `describe_document_classifier`
function
        can be used to get training status and returns a status of TRAINED when
the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
data.
        :param training_key: The prefix used to find training data in the
training
                             bucket. If multiple objects have the same prefix,
all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
```

```
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreateDocumentClassifier中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 刪除 Amazon Comprehend 文檔分類器 AWS

下列程式碼範例示範如何刪除 Amazon Comprehend 文件分類器。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 訓練自訂分類器並對文件進行分類

CLI

AWS CLI

若要刪除自訂文件分類器

下列delete-document-classifier範例會刪除自訂文件分類器模型。

```
aws comprehend delete-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-
classifier/example-classifer-1
```

此命令不會產生輸出。

如需詳細資訊，請參閱亞馬遜開發人員指南中的管 Amazon Comprehend 端點。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteDocumentClassifier中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
 self.classifier_arn)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteDocumentClassifier中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件描述 Amazon Comprehend 文件分類任務 AWS

下列程式碼範例顯示如何描述 Amazon Comprehend 文件分類工作。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 訓練自訂分類器並對文件進行分類

CLI

AWS CLI

描述文件分類工作的步驟

下列`describe-document-classification-job`範例會取得非同步文件分類工作的屬性。

```
aws comprehend describe-document-classification-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

輸出：

```
{
    "DocumentClassificationJobProperties": {
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "exampleclassificationjob",
        "JobStatus": "COMPLETED",
        "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
        "EndTime": "2023-06-14T17:15:58.582000+00:00",
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/mymodel/version/1",
        "InputDataConfig": {
            "S3Uri": "s3://DOC-EXAMPLE-BUCKET/jobdata/",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
```

```
            "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-CLN-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-servicerole"
    }
}
```

如需詳細資訊，請參閱 Amazon 開發人員指南中的自訂分類。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DescribeDocumentClassificationJob中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
```

```
            response =
self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DescribeDocumentClassificationJob中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 描述 Amazon Comprehend 文檔分類器 AWS

下面的代碼示例演示了如何描述 Amazon Comprehend 文檔分類器。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 訓練自訂分類器並對文件進行分類

CLI

AWS CLI

描述文件分類器

下列describe-document-classifier範例會取得自訂文件分類器模型的屬性。

```
aws comprehend describe-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-
classifier/example-classifier-1
```

輸出：

```
{
    "DocumentClassifierProperties": {
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/example-classifier-1",
        "LanguageCode": "en",
        "Status": "TRAINED",
        "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
        "EndTime": "2023-06-13T19:42:31.752000+00:00",
        "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
        "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
        "InputDataConfig": {
            "DataFormat": "COMPREHEND_CSV",
            "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata"
        },
        "OutputDataConfig": {},
        "ClassifierMetadata": {
            "NumberOfLabels": 3,
            "NumberOfTrainedDocuments": 5016,
            "NumberOfTestDocuments": 557,
            "EvaluationMetrics": {
                "Accuracy": 0.9856,
                "Precision": 0.9919,
                "Recall": 0.9459,
                "F1Score": 0.9673,
                "MicroPrecision": 0.9856,
                "MicroRecall": 0.9856,
                "MicroF1Score": 0.9856,
                "HammingLoss": 0.0144
            }
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role",
        "Mode": "MULTI_CLASS"
    }
}
```

如需詳細資訊，請參閱 Amazon Comprehend 開發人員指南中的建立和管理自訂模型。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DescribeDocumentClassifier中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DescribeDocumentClassifier中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件描述 Amazon Comprehend 主題建模任務 AWS

下列程式碼範例說明如何描述 Amazon Comprehend 主題建模工作。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 針對範例資料執行主題建模工作

CLI

AWS CLI

描述主題偵測工作

下列describe-topics-detection-job範例會取得非同步主題偵測工作的屬性。

```
aws comprehend describe-topics-detection-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

輸出：

```
{
    "TopicsDetectionJobProperties": {
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "example_topics_detection",
        "JobStatus": "IN_PROGRESS",
        "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
        "InputDataConfig": {
            "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
```

```
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
testfolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "NumberOfTopics": 10,
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-examplerole"
    }
}
```

如需詳細資訊，請參閱亞馬遜開發人員指南中的亞馬遜開發人員深入解析的非同步分析。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DescribeTopicsDetectionJob中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
```

```
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DescribeTopicsDetectionJob中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件偵測文件中的實體 AWS

下列程式碼範例示範如何使用 Amazon Comprehend 偵測文件中的實體。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 偵測文件元素

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the AmazonComprehend service detect any
    /// entities in submitted text.
    /// </summary>
    public static class DetectEntities
    {
        /// <summary>
        /// The main method calls the DetectEntitiesAsync method to find any
        /// entities in the sample code.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new AmazonComprehendClient();

            Console.WriteLine("Calling DetectEntities\n");
            var detectEntitiesRequest = new DetectEntitiesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

            foreach (var e in detectEntitiesResponse.Entities)
            {
                Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score:
{e.Score}, BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DetectEntities中的。

CLI

AWS CLI

偵測輸入文字中具名實體的步驟

下列detect-entities範例會分析輸入文字，並傳回具名實體。預先訓練的模型的置信度分數也會為每個預測輸出。

```
aws comprehend detect-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
    account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by
 July 31st. Based on your autopay settings, \
    we will withdraw your payment on the due date from your bank account number
 XXXXXX1111 with the routing number XXXXX0000. \
    Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to
 Alice at AnySpa@example.com."
```

輸出：

```
{
    "Entities": [
        {
            "Score": 0.9994556307792664,
            "Type": "PERSON",
            "Text": "Zhang Wei",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9981022477149963,
            "Type": "PERSON",
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9986887574195862,
            "Type": "ORGANIZATION",
            "Text": "AnyCompany Financial Services, LLC",
            "BeginOffset": 33,
```

```
            "EndOffset": 67
        },
        {
            "Score": 0.9959119558334351,
            "Type": "OTHER",
            "Text": "1111-XXXX-1111-XXXX",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9708039164543152,
            "Type": "QUANTITY",
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9987268447875977,
            "Type": "DATE",
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9858865737915039,
            "Type": "OTHER",
            "Text": "XXXXXX1111",
            "BeginOffset": 271,
            "EndOffset": 281
        },
        {
            "Score": 0.9700471758842468,
            "Type": "OTHER",
            "Text": "XXXXX0000",
            "BeginOffset": 306,
            "EndOffset": 315
        },
        {
            "Score": 0.9591118693351746,
            "Type": "ORGANIZATION",
            "Text": "Sunshine Spa",
            "BeginOffset": 340,
            "EndOffset": 352
        },
```

```
        {
            "Score": 0.9797496795654297,
            "Type": "LOCATION",
            "Text": "123 Main St",
            "BeginOffset": 354,
            "EndOffset": 365
        },
        {
            "Score": 0.994929313659668,
            "Type": "PERSON",
            "Text": "Alice",
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9949769377708435,
            "Type": "OTHER",
            "Text": "AnySpa@example.com",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon 開發人員指南中的實體。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DetectEntities中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
```

```
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text)
 {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
 DetectEntitiesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectEntitiesResponse detectEntitiesResult =
 comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }
```

```
        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DetectEntities中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
 places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
```

```
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DetectEntities中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用軟體開發套件偵測文件中的關鍵片語 AWS

下列程式碼範例顯示如何使用 Amazon Comprehend 偵測文件中的關鍵片語。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 偵測文件元素

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
```

```
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the Amazon Comprehend service to
    /// search text for key phrases.
    /// </summary>
    public static class DetectKeyPhrase
    {
        /// <summary>
        /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync
        /// to detect any key phrases in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

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

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectKeyPhrases");
            var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectKeyPhrasesResponse = await
comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
            foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
            {
                Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score},
BeginOffset: {kp.BeginOffset}, EndOffset: {kp.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DetectKeyPhrases中的。

CLI

AWS CLI

偵測輸入文字中的關鍵片語

下列detect-key-phrases範例會分析輸入文字並識別關鍵名詞片語。預先訓練的模型的置信度分數也會為每個預測輸出。

```
aws comprehend detect-key-phrases \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

輸出：

```
{
    "KeyPhrases": [
        {
            "Score": 0.8996376395225525,
            "Text": "Zhang Wei",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9992469549179077,
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.988385021686554,
            "Text": "Your AnyCompany Financial Services",
            "BeginOffset": 28,
            "EndOffset": 62
        },
        {
```

```
            "Score": 0.8740853071212769,
            "Text": "LLC credit card account 1111-XXXX-1111-XXXX",
            "BeginOffset": 64,
            "EndOffset": 107
        },
        {
            "Score": 0.9999437928199768,
            "Text": "a minimum payment",
            "BeginOffset": 112,
            "EndOffset": 129
        },
        {
            "Score": 0.9998900890350342,
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9979453086853027,
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9983011484146118,
            "Text": "your autopay settings",
            "BeginOffset": 172,
            "EndOffset": 193
        },
        {
            "Score": 0.9996572136878967,
            "Text": "your payment",
            "BeginOffset": 211,
            "EndOffset": 223
        },
        {
            "Score": 0.9995037317276001,
            "Text": "the due date",
            "BeginOffset": 227,
            "EndOffset": 239
        },
        {
            "Score": 0.9702621698379517,
            "Text": "your bank account number XXXXXX1111",
```

```
            "BeginOffset": 245,
            "EndOffset": 280
        },
        {
            "Score": 0.9179925918579102,
            "Text": "the routing number XXXXX0000.Customer feedback",
            "BeginOffset": 286,
            "EndOffset": 332
        },
        {
            "Score": 0.9978160858154297,
            "Text": "Sunshine Spa",
            "BeginOffset": 337,
            "EndOffset": 349
        },
        {
            "Score": 0.9706913232803345,
            "Text": "123 Main St",
            "BeginOffset": 351,
            "EndOffset": 362
        },
        {
            "Score": 0.9941995143890381,
            "Text": "comments",
            "BeginOffset": 379,
            "EndOffset": 387
        },
        {
            "Score": 0.9759287238121033,
            "Text": "Alice",
            "BeginOffset": 391,
            "EndOffset": 396
        },
        {
            "Score": 0.8376792669296265,
            "Text": "AnySpa@example.com",
            "BeginOffset": 400,
            "EndOffset": 415
        }
    ]
}
```


如需詳細資訊，請參閱 Amazon 開發人員指南中的關鍵片語。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DetectKeyPhrases中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
```

```
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String
 text) {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
 DetectKeyPhrasesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
 comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DetectKeyPhrases中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DetectKeyPhrases中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用軟體開發套件偵測文件中的個人識別資訊 AWS

下列程式碼範例示範如何使用 Amazon Comprehend 在文件中偵測個人識別資訊 (PII)。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 偵測文件元素

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the Amazon Comprehend service to find
/// personally identifiable information (PII) within text submitted to the
/// DetectPiiEntitiesAsync method.
/// </summary>
public class DetectingPII
{
    /// <summary>
    /// This method calls the DetectPiiEntitiesAsync method to locate any
    /// personally dientifiable information within the supplied text.
    /// </summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();
        var text = @"Hello Paul Santos. The latest statement for your
                    credit card account 1111-0000-1111-0000 was
                    mailed to 123 Any Street, Seattle, WA 98109.";

        var request = new DetectPiiEntitiesRequest
        {
            Text = text,
```

```
                LanguageCode = "EN",
            };

            var response = await
 comprehendClient.DetectPiiEntitiesAsync(request);

            if (response.Entities.Count > 0)
            {
                foreach (var entity in response.Entities)
                {
                    var entityValue = text.Substring(entity.BeginOffset,
 entity.EndOffset - entity.BeginOffset);
                    Console.WriteLine($"{entity.Type}: {entityValue}");
                }
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DetectPiiEntities中的。

CLI

AWS CLI

若要偵測輸入文字中的 pii 實體

下列detect-pii-entities範例會分析輸入文字，並識別包含個人識別資訊 (PII) 的實體。預先訓練的模型的置信度分數也會為每個預測輸出。

```
aws comprehend detect-pii-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

輸出：


```
{
    "Entities": [
        {
            "Score": 0.9998322129249573,
            "Type": "NAME",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9998878240585327,
            "Type": "NAME",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9994089603424072,
            "Type": "CREDIT_DEBIT_NUMBER",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9999760985374451,
            "Type": "DATE_TIME",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9999449253082275,
            "Type": "BANK_ACCOUNT_NUMBER",
            "BeginOffset": 271,
            "EndOffset": 281
        },
        {
            "Score": 0.9999847412109375,
            "Type": "BANK_ROUTING",
            "BeginOffset": 306,
            "EndOffset": 315
        },
        {
            "Score": 0.999925434589386,
            "Type": "ADDRESS",
            "BeginOffset": 354,
```

```
            "EndOffset": 365
        },
        {
            "Score": 0.9989161491394043,
            "Type": "NAME",
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9994171857833862,
            "Type": "EMAIL",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon Comprehend 開發人員指南中的個人識別資訊 (PII)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DetectPiiEntities中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_pii(self, text, language_code):
```

```
        """
        Detects personally identifiable information (PII) in a document. PII can
be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DetectPiiEntities中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件偵測使用 Amazon Comprehend 文件的語法元素 AWS

下列程式碼範例示範如何使用 Amazon Comprehend 偵測文件的語法元素。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 偵測文件元素

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use Amazon Comprehend to detect syntax
    /// elements by calling the DetectSyntaxAsync method.
    /// </summary>
    public class DetectingSyntax
    {
        /// <summary>
        /// This method calls DetectSynaxAsync to identify the syntax elements
        /// in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new AmazonComprehendClient();

            // Call DetectSyntax API
            Console.WriteLine("Calling DetectSyntaxAsync\n");
            var detectSyntaxRequest = new DetectSyntaxRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            DetectSyntaxResponse detectSyntaxResponse = await
comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
            foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
            {
```

```
                Console.WriteLine($"Text: {s.Text}, PartOfSpeech:
 {s.PartOfSpeech.Tag}, BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DetectSyntax中的。

CLI

AWS CLI

若要偵測輸入文字中的語音部分

下列detect-syntax範例會分析輸入文字的語法，並傳回語音的不同部分。預先訓練的模型的置信度分數也會為每個預測輸出。

```
aws comprehend detect-syntax \
    --language-code en \
    --text "It is a beautiful day in Seattle."
```

輸出：

```
{
    "SyntaxTokens": [
        {
            "TokenId": 1,
            "Text": "It",
            "BeginOffset": 0,
            "EndOffset": 2,
            "PartOfSpeech": {
                "Tag": "PRON",
                "Score": 0.9999740719795227
            }
        },
        {
            "TokenId": 2,
            "Text": "is",
```

```
            "BeginOffset": 3,
            "EndOffset": 5,
            "PartOfSpeech": {
                "Tag": "VERB",
                "Score": 0.999901294708252
            }
        },
        {
            "TokenId": 3,
            "Text": "a",
            "BeginOffset": 6,
            "EndOffset": 7,
            "PartOfSpeech": {
                "Tag": "DET",
                "Score": 0.9999938607215881
            }
        },
        {
            "TokenId": 4,
            "Text": "beautiful",
            "BeginOffset": 8,
            "EndOffset": 17,
            "PartOfSpeech": {
                "Tag": "ADJ",
                "Score": 0.9987351894378662
            }
        },
        {
            "TokenId": 5,
            "Text": "day",
            "BeginOffset": 18,
            "EndOffset": 21,
            "PartOfSpeech": {
                "Tag": "NOUN",
                "Score": 0.9999796748161316
            }
        },
        {
            "TokenId": 6,
            "Text": "in",
            "BeginOffset": 22,
            "EndOffset": 24,
            "PartOfSpeech": {
                "Tag": "ADP",
```

```
                "Score": 0.9998047947883606
            }
        },
        {
            "TokenId": 7,
            "Text": "Seattle",
            "BeginOffset": 25,
            "EndOffset": 32,
            "PartOfSpeech": {
                "Tag": "PROPN",
                "Score": 0.9940530061721802
            }
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon 開發人員指南中的語法分析。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DetectSyntax中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest =
 DetectSyntaxRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSyntaxResponse detectSyntaxResult =
 comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
 token.partOfSpeech().tagAsString());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DetectSyntax中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
```

```
            raise
        else:
            return tokens
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DetectSyntax中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用軟體開發套件偵測文件中的主要語言 AWS

下列程式碼範例說明如何使用 Amazon Comprehend 偵測文件中的主要語言。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 偵測文件元素

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example calls the Amazon Comprehend service to determine the
/// dominant language.
/// </summary>
```

```
    public static class DetectDominantLanguage
    {
        /// <summary>
        /// Calls Amazon Comprehend to determine the dominant language used in
        /// the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle.";

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

            Console.WriteLine("Calling DetectDominantLanguage\n");
            var detectDominantLanguageRequest = new
DetectDominantLanguageRequest()
            {
                Text = text,
            };

            var detectDominantLanguageResponse = await
comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
            foreach (var dl in detectDominantLanguageResponse.Languages)
            {
                Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
{dl.Score}");
            }

            Console.WriteLine("Done");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DetectDominantLanguage中的。

CLI

AWS CLI

檢測輸入文本的主要語言

以下內容detect-dominant-language分析輸入文字並識別主要語言。也會輸出預先訓練的模型的置信度分數。

```
aws comprehend detect-dominant-language \
    --text "It is a beautiful day in Seattle."
```

輸出：

```
{
    "Languages": [
        {
            "LanguageCode": "en",
            "Score": 0.9877256155014038
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon Comprehend 開發人員指南中的主導語言。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DetectDominantLanguage中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient,
 String text) {
        try {
            DetectDominantLanguageRequest request =
 DetectDominantLanguageRequest.builder()
                    .text(text)
                    .build();

            DetectDominantLanguageResponse resp =
 comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DetectDominantLanguage中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
```

```
            return languages
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DetectDominantLanguage中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件使用 Amazon Comprehend 文件偵測情緒 AWS

下列程式碼範例顯示如何使用 Amazon Comprehend 偵測文件的情緒。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 偵測文件元素

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service.
/// </summary>
public static class DetectSentiment
```

```
    {
        /// <summary>
        /// This method calls the DetetectSentimentAsync method to analyze the
        /// supplied text and determine the overal sentiment.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

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

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectSentiment");
            var detectSentimentRequest = new DetectSentimentRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectSentimentResponse = await
comprehendClient.DetectSentimentAsync(detectSentimentRequest);
            Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
            Console.WriteLine("Done");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DetectSentiment中的。

CLI

AWS CLI

偵測輸入文字的情緒

下列detect-sentiment範例會分析輸入文字，並傳回流行情緒 (POSITIVE、NEUTRALMIXED、或NEGATIVE) 的推論。

```
aws comprehend detect-sentiment \
    --language-code en \
    --text "It is a beautiful day in Seattle"
```

輸出：

```
{
    "Sentiment": "POSITIVE",
    "SentimentScore": {
        "Positive": 0.9976957440376282,
        "Negative": 9.653854067437351e-05,
        "Neutral": 0.002169104292988777,
        "Mixed": 3.857641786453314e-05
    }
}
```

如需詳細資訊，請參閱 Amazon 開發人員指南中的情緒

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DetectSentiment中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text)
 {
        try {
            DetectSentimentRequest detectSentimentRequest =
 DetectSentimentRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
 comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
 detectSentimentResult.sentimentScore().neutral());

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DetectSentiment中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DetectSentiment中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件列出 Amazon Comprehend 文件分類任務 AWS

下列程式碼範例顯示如何列出 Amazon Comprehend 文件分類工作。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 訓練自訂分類器並對文件進行分類

CLI

AWS CLI

若要清單所有文件分類工作

下列`list-document-classification-jobs`範例會列出所有文件分類工作。

```
aws comprehend list-document-classification-jobs
```

輸出：

```
{
    "DocumentClassificationJobPropertiesList": [
        {
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName": "exampleclassificationjob",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
            "EndTime": "2023-06-14T17:15:58.582000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
```

```
                "S3Uri": "s3://DOC-EXAMPLE-BUCKET/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/1234567890101-CLN-e758dd56b824aa717ceab551f11749fb/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "exampleclassificationjob2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:22:39.829000+00:00",
            "EndTime": "2023-06-14T17:28:46.107000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-BUCKET/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/1234567890101-CLN-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon 開發人員指南中的自訂分類。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListDocumentClassificationJobs中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListDocumentClassificationJobs中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 列出 Amazon Comprehend 文檔分類器 AWS

下列程式碼範例示範如何列出 Amazon Comprehend 文件分類器。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 訓練自訂分類器並對文件進行分類

CLI

AWS CLI

所有文件分類器的清單

下列`list-document-classifiers`範例會列出所有已訓練及訓練中的文件分類器模型。

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

輸出：

```
{
    "DocumentClassifierPropertiesList": [
        {
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier1",
            "LanguageCode": "en",
            "Status": "TRAINED",
            "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
            "EndTime": "2023-06-13T19:42:31.752000+00:00",
            "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
            "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
```

```
                "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata"
            },
            "OutputDataConfig": {},
            "ClassifierMetadata": {
                "NumberOfLabels": 3,
                "NumberOfTrainedDocuments": 5016,
                "NumberOfTestDocuments": 557,
                "EvaluationMetrics": {
                    "Accuracy": 0.9856,
                    "Precision": 0.9919,
                    "Recall": 0.9459,
                    "F1Score": 0.9673,
                    "MicroPrecision": 0.9856,
                    "MicroRecall": 0.9856,
                    "MicroF1Score": 0.9856,
                    "HammingLoss": 0.0144
                }
            },
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        },
        {
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier2",
            "LanguageCode": "en",
            "Status": "TRAINING",
            "SubmitTime": "2023-06-13T21:20:28.690000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
                "S3Uri": "s3://DOC-EXAMPLE-BUCKET/trainingdata"
            },
            "OutputDataConfig": {},
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon Comprehend 開發人員指南中的建立和管理自訂模型。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListDocumentClassifiers中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListDocumentClassifiers中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 列出 Amazon Comprehend 主題建模任務 AWS

下列程式碼範例顯示如何列出 Amazon Comprehend 主題建模工作。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 針對範例資料執行主題建模工作

CLI

AWS CLI

列出所有主題偵測工作

下列`list-topics-detection-jobs`範例會列出所有進行中和已完成的非同步主題偵測工作。

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

輸出：

```
{
    "TopicsDetectionJobPropertiesList": [
        {
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName" "topic-analysis-1"
            "JobStatus": "IN_PROGRESS",
            "SubmitTime": "2023-06-09T18:40:35.384000+00:00",
            "EndTime": "2023-06-09T18:46:41.936000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
                "InputFormat": "ONE_DOC_PER_LINE"
```

```
            },
            "OutputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "topic-analysis-2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
            "EndTime": "2023-06-09T18:50:50.872000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE3",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE3",
            "JobName": "topic-analysis-2",
            "JobStatus": "IN_PROGRESS",
            "SubmitTime": "2023-06-09T18:50:56.737000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
```

```
                "S3Uri": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE3/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

如需詳細資訊，請參閱亞馬遜開發人員指南中的亞馬遜開發人員深入解析的非同步分析。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListTopicsDetectionJobs中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
```

```
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListTopicsDetectionJobs中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件啟動 Amazon Comprehend 文件分類任務 AWS

下列程式碼範例顯示如何啟動 Amazon Comprehend 文件分類工作。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 訓練自訂分類器並對文件進行分類

CLI

AWS CLI

開始文件分類工作

下列start-document-classification-job範例會在--input-data-config標籤所指定位址的所有檔案上以自訂模型啟動文件分類工作。在此範例中，輸入 S3 儲存貯體包含SampleSMStext1.txtSampleSMStext2.txt、和SampleSMStext3.txt。該模型先前針對垃圾郵件和非垃圾郵件的文檔分類進行了培訓，或「ham」，SMS 消息。工作完成後，將放output.tar.gz置在由--output-data-config標籤指定的位置。output.tar.gz包predictions.jsonl含列出每個文件的分類。Json 輸出會列印在每個檔案的一行上，但在此格式化以提高可讀性。

```
aws comprehend start-document-classification-job \
    --job-name exampleclassificationjob \
    --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET-INPUT/jobdata/" \
    --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-
classifier/mymodel/version/12
```

SampleSMStext1.txt 的內容：

```
"CONGRATULATIONS! TXT 2155550100 to win $5000"
```

SampleSMStext2.txt 的內容：

```
"Hi, when do you want me to pick you up from practice?"
```

SampleSMStext3.txt 的內容：

```
"Plz send bank account # to 2155550100 to claim prize!!"
```

輸出：

```
{
    "JobId": "e758dd56b824aa717ceab551fEXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-classification-
job/e758dd56b824aa717ceab551fEXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

predictions.jsonl 的內容：

```
{"File": "SampleSMSText1.txt", "Line": "0", "Classes": [{"Name": "spam", "Score":
 0.9999}, {"Name": "ham", "Score": 0.0001}]}
{"File": "SampleSMStext2.txt", "Line": "0", "Classes": [{"Name": "ham", "Score":
 0.9994}, {"Name": "spam", "Score": 0.0006}]}
{"File": "SampleSMSText3.txt", "Line": "0", "Classes": [{"Name": "spam", "Score":
 0.9999}, {"Name": "ham", "Score": 0.0001}]}
```

