Red Hat

エンタープライズ向け RAG のエンジニアリング

検索拡張生成を大規模に
最適化するための Red Hat ガイド

目次

はじめに

おそらく次のような意見を聞いたことがあるでしょう：「RAG は終わった」。

これは挑発的な主張であり、AI に関する議論における現実の変化を捉えたものです。2 年前、検索拡張生成 (RAG) は AI のトレンドとなったトピックでした。現在のトレンドは、AI エージェント、自律型ワークフロー、マルチステップ推論へと移り変わっています。RAG は踏み台だった、というのがこの議論の主張です。もうその段階を乗り越えたのです。

いいえ、そうではありません。むしろ RAG の重要性は高まっています。シンプルなチャットボットに代わるシステムが、RAG を最も必要としているからです。

RAG がこれまで以上に重要である理由

何が変わったのかを考えてみてください。2024 年では、典型的な RAG システムは、ナレッジベースから数段落を取得してモデルに入力し、モデルが回答を生成する仕組みでした。リスクは低いものでした。答えが少し間違っていた場合は、人間が見直して次に進みました。

現在では、AI エージェントが意思決定を行っています。たとえば、保険会社では請求処理エージェントが保険の適用範囲を評価します。銀行ではコンプライアンス・エージェントが不審な取引にフラグを立てます。調達エージェントがベンダー契約を承認します。これらのシステムはアクションを実行し、実際の結果や実際の金額に影響を与えるワークフローを実行します。

企業が実際に認識している内容に基づいていないエージェントは、自信を持って適切にフォーマットされた決断を下しますが、最終的にはそれは誤った決定になります。RAG は、自律型 AI システムを組織独自のデータに接続するメカニズムであり、エージェントが正確に動作し、監査可能であるために必要なコンテキストを提供します。

これはもはやテキストだけの問題ではありません。エンタープライズデータはマルチモーダルです。グラフが組み込まれたドキュメント、人間の読者向けにデザインされたインフォグラフィック、スキャンされたフォーム、音声録音、データベースとして使用されるスプレッドシートなどがあります。エージェントが必要とするコンテキストは、PDF、Confluence のページ、四半期ごとの収益を示すスライド資料などの複数の場所に分散しており、それらが同時に必要になります。先進的な RAG システムは、この現実に対処できなければなりません。さまざまなデータタイプ、ソース、形式から適切な情報を見つけ、関連付けて提供する必要があります。

ギャップへの対処

RAG は、コンセプト上は一見シンプルであるものの、プロダクションで正しく使用するのは非常に難しいという課題があります。ドキュメントをどのように解析してチャンク化するか、どの埋め込みモデルを選択するか、検索結果をどのようにランク付けするかなど、何十もの相互に依存した決定が伴います。汎用のデフォルト設定などはありません。適切な組み合わせは、もっぱらデータ、ドメイン、ユースケースによって異なります。

このガイドでは、強固なベースライン、厳密な評価、モデルのファインチューニング、そしてこれらの決定の多くを自動的に最適化できる自動 RAG の構築について、実務担当者のプロセス全体を紹介します。ご自身の状況に応じて適切な場所から始めましょう。

01

エンタープライズ RAG が想像よりも難しい理由

当初の RAG のブループリントは、関連するドキュメントを取得し、それをモデルにフィードして回答を生成するという単純なパターンでした。クリーンなデータと単純な質問を使用したデモなら、これはうまく機能します。しかし、実際のエンタープライズデータを使用し、実際のユーザーが操作するプロダクション環境ではうまく機能しません。予測は難しく、診断はさらに難しいからです。

エンタープライズ RAG パイプラインがうまく機能しない原因となる、いくつかの具体的な課題があります。構築したプロトタイプがテストでは良好なパフォーマンスを発揮したものの、実際のクエリでうまくいかない場合は、ここで挙げるいくつかの課題に思い当たるでしょう。

エンタープライズデータの現実

エンタープライズデータは大きく 2 つのカテゴリに分類されます。1 つ目は構造化ソースで、データベースや API など、マシンによる処理が比較的簡単です。データにはスキーマがあり、フィールドには型があり、クエリは予測可能な結果を返します。

2 つ目のカテゴリは非構造化ソースで、これを RAG で使用するのは困難が伴います。ほとんどの組織において、知識は実際には非構造化データとして存在しています。非構造化データに関する最も一般的な課題の例を以下に示します。

人間向けに設計されたドキュメント

エンタープライズのドキュメントはますますリッチ化し、視覚化されています。年次レポートにはインフォグラフィックが含まれています。技術仕様には複数列のレイアウトが使用されています。トレーニングマニュアルには図と本文が組み合わせて使用されています。テーブルは複数のページにまたがっています。これらのドキュメントは、ビジュアルフローを理解できる人間の読者にとっては完全に理解できるものです。しかし、そのコンテンツのインデックスを作成して抽出しようとするマシンにとっては、大きな課題となります。

2 ページにわたって折り返されるテーブルは、プレーンテキストとして解析すると構造が失われます。PDF に埋め込まれた円グラフは、テキスト抽出中にその意味が完全に失われてしまいます。スキャンされたドキュメントでは、埋め込まれたテキストレイヤーが完全になくなったり、物理的な転送プロセスによって文字化けしたりする場合があります。それぞれに異なる解析戦略が必要であり、選択を誤ると、パイプラインに入力されるデータは、チャンク化や埋め込みを始める前にすでに壊れた状態になってしまいます。

