RAG アプリケーションを最適化するためのベストプラクティスの記述

AWS 規範ガイダンス

AWS 規範ガイダンス: RAG アプリケーションを最適化するためのベストプラクティスの記述

Table of Contents

RAG アプリケーションを最適化するためのベストプラクティスの記述

Ivan Cui と Samantha Stuart、Amazon Web Services

2025 年 7 月 (ドキュメント履歴)

大規模言語モデル (LLMsは、人間のようなテキストを理解して生成する能力を備え、人工知能の分野に変革をもたらしました。ただし、トレーニングデータに含まれる知識のみを扱うことができるという大きな制限に直面しています。これは、検索拡張生成 (RAG) が役立つ場所です。これは、LLMsを組織のデータやドキュメントなどの外部のナレッジソースと組み合わせたソリューションを提供します。情報の取得とレスポンスの生成を含む 2 段階のプロセスを通じて、RAG は AI システムがさまざまなソースからのup-to-date情報にアクセスして組み込むことができるため、静的モデル知識と動的な現実世界の情報ニーズとのギャップを埋める、より正確で情報に基づいたレスポンスが得られます。

RAG ベースのアプリケーションで取得するコンテンツを最適化するにはどうすればよいですか? このガイドでは、ナレッジベースのテキストベースのコンテンツのフォーマットと書き込みスタイルを最適化するためのベストプラクティスを提供します。コンテンツを最適化すると、RAG アプリケーションがタスク固有の情報をより正確に理解するのに役立つコンテキストが強化されます。システムが関連性が高く正確なコンテンツを取得すると、LLM のレスポンスの品質が向上します。システムレベルでコンテキスト配信プロセスを最適化することはコンテキストエンジニアリングと呼ばれ、エージェント RAG アーキテクチャの重要な部分を形成します。エージェント RAG では、1 つ以上の追加の LLMs 理由があり、RAG の実行前に受信リクエストを処理します。これにより、複数ステップの情報配信プロセスが容易になります。RAG アーキテクチャがますます複雑になるにつれて、ソースコンテンツの最適化は、LLMs。これらのベストプラクティスは、組織の RAG アプリケーションへの投資を最大化するのに役立ちます。

対象者

このガイドは、1 つ以上の RAG コンポーネントを使用して LLM アプリケーションを構築している AI エンジニア、データサイエンティスト、データエンジニア、またはソフトウェア開発者を対象としています。このガイドの概念と推奨事項を理解するには、ベクトルデータベースと LLMs のプロンプトに精通している必要があります。