如需詳細資訊，請參閱 Amazon 開發人員指南中的自訂分類。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考StartDocumentClassificationJob中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
 can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.
```

```
        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件StartDocumentClassificationJob中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件啟動 Amazon Comprehend 主題建模任務 AWS

下列程式碼範例顯示如何啟動 Amazon Comprehend 主題建模工作。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 針對範例資料執行主題建模工作

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example scans the documents in an Amazon Simple Storage Service
/// (Amazon S3) bucket and analyzes it for topics. The results are stored
/// in another bucket and then the resulting job properties are displayed
/// on the screen. This example was created using the AWS SDK for .NEt
/// version 3.7 and .NET Core version 5.0.
/// </summary>
public static class TopicModeling
{
    /// <summary>
    /// This methos calls a topic detection job by calling the Amazon
    /// Comprehend StartTopicsDetectionJobRequest.
    /// </summary>
    public static async Task Main()
```

```
        {
            var comprehendClient = new AmazonComprehendClient();

            string inputS3Uri = "s3://input bucket/input path";
            InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
            string outputS3Uri = "s3://output bucket/output path";
            string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access
role";
            int numberOfTopics = 10;

            var startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
            {
                InputDataConfig = new InputDataConfig()
                {
                    S3Uri = inputS3Uri,
                    InputFormat = inputDocFormat,
                },
                OutputDataConfig = new OutputDataConfig()
                {
                    S3Uri = outputS3Uri,
                },
                DataAccessRoleArn = dataAccessRoleArn,
                NumberOfTopics = numberOfTopics,
            };

            var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

            var jobId = startTopicsDetectionJobResponse.JobId;
            Console.WriteLine("JobId: " + jobId);

            var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
            {
                JobId = jobId,
            };

            var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);
```

```
            var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new
ListTopicsDetectionJobsRequest());
            foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
            {
                PrintJobProperties(props);
            }
        }

        /// <summary>
        /// This method is a helper method that displays the job properties
        /// from the call to StartTopicsDetectionJobRequest.
        /// </summary>
        /// <param name="props">A list of properties from the call to
        /// StartTopicsDetectionJobRequest.</param>
        private static void PrintJobProperties(TopicsDetectionJobProperties
props)
        {
            Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
            Console.WriteLine($"NumberOfTopics:
{props.NumberOfTopics}\nInputS3Uri: {props.InputDataConfig.S3Uri}");
            Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[StartTopicsDetectionJob](#)中的。

CLI

AWS CLI

若要啟動主題偵測分析工作

下列`start-topics-detection-job`範例會針對位於`--input-data-config`標籤所指定位址的所有檔案啟動非同步主題偵測工作。工作完成後，資料夾會放置在標`--ouput-data-config`籤指定的位置。`output` `output`包含 topic-terms.csv 和 doc-topics.csv 的檔案。第一個輸出檔案 topic-terms.csv 是集合中主題的清單。對於每個主題，根據預設，清單會根據主題

的重量包含最上層的術語。第二個檔案會列出與主題相關聯的文件doc-topics.csv，以及與主題有關的文件比例。

```
aws comprehend start-topics-detection-job \
    --job-name example_topics_detection_job \
    --language-code en \
    --input-data-config "S3Uri=s3://DOC-EXAMPLE-BUCKET/" \
    --output-data-config "S3Uri=s3://DOC-EXAMPLE-DESTINATION-BUCKET/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --language-code en
```

輸出：

```
{
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

如需詳細資訊，請參閱 Amazon Comprehend 開發人員指南中的主題建模。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考StartTopicsDetectionJob中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
```

```
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix,
all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                          file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
```

```
            JobName=job_name,
            DataAccessRoleArn=data_access_role_arn,
            InputDataConfig={
                "S3Uri": f"s3://{input_bucket}/{input_key}",
                "InputFormat": input_format.value,
            },
            OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
        )
        logger.info("Started topic modeling job %s.", response["JobId"])
    except ClientError:
        logger.exception("Couldn't start topic modeling job.")
        raise
    else:
        return response
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件StartTopicsDetectionJob中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 的 Amazon Comprehend 案例 AWS

下列程式碼範例說明如何使 AWS 用軟體開發套件在 Amazon Comprehend 中實作常見案例。這些案例說明如何透過在 Amazon Comprehend 中呼叫多個函數來完成特定任務。每個案例都包含一個連結 GitHub，您可以在其中找到如何設定和執行程式碼的指示。

範例

- 使用 Amazon Comprehend 和開發套件偵測文件元素 AWS
- 使用開發套件在範例資料上執行 Amazon Comprehend 主題建模任務 AWS
- 訓練自訂的亞馬遜分類器，並使用開發套件對文件進行分類 AWS

使用 Amazon Comprehend 和開發套件偵測文件元素 AWS

以下程式碼範例顯示做法：

- 偵測文件中的語言、實體和關鍵片語。

- 偵測文件中的個人識別資訊 (PII)。
- 偵測文件的情緒。
- 檢測文檔中的語法元素。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

創建一個包裝 Amazon Comprehend 操作的類。

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
```

```
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
```

```
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can
be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
```

```
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

呼叫包裝函式類別上的函數，以偵測文件中的實體、片語等。

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend detection demo!")
    print("-" * 88)
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_detect = ComprehendDetect(boto3.client("comprehend"))
    with open("detect_sample.txt") as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print("-" * 88)
    print(sample_text)
    print("-" * 88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]["LanguageCode"]

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])

    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
    print("SentimentScore:")
    pprint(sentiment["SentimentScore"])

    print("Detecting syntax elements.")
    syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(syntax_tokens[:demo_size])
```

```
    print("Thanks for watching!")
    print("-" * 88)
```

- 如需 API 的詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。
  - DetectDominantLanguage
  - DetectEntities
  - DetectKeyPhrases
  - DetectPiiEntities
  - DetectSentiment
  - DetectSyntax

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件在範例資料上執行 Amazon Comprehend 主題建模任務 AWS

以下程式碼範例顯示做法：

- 針對範例資料執行 Amazon Comprehend 主題建模任務。
- 取得有關工作的資訊。
- 從 Amazon S3 擷取任務輸出資料。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立包裝類別以呼叫 Amazon Comprehend 主題建模動作。

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
 stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
 so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix,
 all
                          of them are used.
        :param input_format: The format of the input data, either one document
 per
                          file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
 that
```

```
                                    grants Comprehend permission to read from
the
                                    input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
```

```
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

使用包裝函式類別來執行主題模型工作並取得工作資料。

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    input_prefix = "input/"
    output_prefix = "output/"
    demo_resources = ComprehendDemoResources(
        boto3.resource("s3"), boto3.resource("iam")
    )
    topic_modeler = ComprehendTopicModeler(boto3.client("comprehend"))

    print("Setting up storage and security resources needed for the demo.")
    demo_resources.setup("comprehend-topic-modeler-demo")
    print("Copying sample data from public bucket into input bucket.")
    demo_resources.bucket.copy(
        {"Bucket": "public-sample-us-west-2", "Key": "TopicModeling/Sample.txt"},
        f"{input_prefix}sample.txt",
    )

    print("Starting topic modeling job on sample data.")
```

```
    job_info = topic_modeler.start_job(
        "demo-topic-modeling-job",
        demo_resources.bucket.name,
        input_prefix,
        JobInputFormat.per_line,
        demo_resources.bucket.name,
        output_prefix,
        demo_resources.data_access_role.arn,
    )

    print(
        f"Waiting for job {job_info['JobId']} to complete. This typically takes "
        f"20 - 30 minutes."
    )
    job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = topic_modeler.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from the output Amazon S3 bucket: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = demo_resources.extract_job_output(job)
    lines = 10
    print(f"First {lines} lines of document topics output:")
    pprint(job_output["doc-topics.csv"]["data"][:lines])
    print(f"First {lines} lines of terms output:")
    pprint(job_output["topic-terms.csv"]["data"][:lines])

    print("Cleaning up resources created for the demo.")
    demo_resources.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- 如需 API 的詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。

- DescribeTopicsDetectionJob
- ListTopicsDetectionJobs
- StartTopicsDetectionJob

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

訓練自訂的亞馬遜分類器，並使用開發套件對文件進行分類 AWS

以下程式碼範例顯示做法：

- 創建一個 Amazon Comprehend 多標籤分類器。
- 訓練樣本數據的分類器。
- 對第二組資料執行分類工作。
- 從 Amazon S3 擷取任務輸出資料。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

創建一個包裝類來調用 Amazon Comprehend 文檔分類器操作。

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None
```

```
    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
immediately
        starts training on the data found in the specified Amazon S3 bucket.
Training
        can take 30 minutes or longer. The `describe_document_classifier`
function
        can be used to get training status and returns a status of TRAINED when
the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
data.
        :param training_key: The prefix used to find training data in the
training
                             bucket. If multiple objects have the same prefix,
all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
```

```
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
```

```
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
self.classifier_arn)
            raise


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.
```

```
        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
```

```
        :return: Metadata about the job.
        """
        try:
            response =
 self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

創建一個類來幫助運行場景。

```
class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """
```

```
    def __init__(self, demo_resources):
        """
        :param demo_resources: A ComprehendDemoResources class that manages
 resources
                               for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = "training/"
        self.input_prefix = "input/"
        self.input_format = JobInputFormat.per_line
        self.output_prefix = "output/"

    def setup(self):
        """Creates AWS resources used by the demo."""
        self.demo_resources.setup("comprehend-classifier-demo")

    def cleanup(self):
        """Deletes AWS resources used by the demo."""
        self.demo_resources.cleanup()

    @staticmethod
    def _sanitize_text(text):
        """Removes characters that cause errors for the document parser."""
        return text.replace("\r", " ").replace("\n", " ").replace(",", ";")

    @staticmethod
    def _get_issues(query, issue_count):
        """
        Gets issues from GitHub using the specified query parameters.

        :param query: The query string used to request issues from the GitHub
 API.
        :param issue_count: The number of issues to retrieve.
        :return: The list of issues retrieved from GitHub.
        """
        issues = []
        logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
        response = requests.get(f"{GITHUB_SEARCH_URL}?
{query}&per_page={issue_count}")
        if response.status_code == 200:
            issue_page = response.json()["items"]
            logger.info("Got %s issues.", len(issue_page))
            issues = [
```

```
                {
                    "title": ClassifierDemo._sanitize_text(issue["title"]),
                    "body": ClassifierDemo._sanitize_text(issue["body"]),
                    "labels": {label["name"] for label in issue["labels"]},
                }
                for issue in issue_page
            ]
        else:
            logger.error(
                "GitHub returned error code %s with message %s.",
                response.status_code,
                response.json(),
            )
        logger.info("Found %s issues.", len(issues))
        return issues

    def get_training_issues(self, training_labels):
        """
        Gets issues used for training the custom classifier. Training issues are
        closed issues from the Boto3 repo that have known labels. Comprehend
        requires a minimum of ten training issues per label.

        :param training_labels: The issue labels to use for training.
        :return: The set of issues used for training.
        """
        issues = []
        per_label_count = 15
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:closed+label:{label}",
                per_label_count,
            )
            for issue in issues:
                issue["labels"] = issue["labels"].intersection(training_labels)
        return issues

    def get_input_issues(self, training_labels):
        """
        Gets input issues from GitHub. For demonstration purposes, input issues
        are open issues from the Boto3 repo with known labels, though in practice
        any issue could be submitted to the classifier for labeling.

        :param training_labels: The set of labels to query for.
        :return: The set of issues used for input.
```

```
        """
        issues = []
        per_label_count = 5
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:open+label:{label}",
                per_label_count,
            )
        return issues

    def upload_issue_data(self, issues, training=False):
        """
        Uploads issue data to an Amazon S3 bucket, either for training or for
input.
        The data is first put into the format expected by Comprehend. For
training,
        the set of pipe-delimited labels is prepended to each document. For
        input, labels are not sent.

        :param issues: The set of issues to upload to Amazon S3.
        :param training: Indicates whether the issue data is used for training or
                         input.
        """
        try:
            obj_key = (
                self.training_prefix if training else self.input_prefix
            ) + "issues.txt"
            if training:
                issue_strings = [
                    f"{'|'.join(issue['labels'])},{issue['title']}
{issue['body']}"
                    for issue in issues
                ]
            else:
                issue_strings = [
                    f"{issue['title']} {issue['body']}" for issue in issues
                ]
            issue_bytes = BytesIO("\n".join(issue_strings).encode("utf-8"))
            self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
            logger.info(
                "Uploaded data as %s to bucket %s.",
                obj_key,
                self.demo_resources.bucket.name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't upload data to bucket %s.",
self.demo_resources.bucket.name
            )
            raise

    def extract_job_output(self, job):
        """Extracts job output from Amazon S3."""
        return self.demo_resources.extract_job_output(job)

    @staticmethod
    def reconcile_job_output(input_issues, output_dict):
        """
        Reconciles job output with the list of input issues. Because the input
issues
        have known labels, these can be compared with the labels added by the
        classifier to judge the accuracy of the output.

        :param input_issues: The list of issues used as input.
        :param output_dict: The dictionary of data that is output by the
classifier.
        :return: The list of reconciled input and output data.
        """
        reconciled = []
        for archive in output_dict.values():
            for line in archive["data"]:
                in_line = int(line["Line"])
                in_labels = input_issues[in_line]["labels"]
                out_labels = {
                    label["Name"]
                    for label in line["Labels"]
                    if float(label["Score"]) > 0.3
                }
                reconciled.append(
                    f"{line['File']}, line {in_line} has labels {in_labels}.\n"
                    f"\tClassifier assigned {out_labels}."
                )
        logger.info("Reconciled input and output labels.")
        return reconciled
```

使用已知標籤對一組 GitHub 問題進行分類器訓練，然後向分類器發送第二組 GitHub 問題，以便它們可以被標記。

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_demo = ClassifierDemo(
        ComprehendDemoResources(boto3.resource("s3"), boto3.resource("iam"))
    )
    comp_classifier = ComprehendClassifier(boto3.client("comprehend"))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client
    )
    training_labels = {"bug", "feature-request", "dynamodb", "s3"}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = "doc-example-classifier"
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name,
        "en",
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label,
    )
    print(
        f"Waiting until {classifier_name} is trained. This typically takes "
        f"30-40 minutes."
    )
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
```

```
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
        "issue_classification_job",
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn,
    )
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = comp_classifier.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from Amazon S3: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = comp_demo.extract_job_output(job)
    print("Job output:")
    pprint(job_output)

    print("Reconciling job output with labels from GitHub:")
    reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
    print(*reconciled_output, sep="\n")

    answer = input(f"Do you want to delete the classifier {classifier_name} (y/
n)? ")
    if answer.lower() == "y":
        print(f"Deleting {classifier_name}.")
        comp_classifier.delete()

    print("Cleaning up resources created for the demo.")
    comp_demo.cleanup()
```

```
    print("Thanks for watching!")
    print("-" * 88)
```

- 如需 API 的詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。
  - CreateDocumentClassifier
  - DeleteDocumentClassifier
  - DescribeDocumentClassificationJob
  - DescribeDocumentClassifier
  - ListDocumentClassificationJobs
  - ListDocumentClassifiers
  - StartDocumentClassificationJob

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件的 Amazon Comprehend 跨服務範例 AWS

下列範例應用程式使用 AWS 軟體開發套件來結合 Amazon Comprehend 與其他應用程式。 AWS 服務每個範例都包含一個連結 GitHub，您可以在其中找到如何設定和執行應用程式的指示。

範例

- 建置 Amazon Transcribe 串流應用程式
- 建立 Amazon Lex 聊天機器人來吸引您的網站訪客
- 使用 Amazon SQS 建立可傳送和擷取訊息的 Web 應用程式
- 建立可分析客戶意見回饋並合成音訊的應用程式
- 使用 AWS SDK 檢測從圖像中提取的文本中的實體

建置 Amazon Transcribe 串流應用程式

下面的程式碼範例說明如何建置可即時記錄、轉錄和翻譯直播音訊並透過電子郵件傳送結果的應用程式。

JavaScript

適用於 JavaScript (v3) 的開發套件

說明如何使用 Amazon Transcribe 建置應用程式，該應用程式可即時記錄、轉錄和翻譯直播音訊，並可使用 Amazon Simple Email Service (Amazon SES) 透過電子郵件傳送結果。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立 Amazon Lex 聊天機器人來吸引您的網站訪客

以下代碼示例演示瞭如何創建聊天機器人以吸引您的網站訪問者。

Java

適用於 Java 2.x 的 SDK

示範如何使用 Amazon Lex API 在 Web 應用程式中建立 Chatbot，以吸引您的網站訪客。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

適用於 JavaScript (v3) 的開發套件

示範如何使用 Amazon Lex API 在 Web 應用程式中建立 Chatbot，以吸引您的網站訪客。

如需有關如何設定和執行的完整原始程式碼和指示，請參閱開發人 AWS SDK for JavaScript 員指南中的建立 Amazon Lex 聊天機器人的完整範例。

此範例中使用的服務

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 Amazon SQS 建立可傳送和擷取訊息的 Web 應用程式

下列程式碼範例說明如何使用 Amazon SQS 建立簡訊應用程式。

Java

適用於 Java 2.x 的 SDK

示範如何使用 Amazon SQS API 開發可傳送和擷取訊息的春季 REST API。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Comprehend
- Amazon SQS

Kotlin

適用於 Kotlin 的 SDK

示範如何使用 Amazon SQS API 開發可傳送和擷取訊息的春季 REST API。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Comprehend
- Amazon SQS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立可分析客戶意見回饋並合成音訊的應用程式

下列程式碼範例示範如何建立應用程式，以分析客戶評論卡、從其原始語言進行翻譯、判斷對方情緒，以及透過翻譯後的文字產生音訊檔案。

.NET

AWS SDK for .NET

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署指示，請參閱中的專案 GitHub。

此範例中使用的服務

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

適用於 Java 2.x 的 SDK

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署指示，請參閱中的專案 GitHub。

此範例中使用的服務

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

適用於 JavaScript (v3) 的開發套件

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署指示，請參閱中的專案 GitHub。下列摘錄顯示如何在 AWS SDK for JavaScript Lambda 函數內部使用。

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";
```

```
/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
```

```
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});
```

```
  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

此範例中使用的服務

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

適用於 Ruby 的開發套件

此範例應用程式會分析和存儲客戶的意見回饋卡。具體來說，它滿足了紐約市一家虛構飯店的需求。飯店以實體評論卡的形式收到賓客以各種語言撰寫的意見回饋。這些意見回饋透過 Web 用戶端上傳至應用程式。評論卡的影像上傳後，系統會執行下列步驟：

- 文字內容是使用 Amazon Textract 從影像中擷取。
- Amazon Comprehend 會決定擷取文字及其用語的情感。
- 擷取的文字內容會使用 Amazon Translate 翻譯成英文。
- Amazon Polly 會使用擷取的文字內容合成音訊檔案。

完整的應用程式可透過 AWS CDK 部署。如需原始程式碼和部署指示，請參閱中的專案 GitHub。

此範例中使用的服務

- Amazon Comprehend
- Lambda

- Amazon Polly
- Amazon Textract
- Amazon Translate

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 檢測從圖像中提取的文本中的實體

下列程式碼範例示範如何使用 Amazon Comprehend 偵測 Amazon Textract 從存放在 Amazon S3 中的影像中提取的文字中的實體。

Python

適用於 Python (Boto3) 的 SDK

示範如何使用 Jupyter 筆記本 AWS SDK for Python (Boto3) 中的偵測從影像擷取的文字中的實體。本範例使用 Amazon Textract 從儲存於 Amazon Simple Storage Service (Amazon S3) 和 Amazon Comprehend 中的影像提取文字，以偵測擷取文字中的實體。

此範例是 Jupyter 的筆記型電腦，必須在可以託管的筆記型電腦的環境中運行。有關如何使用 Amazon 運行示例的說明 SageMaker，請參閱 TextractAndComprehendNotebook.ip ynb 中的說明。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Comprehend
- Amazon S3
- Amazon Textract

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Comprehend 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

亞馬遜的安全理解

雲端安全是 AWS 最重視的一環。身為 AWS 客戶的您，將能從資料中心和網路架構的建置中獲益，以滿足組織最為敏感的安全要求。

安全是 AWS 與您共同肩負的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端本身的安全 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計畫的一部分。若要了解適用於 Amazon Comprehend 的合規計劃，請參閱合規計劃的合規計劃AWS服務範圍內的服務。
- 雲端內部的安全 – 您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的請求和適用法律和法規。

本文件可協助您了解如何在使用 Amazon Comprehend 時套用共同的責任模型。下列主題說明如何設定 Amazon Comprehend 以符合您的安全性和合規目標。您也會學到如何使用其他可AWS協助您監控和保護 Amazon Comprehend 資源的服務。

主題

- Amazon Comprehend 域的資料保護
- Amazon Comprehend 的身分和存取管理
- 使用日誌記錄 Amazon Comprehend API 調用 AWS CloudTrail
- Amazon Comprehend 合規驗證
- Amazon Comprehend 中的韌性
- Amazon Comprehend 的基礎設施安全

Amazon Comprehend 域的資料保護

AWS共同責任模型適用於 Amazon Comprehend 中的資料保護。如此模型所述，AWS 負責保護執行所有 AWS 雲端 的全球基礎設施。您負責維護在此基礎設施上託管內容的控制權。您也必須負責您所使用 AWS 服務 的安全組態和管理任務。如需有關資料隱私權的更多相關資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 憑證，並設定個人使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 AWS CloudTrail 設定 API 和使用者活動日誌記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、API 或開發套件AWS 服務使用 Amazon Comprehend 或AWS其他工作時。AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

主題

- Amazon Comprehend 中的 KMS 加密
- 預防跨服務混淆代理人
- 使用 Amazon 虛擬私有雲來保護工作
- Amazon Comprehend 和接口 VPC 端點 () AWS PrivateLink

Amazon Comprehend 中的 KMS 加密

Amazon Comprehend 可與 AWS Key Management Service (AWS KMS) 搭配使用，為您的資料提供增強的加密功能。Amazon S3 已讓您在建立文字分析、主題建模或自訂 Amazon Comprehend 任務時加密輸入文件。與整合AWS KMS可讓您針對 Start* 和 Create* 工作加密儲存磁碟區中的資料，並使用您自己的 KMS 金鑰加密 Start* 任務的輸出結果。

對於AWS Management Console，亞馬遜使用自己的 KMS 金鑰加密自訂模型。對於AWS CLI，Amazon Comprehend 可以使用自己的 KMS 金鑰或提供的客戶受管金鑰 (CMK) 來加密自訂模型。

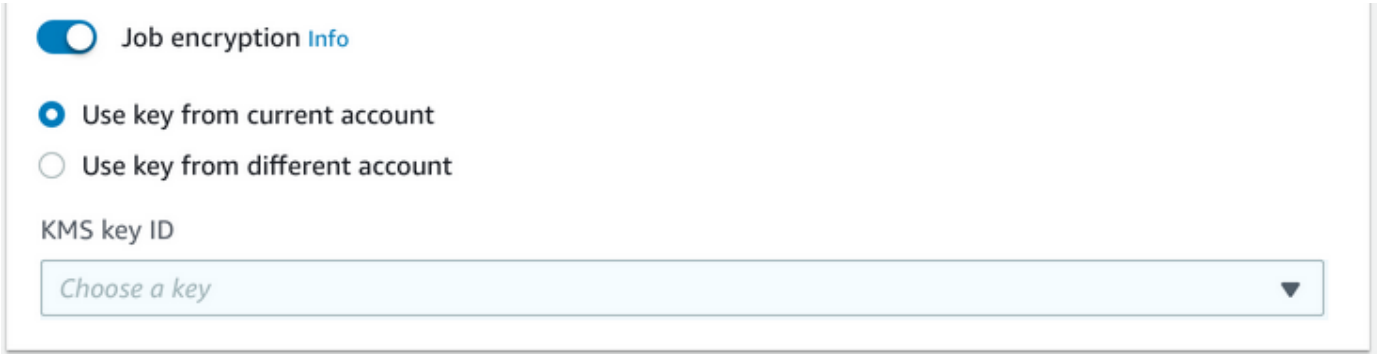
使用 KMS 加密 AWS Management Console

使用控制台時有兩個加密選項可用：

- 磁碟區加密
- 輸出結果加密

啟用磁碟區加密

1. 在「Job 設定」 下，選擇「Job 加密」選項。



2. 選擇 KMS 客戶管理金鑰 (CMK) 來自您目前使用的帳戶，還是來自不同的帳戶。如果您要使用目前帳戶中的金鑰，請從 KMS 金鑰 ID 中選擇金鑰別名。如果您使用不同帳戶的金鑰，則必須輸入金鑰的 ARN。

啟用輸出結果加密

1. 在「輸出設定」 下，選擇「加密」選項。

2. 選擇客戶管理金鑰 (CMK) 來自您目前使用的帳戶，還是來自其他帳戶。如果您要使用目前帳戶中的金鑰，請從 KMS 金鑰識別碼中選擇金鑰識別碼。如果您使用不同帳戶的金鑰，則必須輸入金鑰的 ARN。

如果您先前已在 S3 輸入文件上使用 SSE-KMS 設定加密，這可以為您提供額外的安全性。不過，如果您這麼做，所使用的 IAM 角色必須具有用來加密輸入文件之 KMS 金鑰的`kms:Decrypt`權限。如需詳細資訊，請參閱 使用 KMS 加密所需的權限。

使用 API 作業進行 KMS 加密

所有 Amazon Comprehend `Start*` 和 `Create*` API 操作都支援 KMS 加密的輸入文件。`Describe*`和 `List*` API 操作返回，`OutputDataConfig`如果原始作業`KmsKeyId`提供作為輸入。`KmsKeyId`如果未將其作為輸入提供，則不會返回。

這可以在使用StartEntitiesDetectionJob操作的以下 AWS CLI 示例中看到：

