13
Enabling Artificial Intelligence and Machine Learning

For a long time, organizations could only dream of the competitive advantage they would get if they could accurately forecast demand for their products, personalize recommendations for their customers, and automate complex tasks. And yet, advancements in **Artificial Intelligence** (**AI**) over the past decade or so have made many of these things, and much more, a reality.

AI describes the process of training computers in a way that mimics how humans learn to perform tasks. And **Machine Learning** (**ML**), which is a subset of AI uses a variety of advanced algorithms and, in most cases, large amounts of data to develop and train an ML model. This model can then be used to examine new data and automatically draw insights from that data.

The difference between AI and ML is not always clear, and people tend to use the terms somewhat interchangeably. AI focuses on creating machines to perform tasks that typically require human intelligence, such as problem-solving, decision-making, and understanding language. ML, more specifically, focuses on developing algorithms and statistical models to enable a model to recognize patterns in data and then make predictions based on those patterns. For example, training an ML model on a dataset that has customer data and an indication of which customers defaulted on their debt, and enabling that model to predict whether other customers may default on their debt.

AI and ML offers a wide range of interesting use cases that are expected to have a growing impact on many different aspects of life. For example, doctors are using ML to analyze a patient's retina scans to identify early signs of Alzheimer's disease. And it is the power of AI, and specifically computer vision, that is enabling advances in self-driving vehicles so that a car can navigate itself along a highway or even navigate complicated city streets unaided.

A self-driving car, as an AI system, makes use of underlying ML models for image classification, object tracking, and route planning.

Large Language Models (**LLMs**), such as ChatGPT or Bard, are AI systems that enable users to ask queries using natural language on a wide variety of topics, and get comprehensive responses, as well as generate new content (such as stories, poems, or songs).

AWS offers several services to help developers build their own custom advanced ML models, as well as a variety of pretrained models that can be used for specific purposes. In this chapter, we'll examine why AI and ML matter to organizations, and we'll review a number of the AWS AI and ML services, as well as how these services use different types of data.

In this chapter, we will cover the following topics:

- Understanding the value of ML and AI for organizations
- Exploring AWS services for ML
- Exploring AWS services for AI
- Building generative AI solutions on AWS
- Hands-on – reviewing the reviews with Amazon Comprehend

Before we get started, review the following *Technical requirements* section, which lists the prerequisites for performing the hands-on activity at the end of this chapter.

Technical requirements

In the last section of this chapter, we will go through a hands-on exercise that uses **Amazon SQS** and **AWS Lambda**, to send some text to the **Amazon Comprehend** service so that we can extract insights from it.

As with the other hands-on activities in this book, if you have access to an administrator user in your AWS account, you should have the permissions needed to complete these activities. If not, you will need to ensure that your user is granted access to create Amazon SQS and AWS Lambda resources, as well as at least read-only permissions for Amazon Comprehend APIs.

You can find the code files of this chapter in the GitHub repository using the following link: `https://github.com/PacktPublishing/Data-Engineering-with-AWS-2nd-edition/tree/main/Chapter13`

Understanding the value of AI and ML for organizations

More and more companies, of all sizes, are in various stages in the journey of discovering how AI and ML can positively impact their business. While initially, only the largest of organizations had the money and expertise to invest in AI projects, over time, the required technology has become more affordable and more accessible to non-specialist developers.

Cloud providers, such as AWS, have played a big part in making AI and ML technology more accessible to a wider group of users. Today, a developer with no previous ML education or experience can use a pre-trained AI service such as **Amazon Lex** to create a customer service **chatbot**. This chatbot will allow customers to ask questions using natural language, rather than having to select from a menu of preset

choices. Not all that long ago, anyone wanting to create a chatbot like this would have needed a Ph.D. in ML!

Many large organizations still look to build up data science teams with specialized AI and ML education and experience, and these developers are often involved in cutting-edge research and development. However, organizations of just about any size can use non-specialist developers to harness the power of AI to improve customer experience, financial forecasting, and other aspects of their business.

Let's have a look at some of the ways that ML is having an impact on different types of organizations.

Specialized AI projects

Large organizations in specialized industries make use of advanced AI technologies to develop cutting-edge ML advances. In this section, we'll have a look at a few examples of these technologies.

Medical clinical decision support platform

Cerner, a health information technology services company, has built an AI/ML-powered clinical decision support system to help hospitals streamline their workflows. This solution, built on AWS, uses AI models to predict how busy an emergency room may get on any given day, or time. This helps ensure that the right patients are prioritized for care, that patients are discharged at the right time, and that real-time data is used to create a **Centralized Operations Center** dashboard. This dashboard provides critical, near-real-time information on important metrics for managing hospital workflows, as well as predictions for what these metrics may look like over time.

Cerner has built its *Cerner Machine Learning Ecosystem* platform using **Amazon SageMaker**, as well as other AWS services. As with just about

all AI projects, getting the right data to train the underlying ML model is critical, and data engineers play an important role in this. In addition, data engineers are needed to build pipelines that enable near-real-time data to be ingested from multiple sources and fed into the platform. If the pipeline fails to ingest the right data at the right frequency, then the ML models cannot make the predictions that an organization may have come to depend on.

To learn more about the *Cerner clinical decision support system*, you can watch a pre-recorded webinar, available at https://www.youtube.com/watch?v=TZB8W7BL0eo.

Early detection of diseases

One of the areas of AI and ML that has massive potential for impacting a significant number of people is the early detection of serious diseases.

A January 2023 article in *MIT News* overviews an AI system developed by researchers at MIT that can detect future lung cancer risk (see https://news.mit.edu/2023/ai-model-can-detect-future-lung-cancer-0120). In this article the researches explain how they were surprised that there were lung scans where humans couldn't quite see where the cancer was, and yet the machine learning model they built was able to do better at predicting which lung may eventually develop cancer.

With many terminal diseases, early detection can make a significant difference in the outcome for the patient. For example, early detection, combined with appropriate medical interventions, can significantly increase the chance of survival beyond 5 years for certain cancer patients.

Making sports safer

Another area that AI is having an impact on is improving the safety of athletes for competitive sports. For example, the **National Football League** (**NFL**) in the United States is using Amazon AI and ML services to derive new insights into player injuries, rehabilitation, and recovery.

The NFL has started a project that uses **Amazon SageMaker** to develop a deep learning model to track players on a field, and then detect and classify significant injury events and collisions. There is an expectation that these advanced ML models, along with vast quantities of relevant data (including video data), can be used to significantly improve player safety over time.

> To learn more about how the NFL is using AI to improve player safety, you can watch a short video on YouTube from the AWS re:Invent 2020 conference titled *AWS re:Invent 2020 – Jennifer Langton of the NFL on using AWS to transform player safety* (https://www.youtube.com/watch?v=hXxfCn4tGp4).

Having had a look at a few specialized use cases, let's look at how everyday businesses are using AI and ML to impact their organizations and customers.

Everyday use cases for AI and ML

Just about every business, ranging from those with tens of employees to those with thousands of employees, is finding ways to improve through the use of AI and ML technologies.

One of the big reasons for this is that AI and ML have become more democratized over the past few years. Whereas AI and ML were once solely the domains of experts with years of experience in the field, today, a developer without specialized ML model-building experience can harness the power of these technologies in impactful ways. And

with LLMs like ChatGPT and Amazon Titan, just about every organization can find a use for **generative AI** to improve the productivity of workers.

Let's have a look at a few examples of how AI and ML are widely used across different business sectors.

Forecasting

Just about every organization needs to do forecasting to anticipate a variety of factors that influence their business. This includes financial forecasting (such as sales and profit margin), people forecasting (such as employee turnover, and how many staff are needed for a particular shift), and inventory forecasting (such as how many units we are likely to sell, how many units we need to manufacture next month, and so on).

Forecasting uses historical data over a period (often referred to as time series data) and attempts to predict likely future values over time. Forecasting has been around since long before ML, but traditional forecasts often lacked accuracy due to things such as irregular trends in historical data. Traditional forecasting also often failed to take into account variable factors, such as weather, promotions, and more.