目的

このガイドの推奨事項は、以下を達成するのに役立ちます。

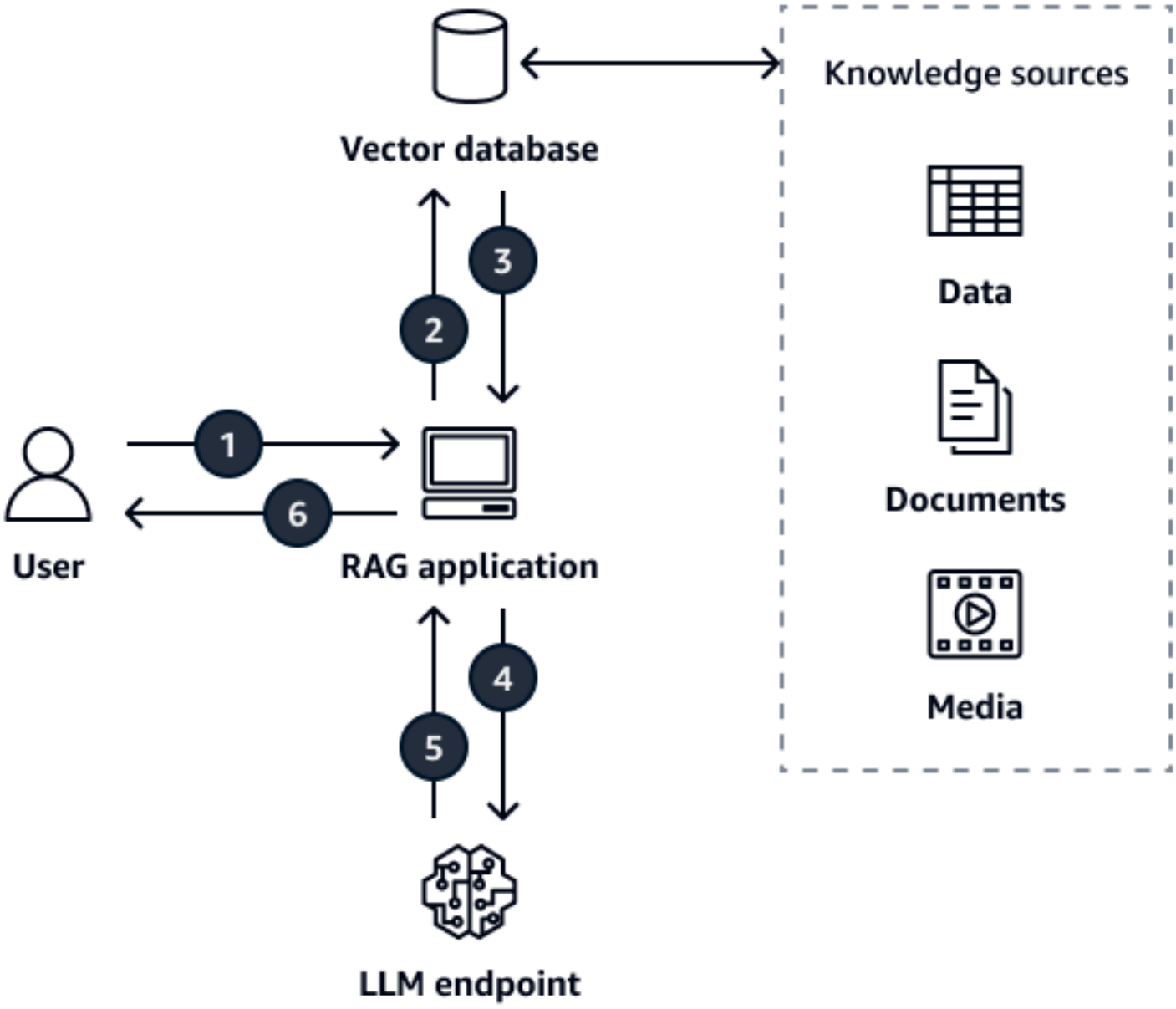
- トークンの使用と冗長性に最適化された、適切に構造化され意味的に豊富なソースドキュメントを提供することで、RAG アプリケーションによって生成されたレスポンスの精度と関連性を向上させます。
- ソースドキュメント内で明確な定義と説明を提供することで、RAG アプリケーションがドメイン固有の知識とコンテキストをよりよく理解できるようにします。
- ソースドキュメント全体で一貫したフォーマットと構造化ガイドラインに従うことで、RAG アプリケーションのメンテナンスとナレッジベースの更新を容易にします。
- 大規模なモノリシックドキュメントを、効率的にインデックス作成および取得できるより小さな自己完結型ユニットに分割することで、RAG ソリューションのスケーラビリティを向上させます。

LLMsと RAG について

ソースドキュメントの品質を向上させることで RAG レスポンスの品質がどのように向上するかを理解するには、LLM の内部動作を理解する必要があります。LLMs の真の能力は、自己注意メカニズムとトランスフォーマーアーキテクチャを使用する能力にあります。これらの高度な手法により、モデルはテキスト内の位置や距離に関係なく、入力シーケンスのさまざまな部分を効果的に処理して関連付けることができます。この機能は、長期的な依存関係やコンテキストの理解にしばしば苦労する従来の言語モデルとはまったく対照的です。さらに、LLMsは前例のない規模でトレーニングされます。最大のモデルの中には、何兆ものパラメータで構成されており、さまざまなソースからテラバイトのテキストデータを取り込みます。この大規模なスケールにより、LLMs言語の深い理解を深め、以前は AI システムにとって困難だった微妙なニュアンス、イディオム、コンテキストキューをキャプチャできます。その結果、コヒーレントで流暢なテキストを生成し、質問への回答、テキスト要約、コード生成などのタスクで優れた機能を示すことができるモデルのクラスが得られます。

これらのモデルを使用するには、Amazon Bedrock などのサービスを利用できます。Amazon Bedrock では、Anthropic、Cohere、Meta など、Amazon やサードパーティープロバイダーのさまざまな基盤モデルにアクセスできます。Amazon Bedrock を使用して、state-of-the-artモデルを試したり、カスタマイズして微調整したり、単一の API を通じて生成 AI を活用したソリューションに組み込むことができます。