```
aws comprehend start-entities-detection-job \
     --region region \
     --data-access-role-arn "data access role arn" \
     --entity-recognizer-arn "entity recognizer arn" \
     --input-data-config "S3Uri=s3://Bucket Name/Bucket Path" \
     --job-name job name \
     --language-code en \
     --output-data-config "KmsKeyId=Output S3 KMS key ID" "S3Uri=s3://Bucket
 Name/Bucket Path/" \
     --volumekmskeyid "Volume KMS key ID"
```

> **Note**
>
> 此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

使用 API 作業的客戶管理金鑰 (CMK) 加密

Amazon Comprehend 自訂模型 API 操作和支援使用客戶受管金鑰的加密`CreateEndpoint`，透過. `CreateEntityRecognizer` `CreateDocumentClassifier` AWS CLI

您需要 IAM 政策來允許主體使用或管理客戶受管金鑰。這些索引鍵會在原則陳述式的`Resource`項目中指定。最佳作法是將客戶管理的金鑰限制為只有主體必須在您的政策陳述式中使用的金鑰。

下列 AWS CLI 範例會使用CreateEntityRecognizer作業建立具有模型加密的自訂實體辨識器：

```
aws comprehend create-entity-recognizer \
     --recognizer-name name \
     --data-access-role-arn data access role arn \
     --language-code en \
     --model-kms-key-id Model KMS Key ID \
     --input-data-config file:///path/input-data-config.json
```

> **Note**
>
> 此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

預防跨服務混淆代理人

混淆代理人問題屬於安全性問題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 AWS 中，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況，AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

我們建議在資源政策中使用aws:SourceArn和aws:SourceAccount全域條件內容金鑰，以限制 Amazon Comprehend 為資源提供其他服務的許可。如果同時使用全域條件內容索引鍵，則在相同政策陳述式中使用 aws:SourceAccount 值和 aws:SourceArn 值中的帳戶時，必須使用相同的帳戶 ID。

防範混淆代理人問題最有效的方法，是使用 aws:SourceArn 全域條件內容金鑰，以及資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 aws:SourceArn 全域條件內容金鑰，同時使用萬用字元 (*) 表示 ARN 的未知部分。例如 arn:aws:*servicename*::*123456789012*:*。

使用來源帳戶

下列範例顯示如何在 Amazon Comprehend 中使用aws:SourceAccount全域條件內容金鑰。

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

```
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
          "Service": "comprehend.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringEquals": {
             "aws:SourceAccount":"111122223333"
        }
    }
  }
}
```

加密模型端點的信任策略

您需要建立信任原則，以建立或更新加密模型的端點。將 aws:SourceAccount 值設定為您的帳戶 ID。如果使用ArnEquals條件，請將該aws:SourceArn值設定為端點的 ARN。

```
{
 "Version": "2012-10-17",
 "Statement": [
    {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
            "Service": "comprehend.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
               "aws:SourceAccount": "111122223333"
            },
            "ArnEquals": {
               "aws:SourceArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier-endpoint/endpoint-name"
            }
        }
    }
  ]
}
```

建立自訂模型

您必須建立信任原則才能建立自訂模型。將 aws:SourceAccount 值設定為您的帳戶 ID。如果您使用ArnEquals條件，請將該aws:SourceArn值設定為自訂模型版本的 ARN。

```
{
 "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "",
          "Effect": "Allow",
          "Principal": {
              "Service": "comprehend.amazonaws.com"
          },
          "Action": "sts:AssumeRole",
          "Condition": {
              "StringEquals": {
                  "aws:SourceAccount": "111122223333"
              },
              "ArnEquals": {
                  "aws:SourceArn": "arn:aws:comprehend:us-west-2:111122223333:
                            document-classifier/smallest-classifier-test/
version/version-name"
              }
          }
      }
  ]
}
```

使用 Amazon 虛擬私有雲來保護工作

Amazon Comprehend 使用各種安全措施，透過我們的任務容器在 Amazon Comprehend 正在使用時儲存資料的容器，確保資料的安全性。但是，任務容器會透過網際網路存取 AWS 資源 (例如存放資料和模型人工因素的 Amazon S3 儲存貯體)。

若要控制對資料的存取，我們建議您建立虛擬私有雲 (VPC) 並進行設定，以便無法透過網際網路存取資料和容器。如需 VPC 在建立和設定方面的資訊，請參閱《Amazon VPC 使用者指南》中的 Amazon VPC 入門的相關文章。使用 VPC 有助於保護您的資料，因為您可以設定 VPC，使其未連接到網際網路。使用 VPC 也可讓您使用 VPC 流程記錄來監控工作容器的所有進出網路流量。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》中的 VPC 流量日誌。

您可以在建立工作時指定 VPC 組態，方法是指定子網路和安全群組。當您指定子網路和安全群組時，Amazon Comprehend 會建立與其中一個子網路中的安全群組相關聯的彈性網路界面 (ENI)。ENI 允許我們的工作容器連接到 VPC 中的資源。如需 ENI 的相關資訊，請參閱 Amazon VPC 使用者指南中的彈性網路介面。

> **Note**
>
> 針對工作，您只能使用預設租用 VPC 來設定子網路，而您的執行個體會在共用硬體上執行。如需 VPC 租用屬性的詳細資訊，請參閱 Amazon EC2 Linux 執行個體使用者指南中的專用執行個體。

設定適用於 Amazon VPC 存取的任務

若要在 VPC 中指定子網路和安全群組，請使用適用 API 的`VpcConfig`請求參數，或在 Amazon Comprehend 主控台中建立任務時提供此資訊。Amazon Comprehend 使用此資訊來建立 ENI，並將它們附加到我們的任務容器。ENI 在您的 VPC 中為我們的工作容器提供了未連接到互聯網的網絡連接。

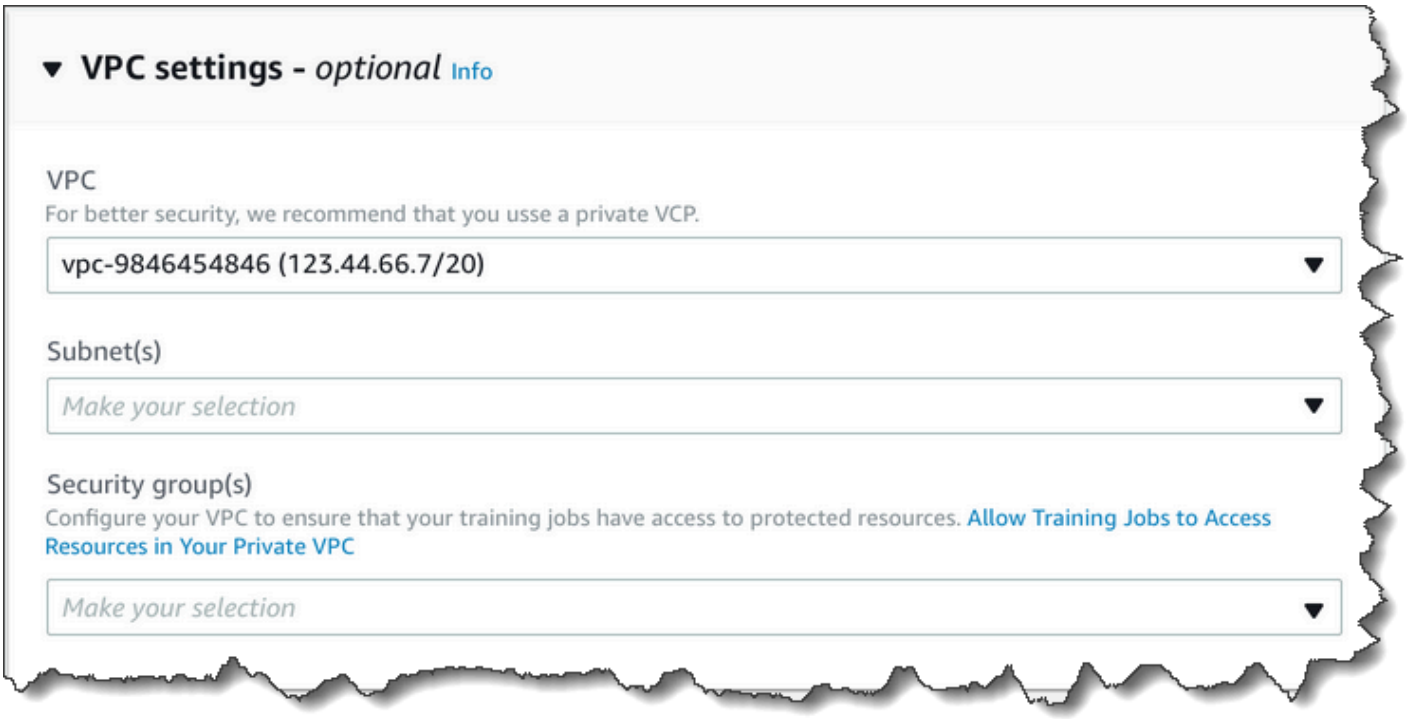
以下 API 包含請`VpcConfig`求參數：

- `Create*`應用程式介 `CreateDocumentClassifier`面： `CreateEntityRecognizer`
- `Start*`API： `StartDocumentClassificationJob`, `StartDominantLanguageDetectionJob`,, `StartEntitiesDetectionJob`, `StartKeyPhrasesDetectionJob`, `StartSentimentDetectionJob`, `StartTargetedSentimentDetectionJob`, `StartTopicsDetectionJob`

以下是您在 API 呼叫中包含的 VpcConfig 參數範例：

```
"VpcConfig": {
      "SecurityGroupIds": [
          " sg-0123456789abcdef0"
          ],
      "Subnets": [
          "subnet-0123456789abcdef0",
          "subnet-0123456789abcdef1",
          "subnet-0123456789abcdef2"
          ]
      }
```

若要從 Amazon Comprehend 主控台設定 VPC，請在建立任務時，從選用的 VPC 設定區段中選擇組態詳細資料。



針對 Amazon Comprehend 任務設定您的 VPC

為您的 Amazon 理解任務設定 VPC 人雲端時，請遵循下列準則。如需如何設定 VPC 的相關資訊，請參閱《Amazon VPC 使用者指南》中的使用 VPC 和子網路的相關文章。

確保子網路具有足夠的 IP 位址

對於工作中的每個執行個體，您的 VPC 子網路至少應具有兩個私有 IP 位址。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 IPv4 的 VPC 與子網路的大小調整的相關文章。

創建一個 Amazon S3 VPC 端點

如果您設定 VPC 使任務容器無法存取網際網路，則除非您建立允許存取的 VPC 端點，否則它們將無法連線到包含您資料的 Amazon S3 儲存貯體。透過建立 VPC 端點，您可以允許工作容器在訓練和分析工作期間存取您的資料。

建立 VPC 端點時，請設定下列值：

- 選擇服務類別作為「AWS 服務」
- 將服務指定為 `com.amazonaws.region.s3`

- 選取閘道作為 VPC 端點類型

如果您使用 AWS CloudFormation 建立虛擬私人 VPC 端點，請遵循 AWS CloudFormation vpcendPoint 說明文件。下列範例顯示範本中的 vpcendPoint 組態。 AWS CloudFormation

```
VpcEndpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Action:
            - s3:GetObject
            - s3:PutObject
            - s3:ListBucket
            - s3:GetBucketLocation
            - s3:DeleteObject
            - s3:ListMultipartUploadParts
            - s3:AbortMultipartUpload
          Effect: Allow
          Resource:
            - "*"
          Principal: "*"
    RouteTableIds:
      - Ref: RouteTable
    ServiceName:
      Fn::Join:
        - ''
        - - com.amazonaws.
          - Ref: AWS::Region
          - ".s3"
    VpcId:
      Ref: VPC
```

我們建議您也建立自訂政策，僅允許來自 VPC 的請求存取 S3 儲存貯體。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的適用於 Amazon S3 的端點。

以下政策允許存取 S3 儲存貯體。編輯此原則，只允許存取工作所需的資源。

```
{
```

```
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:DeleteObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "*"
        }
    ]
}
```

請為端點路由表使用預設的 DNS 設定，如此才能解析標準 Amazon S3 URL (例如 `http://s3-aws-region.amazonaws.com/MyBucket`)。如果您未使用預設 DNS 設定，請確定您用來指定工作中資料位置的 URL 會透過設定端點路由表來解析。如需 VPC 端點路由表的相關資訊，請參閱《Amazon VPC 使用者指南》中的閘道端點路由的相關文章。

預設端點政策允許使用者在我們的任務容器上安裝來自 Amazon Linux 和 Amazon Linux 2 儲存庫的套件。如果不希望使用者從該儲存庫安裝套件，請建立自訂端點政策，明確拒絕至 Amazon Linux 和 Amazon Linux 2 儲存庫的存取。Comprehend 本身不需要任何此類軟件包，因此不會有任何功能影響。以下為拒絕存取上述儲存庫的政策範例：

```
{
    "Statement": [
      {
        "Sid": "AmazonLinuxAMIRepositoryAccess",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::packages.*.amazonaws.com/*",
            "arn:aws:s3:::repo.*.amazonaws.com/*"
        ]
```

```
      }
    ]
}

{
    "Statement": [
        { "Sid": "AmazonLinux2AMIRepositoryAccess",
          "Principal": "*",
          "Action": [
              "s3:GetObject"
              ],
          "Effect": "Deny",
          "Resource": [
              "arn:aws:s3:::amazonlinux.*.amazonaws.com/*"
              ]
         }
    ]
}
```

的權限 **DataAccessRole**

當您將 VPC 與分析工作搭配使用時，DataAccessRole用於Create*和作Start*業的使用者也必須擁有存取輸入文件和輸出值區的 VPC 的權限。

下列原則提供Create*與Start*作業所需的存取權。DataAccessRole

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
              "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

設定 VPC 安全性群組

使用分散式工作時，您必須允許相同工作中不同工作容器之間的通訊。若要執行此操作，請為安全群組設定規則，允許相同安全群組成員彼此間的傳入連線。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的安全群組規則。

Connect 至 VPC 外部的資源

如果您將 VPC 設定為無法存取網際網路，則使用該 VPC 的工作將無法存取 VPC 以外的資源。如果您的工作需要存取 VPC 以外的資源，請使用下列其中一個選項提供存取權：

- 如果您的工作需要存取支援介面 VPC 端點的 AWS 服務，請建立端點以連線至該服務。如需支援介面端點的服務之清單，請參閱《Amazon VPC 使用者指南》中的 VPC 端點的相關文章。如需建立介面 VPC 端點的詳細資訊，請參閱 Amazon VPC 使用者指南中的介面虛擬私人雲端端點 (AWS PrivateLink)。
- 如果您的工作需要存取不支援介面 VPC 端點的 AWS 服務或外部資源的服務 AWS，請建立 NAT 閘道並設定安全性群組以允許輸出連線。如需為 VPC 設定 NAT 閘道的相關資訊，請參閱 Amazon VPC 使用者指南中的案例 2：具有公有和私有子網路 (NAT) 的 V PC。

Amazon Comprehend 和接口 VPC 端點 () AWS PrivateLink

您可以透過建立介面 VPC 端點，在虛擬私人雲端和 Amazon Comprehend 之間建立私有連線。介面端點採用這項技術 AWS PrivateLink，可讓您在沒有網際網路閘道、NAT 裝置、VPN 連線或AWS直 Connect 連線的情況下，私有存取 Amazon Comprehend API。VPC 中的執行個體不需要公有 IP 地址即可與 Amazon Comprehend API 進行通訊。您的 VPC 和 Amazon Comprehend 之間的流量不會離開 Amazon 網絡。

每個介面端點由子網路中的一或多個彈性網路介面來表示。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的介面 VPC 端點 (AWS PrivateLink)。

亞馬遜虛擬私人雲端端點的注意事項

在為 Amazon Comprehend 設定介面虛擬私人雲端節點之前，請務必先查看 Amazon VPC 使用者指南中的界面端點屬性和限制。

並非某個區域中的所有可用區域都提供 Amazon Comprehend 端點。建立端點時，請使用下列命令列出可用區域。

```
aws ec2 describe-vpc-endpoint-services \
  --service-names com.amazonaws.us-west-2.comprehend
```

Amazon Comprehend 支援從您的 VPC 呼叫其所有 API 動作。

為 Amazon Comprehend 創建一個接口 VPC 端點

您可以使用 Amazon Comprehend 虛擬私人雲端主控台或 () 為 Amazon 服務建立 VPC 端點。AWS Command Line Interface AWS CLI如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的建立介面端點。

使用下列服務名稱建立適用於 Amazon Comprehend 的 VPC 端點：

- com.amazonaws.*region*.comprehend

如果您為端點啟用私有 DNS，則可以使用該區域的預設 DNS 名稱向 Amazon Comprehend 發出 API 請求，例如。*comprehend.us-east-1.amazonaws.com*

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的透過介面端點存取服務。

為 Amazon Comprehend 創建 VPC 端點政策

您可以將端點政策附加到 VPC 端點，以控制對 Amazon Comprehend 的存取。此政策會指定下列資訊：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用 VPC 端點控制對服務的存取。

範例：適用於 Amazon Comprehend 動作的 VPC 端點政策

以下是適用於 Amazon Comprehend 的端點政策範例。連接到端點時，此政策會授予所有資源上所有主體對 Amazon Comprehend `DetectEntities` 動作的存取權。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "comprehend:DetectEntities"
         ],
         "Resource":"*"
      }
   ]
}
```

Amazon Comprehend 的身分和存取管理

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全地控制對 AWS 資源的存取權。IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有許可) 來使用 Amazon Comprehend 資源。IAM 是一種您可以免費使用的 AWS 服務。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon Comprehend 如何與 IAM 合作
- Amazon Comprehend 基於身份的政策示例
- AWS亞馬遜管理政策
- 疑難排解 Amazon Comprehend 身分和存取

物件

您的使用方式 AWS Identity and Access Management (IAM) 會有所不同，具體取決於您在亞馬遜領域所做的工作。

服務使用者 — 如果您使用 Amazon Comprehend 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您使用更多 Amazon Comprehend 功能來完成工作時，您可能需要額外的許可。瞭解存取

許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取亞馬遜中的某個功能，請參閱。疑難排解 Amazon Comprehend 身分和存取

服務管理員 — 如果您負責公司的亞馬遜資源，您可能擁有對 Amazon Comprehend 的完整存取權。您的任務是判斷服務使用者應存取哪些 Amazon Comprehend 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與亞馬遜合作搭配使用，請參閱。Amazon Comprehend 如何與 IAM 合作

IAM 管理員 — 如果您是 IAM 管理員，您可能想要了解如何撰寫政策以管理 Amazon Comprehend 存取權的詳細資訊。若要檢視您可 Amazon Comprehend 在 IAM 中使用的基於身分識別的政策範例，請參閱。Amazon Comprehend 基於身份的政策示例

使用身分驗證

身分驗證是使用身分憑證登入 AWS 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者身分，或擔任 IAM 角色進行 驗證 (登入至 AWS)。

您可以使用透過身分來源 AWS IAM Identity Center 提供的憑證，以聯合身分登入 AWS。(IAM Identity Center) 使用者、貴公司的單一登入身分驗證和您的 Google 或 Facebook 憑證都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。您 AWS 藉由使用聯合進行存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入至 AWS 的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

如果您是以程式設計的方式存取 AWS，AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以便使用您的憑證透過密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，您必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重要素驗證 (MFA) 以提高帳戶的安全。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的在 AWS 中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

如果是建立 AWS 帳戶，您會先有一個登入身分，可以完整存取帳戶中所有 AWS 服務 與資源。此身分稱為 AWS 帳戶 根使用者，使用建立帳戶時所使用的電子郵件地址和密碼即可登入並存取。強烈建議您不要以根使用者處理日常作業。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

聯合身分

最佳實務是要求人類使用者 (包括需要管理員存取權的使用者)搭配身分提供者使用聯合功能，使用暫時憑證來存取 AWS 服務。

聯合身分是來自您企業使用者目錄的使用者、Web 身分供應商、AWS Directory Service、Identity Center 目錄或透過身分來源提供的憑證來存取 AWS 服務 的任何使用者。聯合身分存取 AWS 帳戶時，會擔任角色，並由角色提供暫時憑證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分來源中的一組使用者和群組，以便在您的所有 AWS 帳戶和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱《AWS IAM Identity Center 使用者指南》中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您 AWS 帳戶 中的一種身分，具備單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰)的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#rotate-credentials中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的過程變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並取得由角色定義的許可。如需有關聯合角色

的詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-idp.html中的為第三方身分供應商建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱《AWS IAM Identity Center 使用者指南》中的許可集。

- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳户存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人)存取您帳戶中的資源。角色是授予跨帳户存取權的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源 (而非使用角色作為代理)。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。
- 跨服務存取 – 有些 AWS 服務 會使用其他 AWS 服務 中的功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉發存取工作階段 (FAS)：當您使用 IAM 使用者或角色在 AWS 中執行動作時，系統會將您視為主體。當您使用某些服務時，您可能會執行一個動作，而該動作之後會在不同的服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色：服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務 服務。
  - 服務連結角色 – 服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理暫時憑證。這是在 EC2 執行個體內儲存存取金鑰的較好方式。如需指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透過建立政策並將其附加到 AWS 身分或資源，在 AWS 中控制存取。政策是 AWS 中的一個物件，當其和身分或資源建立關聯時，便可定義其許可。AWS 會在主體 (使用者、根使用者或角色工作階段)發出請求時評估這些政策。政策中的許可，決定是否允許或拒絕請求。大部分政策以 JSON 文件形式儲存在 AWS 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授與使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具備該政策的使用者便可以從 AWS Management Console、AWS CLI 或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色)的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策則是獨立的政策，您可以將這些政策附加到 AWS 帳戶 中的多個使用者、群組和角色。受管政策包含 AWS 管理政策和客戶管理政策。如需瞭解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色)擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon Simple Storage Service (Amazon S3)、AWS WAF 和 Amazon VPC 是支援 ACL 的服務範例。若要進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授與您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可範圍。
- 服務控制政策 (SCP) – SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需組織和 SCP 的更多相關資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱《IAM 使用者指南》中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。如需瞭解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，請參閱 IAM 使用者指南中的政策評估邏輯。

Amazon Comprehend 如何與 IAM 合作

在您使用 IAM 管理對亞馬遜的存取權限之前，請先了解哪些 IAM 功能可以搭配 Amazon Comprehend 使用。

您可以與 Amazon Comprehend 合作搭配使用的 IAM 功能

IAM 功能	Amazon Comprehend 支持
身分型政策	是
資源型政策	是
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	是
服務連結角色	否

若要深入瞭解 Amazon Comprehend 和其他AWS服務如何與大多數 IAM 功能搭配運作，請參閱 IAM 使用者指南中的與 IAM 搭配使用的AWS服務。

Amazon Comprehend 基於身份的政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至附加的使用者或角色。如要瞭解您在 JSON 政策中使用的所有元素，請參閱 IAM 使用者指南中的 IAM JSON 政策元素參考。

Amazon Comprehend 基於身份的政策示例

若要檢視以身分識別為基礎的政策範例，請參閱。Amazon Comprehend 基於身份的政策示例

Amazon Comprehend 內基於資源的政策

支援以資源基礎的政策	是

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

若要啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源在不同的 AWS 帳戶 中時，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 存取資源的許可。其透過將身分型政策附加到實體來授予許可。不過，如果資源型政策會為相同帳戶中的主體授與存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策有何差異。