ML has introduced new approaches, techniques and algorithms to forecasting that offer increased accuracy and the ability to take several variable factors into account. AWS offers several services that help bring the power of ML to forecasting problems, as we will discuss later in this chapter.

Personalization

Personalization is all about tailoring communication and content for a specific customer or subscriber. A good example of personalization is the effort Netflix has invested in to provide personalized recommen-

dations about other shows a specific subscriber may be interested in watching, based on the shows they have watched in the past (or shows they have been shown, that they have shown no interest in).

Other examples of where AI is used to power personalized recommendations are the recommended products on the *Amazon.com* storefront, as well as the recommended travel destinations on *booking.com*.

Natural language processing

Natural language processing (**NLP**) is a branch of AI/ML that is used to analyze human language and draw automated insights and context from the text.

A great example of NLP is the Alexa virtual assistant from Amazon. Users can speak to Alexa using natural language, and Alexa uses NLP algorithms to "understand" (in a sense) what the user is asking. While voice recognition systems have been around for a long time, these generally required users to say very specific phrases for the system to interpret what was wanted. With modern NLP approaches, 10 different users could ask the same question in 10 slightly different ways, and the system would be able to interpret what is being asked.

Traditional NLP technology is different to the new LLMs that have become popular since ChatGPT was launched. While LLMs are very good at responding to a very wide variety of prompts (based on their training on very large amounts of data, and their use of new transformer models), they sometimes may hallucinate (making up answers). Traditional NLP models are less likely to make up answers, but they are also more limited in the scope of prompts they can respond to.

Image recognition

Another area where AI and ML is having an impact on many businesses is through the use of image recognition ML models. With these models, images can be analyzed by the model to recognize objects

within them. This can be used for many different types of tasks, such as ensuring employees are wearing appropriate safety gear, or as part of the process of validating the identity of a customer. These models are also able to automatically label images based on what is in the image, such as the breed of dog in a collection of dog photos.

Now that we have reviewed some examples of the typical use cases for AI and ML, we can do a deeper dive into some of the AWS services that enable these use cases.

Exploring AWS services for ML

AWS has three broad categories of AI and ML services, as illustrated in the following diagram (note that only a small sample of AI and ML services are included in this diagram, due to space constraints):

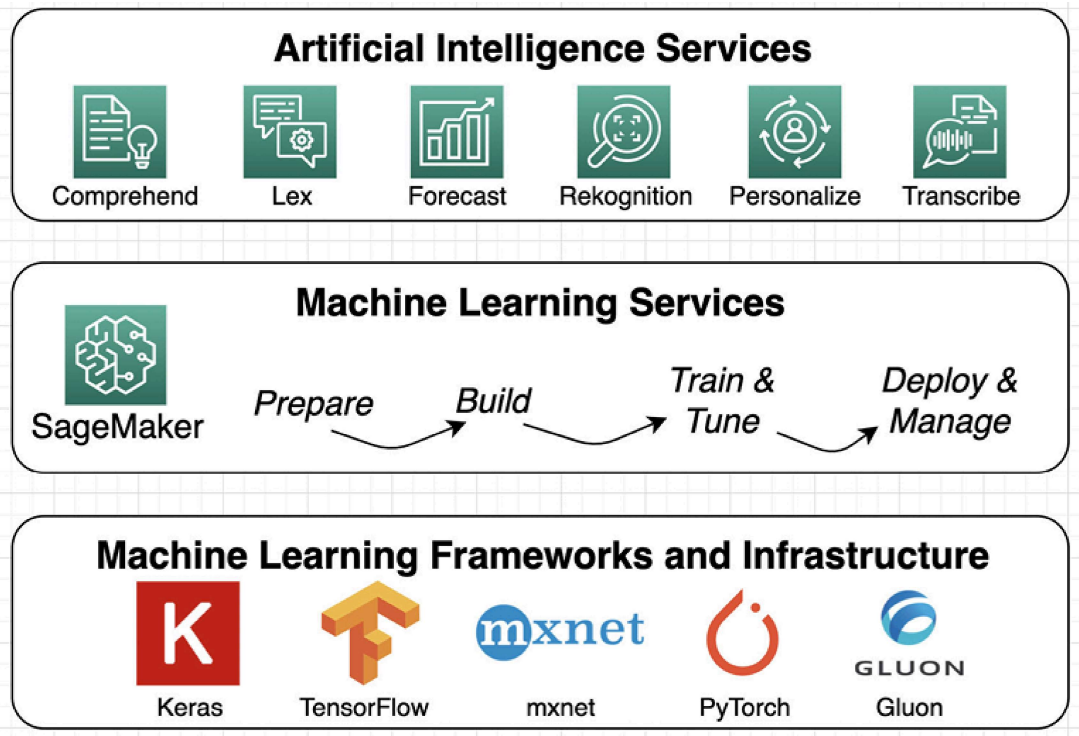

Figure 13.1: Amazon AI/ML stack

In the preceding diagram, we can see a subset of the services that AWS offers in each category – **Pretrained Artificial Intelligence Services**,

Machine Learning Services, and **Machine Learning Frameworks and Infrastructure**.

At the ML framework and infrastructure level, AWS provides **Amazon Machine Images** (**AMIs**) and prebuilt **Docker containers** that have popular **deep learning ML frameworks** pre-installed and optimized for the AWS environment. While these are useful for advanced use cases that require custom ML environments, these use cases are beyond the scope of this book, and require advanced ML expertise.

> For more information on these ML frameworks, refer to the AWS documentation on AWS Deep Learning AMIs (`https://aws.amazon.com/machine-learning/amis/`) and *AWS Deep Learning Containers* (`https://aws.amazon.com/machine-learning/containers/`).

In the remainder of this chapter, we will explore some of the services in the AWS ML services and AWS AI services categories.

AWS ML services

While working in the **Machine Learning Frameworks and Infrastructure** requires advanced ML expertise and experience, AWS makes developing ML models more accessible in the **Machine Learning Services** layer.

In this layer, **Amazon SageMaker** enables users to prepare, build, train, tune, deploy, and manage ML models, without needing to manage the underlying infrastructure. SageMaker is designed to simplify each step of building and deploying an ML model for both data scientists and everyday developers.

SageMaker includes several underlying tools to help with each of the stages of building an ML model.

SageMaker in the ML preparation phase

Several capabilities within SageMaker simplify and speed up the tasks involved in preparing to build an ML model. We covered these services in *Chapter 8*, *Identifying and Enabling Data Consumers*, so review that chapter for more information, but here is a quick reminder of these services.

Amazon SageMaker Ground Truth

The majority of ML models *learn* by being trained on labeled data. That is, the model is effectively given data that includes the attribute the model is designed to predict. Once trained, the model can then predict data where the attribute to be predicted is missing. For example, to train a model that can identify different breeds of dogs in a photo, you would train the model using photos of dogs that are labeled with the breed of dog. Once trained, you could provide a picture of a dog and the model could predict the breed.

SageMaker Ground Truth is a service that uses both ML and/or human curators to label data; for example, labelling the breed of a dog in a photo. This significantly speeds up the process of preparing data to use to train new ML models.

Amazon SageMaker Data Wrangler

The **SageMaker Data Wrangler** service is a visual data preparation tool that data scientists can use to prepare raw data for ML use. The service enables data scientists to select relevant datasets, explore the

data, and then select from over 300 built-in transformations that they can easily apply to the dataset, without writing any code.

SageMaker Data Wrangler also includes visualization templates that enable you to preview the results of transformations in **SageMaker Studio**, a full-fledged **integrated development environment** (**IDE**) for ML.

Amazon SageMaker Clarify

When training an ML model with a training dataset, the dataset may be biased through either a concentration of specific data or because it is missing specific data.

For example, if a dataset is intended to be used to predict responses from people with a wide age range, but the training dataset primarily contains data from people aged 35 – 55, then predictions may be inaccurate for both younger people (under 35) and/or older people (over 55).

