SRA の技術情報誌
GS LetterNeo
Vol.199　2026 年 1 月

生成 AI を利用した ヘルプデスクチャットボット with RAG and MCP

三島 健

技術本部 技術開発室

1 はじめに

本稿では、生成 AI を利用したヘルプデスク用のチャットボットの構築を行い、得られた知見をまとめました。最初に、Amazon Bedrock を使ってチャットボットを構築しました。Bedrock は、Amazon や主要な AI 企業が提供する基盤モデル（FM）を統合 API を通じて利用できる完全マネージド型サービスです。チャットボットは新たな質問を受け付けると、過去のヘルプデスク業務で行われた質問と回答で作ったナレッジベースから関連情報を得て、MCP（Model Context Protocol）サーバから得られる公式情報からも情報を得てドラフト（回答の下書き）を作成し、質問者へ返します。ここで MCP とは、LLM が外部のデータソースと安全かつ標準化された方法でやり取りするための仕組み（プロトコル）で、このプロトコルを使ってエージェントは MCP サーバから情報を得る事ができます。Bedrock で私が作成したチャットボットは、現在、弊社のヘルプデスク部門で実際に使っており、できるだけ早急に正確な回答を少ない作業量で質問者へ返す事に貢献しています。しかし実際に使い続けてもらううちに、新たな要求が出てきました。そこで、それをどうやって実現できるかも検討し別のプロトタイプも作成しました。このプロトタイプは LangChain を使って実装しました。LangChain は LLM を活用したアプリケーション開発を簡単に実装するためのフ

レームワークです。本稿ではポイントのみを解説しますが、詳細は私が書いた引用文献に記載してありますので、興味を持たれた方はそちらもご覧ください。

2 ヘルプデスク部門からの要求条件

弊社のマイクロソフト社製品のヘルプデスクを担当している部門が欲しいチャットボットは以下のような要求条件を満たすものでした。

(1) 過去の質問と回答を参考にドラフトを生成して欲しい。つまり、過去に類似の質問があった場合は過去の回答をベースにドラフトを生成して欲しい。ここで「ドラフト」とは質問者へ返す回答の下書きのことです。チャットボットが生成したドラフトはヘルプデスクの担当者がチェックを行い、ドラフトに修正すべき箇所があった場合は修正を施してから回答を質問者へ返します。

(2) セキュリティを担保してほしい：①過去の質問と回答を AWS にアップロードした場合外部へ漏洩することは防ぎたい，②担当者以外はチャットボットにアクセスできないように制限したい。

(3) マイクロソフト社の公式情報も参考にしてドラフトを生成して欲しい。

(4) 普段 Microsoft teams を業務で使っているので UI が teams だと使いやすい。

3 Amazon Bedrock を使ったチャットボット

２章の要求条件を満たすように、以下のように検討して Bedrock を使って実装しました。AWS のコンソールを使って以下で説明するナレッジベースや MCP サーバとのアクセス機能をエージェントに追加します。エージェントはユーザの指示を理解し LLM を使いタスクを自動化するオーケストレーション機能です。

3.1 ナレッジベース

まずは過去の質問と回答からナレッジベース（知識ベース）を作成します。ナレッジベースとは、組織内に存在する知識や情報を一元化し、簡単に検索しやすい状態にしたデータベースです。入力する質問に意味的に近い過去の質問と回答を探したいため、類似度検索ができるデータベースシステムを利用します。このために、過去の質問と回答を数値ベクトルに変換し（ベクトル化）、その数値をベクトルデータベースに保存します。ベクトル化には

Amazon Titan Text Embeddings V2 を使いました。ベクトルデータベースは OpenSearch Serverless を使いました。

3.2　セキュリティ

前記 Amazon Titan Text Embeddings V2 や OpenSearch Serverless を使うためには過去の質問と回答を保持したファイル（ワード、CSV、PDF など）を S3 へアップロードしておく必要があります。S3 は AWS のクラウド上でデータを安全に保存・管理できるストレージサービスです。S3 を適切に設定すれば、高いセキュリティを実現できます。S3 へのアップロードには、https による暗号化通信を使用します。また、S3 への保存は、デフォルトでは暗号化されて保存されます。つまり、デフォルトではデータは暗号化されますしそもそも外部からアクセスできないようになっています。これを間違えて解除しないように適切に使っていれば問題ないです。