Amazon Comprehend 服務僅支援一種類型的資源型政策 (自訂模型政策)，並附加至自訂模型。此策略定義了可以使用自訂模型的其他帳戶。

若要瞭解如何將以資源為基礎的原則附加至自訂模型，請參閱自訂模型的資源型政策。

Amazon Comprehend 政策行動

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作的名稱通常會和相關聯的 AWS API 操作相同。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些操作需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授與執行相關聯操作的許可。

若要查看 Amazon Comprehend 動作清單，請參閱服務授權參考資料中由 Amazon Comprehend 定義的動作。

Amazon 中的政策動作會在動作之前使用下列前置詞：

```
comprehend
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "comprehend:DetectSentiment",
      "comprehend:ClassifyDocument"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "comprehend:Describe*"
```

請勿使用萬用字元來指定服務的所有動作。在原則中指定權限時，請使用授與最少權限的最佳作法。

若要檢視以身分識別為基礎的政策範例，請參閱。Amazon Comprehend 基於身份的政策示例

Amazon Comprehend 政策資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出作業)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

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

若要查看 Amazon Comprehend 資源類型及其 ARN 的清單，請參閱服務授權參考資料中的 Amazon Comprehend 定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Comprehend 定義的動作。

Amazon Comprehend 政策條件密鑰

支援服務特定政策條件索引鍵	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊)可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。若您為單一條件索引鍵指定多個值，AWS 會使用邏輯 OR 操作評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授與該 IAM 使用者。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件索引鍵和服務特定的條件索引鍵。若要查看 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 AWS 全域條件內容索引鍵。

若要查看 Amazon Comprehend 條件金鑰清單，請參閱服務授權參考資料中的 Amazon Comprehend 條件金鑰。若要了解您可以使用條件金鑰的動作和資源，請參閱 Amazon Comprehend 定義的動作。

若要檢視以身分識別為基礎的政策範例，請參閱。Amazon Comprehend 基於身份的政策示例

在 Amazon Comprehend ACL

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

ABAC 與 Amazon Comprehend

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在 AWS 中，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色)，以及許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件索引鍵，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件索引鍵，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

如需標記 Amazon Comprehend 資源的詳細資訊，請參閱。標記您的 資源

使用臨時登入資料搭 Amazon Comprehend

支援臨時憑證	是

您使用臨時憑證進行登入時，某些 AWS 服務 無法運作。如需詳細資訊，包括那些 AWS 服務 搭配臨時憑證運作，請參閱《IAM 使用者指南》中的可搭配 IAM 運作的 AWS 服務。

如果您使用使用者名稱和密碼之外的任何方法登入 AWS Management Console，則您正在使用臨時憑證。例如，當您使用公司的單一登入(SSO)連結存取 AWS 時，該程序會自動建立臨時憑證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱 IAM 使用者指南中的切換至角色 (主控台)。

您可使用 AWS CLI 或 AWS API，手動建立臨時憑證。接著，您可以使用這些臨時憑證來存取 AWS。AWS 建議您動態產生臨時憑證，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Amazon Comprehend 的轉發存取工作階段

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在 AWS 中執行動作時，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

Amazon Comprehend 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務 服務。

> **Warning**
>
> 變更服務角色的許可可可可可能會中斷 Amazon 功能。只有在 Amazon Comprehend 提供指導時，才能編輯服務角色。

若要使用 Amazon Comprehend 非同步操作，您必須授與 Amazon Comprehend 存取權限，以存取包含您的文件收集的 Amazon S3 儲存貯體。您可以透過信任政策在帳戶中建立資料存取角色，以信任 Amazon Comprehend 服務主體來達到此目的。

如需政策範例，請參閱 非同步作業所需的角色型權限

Amazon Comprehend 的服務連結角色

支援服務連結角色。	否

服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的 AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amazon Comprehend 基於身份的政策示例

依預設，使用者和角色沒有建立或修改 Amazon Comprehend 資源的權限。他們也無法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 執行任務。若要授與使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Amazon Comprehend 定義的動作和資源類型的詳細資訊，包括每個資源類型的 ARN 格式，請參閱服務授權參考中的 Amazon Comprehend 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 Amazon Comprehend 制台
- 允許使用者檢視他們自己的許可
- 執行文件分析動作所需的權限
- 使用 KMS 加密所需的權限
- AWS亞馬遜管理 (預先定義) 政策
- 非同步作業所需的角色型權限
- 允許所有 Amazon Comprehend 動作的許可
- 允許主題建模動作的權限

- 自訂非同步分析工作所需的權限

政策最佳實務

以身分識別為基礎的政策可決定某人是否可以在您的帳戶中建立、存取或刪除 Amazon Comprehend 資源。這些動作可能會讓您的 AWS 帳戶 產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並朝向最低權限許可的目標邁進：如需開始授予許可給使用者和工作負載，請使用 AWS 受管政策，這些政策會授予許可給許多常用案例。它們可在您的 AWS 帳戶 中使用。我們建議您定義特定於使用案例的 AWS 客戶管理政策，以便進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的 AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授予對服務動作的存取權，前提是透過特定 AWS 服務 (例如 AWS CloudFormation)使用條件。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多重要素驗證 (MFA)：如果存在需要 AWS 帳戶中 IAM 使用者或根使用者的情況，請開啟 MFA 提供額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

有關 IAM 中最佳實務的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 最佳安全實務。

使用 Amazon Comprehend 制台

若要存取 Amazon Comprehend 主控台，您必須擁有一組最低限度的許可。這些許可必須允許您列 Amazon Comprehend 和檢視您的. AWS 帳戶 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許其最基本主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

對於 Amazon Comprehend 主控台的最低許可，您可以將*ComprehendReadOnly*AWS受管政策附加到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

若要使用 Amazon Comprehend 主控台，您還需要下列政策所示動作的許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
      "Action": [
          "iam:ListRoles",
          "iam:GetRole",
          "s3:ListAllMyBuckets",
          "s3:ListBucket",
          "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": "*"
  }
  ]
  }
```

Amazon Comprehend 台需要這些額外的許可，原因如下：

- `iam`列出您帳戶可用的 IAM 角色的許可。
- `s3`存取 Amazon S3 儲存貯體和包含用於主題建模之資料的物件的許可。

使用主控台建立非同步批次任務或主題建模工作時，您可以選擇讓主控台為您的工作建立 IAM 角色。若要建立 IAM 角色，使用者必須獲得下列其他許可，才能建立 IAM 角色和政策，以及將政策附加到角色：

```
{
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Action":
          [
          "iam:CreateRole",
          "iam:CreatePolicy",
          "iam:AttachRolePolicy"
```

```
          ],
        "Effect": "Allow",
        "Resource": "*"
      },
      {
        "Action":
          [
          "iam:PassRole"
          ],
        "Effect": "Allow",
        "Resource": "arn:aws:iam::*:role/*Comprehend*"
      }
    ]
  }
```

Amazon Comprehend 台需要這些額外的許可，原因如下：

- iam建立角色和策略以及附加角色和策略的權限。此iam:PassRole動作可讓主控台將角色傳遞給 Amazon Comprehend。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台上，或是使用 AWS CLI 或 AWS API 透過編寫程式的方式完成此動作的許可。

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

執行文件分析動作所需的權限

下列範例政策授予使用 Amazon Comprehend 文件分析動作的許可：

```
{
 "Version": "2012-10-17",
 "Statement": [{
    "Sid": "AllowDetectActions",
    "Effect": "Allow",
    "Action": [
              "comprehend:DetectEntities",
              "comprehend:DetectKeyPhrases",
              "comprehend:DetectDominantLanguage",
              "comprehend:DetectSentiment",
              "comprehend:DetectTargetedSentiment",
              "comprehend:DetectSyntax",
              "textract:DetectDocumentText",
              "textract:AnalyzeDocument"
           ],
    "Resource": "*"
    }
 ]
}
```

該政策具有一個聲明，授予使
用DetectEntities、DetectKeyPhrases、DetectDominantLanguageDetectTargetedSentiment

和DetectSyntax動作的權限。該政策聲明還授予使用兩種 Amazon Textract API 方法的許可。Amazon Comprehend 調用這些方法從圖像文件和掃描的 PDF 文檔中提取文本。您可以為從未針對這些類型的輸入檔案執行自訂推論的使用者移除這些權限。

具有此政策的使用者將無法在您的帳戶中執行批次處理動作或非同步動作。

此政策不指定 Principal 元素，因為您不會在以身分為基礎的政策中，指定取得許可的委託人。當您將政策連接至使用者時，這名使用者是隱含委託人。當您將許可政策連接至 IAM 角色，該角色的信任政策中所識別的委託人即取得許可。

如需顯示所有 Amazon Comprehend API 動作及其套用到的資源的表格，請參閱服務授權參考中適用於 Amazon Comprehend 的動作、資源和條件金鑰。

使用 KMS 加密所需的權限

若要在非同步任務中完全使用 Amazon 金鑰管理服務 (KMS) 進行資料和任務加密，您需要授與下列政策中顯示的動作的許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Action": [
            "kms:CreateGrant"
        ],
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDatakey"
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:ViaService": [
                    "s3.region.amazonaws.com"
                ]
            }
        }
    }
```

```
  ]
}
```

當您使用 Amazon Comprehend 建立非同步任務時，您可以使用存放在 Amazon S3 上的輸入資料。使用 S3 時，您可以選擇加密儲存的資料，這些資料是由 S3 加密，而不是由 Amazon Comprehend 加密。如果您提供將原始輸入資料加密到 Amazon Comprehend 任務所使用之資料存取角色的金鑰的`kms:Decrypt`權限，我們就可以解密和讀取該加密的輸入資料。

您也可以選擇使用 KMS 客戶管理金鑰 (CMK) 來加密 S3 上的輸出結果，以及任務處理期間使用的儲存磁碟區。執行此操作時，您可以對這兩種類型的加密使用相同的 KMS 金鑰，但這不是必要的。建立工作以指定用於輸出加密和磁碟區加密的金鑰時，會有不同的欄位可供使用，您甚至可以使用不同帳戶的 KMS 金鑰。

使用 KMS 加密時，需要`kms:CreateGrant`權限才能進行磁碟區加密，輸出資料加密需要`kms:GenerateDataKey`權限。對於讀取加密的輸入 (就像輸入資料已由 Amazon S3 加密時一樣)，需要`kms:Decrypt`許可。IAM 角色需要視需要授予這些許可。但是，如果金鑰來自與目前使用中不同的帳戶，則該 kms 金鑰的 KMS 金鑰原則也必須將這些權限授與工作的資料存取角色。

AWS亞馬遜管理 (預先定義) 政策

AWS 透過提供獨立的 IAM 政策來解決許多常用案例，這些政策由 AWS 所建立與管理。這些 AWS 受管政策授予常見使用案例的必要許可，讓您免於查詢需要哪些許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

下列AWS受管政策 (您可以附加到帳戶中的使用者) 是 Amazon Comprehend 特有的：

- ComprehendFullAccess— 授予對 Amazon Comprehend 資源的完整存取權，包括執行主題建模任務。包括列出和取得 IAM 角色的權限。
- ComprehendReadOnly— 授予執行除`StartDominantLanguageDetectionJob`、`StartEntitiesDetectionJob`、、和之外的所有 Amazon Comprehend 動作的權`StartSentimentDetectionJob`限`StartTargetedSentimentDetectionJob`。`StartKeyPhrase`
  `StartTopicsDetectionJob`

您必須將下列額外政策套用至將使用 Amazon Comprehend 的任何使用者：

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

```
      [
        {
          "Action":
            [
              "iam:PassRole"
            ],
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/*Comprehend*"
        }
      ]
  }
```

您可以登入 IAM 主控台並在該處搜尋特定政策，以檢閱受管許可政策。

當您使用 AWS SDK 或 AWS CLI 時，這些原則會運作。

您也可以建立自己的自訂 IAM 政策，以允許 Amazon Comprehend 動作和資源的許可。您可以將這些自訂原則附加到需要這些權限的使用者、群組或角色。

非同步作業所需的角色型權限

若要使用 Amazon Comprehend 非同步操作，您必須授與 Amazon Comprehend 存取權限，以存取包含您的文件收集的 Amazon S3 儲存貯體。您可以透過信任政策在帳戶中建立資料存取角色，以信任 Amazon Comprehend 服務主體來達到此目的。如需有關建立角色的詳細資訊，請參閱《AWSIdentity and Access Management 使用者指南》中的建立角色以將權限委派給AWS服務。

以下顯示您建立之角色的信任原則範例。若要協助避免混淆的副手，您可以使用一或多個全域條件內容索引鍵來限制權限範圍。將 `aws:SourceAccount` 值設定為您的帳戶 ID。如果您使用`ArnEquals`條件，請將`aws:SourceArn`值設定為工作的 ARN。使用萬用字元作為 ARN 中的任務編號，因為 Amazon Comprehend 會在任務建立過程中產生此號碼。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "comprehend.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringEquals": {
            "aws:SourceAccount": "111122223333"
```

```
          },
          "ArnEquals": {
            "aws:SourceArn": "arn:aws:comprehend:us-west-2:111122223333:pii-entities-
detection-job/*"
          }
        }
      }
    ]
 }
```

建立角色之後，請為該角色建立存取原則。這應該將 Amazon S3 GetObject 和PutObject許可授予包含您輸入資料的 Amazon S3 儲存貯體，以及 Amazon S3 輸出資料儲存貯體的ListBucket權限。

允許所有 Amazon Comprehend 動作的許可

註冊後AWS，您可以建立管理員使用者來管理您的帳戶，包括建立使用者和管理其權限。

您可以選擇建立具有所有 Amazon Comprehend 動作許可的使用者 (將此使用者視為服務特定的管理員)，以便與 Amazon Comprehend 合作。您可以將以下許可政策附加到此使用者。

```
{
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Sid": "AllowAllComprehendActions",
        "Effect": "Allow",
        "Action":
        [
            "comprehend:*",
            "iam:ListRoles",
            "iam:GetRole",
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:GetBucketLocation",
            "iam:CreateRole",
            "iam:CreatePolicy",
            "iam:AttachRolePolicy",
            "kms:CreateGrant",
            "kms:Decrypt",
            "kms:GenerateDatakey"
      ],
      "Resource": "*"
```

```
    },
    {
      "Action":
        [
          "iam:PassRole"
        ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/*Comprehend*"
    }
  ]
}
```

您可以透過下列方式修改這些加密權限：

- 若要讓 Amazon Comprehend 能夠分析存放在加密 S3 儲存貯體中的文件，IAM 角色必須具有該`kms:Decrypt`權限。
- 若要讓 Amazon Comprehend 能夠加密存放在連接到處理分析任務之運算執行個體的儲存磁碟區上的文件，IAM 角色必須具有權限。`kms:CreateGrant`
- 若要讓 Amazon Comprehend 能夠加密其 S3 儲存貯體中的輸出結果，IAM 角色必須具有權限。`kms:GenerateDataKey`

允許主題建模動作的權限

下列許可政策授予使用者執行 Amazon Comprehend 主題建模操作的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [{
  "Sid": "AllowTopicModelingActions",
  "Effect": "Allow",
  "Action": [
            "comprehend:DescribeTopicsDetectionJob",
            "comprehend:ListTopicsDetectionJobs",
            "comprehend:StartTopicsDetectionJob",
         ],
        "Resource": "*"
        ]
    }
  ]
  }
```

自訂非同步分析工作所需的權限

> **Important**
>
> 如果您有限制模型存取權限的 IAM 政策，您將無法使用自訂模型完成推論工作。您的 IAM 政策應更新為具有用於自訂非同步分析工作的萬用字元資源。

如果您使用StartDocumentClassificationJob 和 StartEntitiesDetectionJobAPI，則必須更新 IAM 政策，除非您目前使用萬用字元做為資源。如果您使用的是預先訓練的模型，則不會影響您，也不需要進行任何變更。StartEntitiesDetectionJob

下列範例原則包含過期的參照。

```
{
    "Action": [
        "comprehend:StartDocumentClassificationJob",
        "comprehend:StartEntitiesDetectionJob",
    ],
    "Resource": [
        "arn:aws:comprehend:us-east-1:123456789012:document-classifier/myClassifier",
        "arn:aws:comprehend:us-east-1:123456789012:entity-recognizer/myRecognizer"
    ],
    "Effect": "Allow"
}
```

這是您需要用來成功執行 StartDocumentClassificationJob 和的更新原則。 StartEntitiesDetectionJob

```
{
    "Action": [
        "comprehend:StartDocumentClassificationJob",
        "comprehend:StartEntitiesDetectionJob",
    ],
    "Resource": [
        "arn:aws:comprehend:us-east-1:123456789012:document-classifier/myClassifier",
        "arn:aws:comprehend:us-east-1:123456789012:document-classification-job/*",
        "arn:aws:comprehend:us-east-1:123456789012:entity-recognizer/myRecognizer",
        "arn:aws:comprehend:us-east-1:123456789012:entities-detection-job/*"
    ],
    "Effect": "Allow"
}
```

AWS亞馬遜管理政策

若要新增許可給使用者、群組和角色，使用 AWS 受管政策比自己撰寫政策更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用 AWS 受管政策。這些政策涵蓋常見的使用案例，並可在您的 AWS 帳戶 中使用。如需有關 AWS 受管政策的詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 服務會維護和更新 AWS 受管政策。您無法更改 AWS 受管政策中的許可。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管政策中移除許可，因此政策更新不會破壞您現有的許可。

此外，AWS 支援跨越多項服務之任務職能的受管政策。例如，ReadOnlyAccess AWS 受管政策提供針對所有 AWS 服務和資源的唯讀存取權限。當服務啟動新功能時，AWS 會為新的操作和資源新增唯讀許可。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的 AWS 受管政策。

AWS 受管政策： ComprehendFullAccess

此政策授予對 Amazon Comprehend 資源的完整存取權，包括執行主題建模任務。此政策也授予 Amazon S3 儲存貯體和 IAM 角色的清單和取得許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "comprehend:*",
                "iam:GetRole",
                "iam:ListRoles",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

AWS 受管政策： ComprehendReadOnly

此政策授予唯讀許可以執行所有 Amazon Comprehend 動作，但下列動作除外：

- `StartDominantLanguageDetectionJob`
- `StartEntitiesDetectionJob`
- `StartKeyPhrasesDetectionJob`
- `StartSentimentDetectionJob`
- `StartTargetedSentimentDetectionJob`
- `StartTopicsDetectionJob`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "comprehend:BatchDetectDominantLanguage",
                "comprehend:BatchDetectEntities",
                "comprehend:BatchDetectKeyPhrases",
                "comprehend:BatchDetectSentiment",
                "comprehend:BatchDetectSyntax",
                "comprehend:ClassifyDocument",
                "comprehend:ContainsPiiEntities",
                "comprehend:DescribeDocumentClassificationJob",
                "comprehend:DescribeDocumentClassifier",
                "comprehend:DescribeDominantLanguageDetectionJob",
                "comprehend:DescribeEndpoint",
                "comprehend:DescribeEntitiesDetectionJob",
                "comprehend:DescribeEntityRecognizer",
                "comprehend:DescribeKeyPhrasesDetectionJob",
                "comprehend:DescribePiiEntitiesDetectionJob",
                "comprehend:DescribeResourcePolicy",
                "comprehend:DescribeSentimentDetectionJob",
                "comprehend:DescribeTargetedSentimentDetectionJob",
                "comprehend:DescribeTopicsDetectionJob",
                "comprehend:DetectDominantLanguage",
                "comprehend:DetectEntities",
                "comprehend:DetectKeyPhrases",
```

```
                "comprehend:DetectPiiEntities",
                "comprehend:DetectSentiment",
                "comprehend:DetectSyntax",
                "comprehend:ListDocumentClassificationJobs",
                "comprehend:ListDocumentClassifiers",
                "comprehend:ListDocumentClassifierSummaries",
                "comprehend:ListDominantLanguageDetectionJobs",
                "comprehend:ListEndpoints",
                "comprehend:ListEntitiesDetectionJobs",
                "comprehend:ListEntityRecognizers",
                "comprehend:ListEntityRecognizerSummaries",
                "comprehend:ListKeyPhrasesDetectionJobs",
                "comprehend:ListPiiEntitiesDetectionJobs",
                "comprehend:ListSentimentDetectionJobs",
                "comprehend:ListTargetedSentimentDetectionJobs",
                "comprehend:ListTagsForResource",
                "comprehend:ListTopicsDetectionJobs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Amazon Comprehend 受管政策的AWS更新

檢視有關 Amazon Comprehend AWS 受管政策更新的詳細資訊，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動警示，請訂閱 Amazon Comprehend 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
ComprehendReadOnly – 更新現有政策	Amazon Comprehend 現在允許在政策中的comprehend:DescribeTargetedSentimentDetectionJob 和comprehend:ListTargetedSentimentDetectionJobs 動作 ComprehendReadOnly	2022年3月30日

變更	描述	日期
ComprehendReadOnly – 更新現有政策	Amazon Comprehend 現在允許在政策中`comprehend:DescribeResourcePolicy` 執行操作 ComprehendReadOnly	2022年2月2日
ComprehendReadOnly – 更新現有政策	Amazon Comprehend 現在允許在政策中的`ListDocumentClassifierSummaries` 和`ListEntityRecognizerSummaries` 動作 ComprehendReadOnly	2021 年 9 月 21 日
ComprehendReadOnly – 更新現有政策	Amazon Comprehend 現在允許在政策中`ContainsPIIEntities` 執行操作 ComprehendReadOnly	2021 年 3 月 26 日
Amazon Comprehend 開始跟踪變化	亞馬遜開始追蹤其AWS受管政策的變更。	2021 年 3 月 1 日

疑難排解 Amazon Comprehend 身分和存取

使用下列資訊可協助您診斷和修正使用 Amazon Comprehend 和 IAM 時可能會遇到的常見問題。

主題

- 我沒有授權在 Amazon Comprehend 執行操作
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪問我AWS 帳戶的 Amazon Comprehend 資源

我沒有授權在 Amazon Comprehend 執行操作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 comprehend:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 comprehend:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 Mateo 政策，允許他使用 comprehend:*GetWidget* 動作存取 *my-example-widget* 資源。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon Comprehend。

有些 AWS 服務 允許您傳遞現有的角色至該服務，而無須建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者marymajor嘗試使用主控台在 Amazon Comprehend 中執行動作時，就會發生下列範例錯誤。但是，該動作要求服務具備服務角色授與的許可。Mary 沒有將角色傳遞至該服務的許可。

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

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我想允許我以外的人訪問我AWS 帳戶的 Amazon Comprehend 資源

您可以建立一個角色，讓其他帳戶中的使用者或您的組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了 Amazon Comprehend 是否支援這些功能，請參閱。Amazon Comprehend 如何與 IAM 合作

- 如需了解如何存取您擁有的所有 AWS 帳戶 所提供的資源，請參閱《IAM 使用者指南》中的將存取權提供給您所擁有的另一個 AWS 帳戶 中的 IAM 使用者。
- 如需了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方擁有的 AWS 帳戶。
- 如需了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策的差異。

使用日誌記錄 Amazon Comprehend API 調用 AWS CloudTrail

Amazon Comprehend 與這項服務整合在一起AWS CloudTrail，可提供使用者、角色或服務在 Amazon Comprehend 中所採取的動作記錄的AWS服務。 CloudTrail 將 Amazon Comprehend 的 API 呼叫擷取為事件。擷取的呼叫包括來自亞馬遜主控台的呼叫，以及對 Amazon Comprehend API 操作的程式碼呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 Amazon Comprehend 的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷向 Amazon Comprehend 提出的請求、提出請求的來源 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，包括如何設定和啟用它，請參閱AWS CloudTrail使用者指南。

Amazon Comprehend 信息 CloudTrail

CloudTrail 在您創建帳戶AWS 帳戶時啟用。當 Amazon Comprehend 中發生受支援的事件活動時，該活動會與事件歷史記錄中的其他AWS服務 CloudTrail 事件一起記錄在事件中。您可以檢視、搜尋和下載 AWS 帳戶 的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

如需您AWS 帳戶的事件的持續記錄 (包括 Amazon Comprehend 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他AWS服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail

- 從多個區域接收 CloudTrail 日誌文件並從多個帳戶接收 CloudTrail 日誌文件

Amazon Comprehend 支援將下列動作記錄為記 CloudTrail 錄檔中的事件：

- BatchDetectDominantLanguage
- BatchDetectEntities
- BatchDetectKeyPhrases
- BatchDetectSentiment
- BatchDetectSyntax
- ClassifyDocument
- CreateDocumentClassifier
- CreateEndpoint
- CreateEntityRecognizer
- DeleteDocumentClassifier
- DeleteEndpoint
- DeleteEntityRecognizer
- DescribeDocumentClassificationJob
- DescribeDocumentClassifier
- DescribeDominantLanguageDetectionJob
- DescribeEndpoint
- DescribeEntitiesDetectionJob
- DescribeEntityRecognizer
- DescribeKeyPhrasesDetectionJob
- DescribePiiEntitiesDetectionJob
- DescribeSentimentDetectionJob
- DescribeTargetedSentimentDetectionJob
- DescribeTopicsDetectionJob
- DetectDominantLanguage
- DetectEntities
- DetectKeyPhrases