コンテキストの断片化

質問に対する答えが 1 カ所にあることはめったにありません。欧州連合 (EU) の請負業者のデータアクセスポリシーに対するクエリに回答するには、人事 (HR) ポリシーの文書、地域の法的フレームワーク、社内のコンプライアンスに関する覚書の情報が必要になる場合があります。これら 3 つのソースは、それぞれ異なる形式で、異なるシステムに格納され、異なるスタイルで記述されている可能性があります。

これはエンタープライズ RAG の最大の課題の 1 つです。つまり、複数のドキュメントや方式に分散した断片的な情報から完全なコンテキストを組み立てなければなりません。マルチモーダルデータを考慮に入れると、状況はさらに複雑になります。関連するコンテキストにテキスト、画像、構造化データ、音声などが含まれている可能性があるからです。単一のソースから単一のテキストチャンクを取得する単純な RAG パイプラインは、これを完全に見逃してしまいます。

急速に進化するデータ

エンタープライズの知識は変化しています。製品機能はリリースごとに変わります。財務データはほぼリアルタイムで更新されます。ポリシーは四半期ごと、あるいはそれ以上の頻度で改訂されます。先月のドキュメントでインデックスを作成した RAG パイプラインが、現在では古くなった回答を返す可能性があります。

これもまた、RAG の存在理由の 1 つです。エンタープライズデータでモデルをファインチューニングすると、スナップショットが作成されます。適切に行えば、RAG は現行のソースから取得できます。しかし「適切に行う」とは、インデックス作成パイプラインが、基盤となるデータの変化の速度に対応しなければならないことを意味します。これはインフラストラクチャの課題で、ほとんどのデモでは考慮されていません。

セキュリティとプロンプトインジェクション

RAG システムがコンテンツを取得してモデルのコンテキストウィンドウに配置するたびに、そのコンテンツは潜在的な攻撃ベクトルとなります。取得したドキュメントに、システム指示のようなテキストが含まれている場合、モデルの意図する動作がオーバーライドされることがあります。これは理論上のリスクではなく、プロダクション RAG システムが対処する必要がある既知の脆弱性です。

優れた RAG 設計では、取得したコンテンツを信頼できない入力として扱います。つまり、拡張コンテキストとシステム指示を明確に区別し、モデルが両者を区別できるようにします。基本的に、優れた RAG 設計にはデータと指示の間にガードレールが存在します。

デモからプロダクションへのギャップ

この問題は複合的です。クリーンで適切にフォーマットされた少数のドキュメントを使用するデモパイプラインは、こうしたデータの問題がないため、うまく機能します。質問はシンプルでデータは整理されており、すべてがモデルのコンテキストウィンドウに適合します。

プロダクションは違います。実際のユーザーは複雑で曖昧な質問をします。ドキュメントは乱雑で、コンテキストは複数のソースに断片化されています。検索システムが返す結果は、もっともらしいが間違っているか、正しい場合でも関連性の低い結果の中に埋もれてしまいます。モデルのコンテキストウィンドウはノイズで満たされ、ハルシネーションが始まります。

その結果、ミーティングで完璧に機能することをステークホルダーが確認したのに、現場では予測できない失敗をするシステムが出来上がります。この結果がもたらす不満は現実のものであり、多くの場合、これがチームにとって RAG を簡単な統合ではなくエンジニアリングの領域として扱い始めるきっかけとなります。

次の章では、プロダクション RAG パイプラインの構築に関わる実践的な決定事項について説明します。まずは、ほとんどのチュートリアルではまったく触れられていない点、つまりインデックス作成パイプラインと取得パイプラインは同じものではないという区別から説明します。

02

初めてのエンタープライズ RAG パイプラインの構築

ほとんどのチュートリアルでは RAG パイプラインを、取り込み、チャンク化、埋め込み、保存、取得、生成からなる単一のフローとして扱います。プロダクション環境においては、エンタープライズデータに最初に触れた時点で、この枠組みは崩れます。実際のエンタープライズ RAG システムは、異なる実行頻度、異なる担当者、異なる障害モードを持つ 3 つの独立したパイプラインで構成されています。これらを混同することは、チームが初期段階で犯しがちな、アーキテクチャ上の最も一般的な間違いの 1 つです。

3 つのパイプラインを備えたアーキテクチャ

- **インデックス作成パイプライン**はデータを準備します。生のエンタープライズ・ドキュメントを解析し、コンテンツを正規化し、メタデータを付加し、チャンク化し、埋め込みを生成し、すべてをベクトルストアに保存します。これはチームが最初に構築するパイプラインであり、ほとんどのチュートリアルで取り上げられるパイプラインです。
- **取得パイプライン**は、アプリケーションが呼び出すものです。ユーザーまたはエージェントがクエリを送信すると、取得パイプラインは準備されたデータを検索し、結果をランク付けしてフィルタリングし、取得したコンテキストを使用してプロンプトを構築し、生成のためにモデルに送信します。
- **メンテナンスパイプライン**は、インデックスが作成されたコーパスを継続的に最新の状態に維持します。ソースドキュメントは、変更されたり、廃止されたり、アクセス規則が更新されたりします。埋め込みモデルはアップグレードされます。チャンク化戦略は高度化されます。メンテナンスパイプラインは、取得をオフラインにすることなく、これらすべてを処理します。多くの場合、このパイプラインは最も重要であり、プラットフォームが提供するツール以上に、組織が高度にカスタマイズする必要があるパイプラインです。

