Invoke Bedrock model for text generation using zero-shot prompt

> *This notebook should work well with the `Python 3` kernel in SageMaker Studio*

Introduction

In this notebook we show you how to use a LLM to generate an email response to a customer who provided negative feedback on the quality of customer service that they received from the support engineer.

We will use Bedrock's Amazon Titan Text large model using the Boto3 API.

The prompt used in this example is called a zero-shot prompt because we are not providing any examples of text alongside their classification other than the prompt.

Note: *This notebook can be run within or outside of AWS environment*.

Context

To demonstrate the text generation capability of Amazon Bedrock, we will explore the use of Boto3 client to communicate with Amazon Bedrock API. We will demonstrate different configurations available as well as how simple input can lead to desired outputs.

Pattern

We will simply provide the Amazon Bedrock API with an input consisting of a task, an instruction and an input for the model under the hood to generate an output without providing any additional example. The purpose here is to demonstrate how the powerful LLMs easily understand the task at hand and generate compelling outputs.

Use case

To demonstrate the generation capability of models in Amazon Bedrock, let's take the use case of email generation.

Persona

You are Bob a Customer Service Manager at AnyCompany and some of your customers are not happy with the customer service and are providing negative feedbacks on the service provided by customer support engineers. Now, you would like to respond to those customers humbly aplogizing for the poor service and regain trust. You need the help of an LLM to generate a bulk of emails for you which are human friendly and personalized to the customer's sentiment from previous email correspondence.

Implementation

To fulfill this use case, in this notebook we will show how to generate an email with a thank you note based on the customer's previous email.We will use the Amazon Titan Text Large model using the Amazon Bedrock API with Boto3 client.

Setup

⚠️ ⚠️ ⚠️ Before running this notebook, ensure you've run the Bedrock basics notebook notebook. ⚠️ ⚠️ ⚠️

In [1]:
```
!pip install boto3
```

```
Requirement already satisfied: boto3 in /home/studio-lab-user/.conda/envs/default/li
b/python3.9/site-packages (1.35.68)
Requirement already satisfied: s3transfer<0.11.0,>=0.10.0 in /home/studio-lab-user/.
conda/envs/default/lib/python3.9/site-packages (from boto3) (0.10.4)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /home/studio-lab-user/.cond
a/envs/default/lib/python3.9/site-packages (from boto3) (1.0.1)
Requirement already satisfied: botocore<1.36.0,>=1.35.68 in /home/studio-lab-user/.c
onda/envs/default/lib/python3.9/site-packages (from boto3) (1.35.68)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /home/studio-lab-user/.cond
a/envs/default/lib/python3.9/site-packages (from botocore<1.36.0,>=1.35.68->boto3)
(1.26.20)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /home/studio-lab-use
r/.conda/envs/default/lib/python3.9/site-packages (from botocore<1.36.0,>=1.35.68->b
oto3) (2.9.0)
Requirement already satisfied: six>=1.5 in /home/studio-lab-user/.conda/envs/defaul
t/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.36.0,>=
1.35.68->boto3) (1.16.0)
```