- DetectPiiEntities
- DetectSentiment
- DetectSyntax
- ListDocumentClassificationJobs
- ListDocumentClassifiers
- ListDominantLanguageDetectionJobs
- ListEndpoints
- ListEntitiesDetectionJobs
- ListEntityRecognizers
- ListKeyPhrasesDetectionJobs
- ListPiiEntitiesDetectionJobs
- ListSentimentDetectionJobs
- ListTargetedSentimentDetectionJobs
- ListTagsForResource
- ListTopicsDetectionJobs
- StartDocumentClassificationJob
- StartDominantLanguageDetectionJob
- StartEntitiesDetectionJob
- StartKeyPhrasesDetectionJob
- StartPiiEntitiesDetectionJob
- StartSentimentDetectionJob
- StartTargetedSentimentDetectionJob
- StartTopicsDetectionJob
- StopDominantLanguageDetectionJob
- StopEntitiesDetectionJob
- StopKeyPhrasesDetectionJob
- StopPiiEntitiesDetectionJob
- StopSentimentDetectionJob
- StopTargetedSentimentDetectionJob
- StopTrainingDocumentClassifier

- StopTrainingEntityRecognizer
- TagResource
- UntagResource
- UpdateEndpoint

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 是否使用 root 使用者認證提出要求。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

範例：Amazon Comprehend 日誌檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範`ClassifyDocument`動作的 CloudTrail 記錄項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AROAICFHPEXAMPLE",
        "arn": "arn:aws:iam::12345678910:user/myadmin2",
        "accountId": "12345678910",
        "accessKeyId": "ASIA3VZEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-10-19T14:22:09Z",
                "mfaAuthenticated": "false"
            }
        }
    },
```

```
    "eventTime": "2023-10-19T17:31:20Z",
    "eventSource": "comprehend.amazonaws.com",
    "eventName": "ClassifyDocument",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "3.21.185.237",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0)
 Gecko/20100101 Firefox/115.0",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "fd916e66-caac-46c9-a1fc-81a0ef33e61b",
    "eventID": "535ca22b-b3a3-4c13-b2c5-bf51ab082794",
    "readOnly": false,
    "resources": [
        {
            "accountId": "12345678910",
            "type": "AWS::Comprehend::DocumentClassifierEndpoint",
            "ARN": "arn:aws:comprehend:us-east-2:12345678910:document-classifier-
endpoint/endpointExample"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "12345678910"
}
```


Amazon Comprehend 合規驗證

第三方稽核員會評估 Amazon Comprehend 的安全性和合規性，做為多個AWS合規計劃的一部分。這些計劃包括 PCI、FedRAMP、HIPAA 等等。您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱在 AWS Artifact 中下載報告。

您在使用 Amazon Comprehend 時的合規責任取決於資料的敏感度、公司的合規目標以及適用的法律和法規。 AWS提供下列資源以協助遵循法規：

- 安全與合規快速入門指南 – 這些部署指南討論在 AWS 上部署以安全及合規為重心基準環境的架構考量和步驟。
- HIPAA 安全與合規架構白皮書 – 本白皮書說明公司可如何運用 AWS 來建立 HIPAA 合規的應用程式。
- AWS 合規資源 – 這組手冊和指南可能適用於您的產業和位置。
- AWS Config – 此 AWS 服務可評定資源組態與內部實務、業界準則和法規的合規狀態。

- AWS Security Hub：此 AWS 服務可供您檢視 AWS 中的安全狀態，可助您檢查是否符合安全產業標準和最佳實務。

如需特定合規計劃範圍內的 AWS 服務清單，請參閱合規計劃內的 AWS 服務。如需一般資訊，請參閱 AWS 合規計劃。

Amazon Comprehend 中的韌性

AWS全球基礎架構是以 AWS 區域 s 和可用區域為基礎建置。 AWS 區域s 提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 s 和可用區域的詳細資訊，請參閱AWS全域基礎結構。

Amazon Comprehend 的基礎設施安全

作為一項受管服務，Amazon Comprehend 受到AWS全球網路安全的保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎架構安全性的最佳做法來設計您的AWS環境，請參閱安全性支柱架構良AWS好的架構中的基礎結構保護。

您可以使用AWS已發佈的 API 呼叫透過網路存取 Amazon Comprehend。用戶端必須支援下列項目：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 以產生暫時安全憑證以簽署請求。

指南和配額

除非另有說明，否則 Amazon Comprehend 的配額是以區域為單位。如果您的應用程式需要，您可以要求增加可調配額。如需配額的相關資訊以及要求增加配額，請參閱 AWS Service Quotas。

主題

支援地區

Amazon Comprehend 服務於下列區域提供： AWS

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (孟買)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- AWS GovCloud (美國西部)

根據預設，Amazon Comprehend 會在每個支援的區域中提供所有 API 操作。如需例外狀況，請參閱文件處理。

如需 API 端點的相關資訊，請參閱亞馬遜網路服務一般參考中的 Amazon Comprehend 區域和端點。

若要檢閱區域中目前的配額，或要求增加可調配額的配額，請開啟「Service Quotas」主控台。

內建模型的配額

Amazon Comprehend 提供內建模型供您分析 UTF-8 文字文件。Amazon Comprehend 提供使用內建模型的同步和非同步作業。

主題

- 即時 (同步) 分析
- 非同步分析

即時 (同步) 分析

本節說明使用內建模型進行即時分析的相關配額。

主題

- 單一文件操作
- 多重文件操作
- 即時 (同步) 要求的要求節流

單一文件操作

Amazon Comprehend API 提供了將單個文檔作為輸入的操作。下列配額適用於這些作業。

單一文件作業的一般配額

下列配額適用於偵測實體、關鍵片語或主要語言的即時分析。對於實體偵測，這些配額適用於內建模型的偵測。如需自訂實體偵測的資訊，請參閱中的配額自訂實體辨識 。

描述	名額 / 指引
最大文件大小	100 KB

單一文件作業的特定作業配額

下列配額適用於偵測情緒、目標情緒和語法的即時分析。

描述	名額 / 指引
最大文件大小	5 KB

多重文件操作

Amazon Comprehend API 提供批次操作，可透過單一 API 請求處理多個文件。下列配額適用於批次作業。

描述	名額 / 指引
最大文件大小	5 KB
每個請求的最大文件	25

如需有關使用批次文件作業的詳細資訊，請參閱多文檔同步處理。

即時 (同步) 要求的要求節流

亞馬遜套用動態節流到同步請求。如果系統處理頻寬可用，Amazon Comprehend 會逐漸增加其處理的請求數量。若要控制應用程式對同步 API 作業的使用，建議您在應用程式中開啟帳單警示或實作速率限制。

非同步分析

本節說明使用內建模型的非同步分析相關配額。

非同步 API 作業每個作業最多可支援 10 個作用中工作。若要檢視每個 API 作業的配額，請參閱 Amazon Comprehend 端點中的 Service Quotas 表和 Amazon Web Services 一般參考中的配額。

對於可調配額，您可以使用 Service Quotas 控制台申請增加配額。

主題

- 非同步作業的一般配額
- 非同步工作的特定作業配額
- 異步請求的請求節流

非同步作業的一般配額

您可以使用主控台或任何 API 作業執行非同步分析工`Start*`作。如需何時使用非同步作業的資訊，請參閱非同步批次處理。下列配額適用於大部分內建模型的 API `Start*` 作業。有關例外情況，請參閱非同步工作的特定作業配額。

描述	名額 / 指引
偵測實體、關鍵片語、PII 和語言的工作中，每個文件的大小上限	1 MB
請求中所有文件的最大總大小	5 GB
請求中所有檔案的最小總大小	500 位元組
檔案數目上限，每個檔案一個文件	1,000,000
最大總行數，每行一個文檔	1,000,000

非同步工作的特定作業配額

本節說明特定非同步作業的配額。如果未在下表中指定配額，則會套用一般配額值。

主題

- 情緒
- 目標情緒
- 事件
- 主題建模

情緒

您使用作業建立的非同步情緒工StartSentimentDetectionJob作具有下列配額。

描述	名額 / 指引
每個輸入文件的大小上限	5 KB

目標情緒

您使用作業建立的非同步目標情緒工StartTargetedSentimentDetectionJob作具有下列配額。

描述	名額 / 指引
支援的文件格式	UTF-8
工作中每個文件的大小上限	10 KB
工作中所有文件的大小上限	三百公釐
檔案數目上限，每個檔案一個文件	30,000
最大總行數，每行一個文件 (適用於請求中的所有檔案)	30,000

事件

您使用作業建立的非同步事件偵測工StartEventsDetectionJob作具有下列配額。

描述	配額
字元編碼	UTF-8
工作中所有檔案的總大小	50 MB
工作中每個文件的大小上限	10 KB
檔案數目上限，每個檔案一個文件	5,000
最大總行數，每行一個文檔（適用於請求中的所有文件）	5,000

主題建模

您使用作業建立的非同步主題模型工StartTopicsDetectionJob作具有下列配額。

描述	名額 / 指引
字元編碼	UTF-8

描述	名額 / 指引
要傳回的主題數目上限	100
一個檔案的最大檔案大小，每個檔案一個文件	100 MB

如需更多資訊，請參閱主題建模

異步請求的請求節流

每個非同步 API 作業支援每秒的請求數目上限 (每個區域、每個帳戶)，以及最多 10 個作用中工作。若要檢視每個 API 作業的配額，請參閱 Amazon Comprehend 端點中的 Service Quotas 表和 Amazon Web Services 一般參考中的配額。

對於可調配額，您可以使用 Service Quotas 控制台申請增加配額。

自訂模型的配額

您可以使用 Amazon Comprehend 建立自己的自訂模型，以進行自訂分類和自訂實體辨識。本節提供與訓練和使用自訂模型相關的準則和配額。如需自訂模型的詳細資訊，請參閱Amazon Comprehend 定制。

主題

- 一般配額
- 端點的配額
- 文件分類
- 自訂實體辨識

一般配額

Amazon Comprehend 會為您可以使用自訂模型進行分析的每種輸入文件類型設定一般大小配額。如需即時分析配額，請參閱即時分析的最大文件大小。如需非同步分析配額，請參閱非同步自訂分析的輸入。

每個非同步 API 作業支援每秒的請求數目上限 (每個區域、每個帳戶)，以及最多 10 個作用中工作。若要檢視每個 API 作業的配額，請參閱 Amazon Comprehend 端點中的 Service Quotas 表和 Amazon Web Services 一般參考中的配額。

對於可調配額，您可以使用 Service Quotas 控制台申請增加配額。

端點的配額

您可以建立端點以使用自訂模型執行即時分析。如需端點的相關資訊，請參閱管理 Amazon Comprehend 端點。

下列配額適用於端點。如需如何要求增加配額的相關資訊，請參閱 AWS Service Quotas。

描述	名額 / 指引
每個帳號每個區域的作用中端點數目上限	20
每個帳戶每個區域的推論單元數量上限	200
每個區域每個端點的推論單元數目上限	50
每個推論單位的最大輸送量 (字元)	每秒
每個推論單位 (文件) 的最大輸送量	每秒 2 次

文件分類

本節說明下列文件分類作業的準則與配額：

- 您從作業開始的分類器訓練工CreateDocumentClassifier作。
- 您從作業開始的非同步文件分類工StartDocumentClassificationJob作。
- 使用此ClassifyDocument作業的同步文件分類請求。

文件分類的一般配額

下表說明與訓練自訂分類器相關的一般配額。

描述	名額 / 指引
類別名稱的最大長度	五千個字元
類數（多類模式）	2—1,000

描述	名額 / 指引
類別數目 (多標籤模式)	2—100
註釋格式	
每個類的最小註釋數量（多類模式）	10
每個類的最小註釋數量（多標籤模式）	10
註釋的最小數目 (多標籤模式)	50
CSV 文件格式	
每班培訓文件的最少數量（多課程模式）	50
每個班級的最少培訓文件數量（多標籤模式）	10
最少培訓文件數量（多標籤模式）	50

純文字文件的分類

您可以使用純文字輸入文件建立和訓練純文字模型。Amazon Comprehend 提供即時和非同步操作，以使用純文字模型對純文字文件進行分類。

培訓

下表說明與使用純文字文件訓練自訂分類器相關的配額。

描述	名額 / 指引
訓練工作中所有檔案的總大小	5 GB
訓練自訂分類器的增強資訊清單檔案數目上限	5
每個增強資訊清單檔案的屬性名稱數目上限	5
屬性名稱的最大長度	63 個字元

即時 (同步) 分析

下表說明與即時純文字文件分類相關的配額。

描述	名額 / 指引
每個同步請求的最大文檔數量	1
文字文件大小上限 (UTF-8 編碼)	10 KB

非同步分析

下表說明與純文字文件非同步分類相關的配額。

描述	名額 / 指引
非同步工作中所有檔案的總大小	5 GB
一個檔案的最大檔案大小，每個檔案一個文件	10 MB
檔案數目上限，每個檔案一個文件	1,000,000
最大總行數，每行一個文檔（適用於請求中的所有文件）	1,000,000

半結構化文件的分類

本節說明半結構化文件之文件分類的準則與配額。若要分類半結構化文件，請使用您使用原生輸入文件訓練的原生文件模型。

使用半結構化文件訓練原生文件模型

下表說明與使用半結構化文件 (例如 PDF 文件、Word 文件和影像檔) 訓練自訂分類器相關的配額。

描述	名額 / 指引
所有文件的最大頁數	10,000
最大註釋檔案大小 (所有 CSV 檔案大小合併)	5 MB

描述	名額 / 指引
文件語料庫大小 (訓練與測試文件)	10 GB
訓練和測試檔案的檔案大小	
圖像文件大小（JPG，PNG，TIFF）。	1 個字節-10 MB。 TIFF 檔案：最多一頁。
PDF 文件的頁面大小	1 個字節-10 MB
Word 文件的頁面大小	1 個字節-10 MB
Amazon Textract 取 API 輸出 JSON 大小	1 個字節-1 MB

即時 (同步) 分析

本節說明與半結構化文件的即時分類相關配額。

下表顯示輸入文件的最大檔案大小。對於所有輸入文件類型，輸入檔案最多為一頁，不超過 10,000 個字元。

檔案類型	大小上限	最大尺寸（控制台）
UTF-8 文字文件	10 KB	10 KB
PDF 文件	10 MB	5 MB
文字文件	10 MB	5 MB
影像檔	10 MB	5 MB
Amazon Textract 取 API 輸出大小	1 MB	N/A

非同步分析

下表說明與非同步分類半結構化文件相關的配額。

描述	名額 / 指引
工作所有輸入文件的最大頁數	25,000
文件語料庫大小	25 GB
影像檔案大小 (JPG、PNG 或 TIFF 格式)	1 個字節-10 MB。 TIFF 檔案：最多一頁。
PDF 文件的頁面大小	1 個字節-10 MB
Word 文件的頁面大小	1 個字節-10 MB
Textract 取 API 輸出 JSON 大小	1 個字節-1 MB。

自訂實體辨識

本節說明下列自訂實體辨識作業的準則和配額：

- 從作業開始的實體辨識器訓練工CreateEntityRecognizer作。
- 從作業開始的非同步實體辨識工StartEntitiesDetectionJob作。
- 使用DetectEntities操作的同步實體識別請求。

純文字文件的自訂實體辨識

Amazon Comprehend 提供非同步和同步操作，可透過自訂實體辨識器分析純文字文件。

培訓

本節說明與訓練自訂實體辨識器以分析純文字文件相關的配額。若要訓練模型，您可以提供實體清單或一組已註解的文字文件。

下表說明與使用實體清單訓練模型相關的配額。

描述	名額 / 指引
每個模型的圖元數	1—25

描述	名額 / 指引
文件大小 UTF-8	千字节
實體清單中的項目數	1 百萬
條目列表中個別條目的長度（後排）	1—5,000
實體列表語料庫大小（所有文檔以純文本組合）	5 KB

下表說明與使用註解文字文件訓練模型相關的配額。

描述	名額 / 指引
每個模型/自訂實體識別器的實體數量	1—25
文件大小 UTF-8	千字节
文件數目 (請參閱純文字註釋)	3—200,000
文件語料庫大小 (所有文件以純文字組合)	5 KB
每個圖元的最小註釋數	25

即時 (同步) 分析

下表說明與純文字文件的即時分析相關配額。

描述	名額 / 指引
每個同步請求的最大文檔數量	1
文字文件大小上限 (UTF-8 編碼)	5 KB

非同步分析

下表說明與純文字文件的非同步實體辨識相關的配額。

描述	名額/指引
文件大小 UTF-8	1 個字節-1 MB
檔案數目上限，每個檔案一個文件	1,000,000
最大總行數，每行一個文檔（適用於請求中的所有文件）	1,000,000
文件語料庫大小 (所有文件以純文字組合)	1 個字節 —5 GB

半結構化文件的自訂實體辨識

Amazon Comprehend 提供非同步和同步操作，可透過自訂實體辨識器分析半結構化文件。您必須使用帶註解的 PDF 文件來訓練模型。

培訓

下表說明與訓練自訂實體辨識器 (CreateEntityRecognizer) 以分析半結構化文件相關的配額。

描述	名額/指引
每個模型/自訂實體識別器的實體數量	1—25
註解檔案大小上限 UTF-8	5 MB
文件數	250—10,000
文件語料庫大小 (所有文件以純文字組合)	5 千兆 — 1 GB
每個圖元的最小註釋數	100
訓練自訂實體辨識器的增強資訊清單檔案數目上限	5
每個增強資訊清單檔案的屬性名稱數目上限	5
屬性名稱的最大長度	63 個字元

即時 (同步) 分析

本節說明與半結構化文件的即時分析相關配額。

下表顯示輸入文件的最大檔案大小。對於所有輸入文件類型，輸入檔案最多為一頁，不超過 10,000 個字元。

檔案類型	大小上限	最大尺寸（控制台）
UTF-8 文字文件	10 KB	10 KB
PDF 文件	10 MB	5 MB
文字文件	10 MB	5 MB
影像檔	10 MB	5 MB
文 Textract 輸出檔案	1 MB	N/A

非同步分析

本節說明非同步分析半結構化文件的配額。

描述	名額/指引
影像大小 (JPG 或 PNG)	1 個字節-10 MB
影像尺寸	1 個字節-10 MB。最多一頁。
文件大小	1 個字節-50 MB
文件大小 (DOCX)	1 個字節 —5 MB
文件大小 UTF-8	1 個字節-1 MB
檔案數目上限，每個檔案一個文件 (影像檔案或 PDF/Word 文件不允許每行一個文件)	500
PDF 文件或 DOCX 文件的最大頁數	100
文本提取後的文檔語料庫大小（純文本，所有文件合併）	1 個字節 —5 GB

如需有關影像限制的詳細資訊，請參閱 Amazon Textract 中的硬性限制

飛輪配額

使用飛輪管理自訂模型版本的訓練和追蹤，以進行自訂分類和自訂實體辨識。若要取得有關飛輪的更多資訊，請參閱飞轮。

飛輪的一般配額

以下配額適用於飛輪和飛輪迭代。

描述	名額/指引
最大飛輪數	50
「建立」狀態下的最大飛輪數	10
每個飛輪訓練資料集的最大數量	50
每個飛輪測試資料集的最大數量	50
具有擷取狀態的資料集數目上限	10
每個帳戶進行中的飛輪迭代次數上限	10

自訂分類模型的資料集配額

當您擷取與自訂分類模型相關聯之飛輪的資料集時，會套用下列配額。

描述	名額/指引
每個班級的最少培訓文件數量（多標籤模式）	50
培訓文件的最大數量	1,000,000
資料集大小下限	500 位元組
資料集大小上限	5 GB
一個檔案的最大檔案大小，每個檔案一個文件	10 MB

自訂實體辨識模型的資料集配額

當您擷取與自訂實體辨識模型相關聯之飛輪的資料集時，會套用下列配額。

描述	名額/指引
最大文件大小	5 KB
培訓文件的最少數量	3
培訓文件的最大數量	200,000
每個圖元的最小註釋數	25
資料集大小上限	200 MB

教學課程和其他資源

Amazon Comprehend 的教學和其他資源。

主題

- 教學：使用亞馬遜分析客戶評論中的見解
- 使用 Amazon S3 物件 Lambda 存取點取得個人識別資訊 (PII)
- 解決方案：使用 Amazon Comprehend 和分析文本 OpenSearch

教學：使用亞馬遜分析客戶評論中的見解

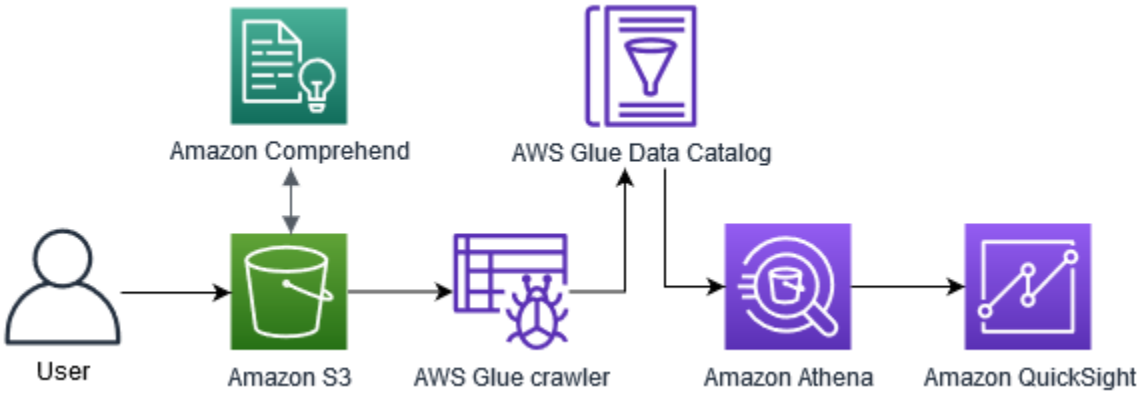
本教學將說明如何將 Amazon Comprehend 與 Amazon 簡單儲存服務AWS Glue、和 Amazon 搭配使用 Amazon Athena， QuickSight以深入瞭解您的文件。Amazon Comprehend 可以從非結構化文字擷取情緒 (文件的情緒) 和實體 (人員、組織、事件、日期、產品、地點、數量和標題的名稱)。

例如，您可以從客戶評論中獲得可行的見解。在本教程中，您將分析有關小說的客戶評論樣本數據集。您可以使用 Amazon Comprehend 情緒分析來判斷客戶對這本小說的感覺是正面還是負面。您也可以使用 Amazon Comprehend 實體分析來探索重要實體的提及，例如相關小說或作者。按照本教程後，您可能會發現 50% 以上的評論是正面的。您可能還會發現客戶正在比較作者並表達對其他經典小說的興趣。

在此自學課程中，您將完成下列工作：

- 在亞馬遜簡單儲存服務 (Amazon S3) 中存放評論的範例資料集。Amazon 簡單存儲服務是一種對象存儲服務。
- 使用 Amazon Comprehend 分析審核文件中的情緒和實體。
- 使用AWS Glue爬行者程式將分析結果儲存在資料庫中。 AWS Glue是一種擷取、轉換和載入 (ETL) 服務，可讓您編目和清理資料以進行分析。
- 執行Amazon Athena查詢以清除資料。 Amazon Athena是無伺服器互動式查詢服務。
- 使用 Amazon 中的資料建立視覺效果 QuickSight。Amazon QuickSight 是一種無伺服器商業智慧工具，可從您的資料擷取洞見。

下圖顯示工作流程。



完成此自學課程的預估時間：1 小時

預估費用：本教學課程中的某些動作會向您收取費用AWS 帳戶。如需上述各項服務費用的相關資訊，請參閱下列定價頁面。

- Amazon S3 定價
- Amazon Comprehend 定價
- AWS Glue 定價
- Amazon Athena 定價
- Amazon QuickSight 定價

主題

必要條件

為了完成本教學，您需要以下項目：

- AWS 帳戶。若要取得有關設定的資訊AWS 帳戶，請參閱設定。

- IAM 實體 (使用者、群組或角色)。若要了解如何為您的帳戶設定使用者和群組，請參閱 IAM 使用者指南中的入門教學課程。
- 下列附加至您的使用者、群組或角色的權限原則。此原則會授與完成此教學課程所需的一些權限。下一個先決條件說明您需要的其他權限。