各パイプラインにはそれぞれ異なる動作プロファイルがあります。インデックス作成は、多くの場合、1回限りまたはスケジュールされたジョブとして、一括して実行されます。取得は、ミリ秒単位が影響を与えるクエリレイテンシーで、継続的に実行されます。メンテナンスは、ドキュメントの編集、アクセス制御の変更、モデルのアップグレードなどのアップストリームイベントによってトリガーされ、段階的に実行されます。

取得の品質に関する問題は、3 つのパイプラインのいずれかで発生している可能性があります。RAG システムのデバッグでは、どのパイプラインを確認するかを把握する必要があります。

パイプライン 1:インデックス作成

パーサーがソースドキュメントを処理できなければ、パイプラインにはこれを補うものは他にありません。インデックス作成では、すべてのダウンストリームステップが結果を継承するため、データの誤りによるコストが最も高くなります。

- **パーサーの選択:**単一のパーサーですべてのユースケースに対応できるわけではありません。複数列の PDF には、スキャンしたドキュメントとは異なる処理が必要です。プログラムで生成された PDF の解析方法は、PDF にエクスポートされたスライド資料とは異なります。パーサーは、扱うドキュメントの種類に一致する必要があり、ほとんどの企業では複数の形式を同時に処理することになります。Red Hat® AI には Docling が統合されています。これは、PDF、HTML、PPTX などの形式で非構造化エンタープライズデータを大規模に処理するオープンソースのドキュメント解析ツールであり、これによってモデルの出力が組織のナレッジベースに基づいたものになります。
- **メタデータの抽出:**コンテンツを抽出する際には、ソースファイル、日付、バージョン、所有者、アクセス制御タグなど、出所も保持する必要があります。このメタデータにより、取得時にフィルタリングが可能になり、モデルはソースを引用できるようになるため、監査可能性が確保されます。これは、更新が必要なものを検出するためにメンテナンスパイプラインが使用するものでもあります。解析中にインデックス作成パイプラインがメタデータを除去した場合、3 つの機能すべてが永続的に失われます。
- **正規化:**生のドキュメントには、ヘッダー、フッター、エンコーディング・アーティファクト、レイアウトのマークアップなどのノイズが含まれています。正規化はこれをクリーンアップし、意味を維持しながらテーブルやグラフなどの構造化された要素を抽出します。
- **チャンク化戦略:**コンテンツは、埋め込むためにチャンクに分割されます。固定サイズのチャンク化は高速ですが、コンテキストを認識しません。セマンティックかつ再帰的なチャンク化により、形式が混在するドキュメントでの再現性が向上します。ドキュメントを認識したチャンク化では、見出し、セクション、テーブルなどの明示的な構造が考慮されます。どのような場合にも合う適切なチャンクサイズは存在しません。適切な組み合わせは、埋め込みモデル、ドキュメントの特性、ユーザーが行うクエリによって異なります。評価 (第 3 章で説明します) することで、いつ変更すべきかがわかります。
- **埋め込みモデルの選択:**埋め込みモデルはチャンクをベクトルに変換します。これらのベクトルの品質によって、取得結果がクエリとコンテンツをどれだけ正確に照合したものになるかが決まります。汎用モデルは、テキスト量の多いユースケースの出発点として適しています。専門性の高い分野では、そのドメイン特有の制約が即座に顕在化します。多言語のデプロイでは、多くの場合、多言語モデルか、使用する言語の組み合わせに応じたファインチューニングのいずれかが必要になります。チャンク化と同様に、これは評価によって対応できる問題です。

パイプライン 2：取得

取得パイプラインは、顧客がその品質を実感できると同時に、構成の小さな変更が大きな効果をもたらす場所でもあります。

- **ベクトルストアの選択：**エンタープライズにおいては多くの場合、インフラストラクチャによって決定されます。組織が PostgreSQL で標準化している場合、最も抵抗が小さいのは pgvector でしょう。Milvus などの専用ベクトルデータベースはベクトル固有のワークロードに対して最適化されたパフォーマンスを提供しますが、選択するデータベースによってまったく新しいインフラストラクチャが導入されるのなら、汎用データベースの拡張機能で十分な場合もよくあります。Red Hat AI は複数のバックエンドをサポートします。
- **取得戦略：**回答の質に直接影響します。主な 3 つのアプローチは、密検索 (ベクトル類似性)、疎検索 (BM25 などのキーワードベース)、その両方を組み合わせたハイブリッド検索です。ほとんどのプロダクションシステムでは、どちらか一方を使用するよりもハイブリッドのほうが優れたパフォーマンスを発揮します。リランキングは、クロスエンコーダーモデルを使用して結果を並べ替えるための 2 番目の処理を追加するものとなり、多くの場合、最小限の労力で大幅な改善が得られます。
- **アクセス制御フィルタリング：**インデックス作成中に捕捉されたメタデータを使用して、取得時に適用されます。ユーザーの権限、グループメンバーシップ、または管轄地域によって、類似性ランキングを実行する前に候補のセットを絞り込むことができます。
- **プロンプト構築：**取得したコンテキストとモデルが出会う場所です。回答の品質とプロンプトインジェクションからの防御の両方において重要なのは、取得したコンテンツとシステム指示をどのように区別するかです。取得したコンテンツを信頼できない入力として扱いましょう。