The same could be applied to datasets that tend to concentrate on a specific gender, sexual orientation, married versus unmarried, or just about any other attribute. To help avoid this type of potential bias in a dataset, **SageMaker Clarify** can examine some specified attributes in a dataset and use advanced algorithms to highlight the existence of potential bias.

SageMaker in the ML build phase

Once data has been labeled and prepared, a data scientist can move on to building ML models. The following capabilities in SageMaker are used to build new ML models.

SageMaker Studio notebooks

Data scientists typically use notebooks to develop the code for their ML models. A notebook is an interactive web-based environment where developers can run their code and immediately see the results of the running code. An interactive notebook is backed by a compute engine that runs a kernel where notebook code is executed.

With **SageMaker Studio Notebooks**, you can quickly launch a new notebook, backed by an EC2 instance type of your choosing. The notebook environment uses **Amazon Elastic File System** (**EFS**), which is network-based storage that persists beyond the life of the instance running the notebook. This enables you to easily start and stop different notebook instances, and have your notebook project files available in each notebook instance.

SageMaker Studio Notebooks also enables users to easily share notebooks, enabling collaborative work between data scientists on a team. In addition, SageMaker Studio Notebooks provides sample projects that can be used as a starting point for developing a new model.

SageMaker Autopilot

For developers that do not have extensive ML experience, **SageMaker Autopilot** can be used to automatically build, train, and tune several different ML models, based on your data.

The developer needs to provide a tabular dataset (rows and columns) and then indicate which column value they want to predict. This could be predicting a number (such as expected spend), a binary category (fraud or not fraud), or a multi-label category (such as favorite fruit, which could be banana, peach, pear, and so on).

SageMaker Autopilot will then build, train, and tune several ML models and provide a model leaderboard to show the results of each model. Users can view each of the models that were generated and explore the results that were generated by each model. From here, a user can select the model that best meets their requirements and deploy it.

SageMaker JumpStart

SageMaker JumpStart provides several preselected end-to-end solutions, ML models, and other resources to help developers and data scientists get their ML projects up and running quickly.

By using these prebuilt resources, developers can easily deploy solutions and models with all the infrastructure components managed for them. Once deployed, the model can be opened with SageMaker Studio Notebooks, and the model can be tested through a notebook environment.

Prebuilt solutions include sample datasets that can be used to test the model, and you can also provide your own dataset to further train and tune the model. Some examples of prebuilt solutions available in JumpStart include the following:

- Foundation models (used to build generative AI solutions)
- Churn prediction
- Credit risk prediction
- Computer vision
- Predictive maintenance

For an example of how to use SageMaker JumpStart to build a custom generative AI solution, see the AWS blog post titled *Fine-tune text-to-image Stable Diffusion models with Amazon SageMaker JumpStart* at `https://aws.amazon.com/blogs/machine-learning/fine-tune-text-to-image-stable-diffusion-models-with-amazon-sagemaker-jumpstart/`. In this blog post, you learn how to use SageMaker JumpStart to fine-tune the Stable Diffusion model by uploading pictures of a dog, and then having the Stable Diffusion model generate new images of the dog in differing situations (such as the dog on a beach, a pencil sketch of the dog, etc).

SageMaker in the ML training and tuning phase

Once you have built an ML model, you need to train the model on a sample dataset, and then further tune and refine the model until you get the results that meet your requirements.

Training a model is core functionality that's built into SageMaker. You point SageMaker to the location of your training data in Amazon S3, and then specify the type and quantity of SageMaker ML instances you want to use for the training job. SageMaker will provision a distributed compute cluster and perform the training, outputting the results to Amazon S3. The training cluster will then automatically be removed.

SageMaker can also automatically tune your ML model by testing the model with thousands of different combinations of algorithm parameters to determine which combination of parameters provides the most accurate results. This process is referred to as **hyperparameter tuning**, and with SageMaker, you can specify the range of hyperparameters that you want to test.

To keep track of the results of different training jobs, SageMaker also includes something called SageMaker Experiments.

SageMaker Experiments

This process of tracking different ML experiments can be made significantly easier using **SageMaker Experiments**. This feature of SageMaker automatically tracks items such as inputs, parameters, and configurations, and stores the result of each experiment. This helps reduce the overhead and time needed to identify the best performing combinations for your ML model.

When running a training job on SageMaker, you can pass in an extra parameter, defining the name of the experiment. By doing this, all the inputs and outputs of the job will automatically be logged.

This data can then be loaded into a pandas DataFrame (a popular Python data structure for working with data), and you can use the built-in analytics features of pandas to analyze your results. Amazon SageMaker Studio also includes integration with SageMaker Experiments, enabling you to run queries on experiment data, and view leaderboards and metrics.

SageMaker in the ML deployment and management phase

Once you have prepared your data, developed your model, and then trained and tuned the model, you are finally ready to deploy the model. There are several different ways that you can select to deploy the model using SageMaker.

For example, if you want to get predictions on a large dataset, you can use SageMaker's batch transform process. Using this, you point SageMaker to the dataset on S3, select the type of compute instance you want to use to power the transform, and then run the transform job, which will make a prediction for each record in the dataset and write out the transformed dataset to S3.

Alternatively, you can deploy an endpoint for your model that can be used by your applications to pass data to the model to get an ML-powered prediction in real time. For example, you can pass information to the endpoint of a specific credit card transaction (date, time, location, vendor, amount, and so on), and the ML model can predict whether this is a fraudulent or genuine transaction.

ML models can become less accurate over time due to changing trends in your customer base, for example, or because of data quality issues

in upstream systems. To help monitor and manage this, you can use SageMaker Model Monitor.

SageMaker Model Monitor

SageMaker Model Monitor can be configured to continuously monitor the quality of your ML models and can send notifications when there are deviations in the model's quality. Model Monitor can detect issues with items such as data quality, model quality, and bias drift.

To resolve issues with model quality, a user may take steps such as retraining the model using updated data or investigating potential quality issues with upstream data preparation systems.

Having briefly covered some of the extensive functionality available for creating custom models using Amazon SageMaker, let's look at some of the AWS AI services that provide prebuilt ML models as a service.

Exploring AWS services for AI

While Amazon SageMaker simplifies building custom ML models, there are many use cases where a custom model is not required, and a generalized ML model that has been pretrained will meet requirements.

For example, if you need to translate from one language into another, that will most likely not require a customized ML model. Existing, generalized models, trained for the languages you are translating between, would work.

You could use SageMaker to develop a French to English translation model, train the model, and then host the model on a SageMaker inference endpoint. But that would take time and would have compute costs associated with each phase of development (data preparation, notebooks, training, and inference).

Instead, it would be massively simpler, quicker, and cheaper to use a pretrained AI service such as **Amazon Translate**, which already incorporates language models trained for this task. This service provides a simple API that can be used to pass in text in one language and receive a translation in a target language. And there would be no ongoing compute costs or commitments – just a small per-character cost for the translation (currently $ 0.000015 per character).

Also, AWS is constantly working to improve the underlying ML algorithms in these pretrained AI services, monitoring data quality, and maintaining the availability of the API endpoints, at no additional cost to you. And if you do need to customize the model (for example, based on specific industry terminology, or a preferred style or tone for the translation), you can provide additional training data for customized translations, although this comes at a slightly higher cost (currently $0.00006 per character).

These types of pretrained AI services have gained in popularity over the past few years, and all of the major cloud providers now offer a range of pretrained ML models that are delivered as an AI service. We don't have space in this chapter to cover all of the AWS AI services, but we'll look at a few of the most popular services in this section.

We started with Amazon Translate as an example of an AWS AI service, so now, let's explore some of the other AI offerings from AWS.

AI for unstructured speech and text

One of the primary benefits of a data lake is the ability to store all types of data, including **unstructured data** such as PDF documents, as well as audio and video files, in the data lake. And while this type of data can be easily ingested and stored in the data lake, the challenge for the data engineer is in how to process and make use of this data.