LLMsパターンのキャプチャとコヒーレントテキストの生成に優れていますが、多くの場合、up-to-dateまたは特殊な情報にアクセスできません。RAG は、LLMs の生成能力と、マテリアライズド LLM プロンプトの一部として外部ソースからの関連情報にアクセスして組み込むことができる取り出しコンポーネントを組み合わせます。外部ソースの例としては、Amazon Bedrock のナレッジベース、Amazon Kendra などのインテリジェント検索システム、Amazon OpenSearch Service などのベクトルデータベースなどがあります。



この図では、次のワークフローについて説明します。

1. ユーザーはクエリを RAG アプリケーションに送信します。
2. RAG アプリケーションは、ドキュメント、データ、メディアなどのナレッジソースを含むベクトルデータベースをクエリします。
3. RAG アプリケーションは、クエリと保存されたドキュメントの間のセマンティック類似性に基づいて、ベクトルデータベースから関連情報を取得します。
4. RAG アプリケーションは、元のプロンプトを取得したコンテキストで拡張し、LLM エンドポイントに送信します。

5. LLM エンドポイントはレスポンスを生成し、RAG アプリケーションに返します。

6. RAG アプリケーションは、生成されたレスポンスをユーザーに返します。

RAG は、その中核として 2 段階のプロセスを採用しています。最初のステージでは、取得モデルは、入力クエリに基づいて関連するドキュメントまたはパッセージを識別して取得します。この取り出しモデルは、従来の情報取り出しシステム、高密度取り出しモデル、またはその両方の組み合わせです。2 番目のステージでは、取得した情報と元のクエリは、完全にマテリアライズされたプロンプトテンプレートとして LLM にフィードされます。LLMs、リトリーバーコンポーネントによって配信されるソースコンテンツの品質に大きく依存します。自己注意メカニズムを適用して、取得したコンテンツがタスクにどのように関連しているかを数学的にエンコードします。次に、LLM はクエリと取得した情報の両方に基づいてレスポンスを生成します。RAG では、取得されたソースドキュメントの品質を制御することは、タスクの LLM の内部表現を改善する直接的な手段を表します。RAG は、LLM のトレーニングデータを関連する外部データで効果的に補強します。このアプローチにより、RAG は LLMs と取り出しシステムの両方の長所を活用できるため、現在および専門知識を組み込んだ、より正確で情報に基づいたレスポンスを生成できます。

ベクトルと埋め込み

ベクトルと埋め込みは、機械学習と自然言語処理の基本概念です。ベクトルは、大きさと方向の両方を持つ量を表す数学オブジェクトです。自然言語処理 (NLP) では、単語、文、またはドキュメントは、多くの場合、高次元ベクトル空間のベクトルとして表されます。一方、埋め込みは、ベクトル間の関係が意味的または構文的な類似性をキャプチャする、低次元のベクトル空間内の単語やドキュメントなどのオブジェクトを表す方法です。例えば、単語埋め込みでは、同様の意味を持つ単語が同様のベクトル表現を持つことができます。これにより、アルゴリズムは言語をより効果的に理解して処理できます。

ベクトルデータベース

生成 AI では、ベクトルデータベースは、ドキュメント、クエリ、またはその他のオブジェクトのベクトル表現を保存および管理するためのデータベースです。ベクトルを効率的に保存および取得するように設計されています。これにより、セマンティック検索や類似度マッチングなどの高速でスケーラブルなオペレーションがサポートされます。ベクトルデータベースは、階層ナビゲーション可能スモールワールド (HNSW) グラフや K 最近傍 (KNN) アルゴリズムなどの特殊なデータ構造を使用してベクトルをインデックス化します。これらのデータ構造により、最も近い近傍をすばやく検索できるため、データベース内で同様のベクトルをすばやく見つけることができます。

セマンティック検索

セマンティック検索は、キーワードを単に一致させるのではなく、クエリのインテントとコンテキストを理解することで、検索結果の関連性を向上させる手法です。技術的には、セマンティック検索では、クエリのベクトル表現とデータベース内のドキュメントを比較して、最も関連性の高い一致を見つける必要があります。セマンティック検索には、次のようなさまざまな取り出し戦略を使用できます。

- HNSW – 最も近い近傍を効率的に検索できるようにベクトルを整理するグラフベースのデータ構造。
- KNN – コサイン類似度などの距離メトリクスに基づいて、クエリベクトルに最も近い K ベクトルを検索するアルゴリズム。
- コサイン類似性 – ゼロ以外の 2 つのベクトル間の角度のコサインを測定する類似性の尺度。多くの場合、セマンティック検索で使用され、高次元空間内のベクトルの方向を比較します。
- Locality-sensitive hashing (LSH) – 同じまたは近くのバケットに類似したベクトルを高い確率でハッシュする手法。これにより、近似近傍検索が可能になり、高次元空間での正確な検索よりも高速になる可能性があります。