密検索からハイブリッド検索に切り替えたり、リランキングステップを追加したりすると、忠実度スコアが 2 桁の割合で変動する可能性があります。これは、構成を体系的にテストする自動 RAG の機能が有効な領域の 1 つです。

パイプライン 3：メンテナンス

メンテナンスパイプラインは、ほとんどのチュートリアルでは取り上げておらず、どの運用チームも、通常は何らかのインシデント発生後のプレッシャーの中で、結果的に構築することになるものです。意図的に構築するほうがコストはかかりません。

- **更新と削除の伝播：**ソースドキュメントは変更され、ポリシーは改訂され、契約は置き換えられ、製品は製造が終了します。メンテナンスパイプラインは、Webhook、変更フィード、または定期的な比較を通じてこれらの変更を検出し、それに応じてベクトルストアを更新します。
- **忘れられる権利とデータレジデンシー：**規制対象の環境では、要求に応じて特定の記録を削除する機能が必要です。多くの場合、サービスレベル契約 (SLA) が定められています。メンテナンスパイプラインは、これらの要求が実行される場所であり、コンプライアンスを証明する監査証跡が生成される場所です。専用のパイプラインがない場合、チームはプロダクション環境に対してアドホックなクエリを実行しますが、これはまさに監査されていないアクセスであり、コンプライアンス違反の指摘につながります。
- **モデル変更時の再埋め込み：**埋め込みモデルをアップグレードすると、ストア内のすべてのベクトルが、受信クエリとは異なるセマンティック空間に存在することになります。検索機能は知らない間に劣化しています。メンテナンスパイプラインはこれを完全な再インデックス付けとして、または段階的な移行として処理し、切り替えが完了するまで新旧の両方の埋め込みを有効とします。
- **戦略変更に伴う再チャンク化：**評価の結果、別のチャンク化アプローチを使用すると検索精度が改善されると判断された場合、メンテナンスパイプラインは完全な再構築を必要とせずに、既存のコーパスに変更を適用します。
- **アクセス制御の同期：**ソースシステムで権限が変更された場合 (メンバーがチームを離れた場合、プロジェクトが制限された場合、該当地域で新しいデータ処理ルールが採用された場合など)、対応するベクトルのメタデータもそれに合わせて変更する必要があります。
- **コーパスドリフトの検出：**これはモデルドリフトとは異なります。ソースコンテンツが進化するにつれて、ベクトルストア内のデータの分布も変化します。以前のスナップショットを使用して構築された評価データセットは、システムが実際に検索しているコーパスを正確に反映しなくなっている可能性があります。メンテナンスパイプラインでは、コーパスドリフトを検出し、再評価サイクルをトリガーします。
- **出所とバージョン管理：**規制対象の業界の顧客にとって、どのドキュメントのどのバージョンがどの時点でインデックスが作成されたか、そしてそれがどの回答に貢献したかを把握することがコンプライアンス要件となります。メンテナンスパイプラインがこの監査証跡を保有します。Red Hat AI は、パイプライン構成全体で実験を追跡するために MLflow を統合しており、同じ手法がコーパス自体にも適用されます。

3 つのパイプラインの相互作用

パイプラインは設計上分離されていますが、ベクトルストアとそのメタデータを通じて状態を共有します。インデックス作成が初期状態を書き込みます。それを取得で読み取ります。メンテナンスにより、ソースシステムからのイベントや、モデルのアップグレードや評価結果などの内部トリガーに応じて、継続的に更新を行います。

実際の運用パターンでは、インデックス作成は、新たにオンボーディングされたデータソースごとに 1 回実行されます。取得はクエリ時に毎回実行されます。メンテナンスはバックグラウンドで継続的に実行されます。これが、数カ月ではなく数年にわたってエンタープライズデータを使用できるアーキテクチャです。