また、IAM を使って誰がこのチャットボットにアクセスできるかを細かく制御することもできます。IAM とは AWS リソースへのアクセス権限を安全に管理するサービスです。今回は teams を使いますので、teams 側でもユーザのアクセス制御は可能になります。

3.3　MCP サーバとの接続

チャットボットはナレッジベースのみならず、マイクロソフト社の公式情報も参考にしてドラフトを作成して欲しいという要求を満たす実装について考えます。

一つの選択肢はマイクロソフト社の公式情報を公開しているホームページからナレッジベースを作成する方法です。この場合、公式情報が変更されホームページの内容に変更があった場合には、ナレッジベースを作り直さなければならず、手間（運用コスト）がかかります。

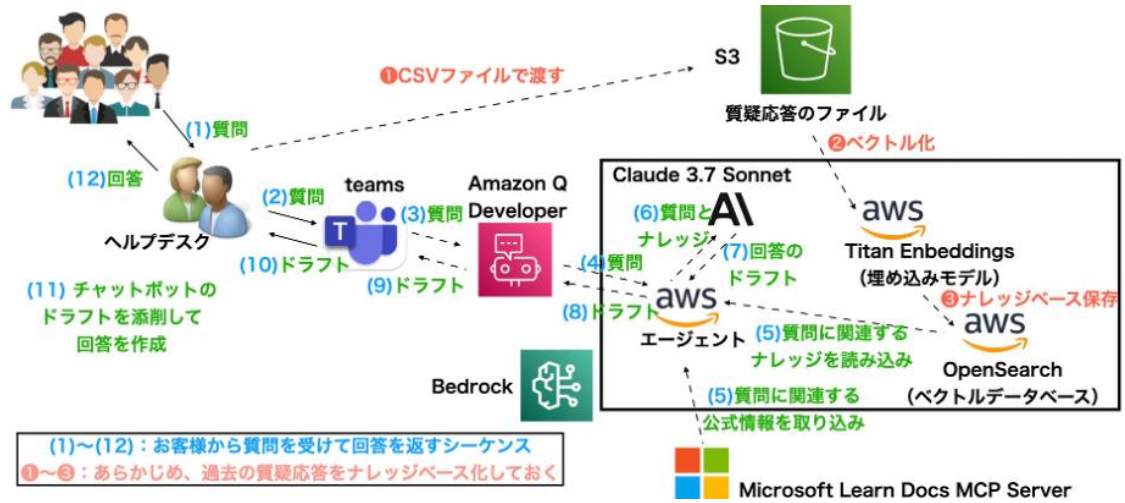
もう一つの選択肢は、マイクロソフト社が公開している MCP サーバを利用する方法です。これは LLM が利用できるマイクロソフト社の情報を返してくれるサーバで、無料で利用することができます。公式情報に変更があれば、その直後から最新の情報を返してくれます。MCP サーバを利用する方がメリットは大きいと考えたため、MCP サーバを使う方法を選択しました。

しかし、Bedrock には MCP サーバと直接やり取りする手段がありません。そこで、Bedrock のエージェントは ActionGroup に設定した Lambda 関数を実行してその情報を取得する機能を使います。ActionGroup とは実行可能なアクションのリストであり、Lambda