RAG アプリケーションに影響するソースデータの課題

最適な検索拡張生成 (RAG) アプリケーションの開発における重要な課題の 1 つは、使用される未加工のデータまたはドキュメントの性質にあります。多くの場合、企業はヒューマンリファレンス用に作成された既存のドキュメントを使用します。これらのドキュメントには、理解を促進するためのハイパーリンクと画像のスクリーンショットが含まれていることがよくあります。ただし、これらの要素は、抜粋トークンの制限によりセマンティック取得を妨げます。これにより、リトリーバーのパフォーマンスが低下します。

最適な RAG アプリケーションにおける最も一般的な raw ドキュメントの課題は次のとおりです。

- 構造化フォーマットとメタデータの欠如 – 未加工のドキュメントには、明確なセクション見出し、サブ見出し、メタデータがない可能性があります。これにより、関連情報の特定と抽出が困難になります。例えば、見出しが明確でない長いドキュメントでは、特定の情報のコンテキストを判断するのが難しい場合があります。
- 非公式で一貫性のない言語 – 未加工のドキュメントには、多くの場合、非公式な言語や一貫性のない用語が含まれています。これにより、RAG モデルが混乱する可能性があります。例えば、ドキュメントで定義されていない略語や LLM で既に知られている略語は、ドキュメント全体で使用される場合があります。
- 冗長性と冗長性 – 未加工のドキュメントは冗長であり、不必要または冗長な情報が含まれている場合があります。これにより、RAG モデルが圧倒され、簡潔さが低くなり、関連する応答が生じる可能性があります。例としては、同じ情報を複数回繰り返すドキュメントや、類似または矛盾する情報を含む複数のドキュメントなどがあります。
- あいまいな用語とフレーズ – 未加工のドキュメントにはあいまいな用語やフレーズが含まれている場合があり、これらは複数の方法で解釈される可能性があります。このあいまいさは、RAG モデルによる誤解や不正確なレスポンスにつながる可能性があります。たとえば、複数の意味を持つ用語を使用するドキュメントでは、意図した意味と一致しないレスポンスが発生する可能性があります。
- グラフィック要素とハイパーリンク要素の挿入 – グラフィック情報とハイパーリンク情報を含む未加工のドキュメントは、人間が使用するのに適しています。ただし、これらの要素は取得トークンの制限を消費する可能性があります。その結果、抜粋が不完全である可能性があります。たとえば、グラフィック URL とハイパーリンク URLs は取得トークンを使用する取得の一部として返され、後続の段落のキー情報が欠落しています。
- ドメイン固有の知識やコンテキストの欠如 – Raw ドキュメントには、正確な生成に必要なドメイン固有の知識やコンテキストがない可能性があります。これにより、RAG モデルが関連性の高い正確なレスポンスを生成する能力が制限される可能性があります。例として、コンテキストを指定

せずに特殊な概念を参照するドキュメントがあります。これにより、特定のドメインで意味のないレスポンスが発生する可能性があります。

このリストは包括的ではありませんが、企業が何が機能していないのか、なぜ機能しないのかを考えるための出発点となります。ドキュメントには、これらの課題が 1 つ以上ある場合があります。RAG アプリケーションを最適化するための鍵は、取得を最適化するベストプラクティスの記述に準拠した一連のドキュメントを使用することです。

RAG アプリケーションのドキュメントのベストプラクティス

取得拡張生成 (RAG) アプリケーションを正常に開発するには、パフォーマンスを最適化するためにさまざまなドキュメント関連の要素を慎重に検討する必要があります。このセクションのベストプラクティスは、多くの組織リーダーによる RAG システムの構築経験に基づいて厳選されています。以下は、RAG アプリケーションの有効性を高めるためのドキュメントの主要なベストプラクティスです。