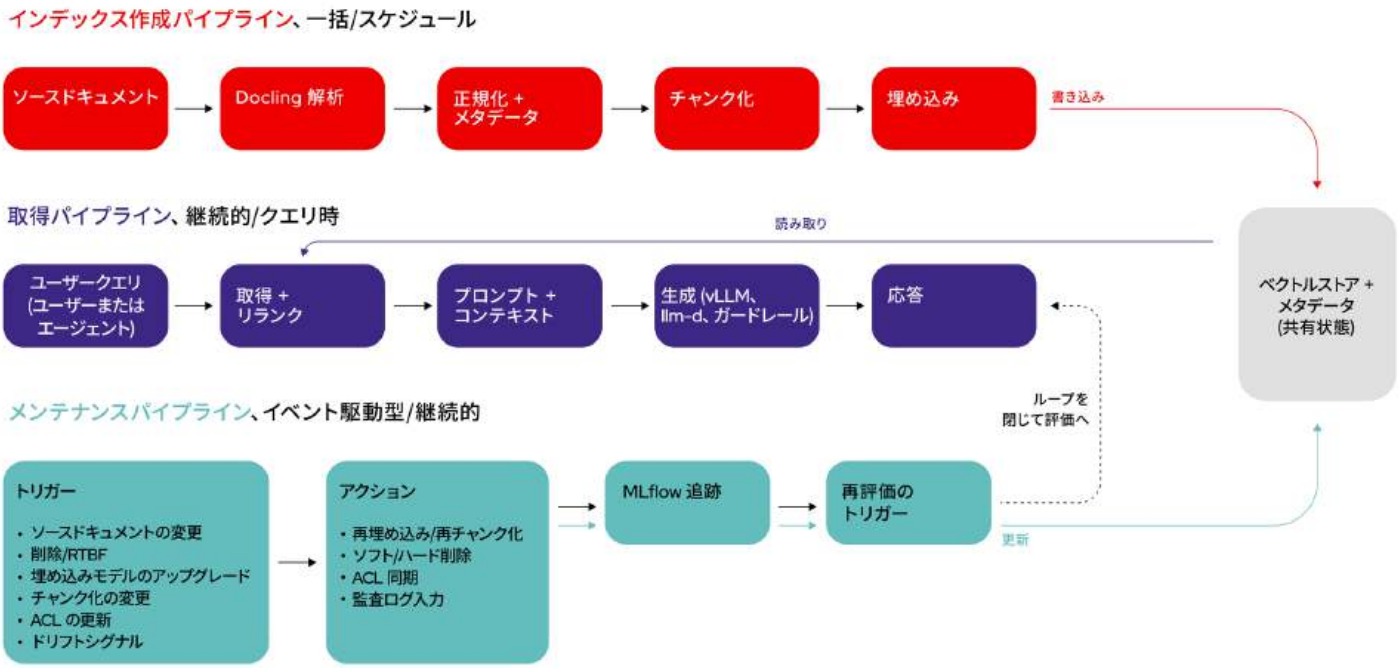


図 1.RAG セットアップで 3 つのパイプラインがどのように相互作用するかを示す実施例。

この時点でパイプラインが確立されました。当然ながら、次の質問は「それがうまく機能しているかどうかは、どうすればわかるのか?」です。第 3 章ではこれについて説明します。

03

RAG の評価

ほとんどのエンタープライズ RAG プロジェクトは、プロダクション環境で何らかの問題が発生するまで評価をスキップします。あるステークホルダーが、回答が適切ではないことを報告します。ユーザーがハルシネーションをスクリーンショットに撮ります。チームは問題が取得、生成、またはその両方のいずれであるかを突き止めようと取り組みますが、比較するためのベースラインがありません。

これには高いコストがかかります。評価データのない RAG パイプラインを修正するのは、当て推量の作業になります。チャンクサイズを変更し、手動でクエリを再実行して、回答の品質が改善されることを願います。評価を行えば、パイプラインのどこで問題が発生しているかを正確に特定し、修正が実際に効果的であったかどうかを測定できます。

重要な 4 つの指標

RAG の評価は、2 つの懸念事項に分けられます。取得が適切かどうか、そして、生成が適切かどうかです。この両方を対象とした 4 つの指標があります。

1.

コンテキスト再現率：適切なドキュメントが取得されているかどうかを測定します。質問に対する回答がナレッジベース内に存在しているのに検索システムで見つからない場合、コンテキスト再現率は低くなります。これはインデックス作成パイプラインに問題があることを示しています。解析の失敗、チャンク化の不備、または埋め込みモデルがドメインを適切に反映していないことが考えられます。

2.

コンテキスト精度：取得されたドキュメントに、実際に関連性があるかどうかを測定します。再現率が高くて精度が低い場合、システムは適切なドキュメントを見つけていますが多数のノイズも返していることを意味します。このノイズによってコンテキストウィンドウが埋め尽くされ、関連するコンテンツが実際には存在していても、生成の品質が低下します。

3.

回答の忠実度：生成された応答が、取得されたコンテキストに基づいているかどうかを測定します。忠実度スコアは、与えられたドキュメントからモデルが実際に導き出した回答の割合と、モデルが捏造した回答の割合を示します。再現率と精度が高くても忠実度が低い場合は、通常、生成モデルがボトルネックになっています。これは、モデルがタスクに対して小さすぎるか、提供されたコンテキストに沿うようにプロンプト構造がモデルに指示していないかのいずれかが原因です。

4.

回答の関連性：応答が実際に質問の回答になっているかどうかを測定します。モデルは、取得したドキュメントに基づいて完全に忠実な回答を生成できますが、それでもユーザーが尋ねた内容の本質を見逃すことがあります。

これら 4 つの指標を組み合わせることで、診断マップが得られます。再現率が低い場合は、インデックス作成パイプラインを確認します。精度が低い場合は、取得の構成とリランキングを確認します。忠実度が低い場合は、生成モデルとプロンプト設計を確認します。関連性が低い場合は、クエリがどのように解釈されているかを確認します。

評価ハーネスの構築

評価にとって実際の障壁となるのは、通常はデータです。テストを行うには、既知の適切な回答を持つ一連の質問が必要ですが、ほとんどのエンタープライズのチームにはそのような質問はありません。

そこで、2 つの方法があります。1 つ目は、ゴールデンデータセットを手動で構築する方法です。実際のユーザーからの代表的な質問を収集し、ドメインエキスパートに参考となる回答を作成してもらい、それをベンチマークとして使用します。これによって高品質の評価データが生成されますが、時間はかかります。

2 つ目は合成データの生成です。Red Hat AI の一部である SDG Hub は、エンタープライズ・ドキュメントから質問と回答のペアを生成します。これにより、手動でアノテーションを付けるコストをかけずに、評価データセットを得られます。合成データは完璧ではありませんが、機能的な評価ループを迅速に提供できるため、まったく評価しないよりは良いでしょう。