関数はアクションを実際に処理するコードです。Lambda 関数の中に MCP クライアントを実装しました。詳細は文献[2]をご覧ください。

3.4　全体構成

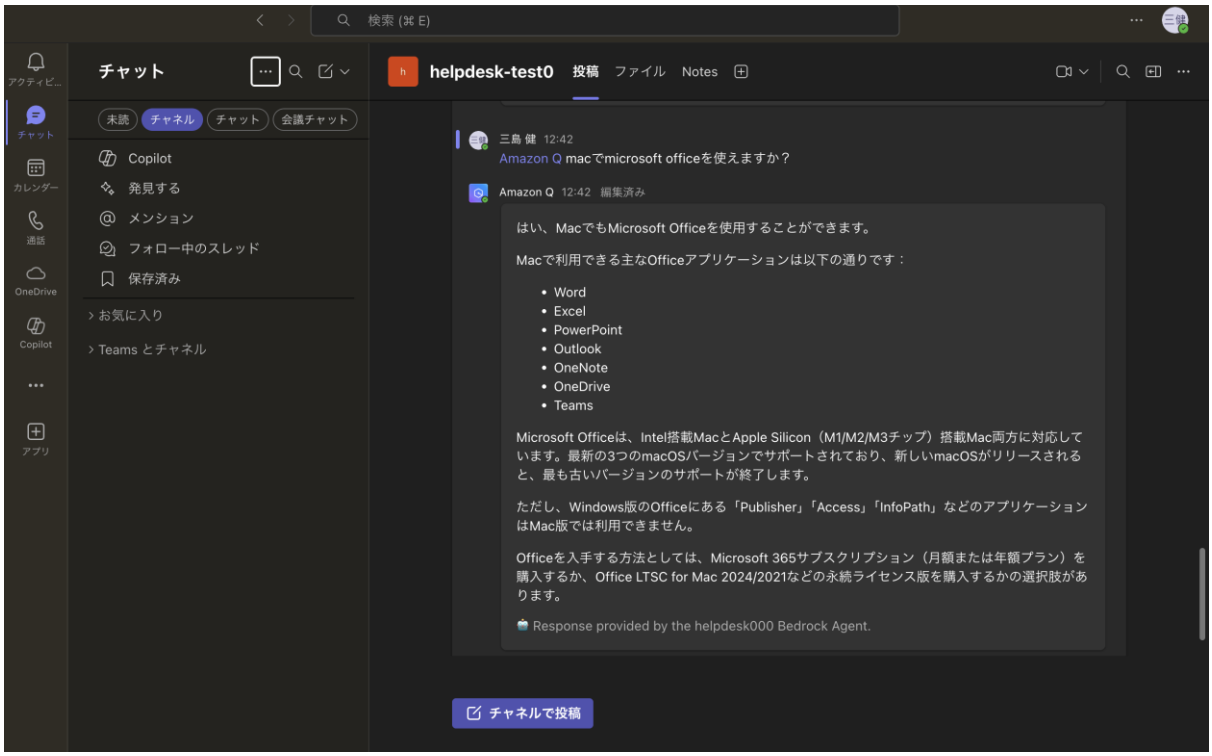
上記 3.1 から 3.3 の検討結果から以下の構成としました。なお、上記 3.1 から 3.3 は要点のみの説明であるため、詳細の検討は文献[1]をご覧ください。上記で説明した Titan と OpenSearch の他に、LLM は Claude 3.7 Sonnet を使いました。エージェントと UI である teams は Amazon Q Developer で接続しました。



4 ヘルプデスク部門で使用中。そして、新たな課題が発生する

現在、ヘルプデスク部門で継続して使用してもらい、小さなトラブルはありましたが、早急に正確な回答を少ない作業量で質問者へ返す事に貢献できて良かったと思っています。トラブルの詳細は文献[1]をご覧ください。

以下のスクリーンショットは teams の画面で本チャットボットを使ってみた例です。まるで teams の他のメンバにメッセージを送るのと同様に、「Amazon Q」というメンバに「mac で microsoft office を使えますか？」と質問を送りました。すると、本チャットボットはナレッジベースの情報と MCP サーバからの情報を使って回答を表示してくれました。