In [2]:
```
%pip install awscli
```

```
Requirement already satisfied: awscli in /home/studio-lab-user/.conda/envs/default/l
ib/python3.9/site-packages (1.36.9)
Requirement already satisfied: colorama<0.4.7,>=0.2.5 in /home/studio-lab-user/.cond
a/envs/default/lib/python3.9/site-packages (from awscli) (0.4.6)
Requirement already satisfied: docutils<0.17,>=0.10 in /home/studio-lab-user/.conda/
envs/default/lib/python3.9/site-packages (from awscli) (0.16)
Requirement already satisfied: botocore==1.35.68 in /home/studio-lab-user/.conda/env
s/default/lib/python3.9/site-packages (from awscli) (1.35.68)
Requirement already satisfied: PyYAML<6.1,>=3.10 in /home/studio-lab-user/.conda/env
s/default/lib/python3.9/site-packages (from awscli) (6.0.1)
Requirement already satisfied: s3transfer<0.11.0,>=0.10.0 in /home/studio-lab-user/.
conda/envs/default/lib/python3.9/site-packages (from awscli) (0.10.4)
Requirement already satisfied: rsa<4.8,>=3.1.2 in /home/studio-lab-user/.conda/envs/
default/lib/python3.9/site-packages (from awscli) (4.7.2)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /home/studio-lab-user/.cond
a/envs/default/lib/python3.9/site-packages (from botocore==1.35.68->awscli) (1.26.2
0)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /home/studio-lab-user/.cond
a/envs/default/lib/python3.9/site-packages (from botocore==1.35.68->awscli) (1.0.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /home/studio-lab-use
r/.conda/envs/default/lib/python3.9/site-packages (from botocore==1.35.68->awscli)
(2.9.0)
Requirement already satisfied: six>=1.5 in /home/studio-lab-user/.conda/envs/defaul
t/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore==1.35.68->
awscli) (1.16.0)
Requirement already satisfied: pyasn1>=0.1.3 in /home/studio-lab-user/.conda/envs/de
fault/lib/python3.9/site-packages (from rsa<4.8,>=3.1.2->awscli) (0.6.1)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
!mkdir ~/.aws
```

```
mkdir: cannot create directory '/home/studio-lab-user/.aws': File exists
```

In [4]:
```
%%writefile ~/.aws/credentials

[default]
aws_access_key_id=AKIA45PU34QGNTCSUR4W
aws_secret_access_key=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
```

```
Overwriting /home/studio-lab-user/.aws/credentials
```

In [5]:
```
%%writefile ~/.aws/config

[default]
region=us-east-1
```

```
Overwriting /home/studio-lab-user/.aws/config
```

In [7]:
```
import json
import os
import sys

import boto3
import botocore

boto3_bedrock = boto3.client("bedrock-runtime")
# - use this for with profile
```

Generate text

Following on the use case explained above, let's prepare an input for the Amazon Bedrock service to generate an email. Note that this prompt would need to be modified with Human:/Assistant: formatting for Claude.

In [8]:
```
# create the prompt
prompt_data = """
Command: Write an email from Bob, Customer Service Manager, to the customer "John D
who provided negative feedback on the service provided by our customer support
engineer"""
```

Let's start by using the Amazon Titan Large model. The Amazon Titan family of models support a large context window of up to 32k tokens and accepts the following parameters:

- `inputText`: Prompt to the LLM
- `textGenerationConfig`: These are the parameters that model will take into account while generating the output.

In [27]:
```
body = json.dumps(
    {
        "inputText": prompt_data,
        "textGenerationConfig": {"topP": 0.95, "temperature": 0.5},
    }
)
```

The Amazon Bedrock API provides you with an API `invoke_model` which accepts the following:

- `modelId`: This is the model ARN for the various foundation models available under Amazon Bedrock
- `accept`: The type of input request
- `contentType`: The content type of the output
- `body`: A json string consisting of the prompt and the configurations

Check documentation for Available text generation model Ids

Invoke the Amazon Titan Text language model

First, we explore how the model generates an output based on the prompt created earlier.

Complete Output Generation

In [24]:
```
# modelId = 'amazon.titan-text-premier-v1:0' # Make sure Titan text premier is avai
modelId = "amazon.titan-tg1-large"
accept = "application/json"
contentType = "application/json"
outputText = "\n"
```

```
try:

    response = boto3_bedrock.invoke_model(
        body=body, modelId=modelId, accept=accept, contentType=contentType
    )
    response_body = json.loads(response.get("body").read())

    outputText = response_body.get("results")[0].get("outputText")

except botocore.exceptions.ClientError as error:

    if error.response["Error"]["Code"] == "AccessDeniedException":
        print(
            f"\x1b[41m{error.response['Error']['Message']}\
                \nTo troubeshoot this issue please refer to the following resources
                 \nhttps://docs.aws.amazon.com/IAM/latest/UserGuide/troubleshoot_ac
                 \nhttps://docs.aws.amazon.com/bedrock/latest/userguide/security-ia
        )

    else:
        raise error
```