評価データが得られたら、ワークフローは簡単です。データセットに対して RAG パイプラインを実行し、LM-Eval を使用して 4 つの指標について各応答をスコアリングし、結果を確認します。コンテキスト再現率が 0.62、忠実度が 0.91 の場合、モデルは適切なコンテキストを与えられた場合にはうまく機能していますが、検索システムが適切なドキュメントを見つけられる割合は約 60% にすぎないことを示しています。これは取得の問題であり、修正対象はインデックス作成パイプラインまたは取得の構成であり、生成モデルではありません。

Red Hat の評価用ツール

Red Hat AI は、RAG パイプラインの構築とテストを支援する幅広いツールを提供します。最も重要なのは次の 4 つです。

- **EvalHub**：アドホックな手動テストに代わる、統合評価コントロールプレーン (プレビュー版) です。モデル、RAG パイプライン、および AI エージェントがプロダクション環境に導入される前に、科学的にベンチマーク、スコアリング、および監査するための包括的なインタフェースを提供します。
- **GuideLLM**：スループットとキャパシティプランニングという、異なる側面に焦点を当てています。パイプラインが正確であれば、GuideLLM はプロダクション環境の負荷におけるパフォーマンスの把握に役立ちます。
- **LM-Eval**：4 つの RAG 指標にわたるベンチマークのほか、論理推論やドメイン固有の精度などのより広範なモデル機能を処理します。パイプラインの変更によって実際に結果が改善されたかどうかを測定するための主要なツールです。
- **AI ガードレール**：モデルの入出力に対して実行時の安全性チェックを行います。LM-Eval はテスト環境での品質を測定しますが、ガードレールはプロダクション環境での品質を監視し、許容範囲を外れた応答にはフラグを立てます。

評価は継続的なプラクティス

この章が提示する最も重要な変化は、評価をデプロイ前の一度限りの関門としてではなく、継続的なものとして扱うことです。

精度の目標を事前に定義しましょう。必要なコンテキスト再現率はどの程度でしょうか？ユースケースで許容される忠実度スコアはどの程度でしょうか？これらの目標があれば、パイプライン内のすべての決定を、目標に照らして測定できます。チャンク化戦略を変更するべきか？別の埋め込みモデルに切り替えるべきか？より大規模な生成モデルを試して追加の推論コストを負担するべきか？いずれにせよ、最善の策は評価を実行することです。

Red Hat AI の評価機能は、ベンチマーク、AI の安全性、AI アーティファクトのパフォーマンス・プロファイル (プロンプト、取得、データセット、RAG 構成など) に関する粒度が小さい実験追跡機能を提供し、これらの結果を検証可能で実証可能な不変の Open Container Initiative (OCI) アーティファクトとしてエクスポートする機能も備えています。これは、EU AI 法などの規制報告要件に対応し、結果の再現性を促進します。

次の第 4 章ではこれらの指標を使用して、一般的な障害モードを診断します。第 5 章では、これらを使用して、ファインチューニングに投資する価値があるかどうかを判断します。また、第 7 章の自動 RAG では、これらをパイプラインの自動最適化の目的関数として使用します。

04

よくある落とし穴とその修正方法

RAG パイプラインのパフォーマンスが低く、その理由が不明な場合、その問題をたどるとほとんどは 2 つの根本原因のいずれかに起因しています。この章では、これらの根本原因を第 3 章の評価指標に対応付けます。そうすることで、何が問題なのかを診断し、適切な場所に労力を集中させることができるようになります。

乱雑なデータが入力されれば、不適切な回答が出力される

エンタープライズデータは、本質的には構造化されていません。パーサーがソースドキュメントからコンテンツを適切に抽出できない場合、ダウンストリームのすべてのステップに問題が継承されます。たとえば、複数列のレイアウトは意味のないテキストに平坦化され、複数のページにまたがるテーブルはその構造が失われ、スキャンされたドキュメントからは文字化けが発生したり何も生成されなかったりします。

入力データがすでに破損していた場合、チャンク化と埋め込みの設定がどれほど正確であっても、結果は望ましくないものになります。

- **指標が低い場合の意味：**ナレッジベースに適切なドキュメントが存在することがわかっている場合でも、コンテキストの再現率が低い。その結果、重要な詳細を見逃した回答や、どこから出てきたのかわからないように見えるものの実際には不正なコンテキストから生じたハルシネーションが返されている。
- **確認すべき場所：**インデックス作成パイプライン。すべてにインデックス作成を行う前に、代表的なサンプルドキュメントを用いて抽出品質を検証します。元のソースに照らして、パイプラインが生成するチャンクのスポットチェックを行います。人間が読んでも意味が理解できないチャンクは、埋め込みモデルにとっても意味を持ちません。解析品質に投資すべきなのはこの点であり、ここで役立つのが Docling です。

構成が間違っている

RAG パイプラインには、チャンク化戦略、チャンクサイズ、重複、埋め込みモデル、取得方法、リランキングなど、相互に依存する多くのパラメーターがあります。適切な組み合わせは、もっぱらデータとユースケースによって異なります。デフォルト値を評価せずに受け入れたり、ダウンストリームへの影響を測定せずに 1 つのパラメーターだけを個別にチューニングしたりすると、パイプラインから一貫性のない結果が生成されます。