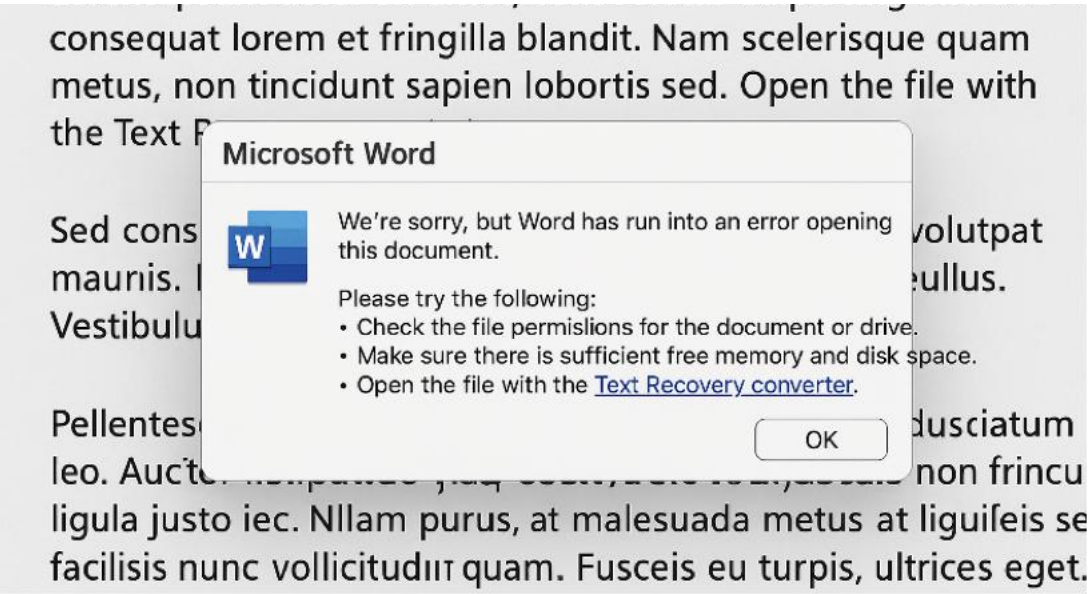
しかし、実際に使っていると新たな課題や新たな要求が生まれました。その中で最も困難な課題は添付画像付きの質問でした。次章で詳しく説明します。

5 新たな課題とは？

３章で説明したヘルプデスクのチャットボットは文章のみの質問を受け付けてドラフトを返すものです。しかし、文章のみではなく、添付画像が付いた質問もあります。

例えば、次のページの画像を添付して、「このエラーの原因と解決策を教えてください。」という質問文です。この画像はワードファイルを開こうとしたら正常に開けずに「We’re sorry, but Word has run into an error opening this document.」というエラーメッセージが出たトラブルです。この画像は Microsoft copilot に生成してもらいました。チャットボットにはこのトラブルの解決方法を回答して欲しいです。３章で作ったチャットボットは文章しか入力できないため、上記質問文のみだと、チャットボットは「このエラー」とは何なのか理解できず、正常な応答が返せません。そこで、最初に「ワードファイルを開こうとしたら、「We’re sorry, but Word has run into an error opening this document.」というエラーメッセージが出ました。このエラーの原因と解決策を教えてください。」という統合した質問を生成した後で、この質問に対するドラフトを生成する必要があります。

ところが、現在（２０２５年）の Bedrock では、エージェントに画像を渡すインターフェースがありません。また、直接エージェントのコンソールで画像を取り込もうとしても、そのような機能はありません。従って、Bedrock では実現は難しいことが分かりました。そこで、LangChain を使って実現する方法を検討しました。



6 LangChain を使ったチャットボット

LangChain は LLM と外部データやツールとを有機的に組み合わせて高度な機能を実現するためのオープンソースのフレームワークです。Bedrock は LLM などの基盤モデルを安全で簡単に利用することができるサービスですが、LangChain は LLM を利用するだけでなく複雑なアプリケーションを柔軟に開発することができます。Bedrock では Bedrock 上で利用できるように設定された基盤モデルしか利用できませんが、LangChain は AWS、Azure, Google、さらには OSS も利用可能です。私は最初は Ollama という LLM の OSS を使っていましたが、私が使えるマシン環境が貧弱で動作が遅かったため、最終的には Bedrock が提供している Claude 4.5 Haiku を LLM として使いました。ベクトルデータベースは OSS の Chroma を使いました。つまり、ローカルで動作する OSS と AWS とを LangChain で統合しています。

6.1　方針

５章で説明した、添付画像付きの質問にチャットボットが正しく回答するためには、以下の機能が必要だと考えました。

(a) 画像から重要な文章（例えば、エラーメッセージ）を取り出すこと
(b) 取り出した文章と元の質問を統合して自然な文章の質問を生成すること
(c) 上記(b)で統合した質問を使ってナレッジベースから関連情報を取り出すこと
(d) 必要ならば、上記(b)で統合した質問を使って MCP サーバから関連情報を取り出すこと
(e) 上記(c)および(d)を使ってドラフトを生成すること