- 見出しとサブ見出しを適切に使用する – 見出しとサブ見出しを明確にしてコンテンツを整理すると、読みやすくなり、RAG モデルがドキュメントの構造を理解するのに役立ちます。この方法により、モデルはドキュメントをより適切にナビゲートし、ドキュメントから情報を抽出できるため、生成されたレスポンスの品質が向上します。
- 番号付けが順番であることを確認する – 番号付きリストを使用する場合は、混乱を避けるために適切な番号付けを維持することが重要です。各リスト項目には、番号をスキップせずに順番に番号が付けられていることを確認します。これにより、コンテンツの明確さと一貫性を維持できます。
- リスト項目間の遷移の追加 – 箇条書きリストまたは番号付きリスト内の項目間の遷移を提供することで、コンテンツを通じて LLM をガイドするのに役立ちます。たとえば、「ステップ 2 を完了したら、...」などのフレーズを使用してアイデアを結び付け、情報の流れを改善できます。
- テーブルの置き換え – テーブルの使用は避けてください。この情報は、複数レベルの箇条書きリストまたはフラットレベルの構文でフォーマットします。フラットレベルの構文は、ネストされたレベルの下位配置なしで、要素または項目を同じ階層レベルで配置します。これらの構造LLMs が情報をダイジェストするのに役立ちます。インデックス付きドキュメントのほとんどは左から右に読み取られるため、フラットレベルの構文では、追加のディメンションを参照することなく、より一貫した情報に従うことができます。この形式は、構造化されたわかりやすい方法で情報を表示するため、RAG アプリケーションにより適しています。
- 効率を高めるためにグラフィカル情報を前処理する – マルチモーダル LLMs はイメージとテキストの両方を取り込むことができます。画像の解像度を下げ、冗長な画像を削除し、グラフィカル要素の内容をテキスト形式で記述します。これらの測定により、意味のあるコンテキストが改善され、トークンが不必要に消費されるのを防ぎ、RAG モデルのアクセシビリティが向上します。
- 一般的なクエリにセッションスターターを追加する –「ソフトウェアの注文方法」などの一般的な質問やタスクに対処する場合は、リーダーをプロセスに移行するセッションスターターを追加します。たとえば、「ソフトウェアを注文する場合は、以下のステップに従います...」を追加できま

す。これにより、高いセマンティックマッチングが作成され、LLM がまとまりのあるレスポンスを構築するのに役立ちます。

- 各セクションに要約を追加する – 各見出しまたはサブ見出し の後に、そのセクションの内容の簡潔で簡潔な要約を追加します。これにより、セマンティックカバレッジが増加し、キーポイントが強化される可能性があります。これにより、埋め込みスペース内の類似度検索の精度が向上し、RAG アプリケーションのパフォーマンスが向上します。これは、ドキュメントが LLM と人間の両方の消費を目的としている場合、またはテーブルとグラフィカルな要素が必要な場合に特に役立ちます。
- 曖昧さの排除 — ドキュメントは簡潔で的を絞ったものにする必要があります。LLMs取得した抜粋に基づいてレスポンスを生成するため、曖昧さを解消することで、モデルが明確で関連情報を使用するのに役立ちます。これにより、より正確で有益なレスポンスが得られます。
- 略語の定義とコンテキストの設定 – LLMs は大量のインターネットデータでトレーニングされており、ほとんどの場合、企業の内部ドキュメントのコンテキストはありません。したがって、コンテキストの設定、略語の定義、会社固有の用語の回避または定義は、LLM がエンタープライズデータを理解するのに役立ちます。これにより、LLM はより正確に質問に回答し、ハルシネーションを防ぐことができます。
- 大きなドキュメントを小さなドキュメントに再構築してタグ付けとインデックス作成を効率化 – 複数のサブトピックを含む大きなドキュメントのインデックス作成を回避します。大きなドキュメントを、明確なタイトルを持つより小さな自己完結型ドキュメントに分割することを検討してください。これにより、インデックス作成とタグ付けが向上します。

よくある質問

RAG アプリケーションのドキュメントを最適化することが重要なのはなぜですか?

Raw ドキュメントは、多くの場合、検索拡張生成 (RAG) アプリケーションなどの高度な AI システムの要件を考慮せずに、人間が使用するために記述されます。ベストプラクティスに従ってドキュメントを最適化すると、モデルに構造化され、あいまいで、関連情報を提供することで、RAG アプリケーションのパフォーマンスと精度が大幅に向上します。

RAG のパフォーマンスを妨げる可能性のある raw ドキュメントの一般的な課題は何ですか?