- **指標が低い場合の意味：**クエリの種類に応じて取得品質が異なる。忠実度スコアが低い。回答が、質問と形式的には関連しているが、厳密には間違っている。またはその逆に、パイプラインがオーバープロビジョニングされているため、不必要に高いコンピュートコストによって高精度が得られている。
- **確認すべき場所：**この場合は体系的な評価が不可欠です。評価データセットに対して、チャンク化戦略、モデルの埋め込み、取得の構成のさまざまな組み合わせをテストします。結果を測定します。1 つの変数を変更し、再評価します。これは手動で行うと面倒な作業です。自動 RAG (第 7 章) が自動化するのは、まさにこの作業です。自動 RAG は、あらゆる組み合わせを実行し、各構成を評価し、精度の目標とコストの制約に最も適合する組み合わせを提示します。

コストと精度のトレードオフ

品質が低い場合のよくある対応として、より大きなモデルを投入して問題を解決しようとする傾向があります。これは役立つ場合もありますが、ボトルネックが生成ではなく取得にある場合は効果がないこともあります。また、適切に構成された小規模なパイプラインのほうが、チューニングが不十分な大規模なパイプラインよりもパフォーマンスが優れていることもあります。

開発プロセスに評価を組み込むことで得られる最も実用的なメリットは、さまざまなパイプライン構成におけるコストと精度を関連付けられることです。これにより、構成を変更すれば問題が解決するにもかかわらずチームが推論に過剰な投資をしたり、モデルがより多くの容量を必要としているのに投資が不足したりすることがなくなります。自動 RAG はこのトレードオフの分析を直接提供し、Red Hat AI は llm-d を提供します。llm-d は、プロダクション環境における分散 LLM 推論のスケーリングと最適化のためのオープンな Kubernetes ネイティブの提供スタックであり、リソース最適化を使用して、不要なオーバープロビジョニングを行うことなく、予測可能でスケーラブルなパフォーマンスを実現します。

次の章では、取得の品質は良好であるものの生成モデルがドメインコンテンツを正確に解釈できない場合の対処方法について説明します。ここで役立つのがファインチューニングです。

05

モデルのカスタマイズによる RAG の改善

RAG とファインチューニングは相反するアプローチであるという誤解がよくありますが、実際にはエンタープライズグレードの RAG パイプラインには通常、両方が必要です。これらはさまざまな問題を解決します。また組み合わせることで、単独で使用するよりも良い結果が得られます。

RAG とファインチューニング：いつ、どちらを使うか

- **RAG**：標準運用手順、製品ドキュメント、ポリシー、価格設定など、頻繁に変更されるデータに最適です。モデルはこれらの情報を記憶する必要はありません。クエリ時に最新バージョンにアクセスする必要があるだけです。これは取得パイプラインが提供するものです。
- **ファインチューニング**：ドメイン固有の推論パターン、用語、応答形式など、安定している知識に最適です。ファインチューニングは、ドメインコンテキスト内の一般的な用語をモデルに学習させ、取得したコンテンツをより正確に解釈できるようにします。埋め込みモデルは金融ヘッジに関するドキュメントを取得する方法を知っているかもしれませんが、生成モデルがヘッジファンド戦略と庭のヘッジ (生け垣) の違いを理解していなければ、回答は正しくないものとなります。このギャップを埋めるのがファインチューニングです。
- **検索拡張ファインチューニング (RAFT)**：ファインチューニングされたモデルと RAG を組み合わせてエンタープライズデータとの密接な連携を実現し、いずれかのテクニック単独よりも高い精度と関連性を提供します。RAFT は、取得したドキュメントに基づいて推論するというタスクに特化してモデルをトレーニングし、モデルは取得したコンテキストのどの部分に関連性があり、どの部分が不要な情報であるか識別することを学習します。ドメイン固有のエンタープライズユースケースでは、RAFT は、いずれのテクニックを単独で使用する場合よりも一貫して優れたパフォーマンスを発揮します。

評価によってファインチューニングの適切なタイミングがわかる

この章が評価の後に配置されているのは、ほとんどのチームは適切な評価を実行して結果を見るまで、ファインチューニングの必要性に気付かないからです。

そのシグナルは明確で、コンテキストの再現率と精度は高いものの、忠実度または回答の関連性が低いというものです。このパターンの場合、取得パイプラインは正常に機能しており、適切なドキュメントが検索され、提供されています。問題は、生成モデルがそれらを十分に解釈して正確な回答を生成することができていないことです。

このパターンが見つかった場合、最初に試すべきアプローチはプロンプトエンジニアリングです。取得したコンテキストを使用するようにモデルに指示する方法を調整することで、ギャップを解消できる場合があります。解消できない場合は、ファインチューニングが次のステップになります。これはすでに構築済みのパイプラインの拡張であり、再構築ではありません。

ラベル付きデータの問題

エンタープライズでのファインチューニングに対する最も一般的な反論は、「ラベル付けされたデータの量が十分ではない」というものです。トレーニングデータセットを手動で構築すると、特にアノテーションが対象分野の専門知識を必要とする特殊なドメインの場合、コストと時間がかかります。