6.2　画像から重要な文章（例えば、エラーメッセージ）を取り出すために LLM を使う

以前から OCR という技術は存在しました。しかし、ここでは単なる OCR では不足です。サンプル画像をご覧ください。たくさんの文字が書いてあります。OCR では、全ての文字を拾ってしまいます。ここでは重要な文章だけを取り出して欲しいのです。つまり、たくさんの文字列の中からどの文章が重要かを判断してもらい、その文章だけを出力して欲しいのです。そこで、LLM を使いました。LLM にはあらかじめシステムプロンプト（LLM がどんなルールに従うかの定義）に「重要な文章、特にエラーメッセージ、を抽出してください。」と設定しておきます。すると LLM は「We’re sorry, but Word has run into an error opening this document.」が重要だと判断し出力します。

6.3　取り出した文章と元の質問を統合して自然な文章の質問を生成するために LLM を使う

6.2 で取り出した文章と元の質問文「このエラーの原因と解決策を教えてください。」を統合して自然な質問文を LLM に生成してもらいます。以下が LLM に生成してもらった統合された質問文です。

> Microsoft Wordでドキュメントを開く際に「We're sorry, but Word has run into an error opening this document.」というエラーが表示されるのですが、この原因と解決策を教えてください。

6.4 統合された質問文に関する情報をナレッジベースから取り出す

3.1 で説明したようにナレッジベースを作るためにはベクトルデータベースが必要です。Bedrock を使って構築する場合は Bedrock が提供している OpenSearch を使いました。LangChain の良いところの一つは、AWS, Azure, Google, OSS など様々なサービスを利用できる事なので、ここでは OSS の Chroma を使ってみました。

6.5 統合された質問文に関する情報を MCP サーバから取り出すために、LLM を使ったエージェントを使う

LangChain のエージェントは LLM を使って MCP サーバのどのツールを使うかを選定し、そのツールに必要な引数を生成し、実行結果を解釈し整理する機能を有しています。これを利用すれば、Bedrock の Lambda 関数 [2] のような煩雑なコーディングは必要ありません。

6.6 ナレッジベースからの情報と MCP サーバからの情報からドラフトを生成してもらうために LLM を使う

最後に、ナレッジベースからの情報と MCP サーバからの情報から LLM にドラフトを生成してもらいます。上記検討結果をまとめて python で実装しました。

6.7 実行結果

実行はローカル環境で以下のようにして行いました。--question オプションの後に質問文を記述し、--image オプションの後に添付ファイル名を指定します。--image オプションは省略することが可能で、省略した場合は 3 章の bedrock 版と同様、このチャットボットは質問文のみからドラフトを生成します。プログラムの中身の解説は省略しますが、このプログラムの中には AWS の SDK を使って Bedrock の LLM である Claude 4.5 Haiku を利用するコードが含まれています。ベクトルデータベースはローカル環境で動作させています。

```
% python3 langchain-chatbot.py --question “このエラーの原因と解決策を教えてください。” --image screenshots/word-error.jpeg
```

実行結果は以下の通りです。Markdown 形式で出力してもらいました。チャットボットは Word ファイルが開けないトラブルだと認識し、その解決方法のドラフトを生成できた事がわかります。

Microsoft Wordドキュメント開封エラーの対処方法

ご質問ありがとうございます。提供いただいた情報に基づいて、段階的な対処方法をご案内いたします。

【第1段階】基本的な確認事項

1. **ファイルのアクセス権限確認**
   - ドキュメントのプロパティを確認し、読み取り専用になっていないか確認
   - ファイルが暗号化されていないか確認（Windows Explorerで「E」マークの有無）
2. **ディスク容量の確認**
   - C:ドライブの空き容量が十分か確認（最低1GB以上推奨）
   - メモリ使用率を確認
3. **ファイルパス長の確認**
   - ファイルパスが259文字以内か確認
   - 長い場合は、より短いパスに移動してから開く

【第2段階】Word固有の対処方法

1. **「開く」と「修復」機能を使用**
   - Word起動 → ファイル → 開く → 対象ファイルを右クリック → 「開いて修復」を選択
2. **テキスト回復コンバーターを使用**
   - ファイル → 開く → ファイルの種類を「テキスト回復コンバーター」に変更
3. **ファイル形式の変更**
   - ファイルを別の形式（.docx → .doc など）で保存し直す