For example, a large enterprise company may have hundreds of thousands of invoices from a variety of vendors, and they may want to perform analysis or fraud detection on those. Or a busy call center may want to automatically transcribe recorded customer calls to perform sentiment analysis and identify unhappy customers.

For these use cases, AWS offers several AI services designed to extract metadata from text or speech sources to make this data available for additional analysis.

Amazon Transcribe for converting speech into text

Amazon Transcribe is an AWS AI service that can produce text transcription from audio and video files. This can be used to generate subtitles for a video file, to provide a transcription of a recording of a meeting or speech, or to get a transcript of a customer service call.

Transcribe uses **automatic speech recognition** (**ASR**), a deep learning process, to enable highly accurate transcriptions from audio files, including the ability to identify different speakers in the transcript and to automatically identify the language of a recording. Transcribe can also detect and remove sensitive personal information (such as credit card numbers or email addresses) from transcripts, as well as words that you don't want to be included in a transcription (such as curses or swear words). Transcribe can also generate a new audio file that replaces these unwanted words with silence.

A data engineer can build a pipeline that processes audio or video files with Transcribe, ensuring that text transcripts from audio sources are generated shortly after new audio sources are ingested into the data lake. Other ML models or AWS AI services can also be built into the pipeline to further analyze the transcript to generate additional metadata.

Amazon Transcribe can work in both batch and streaming mode (although not all features or regions are supported in streaming mode). With batch mode, you point Amazon Transcribe at an audio recording in Amazon S3, while in streaming mode, you send through 'chunks' of the audio that then gets transcribed immediately. Streaming mode can be used for use-cases such as live captioning of meetings or a news broadcast.

Amazon Transcribe also includes functionality targeted at specific types of audio. For example, **Amazon Transcribe Medical** uses an ML model specifically trained to identify medical terminologies such as medicine names, diseases, and conditions. And **Amazon Transcribe Call Analytics** has been specifically designed to understand customer service and sales calls, as well as to identify attributes such as agent and customer sentiment, interruptions, and talk speed.

Amazon Textract for extracting text from documents

Amazon Textract is an AI service that can be used to automatically extract text from unstructured documents, such as PDF or image files. Whether the source document is a scan of printed text or a form that includes printed text and handwriting, Textract can be used to create a semi-structured document for further analysis.

A data engineer may be tasked, for example, with building a pipeline that automatically analyzes uploaded expense receipts to extract relevant information. This may include storing that information in semi-structured files in the data lake, or a different target such as DynamoDB, a relational database, or the Amazon OpenSearch Service.

For example, the following screenshot shows a portion of a hotel receipt bill contained in a PDF file:

DATE	REF NO	DESCRIPTION	CHARGES
4/15/2019	2559498	GUEST ROOM	$179.00
4/15/2019	2559498	STATE TAX	$10.74
4/15/2019	2559498	CITY TAX	$16.11
4/16/2019	2559777	C3 FOOD DRINK	$7.00
4/16/2019	2559811	VS	($212.85)
		BALANCE	$0.00

Hilton Honors(R) stays are posted within 72 hours of checkout. To check your earnings or book your next stay at more than 4,000 hotels and resorts in 100 countries, please visit Honors.com

Thank you for choosing Doubletree! Come back soon to enjoy our warm chocolate chip cookies and relaxed hospitality. For your next trip visit us at doubletree.com for our best available rates!

Figure 13.2: Extract from a PDF document of a hotel invoice

Most traditional analytics tools would not be able to process the data contained within a PDF file, but when this file is sent to the Amazon Textract service, a semi-structured file can be created containing relevant data. For example, the ML model powering Textract can extract information from the preceding table as a CSV file that can be further analyzed in a data engineering pipeline.

The following table shows the CSV file when opened in a spreadsheet application:

DATE	REF NO	DESCRIPTION	CHARGES	
4/15/2019	2559498	GUEST ROOM	$179.00	
4/15/2019	2559498	STATE TAX	$10.74	
4/15/2019	2559498	CITY TAX	$16.11	
4/16/2019	2559777	C3 FOOD DRINK	$7.00	
4/16/2019	2559811	VS	($212.85)	

Figure 13.3: CSV formatted data extracted from a PDF invoice

Textract has been designed to work well with various types of documents, including documents that contain handwritten notes. For example, a medical intake form at a doctor's office, where patients fill out the form by hand, can be sent to Textract to extract data from the form for further processing.

Textract also has advanced functionality for handling specific types of documents, such as the **Textract Analyze Lending** feature that can automate the extraction of information from loan packages, automatically splitting the document package by document type. Textract can also automatically detect signatures, extract data from forms and tables, and process identity documents (such as U.S. passports and drivers licenses). And with a feature called **Textract Queries**, you can enable natural language queries about data extracted from documents. For example, you can enable a user to ask questions such as "*What is the customer name*", and have Textract automatically provide the relevant answer based on the processing of the document.

Amazon Comprehend for extracting insights from text

We have looked at how Amazon Transcribe can create electronic text from speech, as well as how Amazon Textract can create semi-structured documents from scanned documents and images. Now, let's look at how to extract additional insights from text.

Amazon Comprehend is an AI service that uses advanced ML models to generate additional insights from text documents, such as sentiment, topics, place names, PII information, key phrases, dominant language detection, and more. With Comprehend, you can build a near-real-time pipeline that passes in 1 to 25 documents in a single API call for analysis, or build a batch pipeline that configures Comprehend to analyze all documents in an S3 bucket.

When you call the API or run an asynchronous batch job, you specify the type of comprehension that you want in the results. For example, you can have Comprehend analyze text to detect the dominant language, entities, key phrases, PII data, sentiment, or topics (each type of comprehension has a different API call).

Comprehend can be used for several use cases, such as identifying important entities in lengthy legal contracts (such as location, people, and companies), or understanding customer sentiment when customers interact with your call center. As a data engineer, you may be tasked with building a pipeline that uses Amazon Transcribe to convert the audio of recorded customer service calls into text, and then run that text through Comprehend to capture insight into customer sentiment for each call.

Another use case could be to analyze social media posts to identify which organizations were being referenced in a post, and what the sentiment of the review was. For example, we could analyze the following fictional post made to a social media platform:

> *"I went to Jack's Cafe last Monday, and the pancakes were amazing! You should try this place, it's new in downtown Westwood. Our server, Regina, was amazing."*

When Amazon Comprehend analyzes this text, it returns the following insights:

1. **Entities detected**:
    1. Jack's café, Organization, 93% confidence
    2. Westwood, Location, 71% confidence
    3. Regina, Person, 99% confidence
    4. last Monday, Date, 94% confidence
2. **Sentiment**:
    1. Positive, 99% confidence

As we can see from the previous results, Comprehend can accurately detect entities and sentiment. At the end of this chapter, we will go through an exercise with Amazon Comprehend to determine customer sentiment from online reviews, which will allow you to get hands-on with how Amazon Comprehend works.

Note that there is also a specialty version of Comprehend, called **Amazon Comprehend Medical**, that has been designed to extract medical information from electronic text, such as medical conditions, medications, treatments, and protected health information. You can also train a **Comprehend custom entity detection** model using your data to recognize specialized entities (such as a model trained to recognize different makes and models of cars and motorbikes).

AI for extracting metadata from images and video

In the previous section, we reviewed AI services for processing text – including audio transcribed into text (Amazon Transcribe), images and scanned documents converted into text (Amazon Textract), and insights drawn out of electronic text (Amazon Comprehend).

In this section, we will change focus and look at how we can extract insights out of videos and images using the power of AI.

Amazon Rekognition

Amazon Rekognition uses the power of pretrained ML models to extract metadata from images and videos, enabling users to get rich insights from this unstructured content.

With traditional data warehouses and databases, the ability to store unstructured data, such as images and videos, was very limited. In addition, until recently, it was difficult to extract rich metadata from these unstructured sources, without having humans manually label data. And, as you can imagine, this was a very time-consuming and error-prone process.