```
{
  "Version": "2012-10-17",
  "Statement":
  [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action":
      [
        "comprehend:*",
        "ds:AuthorizeApplication",
        "ds:CheckAlias",
        "ds:CreateAlias",
        "ds:CreateIdentityPoolDirectory",
        "ds:DeleteDirectory",
        "ds:DescribeDirectories",
        "ds:DescribeTrusts",
        "ds:UnauthorizeApplication",
        "iam:AttachRolePolicy",
        "iam:CreatePolicy",
        "iam:CreatePolicyVersion",
        "iam:CreateRole",
        "iam:DeletePolicyVersion",
        "iam:DeleteRole",
        "iam:DetachRolePolicy",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetRole",
        "iam:ListAccountAliases",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy",
        "iam:ListPolicies",
        "iam:ListPolicyVersions",
        "iam:ListRoles",
        "quicksight:*",
        "s3:*",
        "tag:GetResources"
      ],
```

```
      "Resource": "*"
    },
    {
      "Action":
      [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource":
      [
        "arn:aws:iam::*:role/*Comprehend*"
      ]
    }
  ]
}
```

使用先前的政策建立 IAM 政策，並將其附加到您的群組或使用者。如需建立 IAM 政策的相關資訊，請參閱 IAM 使用者指南中的建立 IAM 政策。如需附加 IAM 政策的相關資訊，請參閱 IAM 使用者指南中的新增和移除 IAM 身分許可。

- 附加至 IAM 群組或使用者的受管政策。除了先前的策略之外，您還必須將下列AWS受管理策略附加到您的群組或使用者：
  - `AWSGlueConsoleFullAccess`
  - `AWSQuicksightAthenaAccess`

  這些受管政策授予您使用AWS GlueAmazon Athena、和 Amazon 的許可 QuickSight。如需附加 IAM 政策的相關資訊，請參閱 IAM 使用者指南中的新增和移除 IAM 身分許可。

第 1 步：將文檔添加到 Amazon S3

在開始 Amazon Comprehend 分析任務之前，您需要將客戶評論的範例資料集存放在亞馬遜簡單儲存服務 (Amazon S3) 中。Amazon S3 將您的資料託管在稱為儲存貯體的容器中。Amazon Comprehend 可以分析儲存在值區中的文件，並將分析結果傳送至儲存貯體。在此步驟中，您會建立 S3 儲存貯體、在儲存貯體中建立輸入和輸出資料夾，然後將範例資料集上傳到儲存貯體。

主題

- 必要條件
- 下載範例資料
- 建立 Amazon S3 儲存貯體

- (僅限主控台) 建立資料夾
- 上傳輸入數據

必要條件

開始之前，請檢閱教學：使用亞馬遜分析客戶評論中的見解並完成先決條件。

下載範例資料

下面的示例數據集包含從較大的數據集「Amazon 評論-完整」，該數據集與「文本分類的字符級卷積網絡」一文一起發布的 Amazon 評論（翔章等人，2015）。將資料集下載到您的電腦。

若要取得範例資料

1. 將 zip 文件下載到您的計算機tutorial-reviews-data。
2. 解壓縮您的計算機上的 zip 文件。有兩個檔案。該文件`THIRD_PARTY_LICENSES.txt`是由翔張等人發布的數據集的開源許可證。檔案`amazon-reviews.csv`是您在自學課程中分析的資料集。

建立 Amazon S3 儲存貯體

下載範例資料集後，請建立 Amazon S3 儲存貯體來存放輸入和輸出資料。您可以使用 Amazon S3 主控台或 AWS Command Line Interface (AWS CLI) 建立 S3 儲存貯體。

創建一個 Amazon S3 存儲桶（控制台）

在 Amazon S3 主控台中，您可以建立名稱在所有儲存貯體中都是唯一的AWS。

若要建立 S3 儲存貯體 (主控台)

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在「值區」中選擇「建立值區」。
3. 在值區名稱中，輸入說明值區用途的全域唯一名稱。
4. 在「區域」中，選擇您要建立值AWS區的「區域」。您選擇的區域必須支援 Amazon Comprehend 域。若要減少延遲，請選擇距離您所在地理位置最近的AWS區域 (Amazon Comprehend 支援)。如需支援 Amazon Comprehend 的區域清單，請參閱全球基礎設施指南中的區域表。

5. 保留「物件擁有權」、「區塊公開存取」、「值區版本控制」和「標籤」的預設值區設定。
6. 對於預設加密，請選擇停用。

> **Tip**
>
> 雖然本教學課程不使用加密，但您可能會想要在分析重要資料時使用加密。對於 end-to-end 加密，您可以加密儲存貯體中的靜態資料，也可以在執行分析工作時加密。如需有關使用加密的詳細資訊AWS，請參閱什麼是AWS Key Management Service？ 在AWS Key Management Service開發人員指南中。

7. 檢閱值區組態，然後選擇 [建立值區]。

創建一個 Amazon S3 存儲桶（AWS CLI）

開啟之後AWS CLI，您可以執行`create-bucket`命令來建立儲存輸入和輸出資料的值區。

要創建一個 Amazon S3 存儲桶（AWS CLI）

1. 若要建立值區，請在中執行下列命令AWS CLI。將##########的名稱，該名稱在所有存儲桶中都是唯一的。AWS

```
aws s3api create-bucket --bucket DOC-EXAMPLE-BUCKET
```

根據預設，指`create-bucket`令會在「`us-east-1`AWS地區」中建立值區。若要在以外的值區中建立值區`us-east-1`，請新增`LocationConstraint`參數以指定您的「地區」。AWS 區域例如，下列指令會在「`us-west-2`區域」中建立值區。

```
aws s3api create-bucket --bucket DOC-EXAMPLE-BUCKET
--region us-west-2 --create-bucket-configuration LocationConstraint=us-west-2
```

請注意，只有特定區域支 Amazon Comprehend。如需支援 Amazon Comprehend 的區域清單，請參閱全球基礎設施指南中的區域表。

2. 若要確保您的值區已成功建立，請執行下列命令。此命令會列出與您帳戶相關聯的所有 S3 儲存貯體。

```
aws s3 ls
```

(僅限主控台) 建立資料夾

接下來，在 S3 儲存貯體中建立兩個資料夾。第一個文件夾用於輸入數據。第二個文件夾是 Amazon Comprehend 發送分析結果的地方。如果您使用 Amazon S3 主控台，則必須手動建立資料夾。如果您使用AWS CLI，則可以在上傳範例資料集或執行分析工作時建立資料夾。因此，我們提供了一個僅為控制台用戶創建文件夾的過程。如果您使用 AWS CLI，您將在上傳輸入數據和中建立資料夾步驟 3：在 Amazon S3 中的文件上執行分析任務。

在 S3 儲存貯體 (主控台) 中建立資料夾

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在「值區」中，從值區清單中選擇您的值區。
3. 在「概觀」標籤中，選擇「建立資料夾」。
4. 對於新資料夾名稱，請輸入`input`。
5. 對於加密設定，請選擇「無」(使用值區設定)。
6. 選擇儲存。
7. 重複步驟 3 到步驟 6，為分析工作的輸出建立另一個資料夾，但在步驟 4 中，輸入新的資料夾名稱`output`。

上傳輸入數據

現在您已經擁有值區，請上傳範例資料集`amazon-reviews.csv`。您可以使用 Amazon S3 主控台或 AWS CLI.

將示例文檔上傳到存儲桶（控制台）

在 Amazon S3 主控台中，將範例資料集檔案上傳到輸入資料夾。

若要上傳範例文件 (主控台)

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在「值區」中，從值區清單中選擇您的值區。
3. 選擇資`input`料夾，然後選擇 [上傳]。
4. 選擇添加文件，然後選擇計算機上的`amazon-reviews.csv`文件。

5. 將其他設定保留為其預設值。
6. 選擇上傳。

將範例文件上傳至值區 (AWS CLI)

在 S3 儲存貯體中建立輸入資料夾，並使用cp指令將資料集檔案上傳至新資料夾。

若要上載樣本文件 (AWS CLI)

1. 若要將amazon-reviews.csv檔案上傳至值區中的新資料夾，請執行下列AWS CLI命令。將*##############*的名稱。Amazon S3 會/input/在最後新增路徑，在儲存貯體中自動建立名為input的新資料夾，並將資料集檔案上傳到該資料夾。

```
aws s3 cp amazon-reviews.csv s3://DOC-EXAMPLE-BUCKET/input/
```

2. 若要確保檔案已成功上傳，請執行下列命令。此指令會列出值區資input料夾的內容。

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/input/
```

現在，您有一個 S3 存儲桶，該amazon-reviews.csv文件夾中的文件名為input. 如果您使用控制台，則存儲桶中也有一個output文件夾。如果您使用AWS CLI，則會在執行 Amazon Comprehend 分析任務時建立輸出資料夾。

步驟 2：(僅限 CLI) 為 Amazon Comprehend 創建 IAM 角色

只有當您使用 AWS Command Line Interface (AWS CLI) 完成此自學課程時，才需要執行此步驟。如果您使用 Amazon Comprehend 主控台執行分析任務，請跳至。步驟 3：在 Amazon S3 中的文件上執行分析任務

若要執行分析任務，Amazon Comprehend 需要存取包含範例資料集且包含工作輸出的 Amazon S3 儲存貯體。IAM 角色可讓您控制AWS服務或使用者的許可。在此步驟中，您會為 Amazon Comprehend 創建 IAM 角色。然後，您建立並附加到此角色，以資源為基礎的政策，以授予 Amazon Comprehend 存取您的 S3 儲存貯體的存取權。在此步驟結束時，Amazon Comprehend 將擁有必要的許可，以存取您的輸入資料、存放輸出以及執行情緒和實體分析任務。

如需將 IAM 與亞馬遜合作搭配使用的詳細資訊，請參閱。Amazon Comprehend 如何與 IAM 合作

主題

- 必要條件
- 建立 IAM 角色
- 將身分與存取權管理政策附加至 IAM 角色

必要條件

開始之前，請執行以下動作：

- 完成 第 1 步：將文檔添加到 Amazon S3。
- 使用代碼或文本編輯器來保存 JSON 政策並跟踪您的 Amazon 資源名稱（ARN）。

建立 IAM 角色

若要存取您的 Amazon 簡單儲存服務 (亞馬遜 S3) 儲存貯體，亞馬遜需要擔任 AWS Identity and Access Management (IAM) 角色。IAM 角色宣告 Amazon Comprehend 為受信任的實體。在 Amazon Comprehend 擔任該角色並成為受信任的實體之後，您可以將儲存貯體存取權限授與 Amazon Comprehend。在此步驟中，您會建立將 Amazon Comprehend 標示為受信任實體的角色。您可以使用AWS CLI或 Amazon Comprehend 主控台建立角色。若要使用主控台，請跳至步驟 3：在 Amazon S3 中的文件上執行分析任務。

Amazon Comprehend 主控台可讓您選取角色名稱包含「Comprehend」且信任政策包含的角色。comprehend.amazonaws.com如果您希望主控台顯示這些角色，請設定 CLI 建立的角色以符合這些準則。

若要為 Amazon Comprehend (AWSCLI) 建立 IAM 角色

1. 將下列信任原則儲存為電腦上程式碼或文字編輯器comprehend-trust-policy.json中呼叫的 JSON 文件。此信任政策將 Amazon Comprehend 宣告為受信任的實體，並允許其擔任 IAM 角色。

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

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

2. 若要建立 IAM 角色，請執行下列AWS CLI命令。此命令會建立名為的 IAM 角色，AmazonComprehendServiceRole-access-role並將信任政策附加至該角色。*path/*以本機電腦的 JSON 文件路徑取代。

```
aws iam create-role --role-name AmazonComprehendServiceRole-access-role
--assume-role-policy-document file://path/comprehend-trust-policy.json
```

> **Tip**
>
> 如果您收到剖析參數訊息時發生錯誤，表示 JSON 信任原則檔的路徑可能不正確。根據您的主目錄提供檔案的相對路徑。

3. 複製 Amazon 資源名稱（ARN）並將其保存在文本編輯器中。ARN 的格式類似於*arn:aws:iam::123456789012:role/AmazonComprehendServiceRole-access-role*。您需要此 ARN 才能執行 Amazon Comprehend 分析任務。

將身分與存取權管理政策附加至 IAM 角色

若要存取您的 Amazon S3 儲存貯體，Amazon Comprehend 需要列出、讀取和寫入的許可。若要授予 Amazon Comprehend 所需的許可，請建立 IAM 政策並將其附加到您的 IAM 角色。IAM 政策允許 Amazon Comprehend 從儲存貯體擷取輸入資料，並將分析結果寫入儲存貯體。建立政策後，您可以將其附加到 IAM 角色。

若要建立 IAM 政策 (AWS CLI)

1. 將下列原則儲存為名為的 JSON 文件comprehend-access-policy.json。它會授予 Amazon Comprehend 對指定 S3 儲存貯體的存取權。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

2. 若要建立 S3 儲存貯體存取政策，請執行下列AWS CLI命令。*path/*以本機電腦的 JSON 文件路徑取代。

```
aws iam create-policy --policy-name comprehend-access-policy
--policy-document file://path/comprehend-access-policy.json
```

3. 複製存取原則 ARN 並將其儲存在文字編輯器中。ARN 的格式類似於*arn:aws:iam::123456789012:policy/comprehend-access-policy*。您需要此 ARN 才能將存取政策附加到 IAM 角色。

將 IAM 政策附加到您的 IAM 角色 (AWS CLI)

- 執行下列命令。取代*policy-arn*為您在上一個步驟中複製的存取原則 ARN。

```
aws iam attach-role-policy --policy-arn policy-arn
```

```
--role-name AmazonComprehendServiceRole-access-role
```

您現在擁有一個名為的 IAM 角色，AmazonComprehendServiceRole-access-role該角色具有適用於 Amazon Comprehend 的信任政策，以及授予對 S3 儲存貯體的 Amazon Comprehend 存取權限的存取政策。您也將 IAM 角色的 ARN 複製到文字編輯器。

步驟 3：在 Amazon S3 中的文件上執行分析任務

將資料存放在 Amazon S3 之後，您就可以開始執行 Amazon Comprehend 分析任務。情緒分析工作決定文件的整體情緒 (正面、負面、中性或混合)。實體分析工作從文件中萃取真實世界物件的名稱。這些物件包括人員、場所、標題、事件、日期、數量、產品和組織。在此步驟中，您會執行兩個 Amazon Comprehend 分析任務，以從範例資料集擷取情緒和實體。

主題

- 必要條件
- 分析情緒和實體

必要條件

開始之前，請執行以下動作：

- 完成 第 1 步：將文檔添加到 Amazon S3。
- (選擇性) 如果您正在使用AWS CLI，請完成步驟 2：(僅限 CLI) 為 Amazon Comprehend 創建 IAM 角色並準備好您的 IAM 角色 ARN。

分析情緒和實體

您執行的第一個工作會分析範例資料集中每個客戶審核的情緒。第二個工作會擷取每個客戶審核中的實體。您可以使用 Amazon Comprehend 主控台或執行 Amazon Comprehend 分析任務。AWS CLI

> **Tip**
>
> 請確定您所在的AWS區域支援 Amazon Comprehend。如需詳細資訊，請參閱《全球基礎架構指南》中的「區域」表。

分析情緒和實體（控制台）

使用亞馬遜主控台時，您一次建立一個任務。您必須重複下列步驟，才能同時執行情緒和實體分析工作。請注意，對於第一個任務，您會建立 IAM 角色，但對於第二個任務，您可以重複使用第一個任務的 IAM 角色。只要您使用相同的 S3 儲存貯體和資料夾，就可以重複使用 IAM 角色。

若要執行情緒和實體分析工作 (主控台)

1. 確保您位於建立 Amazon Simple Storage Service (Amazon S3) 貯體的相同區域。如果您位於其他區域，請在導覽列中，從AWS區域選取器中選擇您建立 S3 儲存貯體的區域。
2. 在以下位置打開 Amazon Comprehend 控制台 https://console.aws.amazon.com/comprehend/
3. 選擇啟動 Amazon Comprehend.
4. 在導覽窗格中，選擇 [分析工作]。
5. 選擇 建立任務。
6. 在 [Job 設定] 區段中，執行下列動作：
   a. 針對名稱，輸入 `reviews-sentiment-analysis`。
   b. 針對 [分析類型] 選擇 [情緒]。
   c. 選擇「英文」做為「語言」。
   d. 將 Job 加密設定保持為停用狀態。
7. 在「輸入資料」區段中，執行下列操作：
   a. 對於資料來源，選擇我的文件。
   b. 對於 S3 位置，請選擇瀏覽 S3，然後從儲存貯體清單中選擇儲存貯體。
   c. 在 S3 儲存貯體中，對於物件，選擇您的`input`資料夾。
   d. 在資料`input`夾中，選擇範例資料集，`amazon-reviews.csv`然後選擇 [選擇]。
   e. 在「輸入格式」中，選擇「每行一個文件」。
8. 在「輸出資料」區段中，執行下列操作：
   a. 對於 S3 位置，請選擇瀏覽 S3，然後從儲存貯體清單中選擇儲存貯體。
   b. 在 S3 儲存貯體中，針對物件選擇`output`資料夾，然後選擇 [選擇]。
   c. 將加密保持關閉狀態。
9. 在「存取權限」區段中，執行下列動作：
   a. 對於 IAM 角色，請選擇建立 IAM 角色。

b. 對於存取權限，請選擇輸入和輸出 S3 儲存貯體。

c. 對於「名稱尾碼」，輸入comprehend-access-role。此角色可讓您存取 Amazon S3 儲存貯體。

10. 選擇 建立任務。

11. 重複步驟 1-10 以建立圖元分析工作。進行下列變更：

a. 在 Job 設定中，對於名稱，輸入reviews-entities-analysis。

b. 在 Job 設定中，對於分析類型，選擇實體。

c. 在 [存取權限] 中，選擇 [使用現有的 IAM 角色]。在 [角色名稱] 中，選擇 AmazonComprehendServiceRole-comprehend-access-role (這與您為情緒工作建立的角色相同)。

分析情緒和實體（ ）AWS CLI

您可以使用start-sentiment-detection-job和命start-entities-detection-job令來執行情緒和實體分析工作。執行每個命令後，會顯AWS CLI示一個 JSON 物件，其JobId值可讓您存取有關工作的詳細資訊，包括輸出 S3 位置。

若要執行情緒和實體分析工作 (AWSCLI)

1. 在中執行下列命令，以啟動情緒分析工作AWS CLI。以先前複製到文字編輯器的 IAM 角色 ARN 取*arn:aws:iam::123456789012:role/comprehend-access-role*代。如果您的預設 AWS CLI區域與建立 Amazon S3 儲存貯體的區域不同，請包含--region參數，並以儲存貯體所在的區域取代*us-east-1*。

```
aws comprehend start-sentiment-detection-job
--input-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/input/
--output-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/output/
--data-access-role-arn arn:aws:iam::123456789012:role/comprehend-access-role
--job-name reviews-sentiment-analysis
--language-code en
[--region us-east-1]
```

2. 提交工作後，複製JobId並將其儲存至文字編輯器。您將需JobId要從分析工作中尋找輸出檔案。

3. 執行下列命令以啟動實體分析工作。

```
aws comprehend start-entities-detection-job
--input-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/input/
```

```
--output-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/output/
--data-access-role-arn arn:aws:iam::123456789012:role/comprehend-access-role
--job-name reviews-entities-analysis
--language-code en
[--region us-east-1]
```

4. 提交工作後，複製JobId並將其儲存至文字編輯器。
5. 檢查工作狀態。您可以透過追蹤工作來檢視工作進度JobId。

   若要追蹤情緒分析工作的進度，請執行下列命令。以執*sentiment-job-id*行情緒分析後複製的項JobId目取代。

   ```
   aws comprehend describe-sentiment-detection-job
   --job-id sentiment-job-id
   ```

   若要追蹤實體分析工作，請執行下列命令。*entities-job-id*用執行圖元分析後複製的項JobId目取代。

   ```
   aws comprehend describe-entities-detection-job
   --job-id entities-job-id
   ```

   顯示為需要幾分鐘JobStatus的時間COMPLETED。

您已完成情緒和實體分析工作。在您繼續下一個步驟之前，這兩項工作都應完成。工作可能需要幾分鐘的時間才能完成。

步驟 4：準備用於資料視覺化的 Amazon Comprehend 輸出

若要準備用於建立資料視覺效果的情緒和實體分析工作的結果，請使用 AWS Glue 和 Amazon Athena。在此步驟中，您將擷取 Amazon Comprehend 結果檔案。然後，您可以建立探索資料的 AWS Glue 爬蟲程式，並在. AWS Glue Data Catalog之後，您可以使用 Amazon Athena無伺服器互動式查詢服務來存取和轉換這些資料表。完成此步驟後，您的 Amazon Comprehend 結果就會變得乾淨，而且可以進行視覺化。

對於 PII 實體偵測工作，輸出檔案為純文字，而非壓縮封存。輸出檔案名稱與輸入檔案名稱相同，尾端.out附加。您不需要提取輸出文件的步驟。跳過以將資料載入 AWS Glue Data Catalog。

主題

- 必要條件
- 下載輸出
- 提取輸出文件
- 上傳解壓縮的檔案
- 將數據加載到 AWS Glue Data Catalog
- 準備資料以進行分析

必要條件

開始之前，請完成步驟 3：在 Amazon S3 中的文件上執行分析任務。

下載輸出

Amazon Comprehend 使用 Gzip 壓縮來壓縮輸出文件並將其另存為 tar 存檔。提取輸出文件的最簡單方法是在本地下載`output.tar.gz`存檔。

在此步驟中，您會下載情緒和實體輸出封存檔。

下載輸出文件（控制台）

若要尋找每個任務的輸出檔案，請返回 Amazon Comprehend 主控台中的分析任務。分析任務提供輸出的 S3 位置，您可以在其中下載輸出檔案。

若要下載輸出檔案 (主控台)

1. 在亞馬遜主控台的導覽窗格中，返回分析任務。
2. 選擇您的情緒分析工作`reviews-sentiment-analysis`。
3. 在「輸出」下，選擇「輸出資料位置」旁邊顯示的連結。這會將您重新導向至 S3 儲`output.tar.gz`存貯體中的存檔。
4. 在「概覽」標籤中，選擇「下載」。
5. 在您的電腦上，將歸檔重新命名為`sentiment-output.tar.gz`。由於所有輸出檔案的名稱相同，因此可協助您追蹤情緒和實體檔案。
6. 重複步驟 1-4，尋找並下載`reviews-entities-analysis`工作的輸出。在您的電腦上，將歸檔重新命名為`entities-output.tar.gz`。

下載輸出檔案 (AWS CLI)

若要尋找每個工作的輸出檔案，請使用分析工作JobId中的來尋找輸出的 S3 位置。然後，使用命cp令將輸出文件下載到您的計算機。

若要下載輸出檔案 (AWS CLI)

1. 若要列出情緒分析工作的詳細資料，請執行下列命令。以您儲存的JobId情緒取*sentiment-job-id*代。

   ```
   aws comprehend describe-sentiment-detection-job --job-id sentiment-job-id
   ```

   如果遺失追蹤JobId，您可以執行下列命令來列出所有情緒工作，並依名稱篩選工作。

   ```
   aws comprehend list-sentiment-detection-jobs
   --filter JobName="reviews-sentiment-analysis"
   ```

2. 在物OutputDataConfig件中，尋找S3Uri值。該S3Uri值應類似於以下格式：*s3://DOC-EXAMPLE-BUCKET/.../output/output.tar.gz*。將此值複製到文字編輯器。

3. 若要將情緒輸出歸檔下載到您的本機目錄，請執行下列命令。將 S3 儲存貯體路徑取代為S3Uri您在上一個步驟中複製的路徑。以本*path/*機目錄的資料夾路徑取代。此名稱sentiment-output.tar.gz會取代原始封存檔名稱，以協助您追蹤情緒和實體檔案。

   ```
   aws s3 cp s3://DOC-EXAMPLE-BUCKET/.../output/output.tar.gz
   path/sentiment-output.tar.gz
   ```

4. 若要列出實體分析工作的詳細資料，請執行下列命令。

   ```
   aws comprehend describe-entities-detection-job
   --job-id entities-job-id
   ```

   如果您不知道您的JobId，請執行下列命令以列出所有實體工作，並依名稱篩選工作。

   ```
   aws comprehend list-entities-detection-jobs
   --filter JobName="reviews-entities-analysis"
   ```

5. 從實體工作說明中的OutputDataConfig物件複製S3Uri值。

6. 若要將實體輸出歸檔下載到本機目錄，請執行下列命令。將 S3 儲存貯體路徑取代為S3Uri您在上一個步驟中複製的路徑。以本*path/*機目錄的資料夾路徑取代。此名稱entities-output.tar.gz會取代原始壓縮檔名稱。

```
aws s3 cp s3://DOC-EXAMPLE-BUCKET/.../output/output.tar.gz
path/entities-output.tar.gz
```

提取輸出文件

在您可以存取 Amazon Comprehend 結果之前，請先解壓縮情緒和實體存檔。您可以使用本機檔案系統或終端機來解壓縮封存檔案。

解壓縮輸出文件（GUI 文件系統）

如果您使用 macOS，請按兩下 GUI 檔案系統中的歸檔，以從歸檔中解壓縮輸出檔案。

如果您使用 Windows，則可以使用第三方工具（例如 7-Zip）在 GUI 文件系統中提取輸出文件。在 Windows 中，您必須執行兩個步驟來存取歸檔中的輸出檔案。首先解壓縮存檔，然後解壓縮存檔。

將情緒檔案重新命名為sentiment-output，將實體檔案重新命名entities-output為，以區分輸出檔案。

提取輸出文件（終端）

如果您使用 Linux 或 macOS，則可以使用標準終端機。如果您使用 Windows，您必須能夠存取 Unix 風格的環境，例如 Cygwin，才能執行 tar 指令。

若要從情緒封存擷取情緒輸出檔案，請在本機終端機中執行下列命令。

```
tar -xvf sentiment-output.tar.gz --transform 's,^,sentiment-,'
```

請注意，該--transform參數將前綴添加sentiment-到歸檔內的輸出文件中，將文件重命名為sentiment-output。這可讓您區分情緒和實體輸出檔案，並防止覆寫。

要從實體存檔中提取實體輸出文件，請在本地終端中運行以下命令。

```
tar -xvf entities-output.tar.gz --transform 's,^,entities-,'
```

--transform參數會將字首加入entities-至輸出檔案名稱。

> **Tip**
>
> 若要節省 Amazon S3 中的儲存成本，您可以在上傳檔案之前使用 Gzip 再次壓縮檔案。解壓縮和解壓縮原始檔案很重要，因為 AWS Glue 無法自動從 tar 歸檔中讀取數據。但是， AWS Glue 可以從 Gzip 格式的文件中讀取。

上傳解壓縮的檔案

解壓縮檔案後，將檔案上傳至您的儲存貯體。您必須 AWS Glue 將情緒和實體輸出檔案儲存在不同的資料夾中，才能正確讀取資料。在值區中，為擷取的情緒結果建立資料夾，並為擷取的實體結果建立第二個資料夾。您可以使用 Amazon S3 主控台或 AWS CLI.