【第3段階】システムレベルの対処方法

1. **クリーンブート環境での動作確認**
   - 不要なスタートアップアプリを無効化して再起動
   - 常駐ソフトウェアの干渉を排除
2. **別ネットワークでの再現性確認**
   - 社内ネットワークを経由しないネットワークで試行
3. **別Windowsプロファイルでの再現性確認**
   - 別のユーザーアカウントでログインして試行

【第4段階】Office再インストール

上記で改善しない場合：

- コントロールパネル → プログラムと機能 → Officeをアンインストール
- https://portal.office.com/account#installs から再インストール

ご不明な点やエラーが改善しない場合は、以下の情報をお知らせください：

- エラーメッセージの全文とエラーコード
- 問題が発生するファイルの保存場所
- 上記で試行済みの対処方法

ご不便をおかけして申し訳ございません。ご協力よろしくお願いいたします。

このチャットボットは (a) 画像から重要な文章を取り出す、(b) 取り出した文章と元の質問を統合して自然な文章の質問を生成する、(d) 上記(b)で統合した質問を使って MCP サーバから関連情報を取り出す、(e) 得られた情報を使ってドラフトを生成する、という 4 つの異なる作業に LLM を利用しています。これらを組み合わせることで Bedrock では作成が困難であった添付画像付きの質問にもドラフトが生成できるようになりました。

7 Bedrock 版チャットボットと LangChain 版チャットボットの比較

以上のようにして、Bedrock 版のチャットボットと LangChain 版のチャットボットを作成しました。比較してみると、以下のような違いがあります。

	Bedrock 版チャットボット	LangChain 版チャットボット
利用できる基盤モデル	Bedrock で用意されたモデルのみ	AWS, Azure, Google, OSS
MCP サーバとの接続の煩雑さ	煩雑（MCPプロトコルを処理するMCPクライアントを実装する必要がある）	簡単（エージェントは MCP プロトコルを扱える）
セキュリティ	デフォルトで暗号化されている、IAM でアクセス制御可能	自分で実装が必要
クラウドかオンプレミスか？	AWS	オンプレミスのみも可能、オンプレミスとクラウドの混合も可能
柔軟性、拡張性	低い	高い

シンプルなチャットボットを作りたい場合は Bedrock で十分だと思います。しかし、6 章で説明したように質問文の他に添付画像が付いているケースや MCP サーバともやり取りしたいケースなど、複雑な構成を実現したい場合は LangChain を使ったほうが実現しやすそうです。しかし、LangChain 版の場合は、セキュリティを強化する方法については別途検討し実装する必要がありそうです。LangChain であれば、オンプレミスのみ（ローカル環境のみでクラウドを使わない）でもクラウドとローカル環境の混合も可能ですし、OSS のみで実現することも可能です。要求条件を検討してどちらが良いかを決定する必要があります。

以下の私が書いた引用文献に詳細を記載してありますので、ご興味がある方はこちらもご覧ください。Bedrock 版のチャットボット全体 [1]、Bedrock 版チャットボットで MCP サーバと接続する方法 [2]、LangChain 版のチャットボット [3]、の解説となっています。

引用文献

[1] 三島健, "Amazon Bedrock with RAG&MCP を使ってヘルプデスク用チャットボットを構築して実際に導入してみた," https://qiita.com/ken340/items/be00c816a3bc661e5db0.

[2] 三島健, "Bedrock で作ったチャットボットに MCP サーバを繋ぐ方法," https://qiita.com/ken340/items/fabec06df3f82e67a387

[3] 三島健, " Bedrock を使ったチャットボットを作って気がついた課題を LangChain で実装してみた," https://qiita.com/ken340/items/af22fa4ac8a7ccd99985

GSLetterNeo Vol.199
2026 年 1 月 20 日発行
発行者　株式会社 SRA 技術本部 先端技術研究室

編集者　熊澤努　方 学芬
バックナンバー https://www.sra.co.jp/public/sra/gsletter/
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8

夢を。Yawaraka Innovation
やわらかいのべーしょん