For organizations that stored a lot of images or videos, they needed to manually build catalogs to tag the media appropriately. For example, these organizations would need someone to manually identify celebri-

ties in photos or add labels to an image to tag what was shown in the image.

As ML technologies advanced, these organizations could build and train ML models to automatically tag images (or stills from a video), but this still required deep expertise and an extensive labeled catalog for training the ML model.

With new pretrained AI services, such as **Amazon Rekognition**, vendors do the hard work of building and training the ML models, and users can then use a simple API to automatically extract metadata from images. And, with **Amazon Rekognition Video**, users can also gain these insights from video files. When passed a video file for analysis, the results that are returned include a timestamp of where the object was detected, enabling an index of identified objects to be created.

For example, the following photo could be sent to the Amazon Rekognition service to automatically identify elements in the photo:

Figure 13.4: Photo of a dog and a Jeep in the snow

When passed to Amazon Rekognition, the service can automatically identify objects in the photo. The following is a partial list of the identified objects (with the confidence level of the ML model shown in brackets):

- Outdoors (99.8%)
- Dog (98.3%)
- Winter (96.8%)
- Car (96.6%)
- Snow (90.7%)
- Wheel (87.8%)

A data engineer could use this type of service to build a data pipeline that ingests images and/or video, and then calls the Amazon

Rekognition service for each file, building an index of objects in each file, and storing that in DynamoDB, for example.

Some of the features included in Amazon Rekognition include label detection (such as objects, activities and landmarks), dominant color detection, facial recognition (including facial comparison and search), celebrity detection, unsafe images (used for content moderation), text in images, and more.

The AI services we have discussed so far are used to extract data from unstructured files such as PDF scans and image and video files. Now, let's take a look at AWS AI services that can be used to make predictions based on semi-structured data.

AI for ML-powered forecasts

A common business need is to forecast future values, whether these be the number of staff an entertainment venue is likely to need next month, or how much revenue an organization is likely to receive on a specific product line over the next 12 months.

For many years, organizations would use formulas to forecast future values, based on historical data that they had built up. However, these formulas often did not take into account seasonal trends and other third-party factors that could significantly influence the actual values that are realized.

Modern forecasting tools, such as Amazon Forecast, can provide significantly more accurate forecasts by using the power of ML.

Amazon Forecast

Amazon Forecast is a powerful pretrained AI service for predicting future time series data, based on complex relationships between multiple datasets. Using Forecast, a developer can train and build a customized forecast ML model, without needing ML expertise.

To train the custom model, a user would provide historical data for the attribute that they want to predict (for example, daily sales at each store over the past 12 months). In addition, they can include related datasets, such as a dataset listing the total number of daily visitors to each store.

If the primary dataset also includes geolocation data (identifying, for example, the location of the store) and timezone data, Amazon Forecast can automatically use weather information to help further improve prediction accuracy.

For example, the model can take into account how the weather has affected sales in the past, and use the latest 14-day weather forecast to optimize predictions for the upcoming period based on the weather forecast.

A data engineer may be involved in building a pipeline that uses Amazon Forecast. The following could be some of the steps in a pipeline that the data engineer architects and implements:

1. Use an **AWS Glue** job to create an hourly aggregation of sales for each store, storing the results in Amazon S3.
2. Use **AWS Step Functions** to call Lambda functions that clean up previous predictions, and generate new predictions based on the latest data. Use a Lambda function to create an export job to export the newly generated predictions to Amazon S3.
3. Use **a Redshift COPY** job to load the newly generated predictions from Amazon S3 to Amazon Redshift for further analysis.

> Refer to the AWS blog post titled *Automating your Amazon Forecast workflow with Lambda, Step Functions, and CloudWatch Events rule* (**https://aws.amazon.com/blogs/machine-learning/automating-your-amazon-forecast-workflow-with-lambda-step-functions-and-cloudwatch-events-**

rule/) for more details on building a pipeline that incorporates Amazon Forecast.

AI for fraud detection and personalization

The AI services we discussed previously are often incorporated into data engineering pipelines as these services are useful for advanced analytics (such as extracting metadata from images, text transcripts from audio files, or making forecasts). However, other AI services are often used as a part of transactional systems, rather than data engineering pipelines, which we will briefly look at in this section.

Amazon Fraud Detector

Amazon Fraud Detector is an AI service that helps organizations detect potentially fraudulent transactions and fake account registrations.

Fraud Detector enables an organization to upload its historical data regarding fraudulent transactions. It then adds this to a model trained with fraud data from Amazon and AWS to optimize fraud detection.

Using Fraud Detector, an organization can build fraud prediction into their checkout process, getting a prediction within milliseconds as part of the checkout process.

Amazon Personalize

Amazon Personalize is an AI service that helps organizations provide personalized recommendations to their customers. Using Personalize, developers can easily integrate personalized product recommendations, marketing initiatives, and other ML-powered personal recommendations into existing customer-facing systems.

With Personalize, developers can design systems that capture live events from users (such as data extracted from a website click-stream) and combine this with historical user profile information to recom-

mend the most relevant items for a user. This can be used to recommend other products a customer may be interested in, or the next movie or TV show a customer may like to watch.

Before we get to the hands-on section of this lab, let's have a look at how you can build generative AI solutions (such as those made popular with ChatGPT and Bard) on AWS.

Building generative AI solutions on AWS

In November 2022, a company called OpenAI launched ChatGPT, a chatbot built on a new type of ML technology. This very quickly went viral and became a world-wide sensation, as people were amazed at the ability of this new chatbot to have conversations on just about any topic. OpenAI explained the abilities of this new chatbot as follows:

> *We've trained a model called ChapGPT which interacts in a conversational way. The dialogue format makes it possible for ChaptGPT to answer follow-up questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests.*

Using AWS services, it is possible for anyone to create new solutions or services that take advantage of the same type of technology that powers ChatGPT. But before we get into the AWS services that can be used for generative AI, let's have a brief look at the details of the technology behind these new machine learning services.

Understanding the foundations of generative AI technology

The technology behind ChatGPT is a model known as **Generative Pretrained Transformer** (**GPT**), which is based on the **Transformer**

architecture, a type of neural network designed for processing sequential data, such as text. This technology is an example of a LLM, which is designed to both interpret, and generate, conversational text in natural language.

A **foundational model** (**FM**) is a model, such as ChatGPT, that can be used as a foundation for building more specialized models and applications. Beyond ChatGPT, there are many other companies and organizations that have created FM (such as Jurassic-2 from AI21labs, Claude from Anthopic, and Bard from Google) that can be used to build advanced AI applications. These models have been trained on a massive quantity of data (with many being trained on all publicly available internet sites), enabling them to respond to a wide range of queries and to handle many different tasks. Some models, such as ChatGPT, are primarily used for text based interactions, while others, such as DALL-E and Stable Diffusion, are orientated for visual based tasks (such as generating new images).

Text-based models can be used for use-cases such as effectively summarizing a large amount of text, answering questions on just about any topic, translating between languages, or generating content (such as marketing copy, stories, or even poems and songs). Image-based models can be used for generating images (creating realistic images based on whatever prompt is provided), detecting objects within an image, or modifying an existing image (such as changing the style of an image, or changing the background).

Both text-based and image-based solutions that use this new ML technology can be built on AWS, as we look at in the following sections.

Building on foundational models using Amazon SageMaker JumpStart

Amazon SageMaker JumpStart (which we covered earlier in this chapter) includes options that can accelerate the process of building new

solutions that harness the power of LLMs, as well as large-scale image models. Using JumpStart, you can use either publicly available models, or proprietary FMs, to build new solutions.

When building a custom solution using SageMaker and FM, any data you use to train the model, as well as prompts you provide when using the model, are kept private. This ensures that any data that may be confidential to your organization is not exposed to a service provider, where they may potentially be able to incorporate the data you provided in the training of future versions of their model.