In [25]:
```
# The relevant portion of the response begins after the first newline character
# Below we print the response beginning after the first occurence of '\n'.

email = outputText[outputText.index("\n") + 1 :]
print(email)
```

```
Bob, Customer Service Manager, has been with the company for over 15 years and has y
ears of experience in resolving customer issues. He is the point of contact for any
customer feedback or complaints.
Here is the email from Bob:

John Doe

Customer Service Manager

[Company Name]

Dear Mr. Doe,

I would like to express my regret for the poor service you received from our custome
r support engineer. We pride ourselves on our customer service, and we take feedback
from our customers seriously.

We are currently investigating the issue, and we will take steps to prevent this fro
m happening again. We apologize for any
```

Streaming Output Generation

Above is an example email generated by the Amazon Titan Large model by understanding the input request and using its inherent understanding of the different modalities. This request to the API is synchronous and waits for the entire output to be generated by the model.

Bedrock also supports that the output can be streamed as it is generated by the model in form of chunks. Below is an example of invoking the model with streaming option. `invoke_model_with_response_stream` returns a `ResponseStream` which you can read from.

You may want to enable scrolling on your output cell below:

In [26]:
```
output = []
try:

    response = boto3_bedrock.invoke_model_with_response_stream(
        body=body, modelId=modelId, accept=accept, contentType=contentType
    )
    stream = response.get("body")

    i = 1
    if stream:
        for event in stream:
            chunk = event.get("chunk")
            if chunk:
                chunk_obj = json.loads(chunk.get("bytes").decode())
                text = chunk_obj["outputText"]
                output.append(text)
                print(f"\t\t\x1b[31m**Chunk {i}**\x1b[0m\n{text}\n")
                i += 1

except botocore.exceptions.ClientError as error:

    if error.response["Error"]["Code"] == "AccessDeniedException":
        print(
            f"\x1b[41m{error.response['Error']['Message']}\
                \nTo troubeshoot this issue please refer to the following resources
                 \nhttps://docs.aws.amazon.com/IAM/latest/UserGuide/troubleshoot_ac
                 \nhttps://docs.aws.amazon.com/bedrock/latest/userguide/security-ia
        )

    else:
        raise error
```

```
                **Chunk 1**

Dear John Doe,
Thank you for providing feedback regarding your recent experience with our customer
support. I apologize that we did not meet your expectations. We take customer feedba
ck seriously and will review your case to identify areas that require

                **Chunk 2**
 improvement.

Please be assured that we will make every effort to provide you with the best servic
e possible. Could you please provide more details about the issues you encountered?
Any additional information you can provide will help us to further understand your c
oncerns.

We value your opinion and want to make sure we have the opportunity to address your
issues. Our goal is to improve our service and exceed your expectations. Once again,
thank
```

The above helps to quickly get output of the model and let the service complete it as you read. This assists in use-cases where there are longer pieces of text that you request the model to generate. You can later combine all the chunks generated to form the complete output and use it for your use-case

Conclusion

You have now experimented with using `boto3` SDK which provides a vanilla exposure to Amazon Bedrock API. Using this API you have seen the use case of generating an email responding to a customer due to their negative feedback.

Take aways

- Adapt this notebook to experiment with different models available through Amazon Bedrock such as Anthropic Claude and AI21 Labs Jurassic models.
- Change the prompts to your specific usecase and evaluate the output of different models.
- Play with the token length to understand the latency and responsiveness of the service.
- Apply different prompt engineering principles to get better outputs.

Thank You

```
In [ ]:
```