將擷取的檔案上傳到 Amazon S3 (主控台)

在 S3 儲存貯體中，為擷取的情緒結果檔案建立一個資料夾，為實體結果檔案建立一個資料夾。然後，將提取的結果文件上傳到各自的文件夾中。

將擷取的檔案上傳到 Amazon S3 (主控台)

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在「值區」中，選擇值區，然後選擇「建立檔案夾」。
3. 針對新資料夾名稱，輸入`sentiment-results`並選擇 [儲存]。此資料夾將包含擷取的情緒輸出檔案。
4. 在值區的「概覽」索引標籤中，從值區內容清單中選擇新的資料夾`sentiment-results`。選擇上傳。
5. 選擇 [新增檔案]，從本機電腦選擇`sentiment-output`檔案，然後選擇 [下一步]。
6. 保留 [管理使用者]、[其他 AWS 帳戶使用者存取] 和 [管理公用權限] 的選項做為預設值。選擇下一步。
7. 對於儲存空間類別，請選擇標準。將「加密」、「中繼資料」和「標籤」的選項保留為預設值。選擇下一步。
8. 檢閱上傳選項，然後選擇 [上傳]。
9. 重複步驟 1-8 以建立名為的資料夾`entities-results`，然後將`entities-output`檔案上傳至該資料夾。

將解壓縮的檔案上傳到 Amazon S3 (AWS CLI)

您可以在 S3 儲存貯體中建立資料夾，同時使用cp指令上傳檔案。

將擷取的檔案上傳到 Amazon S3 (AWS CLI)

1. 執行下列命令，建立情緒資料夾並將情緒檔案上傳至該資料夾。以擷取*path/*的情緒輸出檔案的本機路徑取代。

```
aws s3 cp path/sentiment-output s3://DOC-EXAMPLE-BUCKET/sentiment-results/
```

2. 創建一個實體輸出文件夾，並通過運行以下命令將您的實體文件上傳到它。以擷取實體輸出檔案的本機路徑取代*path/*。

```
aws s3 cp path/entities-output s3://DOC-EXAMPLE-BUCKET/entities-results/
```

將數據加載到 AWS Glue Data Catalog

要將結果放入數據庫中，您可以使用 AWS Glue 爬蟲。 AWS Glue 爬行者程式會掃描檔案並探索資料的結構描述。然後將資料以表格排列 AWS Glue Data Catalog (無伺服器資料庫) 中的資料。您可以使用 AWS Glue 主控台或. AWS CLI

將數據加載到 AWS Glue Data Catalog （ 控制台 ）

建立可分別掃描您sentiment-results和entities-results資料夾的 AWS Glue 爬蟲程式。新的 IAM 角色授 AWS Glue 予搜尋器存取 S3 儲存貯體的權限。您可以在設定爬行者程式時建立此 IAM 角色。

若要將資料載入 AWS Glue Data Catalog (主控台)

1. 確保您位於支持的地區 AWS Glue。如果您位於其他地區，請在導覽列中，從「地區」選取器中選擇支援的「地區」。如需支援的區域清單 AWS Glue，請參閱《全球基礎架構指南》中的區域表。
2. 開啟主 AWS Glue 控台，網址為 https://console.aws.amazon.com/glue/。
3. 在瀏覽窗格中，選擇 [爬行者程式]，然後選擇 [新增爬行者程式]。
4. 針對爬行者程式名稱，輸入，comprehend-analysis-crawler然後選擇下一步。
5. 對於爬行者程式來源類型，請選擇資料存放區，然後選擇下一步。
6. 對於「加入資料倉庫」，請執行以下作業：

    a. 對於 Choose a data store (選擇資料存放區)，選擇 S3。

b. 將「連線」留空。

c. 針對中的 [編目資料]，選擇 [我的帳戶中的指定路徑]

d. 在包含路徑中，輸入情緒輸出資料夾的完整 S3 路徑：`s3://DOC-EXAMPLE-BUCKET/sentiment-results`。

e. 選擇下一步。

7. 對於「新增其他資料倉庫」，選擇「是」，然後選擇「下一步 重複步驟 6，但輸入實體輸出資料夾的完整 S3 路徑：`s3://DOC-EXAMPLE-BUCKET/entities-results`。
8. 對於「新增其他資料倉庫」，選擇「否」，然後選擇「下一步
9. 對於 「選擇 IAM 角色」，請執行以下操作：

a. 選擇建立 IAM 角色。

b. 針對 IAM 角色，請輸入，`glue-access-role`然後選擇 [下一步]。

10. 針對「建立此爬行者程式」的排程，請選擇視需求執行，然後選擇下一步。
11. 對於設定爬行者程式的輸出，請執行下列動作：

a. 在「資料庫」中，選擇「新增資料

b. 對於 Database name (資料庫名稱)，輸入 `comprehend-results`。該數據庫將存儲您的 Amazon Comprehend 輸出表。

c. 將其他選項保留在其預設設定上，然後選擇「下一步」。

12. 複查爬行者程式資訊，然後選擇完成。
13. 在 Glue 主控台的 [爬行程式] 中，選擇**`comprehend-analysis-crawler`**並選擇 [執行爬行者程式]。爬行者程式可能需要幾分鐘的時間才能完成。

將數據加載到 AWS Glue Data Catalog （ AWS CLI ）

為建立 IAM 角色 AWS Glue ，提供存取 S3 儲存貯體的權限。然後，在中建立資料庫 AWS Glue Data Catalog。最後，建立並執行可將資料載入資料庫資料表的爬蟲程式。

若要將資料載入 AWS Glue Data Catalog (AWS CLI)

1. 若要為其建立 IAM 角色 AWS Glue，請執行下列動作：

a. 將下列信任原則儲存為電腦`glue-trust-policy.json`上呼叫的 JSON 文件。

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

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

b. 若要建立 IAM 角色，請執行下列命令。*path/*以本機電腦的 JSON 文件路徑取代。

```
aws iam create-role --role-name glue-access-role
--assume-role-policy-document file://path/glue-trust-policy.json
```

c. AWS CLI 列出新角色的 Amazon 資源編號 (ARN) 時，請將其複製並儲存到文字編輯器。

d. 將下列 IAM 政策儲存為電腦glue-access-policy.json上呼叫的 JSON 文件。此原則會 AWS Glue 授與檢索結果資料夾的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/sentiment-results*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/entities-results*"
            ]
        }
    ]
}
```

e. 若要建立 IAM 政策，請執行下列命令。*path/*以本機電腦的 JSON 文件路徑取代。

```
aws iam create-policy --policy-name glue-access-policy
--policy-document file://path/glue-access-policy.json
```

f. AWS CLI 列出存取原則的 ARN 時，請將其複製並儲存至文字編輯器。

g. 執行下列命令，將新政策附加至 IAM 角色。取代*policy-arn*為您在上一個步驟中複製的 IAM 政策 ARN。

```
aws iam attach-role-policy --policy-arn policy-arn
--role-name glue-access-role
```

h. 執行下列命令，將 AWS 受管政策附加AWSGlueServiceRole到 IAM 角色。

```
aws iam attach-role-policy --policy-arn
arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
--role-name glue-access-role
```

2. 通過運行以下命令創建一個 AWS Glue 數據庫。

```
aws glue create-database
--database-input Name="comprehend-results"
```

3. 執行下列命令來建立新的 AWS Glue 爬行者程式。*glue-iam-role-arn*以 AWS Glue IAM 角色的 ARN 取代。

```
aws glue create-crawler
--name comprehend-analysis-crawler
--role glue-iam-role-arn
--targets S3Targets=[
{Path="s3://DOC-EXAMPLE-BUCKET/sentiment-results"},
{Path="s3://DOC-EXAMPLE-BUCKET/entities-results"}]
--database-name comprehend-results
```

4. 執行下列命令來啟動搜尋器。

```
aws glue start-crawler --name comprehend-analysis-crawler
```

爬行者程式可能需要幾分鐘的時間才能完成。

準備資料以進行分析

現在您已經有一個資料庫填入了 Amazon Comprehend 結果。但是，結果是巢狀的。若要取消巢狀化，您可以在中執行幾個 SQL 陳述式 Amazon Athena。 Amazon Athena 這是一種互動式查詢服務，可讓您使用標準 SQL 輕鬆分析 Amazon S3 中的資料。Athena 是無伺服器服務，因此無需管理基礎架

構，而且具有 pay-per-query 定價模式。在此步驟中，您會建立可用於分析和視覺化的已清除資料的新表格。您可以使用 Athena 主控台準備資料。

準備資料

1. 前往 https://console.aws.amazon.com/athena/ 開啟 Athena 主控台。
2. 在查詢編輯器中，選擇 Settings (設定)，然後選擇 Manage (管理)。
3. 針對查詢結果的位置，輸入`s3://DOC-EXAMPLE-BUCKET/query-results/`。這會在值區中建立名為`query-results`的新資料夾，以儲存您執行的 Amazon Athena 查詢輸出。選擇 儲存。
4. 在查詢編輯器中，選擇「編輯器」。
5. 在「資料庫」中，選擇您建立`comprehend-results`的 AWS Glue 資料庫。
6. 在表部分，你應該有兩個名為`sentiment_results`和的表`entities_results`。預覽表格，確定爬行者程式已載入資料。在每個表格的選項 (表格名稱旁的三個點) 中，選擇「預覽表格」。簡短的查詢會自動執行。檢查「結果」窗格，以確保表格包含資料。

   > **Tip**
   >
   > 如果表沒有任何數據，請嘗試檢查 S3 存儲桶中的文件夾。請確定實體結果有一個資料夾，另一個資料夾可用於情緒結果。然後，嘗試運行一個新的 AWS Glue 爬蟲。

7. 若要取消巢狀`sentiment_results`資料表，請在 [查詢編輯器] 中輸入下列查詢，然後選擇 [執行]。

   ```
   CREATE TABLE sentiment_results_final AS
   SELECT file, line, sentiment,
   sentimentscore.mixed AS mixed,
   sentimentscore.negative AS negative,
   sentimentscore.neutral AS neutral,
   sentimentscore.positive AS positive
   FROM sentiment_results
   ```

8. 若要開始取消嵌套實體資料表，請在 [查詢編輯器] 中輸入下列查詢，然後選擇 [執行]。

   ```
   CREATE TABLE entities_results_1 AS
   SELECT file, line, nested FROM entities_results
   CROSS JOIN UNNEST(entities) as t(nested)
   ```

9. 若要完成取消嵌套實體資料表，請在 [查詢編輯器] 中輸入下列查詢，然後選擇 [執行查詢]。

```
CREATE TABLE entities_results_final AS
SELECT file, line,
nested.beginoffset AS beginoffset,
nested.endoffset AS endoffset,
nested.score AS score,
nested.text AS entity,
nested.type AS category
FROM entities_results_1
```

您的sentiment_results_final表格應如下所示，其中包含名為「檔案」、「線條」、「情緒」、「混合」、「負面」、「中性」和「正面」的欄。該表應該有每個單元格一個值。情緒欄描述特定檢閱最有可能的整體情緒。混合、負面、中立和正面的欄會為每種情緒類型提供分數。

Results

	file	line	sentiment	mixed	negative	neutral	positive
1	amazon-reviews.csv	6	MIXED	0.9862896203994751	0.0015502438182011247	1.6660270921420306E-4	0.0119935879483
2	amazon-reviews.csv	8	POSITIVE	0.0012987082591280341	0.01186690479516983	0.174478679895401	0.8123556375503
3	amazon-reviews.csv	11	POSITIVE	6.5368581090297084E-6	0.0013866390800103545	0.007405391428619623	0.9912014007568
4	amazon-reviews.csv	13	POSITIVE	4.7155481297522783E-4	0.24615342915058136	0.017713148146867752	0.7356618046760
5	amazon-reviews.csv	14	POSITIVE	1.5821871784282848E-5	0.06828905642032623	0.014075091108679771	0.9176200628280
6	amazon-reviews.csv	16	MIXED	0.9864791035652161	8.548551704734564E-4	1.0789262159960344E-4	0.0125581491738
7	amazon-reviews.csv	20	NEGATIVE	1.1621621524682269E-4	0.9815887212753296	0.004688907880336046	0.0136061981320
8	amazon-reviews.csv	21	POSITIVE	4.6635737817268 82E-5	0.009533549658954144	0.0015825830632820725	0.9888372421264
9	amazon-reviews.csv	23	POSITIVE	1.7699007003102452E-4	0.40269607305526733	0.0018250439316034317	0.5953019261360
10	amazon-reviews.csv	25	POSITIVE	1.8434448065818287E-6	1.15832663141191E-4	0.0010993879986926913	0.9987829327583

您的**entities_results_final**表應該如下所示，其中包含名為文件，行，開始偏移，內容集，得分，實體和類別的列。該表應該有每個單元格一個值。分數欄會指出 Amazon 對偵測到的實體的信賴度。類別會指出偵測到 Comprehend 實體類型。

Results

	file	line	beginoffset	endoffset	score	entity	category
1	amazon-reviews.csv	0	15	22	0.9885989378545348	English	OTHER
2	amazon-reviews.csv	2	24	28	0.9699371997593782	2 me	QUANTITY
3	amazon-reviews.csv	2	94	95	0.6523066984191679	2	QUANTITY
4	amazon-reviews.csv	2	125	126	0.713791396412543	2	QUANTITY
5	amazon-reviews.csv	4	30	36	0.9957169942979278	kindle	COMMERCIAL_ITEM
6	amazon-reviews.csv	5	1	10	0.9979111763962706	Hawthorne	PERSON
7	amazon-reviews.csv	5	135	142	0.5065408081314243	Puritan	OTHER
8	amazon-reviews.csv	5	143	148	0.7702269458801602	Salem	LOCATION
9	amazon-reviews.csv	5	211	229	0.999675563687763	The Scarlet Letter	TITLE
10	amazon-reviews.csv	5	233	236	0.8944631322676461	one	QUANTITY

現在您已將 Amazon Comprehend 結果載入到表格中，您可以從資料中視覺化並擷取有意義的見解。

第 5 步：可視化 Amazon Comprehend 輸出在 Amazon QuickSight

將 Amazon Comprehend 結果存儲在表中後，您可以連接到 Amazon 的數據並將其視覺化。QuickSightAmazon QuickSight 是用於視覺化資料的 AWS 受管商業智慧 (BI) 工具。Amazon QuickSight 可讓您輕鬆連線到資料來源並建立強大的視覺效果。在此步驟中，您可以將 Amazon 連接 QuickSight 到資料、建立可從資料擷取洞察的視覺效果，以及發佈視覺化的儀表板。

主題

- 必要條件
- 給 Amazon QuickSight 訪問
- 匯入資料集
- 建立情緒視覺效果
- 建立實體視覺效果
- 發佈儀表板
- 清除

必要條件

開始之前，請完成步驟 4：準備用於資料視覺化的 Amazon Comprehend 輸出。

給 Amazon QuickSight 訪問

要導入數據，Amazon QuickSight 要求訪問您的 Amazon Simple Storage Service (Amazon S3) 存儲桶和 Amazon Athena 表。若要讓 Amazon QuickSight 存取您的資料，您必須以 QuickSight 管理員身分登入，並擁有編輯資源許可的存取權。如果您無法完成下列步驟，請從概觀頁面檢閱 IAM 必要條件教學：使用亞馬遜分析客戶評論中的見解。

為了讓 Amazon QuickSight 訪問您的數據

1. 打開 Amazon QuickSight 控制台。
2. 如果這是您第一次使用 Amazon QuickSight，主控台會提示您透過提供電子郵件地址建立新的管理員使用者。在「電子郵件地址」中，輸入與您相同的電子郵件地址 AWS 帳戶。選擇 繼續。
3. 登入後，請在導覽列中選擇您的設定檔名稱，然後選擇「管理」 QuickSight。您必須以系統管理員身分登入才能檢視「管理 QuickSight」選項。
4. 選擇安全性和權限。
5. 若要QuickSight 存取 AWS 服務，請選擇 [新增或移除]。
6. 選擇 Amazon S3。
7. 從選取的 Amazon S3 儲存貯體中，為 S3 儲存貯體和 Athena 工作群組的寫入許可選擇 S3 儲存貯體。
8. 選擇 Finish (完成)。
9. 選擇更新。

匯入資料集

在建立視覺效果之前，您必須將情緒和實體資料集新增至 Amazon QuickSight。您可以使用 Amazon QuickSight 控制台執行此操作。您可以從中匯入非巢狀情緒和非巢狀實體表格。 Amazon Athena

若要匯入資料集

1. 打開 Amazon QuickSight 控制台。
2. 在導覽列的 [資料集] 中，選擇 [新增資料集]。
3. 對於「建立資料集」，請選擇 Athena。
4. 對於資料來源名稱，輸入`reviews-sentiment-analysis`並選擇建立資料來源。
5. 在 Database (資料庫) 中，選擇 `comprehend-results` 資料庫。
6. 針對「表格」，選擇情緒表，`sentiment_results_final`然後選擇 「選取」。

7. 選擇匯入至 SPICE 以加快分析速度，並選擇視覺化。SPICE QuickSight 是記憶體內計算引擎，可在建立視覺效果時提供比直接查詢更快的分析。
8. 返回 Amazon 主 QuickSight 控台並選擇資料集。重複步驟 1-7 以建立實體資料集，但請進行下列變更：

   a. 對於資料來源名稱，輸入`reviews-entities-analysis`。

   b. 對於「表格」，請選擇實體表格`entities_results_final`。

建立情緒視覺效果

現在您可以在 Amazon 中存取資料 QuickSight，就可以開始建立視覺效果了。您可以使用 Amazon Comprehend 情緒資料建立圓形圖。圓形圖會顯示評論的正面、中性、混合和負面的比例。

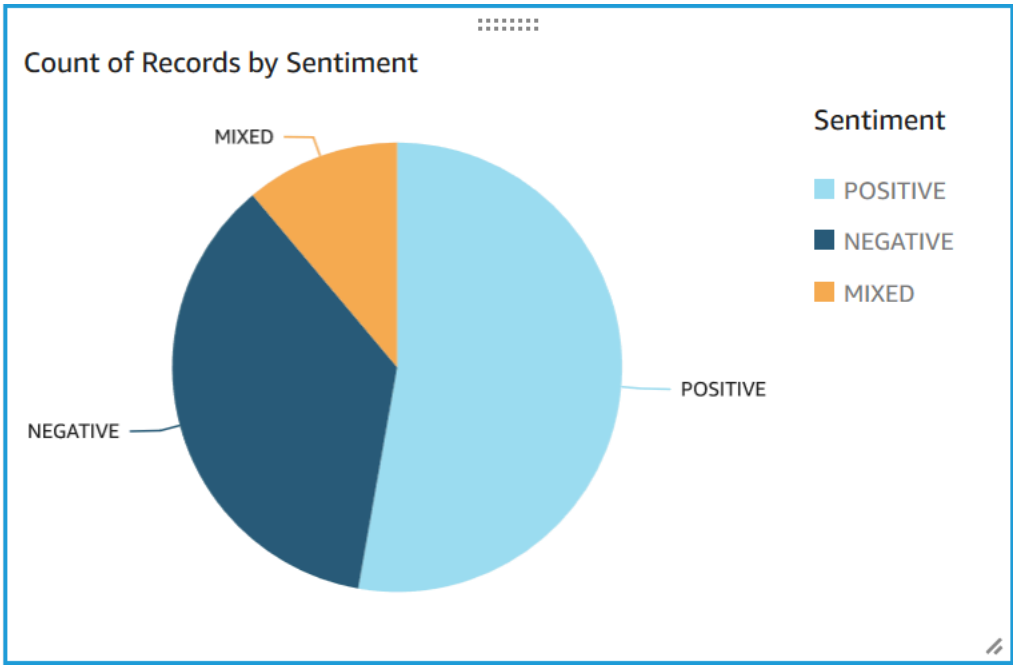
視覺化情緒資料

1. 在 Amazon QuickSight 主控台中，選擇 [分析]，然後選擇 [新增分析]。
2. 從 [您的資料集] 中選擇情緒資料集，`sentiment_results_final`然後選擇 [建立分析]。
3. 在視覺化編輯器的 [欄位] 清單中，選擇情緒。

> **Note**
>
> 「欄位」清單中的值取決於您在其中建立表格時所使用的欄名稱 Amazon Athena。如果您在 SQL 查詢中變更提供的欄名稱，[欄位] 清單名稱將會與這些視覺化範例中使用的名稱不同。

4. 針對 [視覺類型]，選擇 [圓形圖]。

會顯示類似下列的圓餅圖，其中包含正面、中性、混合和負面區段。若要查看區段的計數和百分比，請將游標暫留在該區段上。



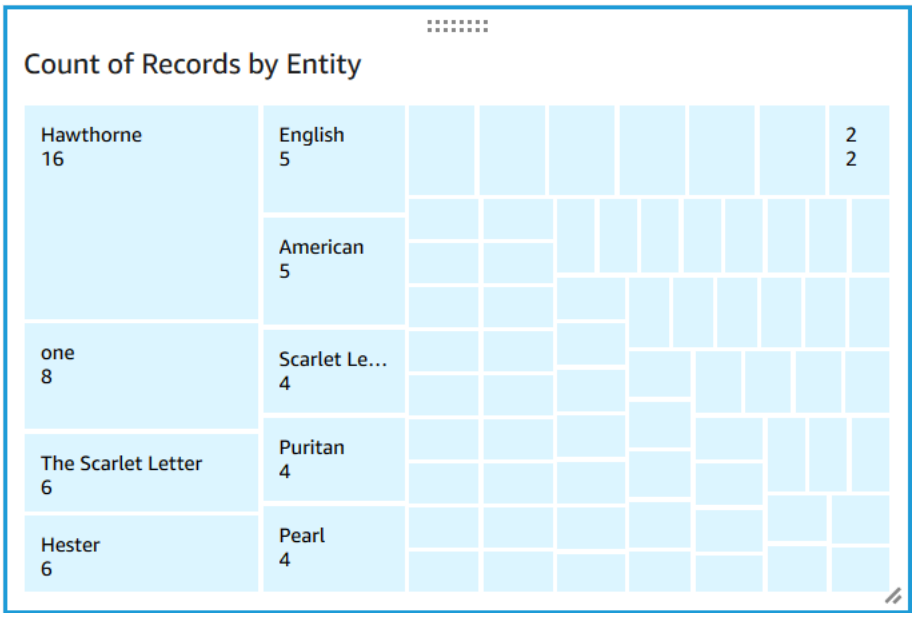
建立實體視覺效果

現在，使用實體資料集建立第二個視覺效果。您可以在資料中建立不同實體的樹狀對應。樹狀結構圖中的每個區塊都代表一個實體，而區塊的大小與實體出現在資料集中的次數相關。

若要視覺化實體資料

1. 在「視覺化」控制項窗格的「資料集」旁邊，選擇「新增」、「編輯」、「取代」和「移除資料集」圖示。
2. 選擇新增資料集。
3. 在 [選擇要新增的資料集] 中，`entities_results_final`從資料集清單中選擇您的實體資料集，然後選擇 [選取]。
4. 在「視覺化」控制項窗格中，選擇「資料集」下拉式功能表，然後選擇實體資料集`entities_results_final`。
5. 在 [欄位] 清單中選擇 [實體]。
6. 對於「視覺」類型，請選擇「樹狀貼圖

圓餅圖旁邊會顯示類似下列內容的樹狀圖。若要查看特定圖元的計數，請將游標暫留在圖塊上。



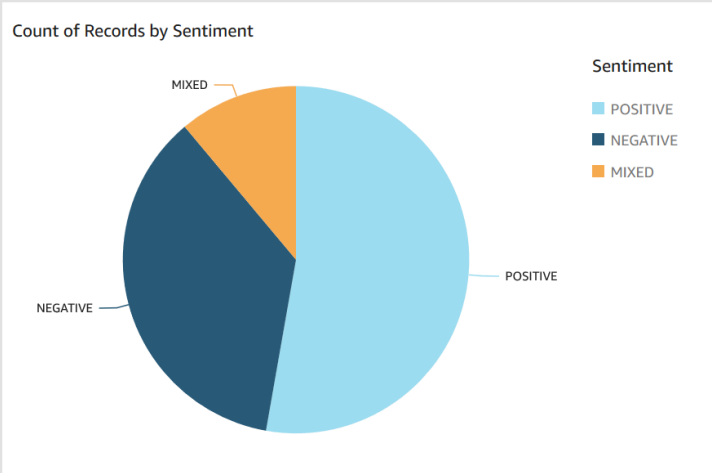
發佈儀表板

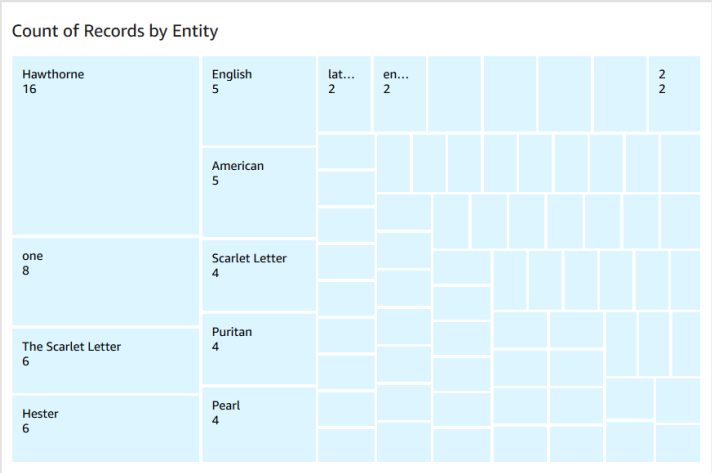
建立視覺效果後，您可以將其發佈為儀表板。您可以使用儀表板執行各種任務，例如與中的用戶共享 AWS 帳戶，將其保存為 PDF 或通過電子郵件發送為報告（僅限於 Amazon 的企業版 QuickSight）。在此步驟中，您將視覺效果發佈為帳戶中的儀表板。

若要發佈您的儀表板

1. 在導覽列中，選擇「分享」。
2. 選擇 (發佈儀表板)。
3. 選擇 [發佈新儀表板為]，然後輸入儀表板`comprehend-analysis-reviews`的名稱。
4. 選擇 (發佈儀表板)。
5. 選擇右上角的關閉按鈕，以關閉與使用者共用儀表板窗格。
6. 在 Amazon 主 QuickSight 控台的導覽窗格中，選擇儀表板。新儀表板的縮圖`comprehend-analysis-reviews`應會出現在「儀表板」下方。選擇儀表板以進行檢視。

您現在擁有一個包含情緒和實體視覺效果的儀表板，看起來與下列範例類似。





> **Tip**
>
> 如果您想要編輯儀表板中的視覺效果，請返回「分析」並編輯您要更新的視覺效果。然後，再次將儀表板發佈為新儀表板或取代現有儀表板。

清除

完成此自學課程後，您可能會想要清除任何不想再使用的 AWS 資源。使用中的 AWS 資源可能會繼續在您的帳戶中產生費用。

下列動作有助於防止產生持續的費用：

- 取消您的 Amazon QuickSight 訂閱。Amazon QuickSight 是每月訂閱服務。若要取消訂閱，請參閱 Amazon QuickSight 使用者指南中的取消訂閱。
- 刪除您的 Amazon S3 存儲桶。Amazon S3 向您收取存儲費用。若要清理 Amazon S3 資源，請刪除儲存貯體。如需刪除儲存貯體的詳細資訊，請參閱如何刪除 S3 儲存貯體？ 在 Amazon 簡單存儲服務用戶指南。刪除值區之前，請務必先儲存所有重要檔案。
- 清除您的 AWS Glue Data Catalog. 您每月的儲存 AWS Glue Data Catalog 費用。您可以刪除資料庫，以防止產生持續的費用。如需有關管理資 AWS Glue Data Catalog 料庫的詳細資訊，請參閱開AWS Glue 發人員指南中的使用 AWS Glue 主控台上的資料庫。在清除任何資料庫或表格之前，請務必先匯出資料。

使用 Amazon S3 物件 Lambda 存取點取得個人識別資訊 (PII)

使用 Amazon S3 物件 Lambda 存取點取得個人識別資訊 (PII)，以設定如何從 Amazon S3 儲存貯體擷取文件。您可以控制存取包含 PII 的文件，並從文件編輯 PII。如需 Amazon Comprehend 如何在文件中偵測 PII 的詳細資訊，請參閱。偵測 PII 實體Amazon S3 物件 Lambda 存取點使用AWS Lambda函數來自動轉換標準 Amazon S3 GET 請求的輸出。如需詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的使用 S3 物件 Lambda 轉換物件。

當您建立適用於 PII 的 Amazon S3 物件 Lambda 存取點時，系統會使用 Amazon Comprehend Lambda 函數處理文件，以控制存取包含 PII 的文件，以及從文件編輯 PII。

當您建立適用於 PII 的 Amazon S3 物件 Lambda 存取點時，會使用下列 Amazon Comprehend Lambda 函數處理文件：

- `ComprehendPiiAccessControlS3ObjectLambda`-使用存放在 S3 儲存貯體中的 PII 控制文件的存取。如需此 Lambda 函數的詳細資訊，請登入AWS Management Console以檢視中的 ComprehendPiiAccessControlS3 ObjectLambda 函數AWS Serverless Application Repository。
- `ComprehendPiiRedactionS3ObjectLambda`-從 Amazon S3 儲存貯體中的文件編輯 PII。如需此 Lambda 函數的詳細資訊，請登入AWS Management Console以檢視中的 ComprehendPiiRedactionS3 ObjectLambda 函數AWS Serverless Application Repository。

如需有關如何從部署無伺服器應用程式的資訊AWS Serverless Application Repository，請參閱《AWS 無伺服器應用程式儲存庫開發人員指南》中的部署應用

主題

- 控制存取含有個人識別資訊 (PII) 的文件
- 從文件中編輯個人識別資訊 (PII)

控制存取含有個人識別資訊 (PII) 的文件

您可以使用 Amazon S3 物件 Lambda 存取點來控制對具有個人識別資訊 (PII) 之文件的存取。

為確保只有授權使用者能夠存取包含存放在 Amazon S3 儲存貯體中之 PII 的文件，請使用此`ComprehendPiiAccessControlS3ObjectLambda`功能。此 Lambda 函數會在處理文件物件上的標準 Amazon S3 GET 請求時使用此ContainsPiiEntities作業。

例如，如果您的 S3 儲存貯體中有包含 PII 的文件，例如信用卡號碼或銀行帳戶資訊，則可以設定此ComprehendPiiAccessControlS3ObjectLambda功能偵測這些 PII 實體類型，並限制對未授權使用者的存取。如需支援的 PII 實體類型的詳細資訊，請參閱PII 通用實體類型。

如需此 Lambda 函數的詳細資訊，請登入AWS Management Console以檢視中的 ComprehendPiiAccessControlS3 ObjectLambda 函數AWS Serverless Application Repository。

建立 Amazon S3 物件 Lambda 存取點以控制文件的存取

下列範例會建立 Amazon S3 物件 Lambda 存取點，以控制對包含社會安全號碼之文件的存取。

使用建立 Amazon S3 物件 Lambda 存取點 AWS Command Line Interface

建立 Amazon S3 物件 Lambda 存取點組態，並將組態儲存在名為 config. json 的檔案中。