SageMaker JumpStart enables users to easily build solutions using state-of-the-art foundation models for a variety of use cases, such as code generation, summarization, content writing, image generation, and much more. With SageMaker JumpStart, AWS provides and maintains foundation models that can be integrated into your machine learning lifecycle using popular publicly available models (such as HuggingFace, Stable Diffusion, and more) as well as proprietary models that you can subscribe to (such as models from AI21 Labs, Cohere, and LightOn).

For more information on selecting an appropriate model for your use case, see the AWS documentation titled *Choose a foundation model* at `https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-foundation-models-choose.html`.

However, if your use case would benefit from a pre-built solution that is accessible via API, instead of building with SageMaker JumpStart, then keep reading as we explore the Amazon Bedrock service.

Building on foundational models using Amazon Bedrock

In September 2023, AWS launched a new service for building generative AI on AWS called **Amazon Bedrock**. This service makes FMs from

multiple sources available via an API, providing an easy way for customers to build and scale generative AI-based applications. Bedrock includes FMs from leading AI companies like AI21 Labs, Anthropic, Cohere, Meta, Stability AI, and Amazon.

Amazon Bedrock provides a serverless environment, meaning that users do not need to manage any infrastructure. With Bedrock, customers can privately customize (or fine-tune) popular FMs with their own data, and then easily integrate and deploy these models into their existing applications.

Along with the announcement of Amazon Bedrock, Amazon also announced the launch of Amazon Titan Embeddings, an LLM that translates text into a numerical representation, which can be used in ML applications for personalization and search. In addition, Amazon announced the preview of LLMs for text use cases, including Amazon Titan Text Express (offering a balance of price and performance), and Amazon Titan Text Lite (offering an affordable and compact model, ideal for basic tasks and fine-tuning).

Common use cases for LLMs

LLMs by design are very versatile, meaning that this technology can be applied to many different potential use cases. For example, you could create a chatbot application that makes use of LLM technology and acts like an unhappy customer. You could then use this as part of the training for your call center staff, to train them on dealing with unhappy customers and evaluate how well they handle the “customer.”

You could also build a Q&A chatbot that is connected to some of your internal datasets (such as FAQ documents), enabling users to ask questions using natural language and to get answers that come from your official corporate documents.

There are many other common use cases, as well as many use cases that are yet to be discovered. To review some of the common use cases

and understand how you can use Amazon Bedrock for these use cases, see the Amazon Bedrock workshop at https://github.com/aws-samples/amazon-bedrock-workshop.

Having learned more about AWS services for AI and ML, let's now get hands-on with one of the pretrained AI services we discussed earlier in this chapter, Amazon Comprehend.

Hands-on – reviewing reviews with Amazon Comprehend

Imagine that you work for a large hotel chain and have been tasked with developing a process for identifying negative reviews that have been posted on your website. This will help the customer service teams follow up with the customer.

If your company was getting hundreds of reviews every day, it would be time-consuming to have someone read the entire review every time a new review was posted. Luckily, you have recently heard about Amazon Comprehend, so you decide to develop a small **Proof of Concept (PoC)** test to see whether Amazon Comprehend can help.

If your PoC is successful, you will want to have a decoupled process that receives reviews once they have been posted, calls Amazon Comprehend to determine the sentiment of the review, and then takes a follow-up action if the review is negative or mixed. Therefore, you decide to build your PoC in the same way, using Amazon **Simple Queue Service** (**SQS**) to receive reviews and have this trigger a Lambda function to perform analysis with Comprehend.

Setting up a new Amazon SQS message queue

Create a new Amazon SQS message queue for receiving reviews by following these steps:

1. Log in to **AWS Management Console** and navigate to the **Amazon SQS** service at https://console.aws.amazon.com/sqs/v2/. Make sure you are in the same region you have been using for the other hands-on activities in this book.
2. Click on **Create queue**.
3. Leave the defaults for the **Standard** queue as-is and fill in the **Name** field for your queue (such as `website-reviews-queue`):

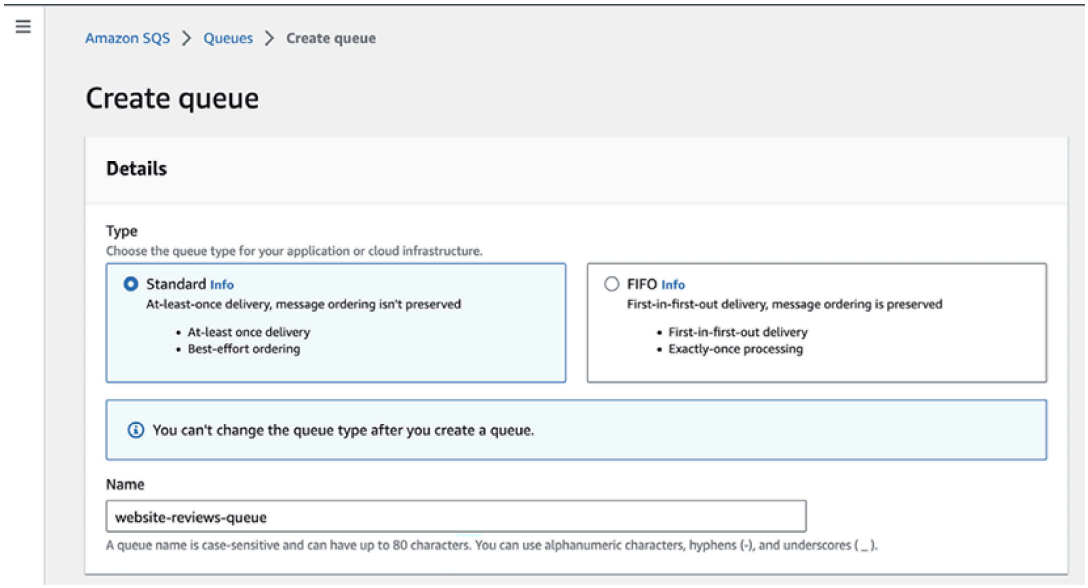

Figure 13.5: Creating a new Amazon SQS message queue

4. Leave all other options as their default values and click on **Create queue** at the bottom of the page.

Now that our queue has been created, we want to create a Lambda function that will read items from the queue and submit the website review text to Amazon Comprehend for analysis.

Creating a Lambda function for calling Amazon Comprehend

The following steps will create a new Lambda function to call Amazon Comprehend to analyze the text that's passed in from the SQS queue:

1. In the Amazon Management Console, navigate to the **AWS Lambda** service at `http://console.aws.amazon.com/lambda/home`.
2. Click on **Create function**.
3. Select the option to **Author from scratch**.
4. Provide a **Function name** value (such as `website-reviews-analysis-function`) and select the most recent version of **Python** for **Runtime**.
5. Expand the **Change default execution role** section, and for **Execution role**, select **Create a new role from AWS policy templates**.
6. Provide a **Role name value** (such as `website-reviews-analysis-role`).
7. For **Policy templates**, search for `SQS` and add **Amazon SQS poller permissions**.
8. Leave everything else as the defaults and click on **Create function**. Having created our function, we can add our custom code, which will receive the SQS message, extract the review text from the message, and then send it to Amazon Comprehend for sentiment and entity analysis.

9. Scroll down a little, and replace the `lambda_function` code, in the **Code source** section, with the following block of code:

```
import boto3
import json
comprehend = boto3.client(service_name='comprehend',
                region_name='us-east-2')
def lambda_handler(event, context):
    for record in event['Records']:
        payload = record["body"]
        print(str(payload))
```

10. In this preceding block of code, we imported the required libraries (`boto3`, and `json`) and initialized the Comprehend API, which is part of `boto3`. Make sure that you modify the preceding

Comprehend API initialization code to reflect the region you are using for these exercises (in the example above, we use `us-east-2`, which is the Ohio region). Then, we defined our Lambda function and read in the records that we received from Amazon SQS. Finally, we loaded `body` of `rescord` into a variable called `payload`. Continue your Lambda function with the following block of code:

```
        print('Calling DetectSentiment')
        response = comprehend.detect_sentiment(Text=payload,
                    LanguageCode='en')
        sentiment = response['Sentiment']
        sentiment_score = response['SentimentScore']
        print(f'SENTIMENT: {sentiment}')
        print(f'SENTIMENT SCORE: {sentiment_score}')
```

In this preceding block of code, we called the Comprehend API for sentiment detection, passed in the review text (`payload`), and specified that the text is in English. In the response we receive from Comprehend, we extracted the `sentiment` property (positive, mixed, or negative), as well as the `SentimentScore` property.