主な課題には、構造化されたフォーマットとメタデータの欠如、非公式または一貫性のない言語、冗長性と冗長性、あいまいな用語とフレーズ、ハイパーリンク要素の包含、ドメイン固有のコンテキストの欠如などがあります。これらの問題は、RAG モデルを混乱させ、不正確または無関係な応答につながる可能性があります。詳細については、このガイドの「RAG アプリケーションに影響するソースデータの課題」を参照してください。

見出しとサブ見出しを使用すると、どのように RAG のパフォーマンスを向上させることができますか?

明確な見出しとサブ見出しは、RAG モデルがコンテンツの構造とコンテキストを理解するのに役立ちます。これにより、ドキュメントをより適切にナビゲートして関連情報を抽出し、生成されたレスポンスの品質を向上させることができます。詳細については、このガイドの「RAG アプリケーションのドキュメントのベストプラクティス」を参照してください。

テーブル情報をフラットレベルの構文に置き換えることが推奨されるのはなぜですか?

RAG モデルは 2 次元構造を理解する必要があるため、テーブルを解釈するのは難しい場合があります。フラットレベルの構文または箇条書きリストでテーブル情報を表示すると、モデルが情報をより簡単に処理できるため、パフォーマンスが向上します。詳細については、このガイドの「RAG アプリケーションのドキュメントのベストプラクティス」を参照してください。

概要を追加すると、RAG のパフォーマンスを向上させるにはどうすればよいですか?

各セクションまたはサブセクションの先頭に簡潔な概要を含めると、セマンティック力バレッジが向上し、重要なポイントが強化される可能性があります。これにより、埋め込みスペース内の類似度検索の精度が向上し、最終的に RAG アプリケーションのパフォーマンスが向上します。詳細については、このガイドの「RAG アプリケーションのドキュメントのベストプラクティス」を参照してください。

LLMs の略語を定義し、コンテキストを設定することが重要なのはなぜですか?

LLMs は幅広いデータに基づいてトレーニングされていますが、企業固有の略語や用語のコンテキストがありません。略語の定義とコンテキストの提供はLLMs がより正確に理解し、応答するのに役立ちます。これにより、幻覚や誤解を防ぐことができます。詳細については、このガイドの「RAG アプリケーションのドキュメントのベストプラクティス」を参照してください。

次のステップとリソース

このガイドでは、RAG アプリケーションに関する一連のドキュメントレベルの課題と、それらを軽減するためのベストプラクティスについて説明します。これらの学習は、業界のリーダーとのインタビューとディスカッションから導き出され、企業のユースケースに支えられています。

RAG アプリケーション用にドキュメントの最適化を開始するには、既存のドキュメントの監査を実行することをお勧めします。RAG アプリケーションに課題をもたらす領域を特定します。例としては、構造の欠如、あいまいな言語、グラフィカル要素の過剰な使用などがあります。頻繁にアクセスされる、または事業運営にとって重要なドキュメントに優先順位を付けます。対象分野のエキスパートと協力して、このガイドのベストプラクティスを実装します。ドキュメントが明確な見出し、簡潔な言語、コンテキスト設定要素で再構成されていることを確認します。新しいドキュメントについては、一貫性を確保し、作成者がベストプラクティスに従うのに役立つガイドラインとテンプレートを確立します。さらに、生成 AI を使用してドキュメントを再構築するなど、ドキュメント最適化プロセスの側面を自動化できるツールやサービスへの投資を検討してください。ドキュメントの最適化に積極的にアプローチすることで、RAG アプリケーションの可能性を最大限に引き出し、組織全体でより正確で洞察力のある結果を生み出すことができます。

以下のリソースは、組織内の RAG アプリケーションを理解して構築するのに役立ちます。

リソース

AWS ドキュメント

- RAG ユースケース用の AWS ベクトルデータベースの選択 (AWS 規範ガイダンス)
- Terraform と Amazon Bedrock を使用して AWS に RAG ユースケースをデプロイする (AWS 規範ガイダンス)
- RAG と ReAct プロンプトを使用して高度な生成 AI チャットベースのアシスタントを開発する (AWS 規範ガイダンス)
- Amazon Bedrock ナレッジベースを使用してデータを取得し、AI レスポンスを生成する (Amazon Bedrock ドキュメント)
- 取得拡張生成 (Amazon SageMaker AI ドキュメント)
- で拡張生成オプションとアーキテクチャを取得する AWS (AWS 規範ガイダンス)

その他の AWS リソース

- 高度な RAG と Amazon Bedrock を使用してマルチモーダルアシスタントを作成する (AWS ブログ記事)

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィード をサブスクライブできます。

変更	説明	日付
初版発行	—	2025 年 7 月 24 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。