```
{
    "SupportingAccessPoint": "s3-default-access-point-name-arn",
    "TransformationConfigurations": [
        {
            "Actions": [
                "s3:GetObject"
            ],
            "ContentTransformation": {
                "AwsLambda": {
                    "FunctionArn": "comprehend-pii-access-control-s3-object-lambda-
arn",
                    "FunctionPayload": "{\"pii_entities_types\": \"SSN\"}"
                }
            }
        }
    ]
}
```

下列範例會根據config.json檔案中定義的組態建立 Amazon S3 物件 Lambda 存取點。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
aws s3control create-banner-access-point \
    --region region \
    --account-id account-id \
    --name s3-object-lambda-access-point \
```

```
    --configuration file://config.json
```

叫用 Amazon S3 物件 Lambda 存取點以控制文件的存取

下列範例會叫用 Amazon S3 物件 Lambda 存取點來控制文件的存取。

使用叫用 Amazon S3 物件的 Lambda 存取點 AWS Command Line Interface

下列範例會使用叫用 Amazon S3 物件 Lambda 存取AWS CLI點。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
aws s3api get-object \
    --region region \
    --bucket s3-object-lambda-access-point-name-arn \
    --key object-prefix-key output-file-name
```

從文件中編輯個人識別資訊 (PII)

您可以使用 Amazon S3 物件 Lambda 存取點來編輯文件中的個人識別資訊 (PII)。

若要從 S3 儲存貯體中存放的文件編輯 PII 實體類型，請使用函數`ComprehendPiiRedactionS3ObjectLambda`。此 Lambda 函數在處理文件物件上的標準 Amazon S3 GET 請求時使用ContainsPiiEntities和DetectPiiEntities操作。

例如，如果 S3 儲存貯體中的文件包含 PII，例如信用卡號碼或銀行帳戶資訊，您可以設定偵測 PII 的`ComprehendPiiRedactionS3ObjectLambda`功能，然後傳回編輯 PII 實體類型的這些文件副本。如需支援的 PII 實體類型的詳細資訊，請參閱PII 通用實體類型。

如需此 Lambda 函數的詳細資訊，請登入AWS Management Console以檢視中的 ComprehendPiiRedactionS3 ObjectLambda 函數AWS Serverless Application Repository。

建立 Amazon S3 物件 Lambda 存取點，以便從文件編輯 PII

下列範例會建立 Amazon S3 物件 Lambda 存取點，以便從文件兌換信用卡號碼。

使用建立 Amazon S3 物件 Lambda 存取點 AWS Command Line Interface

建立 Amazon S3 物件 Lambda 存取點組態，並將組態儲存在名為的檔案中`config.json`。

```
{
    "SupportingAccessPoint": "s3-default-access-point-name-arn",
    "TransformationConfigurations": [
        {
            "Actions": [
                "s3:GetObject"
            ],
            "ContentTransformation": {
                "AwsLambda": {
                    "FunctionArn": "comprehend-pii-redaction-s3-object-lambda-arn",
                    "FunctionPayload": "{\"pii_entities_types\": \"CREDIT_DEBIT_NUMBER
\"}"
                }
            }
        }
    ]
}
```

下列範例示範根據中定義的組態建立 Amazon S3 物件 Lambda 存取點 `config.json`

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
aws s3control create-access-point-for-object-lambda \
    --region region \
    --account-id account-id \
    --name s3-object-lambda-access-point \
    --configuration file://config.json
```

叫用 Amazon S3 物件 Lambda 存取點以從文件編輯 PII

下列範例會叫用 Amazon S3 物件 Lambda 存取點，以便從文件編輯 PII。

使用叫用 Amazon S3 物件的 Lambda 存取點 AWS Command Line Interface

下列範例會使用叫用 Amazon S3 物件 Lambda 存取AWS CLI點。

此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
aws s3api get-object \
    --region region \
```

```
    --bucket s3-object-lambda-access-point-name-arn \
    --key object-prefix-key output-file-name
```

解決方案：使用 Amazon Comprehend 和分析文本 OpenSearch

AWS提供使用 Amazon Comprehend 和服務的文字分析的參考實作。 OpenSearch Amazon Comprehend 提供文字分析，並 OpenSearch 提供文件索引、搜尋和視覺化功能。

如需詳細資訊，請參閱使用 OpenSearch 和 Amazon 理解分析文字。

API 參考

API 參考現在是個別的文件。如需詳細資訊，請參閱 Amazon Comprehend API 參考。

Amazon Comprehend 的文檔歷史

下表說明此發行版本 Amazon Comprehend 文件。

變更	描述	日期
使用本機文件進行自訂分類器訓練	Amazon Comprehend 現在支援使用原生文件的自訂分類器訓練。如需詳細資訊，請參閱訓練分類模型。	2023 年 4 月 19 日
用於管理自訂模型的飛輪	Amazon Comprehend 現在支援飛輪，可協助您管理自訂模型模型版本的訓練和追蹤。如需詳細資訊，請參閱亞馬遜概念中的飛輪。	2023 年 2 月 28 日
更新的 IAM 安全主題	更新 IAM 安全主題以包含聯合身分。如需詳細資訊，請參閱亞馬遜的 Identity and Access Management。	2022 年 12 月 22 日
使用自訂模型進行推論的單一步驟處理	Amazon Comprehend 現在會在執行自訂分類或自訂實體辨識之前，自動針對影像、PDF 或 Word 輸入文件執行文字擷取。如需詳細資訊，請參閱亞馬遜中的文件處理。	2022 年 12 月 1 日
針對目標情緒的同步 API	Amazon Comprehend 現在支援針對目標情緒的同步 API 和主控台即時分析。目標情緒會決定與文件中特定實體相關聯的情緒。如需詳細資訊，請參閱 Amazon Comprehend 中的目標情緒。	2022 年 9 月 21 日

降低訓練辨識者的最低註解	Amazon Comprehend 已降低使用純文字 CSV 註解檔訓練辨識器的最低需求。您現在可以建立一個自訂實體辨識模型，每個圖元類型最少包含三個註釋的文件和至少 25 個註記。如需詳細資訊，請參閱準備訓練資料。	2022 年 8 月 3 日
增加即時 API 的輸入文件大小	現在，針對大部分的即時 API，Amazon Comprehend 最高可支援 100 KB 的輸入文件。如需詳細資訊，請參閱準則和配額。	2022 年 7 月 18 日
其他 PII 實體類型	Amazon Comprehend 現在可以偵測到其他 PII 實體類型。如需詳細資訊，請參閱在 Amazon Comprehend 中偵測 PII 實體。	2022 年 5 月 20 日
目錄重組	重新架構 Amazon Comprehend 發人員指南目錄，以便於瀏覽。如需詳細資訊，請參閱什麼是 Amazon Comprehend。	2022 年 4 月 7 日
目標情緒	Amazon Comprehend 現在支援目標情緒分析，可決定與文件中特定實體相關聯的情緒。如需詳細資訊，請參閱 Amazon Comprehend 中的目標情緒。	2022 年 3 月 9 日

新功能	Amazon Comprehend 現在允許您分析圖像以進行自定義實體識別。如需詳細資訊，請參閱在 Amazon Comprehend 中偵測自訂實體。	2022 年 2 月 28 日
新功能	您現在可以在之間複製訓練過的自訂模型AWS 帳戶。如需詳細資訊，請參閱在 Amazon Comprehend 中的帳戶之間複製自訂模型。	2022 年 2 月 2 日
新功能	您現在可以使AWS Trusted Advisor用檢視建議，以協助您最佳化 Amazon Comprehend 端點的成本和安全性。如需詳細資訊，請參閱Trusted Advisor搭配 Amazon Comprehend 使用。	2021 年 9 月 29 日
新功能	Amazon Comprehend 已推出 Comprehend 自訂功能套件，讓您能夠建立新模型版本、在特定測試集上持續測試，以及執行即時移轉至新模型端點，藉此持續改善模型。	2021 年 9 月 21 日
新功能	Amazon Comprehend 現在允許您分析 PDF 和 Word 文檔以進行自定義實體識別。使用 PDF 和 Word 格式，您可以從包含標題，列表和表格的文檔中提取信息。	2021 年 9 月 14 日

新功能	Amazon Comprehend 推出了新的端點概觀功能，可為您提供端點的全球視圖。在端點概觀頁面中，您可以在一個位置查看所有端點，以了解端點使用情況與實際資源使用情況。	2021 年 8 月 24 日
新功能	Amazon Comprehend Medical 現在可讓您建立介面 VPC 端點，與 Virtual Private Cloud (VPC) 端 (VPC) 建立私有連線。如需詳細資訊，請參閱 VPC 端點 (PrivateLink)。	2021年6月13日
語言擴展	Amazon Comprehend 為主要語言功能增加了四種額外的語言：豪薩語 (ha)、老撾語 (lo)、馬耳他語 (mt) 和奧羅莫 (om)。如需詳細資訊，請參閱支援 Amazon Comprehend 語言。	2021 年 5 月 10 日
新功能	使用 Amazon Comprehend，您現在可以使用客戶受管金鑰 (CMK) 來加密自訂模型。如需詳細資訊，請參閱亞馬遜中的 KMS 加密。	2021 年 3 月 31 日
新功能	您現在可以使用 Amazon S3 物件 Lambda 存取點來設定如何從 Amazon S3 儲存貯體擷取包含個人識別資訊 (PII) 的文件。您可以控制包含 PII 的文件存取，並從文件編輯 PII。如需詳細資訊，請參閱使用 Amazon S3 物件 Lambda 存取點取得個人識別資訊 (PII)。	2021 年 3 月 18 日

新功能	您現在可以使用個人識別資訊 (PII) 來標記文件。Amazon Comprehend 可以分析您的文件是否存在 PII，並傳回已識別之 PII 實體類型的標籤，例如姓名、地址、銀行帳戶號碼或電話號碼。如需詳細資訊，請參閱使用 PII 標示文件。	2021 年 3 月 11 日
新功能	使用 Amazon Comprehend，您現在可以偵測一組文件中的事件。當您建立非同步事件偵測任務時，Amazon Comprehend 可以偵測支援的金融事件類型。如需詳細資訊，請參閱偵測事件。	2020 年 11 月 24 日
新功能	Amazon Comprehend 現在可讓您針對自訂實體辨識器端點使用自 auto 擴展。透過 auto 動擴充功能，您可以自動設定端點佈建以符合您的容量需求。如需詳細資訊，請參閱隨端點自動調整比例。	2020 年 9 月 28 日
新功能	若要訓練自訂分類器或實體辨識器，您現在可以提供擴增資訊清單檔案，這些檔案是由 Amazon SageMaker Ground Truth 產生的標籤資料集。有關這些檔案的詳細資訊以及範例，請參閱多重類別模式、多標籤模式和註釋。	2020 年 9 月 22 日

新增教學	Amazon Comprehend 現在提供一個教學課程，可引導您完成分析客戶評論和視覺化分析結果的多服務工作流程。如需詳細資訊，請參閱教學課程：分析評論中的見解。	2020 年 9 月 17 日
新功能	使用 Amazon Comprehend，您現在可以偵測文字中包含個人識別資訊 (PII) 的實體，例如地址、銀行帳戶號碼或電話號碼。Amazon Comprehend 可以在您的文字中提供每個 PII 實體的位置，也可以提供編輯 PII 的文字副本。如需詳細資訊，請參閱偵測個人識別資訊 (PII)。	2020 年 9 月 17 日
新功能	以前，您最多只能在 12 個自訂實體上訓練模型。現在，Amazon Comprehend 可讓您一次訓練多達 25 個自訂實體的模型。如需詳細資訊，請參閱自訂實體辨識。	2020 年 8 月 12 日
語言擴展	Amazon Comprehend 為自訂實體辨識功能新增了五種其他語言：德文 (德文)、西班牙文、法文 (fr)、義大利文 (it) 和葡萄牙文 (pt)。如需詳細資訊，請參閱支援 Amazon Comprehend 語言。	2020 年 8 月 12 日

新功能	Amazon Comprehend 現在可讓您透過建立介面 VPC 端點，與 Virtual Private Cloud (VPC) 端 (VPC) 建立私有連線。如需詳細資訊，請參閱 VPC 端點 (AWS PrivateLink)。	2020 年 8 月 11 日
新功能	透過 Amazon Comprehend，您現在可以透過執行即時分析，快速偵測個別文字文件中的自訂實體。如需詳細資訊，請參閱使用 Amazon 理解即時偵測自訂實體。	2020 年 7 月 9 日
添加了新功能	Amazon Comprehend 現在針對文件提供非同步自訂分類中的第二種模式支援，可在將自訂類別套用至文件時提供更大的彈性。雖然多重類別模式只會將單一類別與每個文件產生關聯，但新的多標籤模式可以關聯多個文件。例如，電影可以同時分類為科幻小說和動作。如需詳細資訊，請參閱自訂分類中的多類別和多標籤模式。	2019 年 12 月 19 日
添加了新功能	Amazon Comprehend 現在可為包含非結構化文字的文件提供即時自訂分類支援。客戶可以使用即時自訂分類，根據自己的業務規則同步理解、標記和路由資訊。如需詳細資訊，請參閱使用自訂分類進行即時分析。	2019 年 11 月 25 日

增加了新語言	Amazon Comprehend 為其數個功能新增了六種額外的語言：阿拉伯文 (ar)、印地文 (hi)、日文 (ja)、韓文 (ko)、簡體中文 (zh) 和繁體中文 (zh-TW)。這些新語言僅支援「判斷情緒」、「偵測關鍵片語」和「非自訂偵測實體」作業。如需詳細資訊，請參閱支援的語言。	2019 年 11 月 6 日
新功能	以前，您只能在單一自訂實體上訓練模型。因此，您只能使用實體辨識作業搜尋該實體。Amazon Comprehend 已經改變了這一點，您現在可以一次最多在 12 個自訂實體上訓練模型。如需詳細資訊，請參閱自訂實體辨識	2019 年 7 月 9 日
新功能	Amazon Comprehend 現在提供了一個多類別混淆矩陣，可在訓練自訂分類器時增加分析指標的能力。目前僅使用 API 支援此功能。如需詳細資訊，請參閱標記資源	2019 年 4 月 5 日
新功能	Amazon Comprehend 為自訂分類器和自訂實體辨識器提供標籤，這些標籤可用作中繼資料，讓您以比以往更精細的控制層級來組織、篩選和控制對資源的存取。如需詳細資訊，請參閱標記資源	2019 年 4 月 3 日

新功能	Amazon S3 已經讓您加密您的輸入文件，而 Amazon Comprehend 將其擴展得更遠。透過使用您自己的 KMS 金鑰，您不僅可以加密任務的輸出結果，還可以加密連接至處理分析工作之運算執行個體的儲存磁碟區上的資料。結果是 end-to-end安全性。如需詳細資訊，請參閱 Amazon Comprehend 中的 KMS 加密	2019 年 3 月 28 日
新功能	自訂實體辨識功能可讓您識別不支援的新實體類型做為預設泛型實體類型之一，藉此擴展 Amazon Comprehend 的功能。這表示您可以分析文件，並擷取符合您特定需求的產品代碼或業務特定實體等實體。如需詳細資訊，請參閱自訂實體辨識	2018 年 11 月 16 日
新功能	您可以使用 Amazon Comprehend 建立自己的模型以進行自訂分類，並將文件指派給類別或類別。如需詳細資訊，請參閱文件分類。	2018 年 11 月 15 日
區域擴展	Amazon Comprehend 現已在歐洲（法蘭克福）（歐盟中部-1）上市。	2018 年 10 月 10 日

語言擴展	除了英文和西班牙文外，Amazon Comprehend 現在還可以檢查法文、德文、義大利文和葡萄牙文的文件。如需詳細資訊，請參閱支援 Amazon Comprehend 語言。	2018 年 10 月 10 日
區域擴展	Amazon Comprehend 現已在亞太區域 (雪梨) (四月東南 -2) 推出。	2018 年 8 月 15 日
新功能	Amazon Comprehend 現在會剖析文件，以探索文件的語法以及每個字詞的語音部分。如需詳細資訊，請參閱語法。	2018 年 7 月 17 日
新功能	Amazon Comprehend 現在支援語言、關鍵片語、實體和情緒偵測的非同步批次處理。如需詳細資訊，請參閱非同步批次處理。	2018 年 6 月 27 日
新的指南	這是 Amazon Comprehend 開發人員指南的第一個版本。	2017 年 11 月 29 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。