11. Now, let's look at our last block of code:

```
        print('Calling DetectEntities')
        response = comprehend.detect_entities(Text=payload,
                    LanguageCode='en')
        #print(response['Entities'])
        for entity in response['Entities']:
            entity_text = entity['Text']
            entity_type = entity['Type']
```

And finally, to correctly print over two lines, we need the following code:

```
            print(
                f'ENTITY: {entity_text}, '
                f'ENTITY TYPE: {entity_type}'
                )
    return
```

In this final part of our code, we called the Comprehend API for entity detection, again passing in the same review text (`payload`). Multiple entities may be detected in the text, so we looped through the response and printed out some information about each entity.

12. Then, we returned without any error, which indicates success, which means the message will be deleted from the SQS message queue. Note that for a production implementation of this code, you would want to add error-catching code to raise an exception if there were any issues when calling the Comprehend API.
13. Click **Deploy** in the **Lambda** console to deploy your code.

Now, we just need to add permissions to our Lambda function to access the Comprehend API and add our function as a trigger for our SQS queue. Then, we can test it out.

Adding Comprehend permissions for our IAM role

When we created our Lambda function, we were able to select from a preset list of common permissions to add permission for our Lambda function to poll an SQS message queue. However, our function also needs to call the Comprehend API, so let's add permission for that as well:

1. In the **AWS Lambda console**, with your website reviews analysis function open, click on the **Configuration** tab along the top, and then the **Permissions** tab on the left.

2. The name of the role you specified when creating the Lambda function will be shown as a link. Click on **Role name** (such as **website-reviews-analysis-role**) to open the **IAM** console so that we can edit the permissions, as shown below.

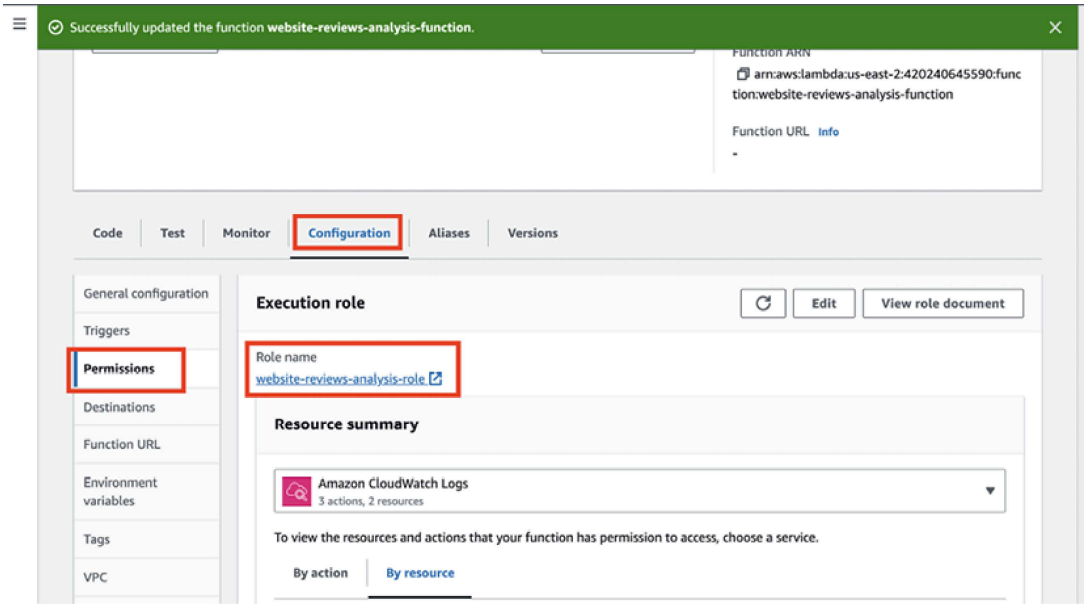

Figure 13.6: Lambda Permissions > Configuration tab > Execution role

3. In the **IAM** console, click on the **Add permissions** drop-down, and then click **Attach policies**.
4. Search for a policy called **ComprehendReadOnly**, which has sufficient permissions to enable us to call the Comprehend API from our Lambda function.
5. Select the checkbox for **ComprehendReadOnly**, and then click on **Add permissions**.

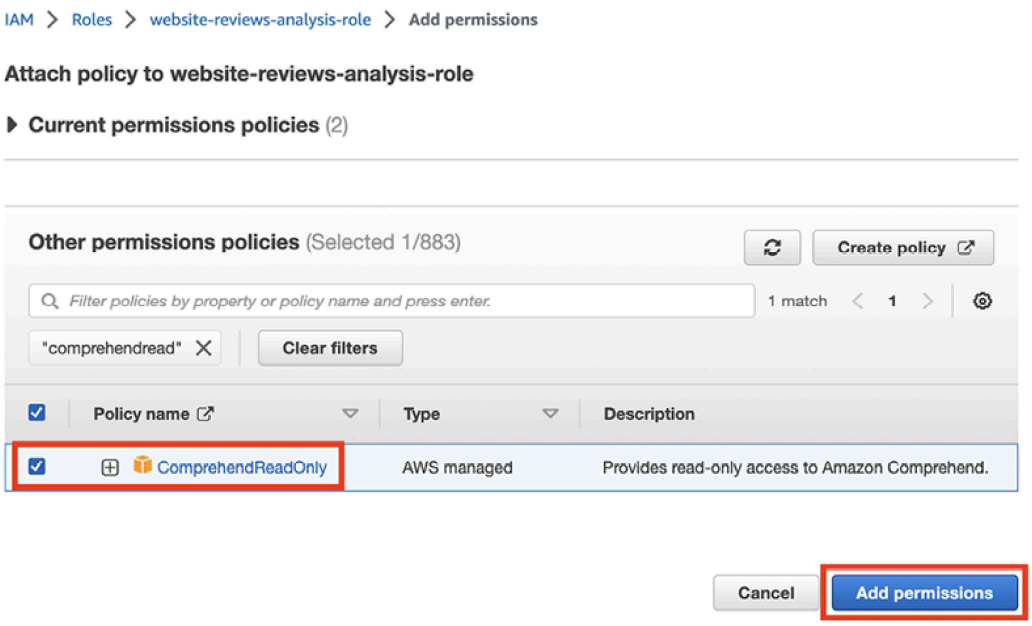

Figure 13.7: Finding and selecting the required Comprehend permissions in IAM

We are just about ready to test our function. Our last step is to link our SQS queue and our Lambda function.

Adding a Lambda function as a trigger for our SQS message queue

With the following steps, we'll configure our Lambda function to be able to pick up new messages that are added to our SQS message queue for processing:

1. Navigate back to the **Amazon SQS** message queue console at `https://console.aws.amazon.com/sqs/v2/`.
2. Click on the name of the SQS queue you previously created (such as **website-reviews-queue**).
3. Click on the **Lambda triggers** tab, and then click **Configure Lambda function trigger**.
4. Make sure that **Region** is set to the region you have been using for the exercises in this chapter, and then select your Lambda function

from the drop-down list (such as `website-reviews-analysis-function`).

5. Click **Save** to link your SQS queue and Lambda function.

And with that, we are now ready to test out our solution and see how Amazon Comprehend performs.

Testing the solution with Amazon Comprehend

Using the following steps, test the solution and get Amazon Comprehend to analyze the text you have provided for both sentiment and entity detection:

1. Ensure that you are still on the **Amazon SQS** console and that your SQS queue is open.
2. At the top right, click on **Send and receive messages**:

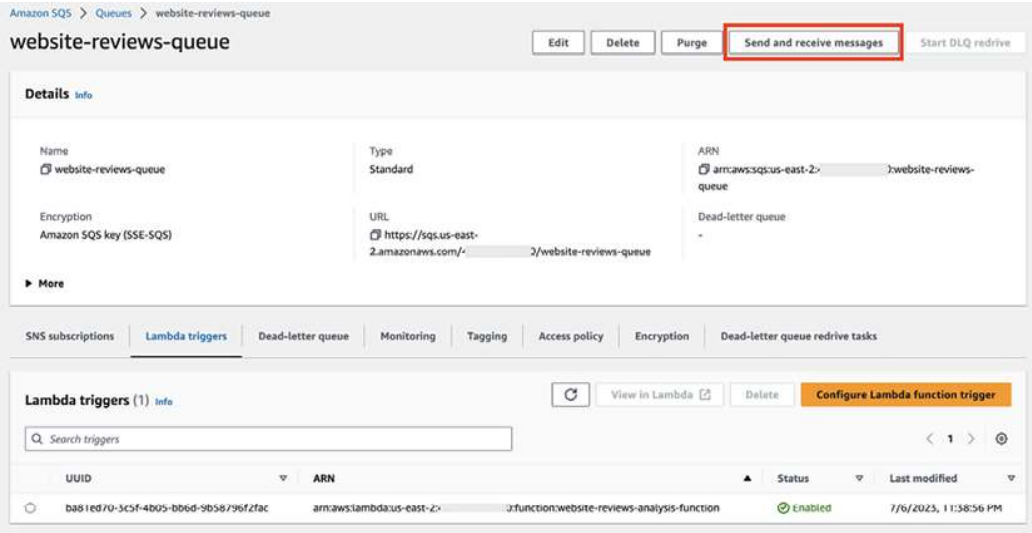

Figure 13.8: Amazon SQS queue detail view

We can now send a message directly to our SQS queue, which will trigger our Lambda function to process the message and send it to Amazon Comprehend. When moved to production, we would build integration into our website to automatically send all new reviews to our Amazon SQS message queue as the reviews are posted.

3. Paste the following text (or your own, similar text) into the **Message Body** section of **Send and receive messages**:
   "I recently stayed at the Kensington Hotel in downtown Cape Town and was very impressed. The hotel is beautiful, the service from the staff is amazing, and the sea views cannot be beaten. If you have the time, stop by Elizabeth's Kitchen, a coffee shop not far from the hotel, to get a coffee and try some of their delicious cakes and baked goods."

4. Then, click on the **Send message** option at the top right.
5. To view the results of the Comprehend analysis, we can review the output of our Lambda function in CloudWatch Logs. If it's not already open in a separate browser tab, open a new browser tab and navigate back to your **Lambda function**. Click on the **Monitor** tab, and then click **View CloudWatch logs**. This will open the **CloudWatch** console in a new browser tab.
6. The **CloudWatch** console should have opened at the log group for your Lambda function (for example, the log group named `/aws/lambda/website-reviews-analysis-functions`). Click on the latest log stream to open the log, which should look similar to the following screenshot.

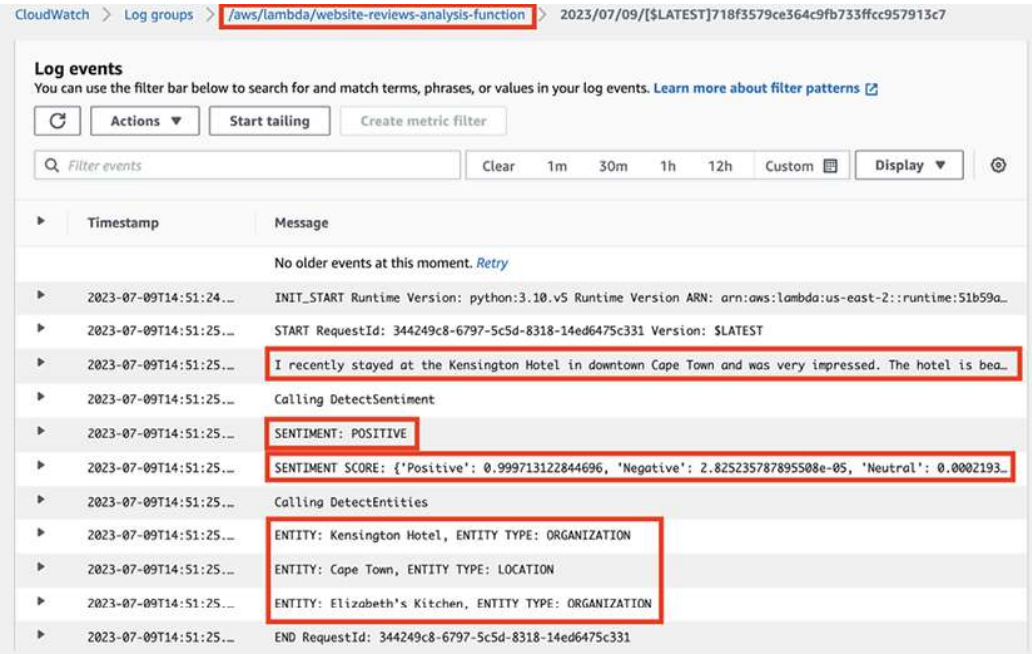

Figure 13.9: Amazon CloudWatch logs for our Lambda function
In the CloudWatch logs, you can see the output of our Lambda function. This includes the text that was analyzed, the sentiment (**POSITIVE**), the sentiment score, as well as the three entities Comprehend detected in our text (the hotel and coffee shop names, and the city location).

7. Go back to your browser tab for the **Amazon SQS console** and click **Clear content** and then provide a negative review for the message body. You can either write your own fictional negative review or copy and paste a negative review that you find via Google. Send the message via SQS and review the analysis results in CloudWatch to see how Comprehend detects the negative sentiment, and see which other entities Comprehend can detect.

After testing and validating that Amazon Comprehend can reliably detect sentiment from published reviews, you may decide to move forward with implementing this solution in production. If you do decide to do this, you could use Amazon Step Functions to build a workflow that runs a Lambda function to do the sentiment analysis. Then, depending on the results (positive, negative, neutral, or mixed), the Step Function state machine could run different Lambda functions based on the next steps (such as sending a negative review to customer service to follow up with the customer or sending a mixed review to a manager to decide on the next steps).

With this hands-on exercise, you got to experiment with how Amazon Comprehend can detect both sentiment and entitles in written text. If you have time, you can explore the functionality of other Amazon AI services directly in the console. This includes Amazon Rekognition, Amazon Transcribe, Amazon Textract, and Amazon Translate.

Summary

In this chapter, you learned more about the broad range of AWS ML and AI services that AWS provides, and had the opportunity to get

hands-on with Amazon Comprehend, an AI service for extracting insights from written text.

We discussed how ML and AI services can apply to a broad range of use cases, both specialized (such as detecting cancer early) and general (business forecasting or personalization).

We examined different AWS services related to ML and AI. We looked at how different Amazon SageMaker capabilities can be used to prepare data for ML, build models, train and fine-tune models, and deploy and manage models. SageMaker makes building custom ML models much more accessible to developers without existing expertise in ML.

We then looked at a range of AWS AI services that provide prebuilt and trained models for common use cases. We looked at services for transcribing text from audio files (Amazon Transcribe), for extracting text from forms and handwritten documents (Amazon Textract), for recognizing images (Amazon Rekognition), and for extracting insights from text (Amazon Comprehend). We also briefly discussed other business-focused AI services, such as Amazon Forecast and Amazon Personalize.

Finally, we had a brief look at how you can build generative AI solutions based on Foundation Models, using either Amazon SageMaker JumpStart, or the Amazon Bedrock service (a new service that AWS has announced, but not yet released as of the time of writing).

We're near the end of a journey that has had us look, at a high level, at several tasks, activities, and services that are part of the life of a data engineer. In the final part of this book, we will bring a lot of the previous concepts together and look at how you can build a modern data platform on AWS.

Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

https://discord.gg/9s5mHNyECd