プライベートデータが利用できない場合や不足している場合、Red Hat AI は合成データを生成するためのモジュール式のワークフローを提供します。SDG Hub (Red Hat AI の一部) は、既存のエンタープライズ・ドキュメントから合成トレーニングデータを生成することでこれに対処します。これにより、チームはデータセットを拡張および改良できるようになります。SDG Hub を使用して PDF、テキストファイル、構造化データをフィードし、ドメインに合わせた質問と回答のペアの厳選されたデータセットを生成することができます。その結果、手動で作成すると数週間かかるトレーニングデータが、ほんのわずかな時間で作成できます。

合成データは、あらゆる場合においてエキスパートがラベル付けしたデータに完全に代わるものではありませんが、チームが初期段階での問題を克服し、機能するファインチューニングのループへと移行するのに役立ちます。後で、手動でラベル付けした例を使用していつでも補強できます。

Training Hub によるファインチューニング

エンタープライズ向けにモデルをファインチューニングするには、いくつかの要素が関係します。まず、ドメインに適したベースモデルを選択し、LoRA (低ランク適応) などのパラメーター効率の良い手法とフルファインチューニングのどちらを選択するかを決め、ジョブを実行するためのコンピュート・インフラストラクチャを管理し、最後に、ファインチューニングしたモデルが実際に改善されたことを検証してからプロダクション環境に導入します。このすべてを異なるツールや環境にまたがって調整することは、特に初めてのファインチューニング作業を行うチームにとっては摩擦が生じる要因となります。

Red Hat AI により、Training Hub を使用してワークフロー全体を 1 カ所で管理できます。

1. 検証済みのモデルリポジトリからベースモデルを選択します。
2. 必要なカスタマイズの規模に応じて LoRA とフルファインチューニングのいずれかを選択し、ジョブを構成します。
3. Red Hat AI 分散トレーニング・インフラストラクチャで実行します。
4. LM-Eval を使用して、ファインチューニングされたモデルを、RAG パイプラインに渡す前に評価します。

この最後のステップが重要です。ファインチューニングされたモデルは、最初にファインチューニングを行うきっかけとなった指標の評価スコアが、明らかに向上するはずです。忠実度が問題だった場合、それが向上します。向上しなかった場合はトレーニングデータまたは構成を調整する必要があります。その際、第 3 章で説明した評価フレームワークを参考に調整を繰り返してください。

エージェント型 RAG

RAG を補完できるツールはファインチューニングだけではありません。

標準の RAG は固定された直線的なパイプラインに従いますが、エージェント型 RAG はエージェント型ワークフローを統合してアクティブな意思決定エコシステムを作成します。これらのワークフローは、LLM が複雑な問題に体系的に取り組むための段階的な計画を構築するのに役立ち、取得を固定された手順ではなく動的なツールとして扱います。

自律型推論

この中核となるのが ReAct (思考、行動、観察) パターンです。これにより、システムは 1 回限りの検索から自律型推論へと移行し、何をいつ、どの程度の頻度で取得するかをエージェントが決定します。

1. **思考：**エージェントは不足しているデータまたは必要なデータを分析し、次の手順を計画します。
2. **行動：**契約や請求を取得するためにモデルコンテキストプロトコル (MCP) ツールを呼び出すなど、適切なツールを呼び出します。
3. **観察：**結果を評価します。コンテキストが十分であれば、最終的な応答が生成されます。そうでなければ、ステップ 1 に戻ります。
4. **繰り返し：**このパターンを、エージェントが意思決定を行うのに十分な検証済みコンテキストを取得するまで繰り返します。

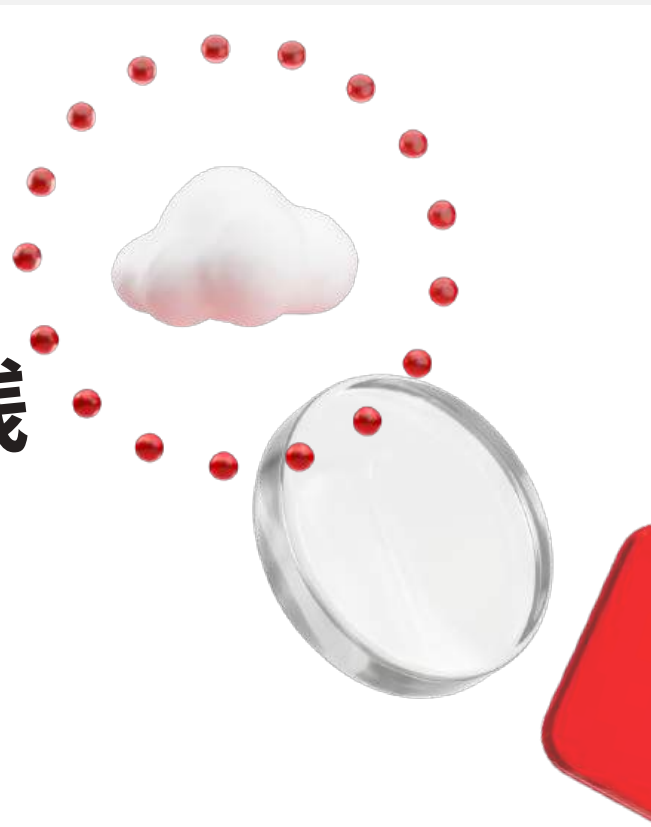
エージェント型 RAG の重要な柱

- **自律型ツールの選択：**LLM は、どの RAG ツールをどの順序で呼び出すかをその場で独自に決定します。
- **マルチソース検索：**システムは単一のセッションで、統合されていないデータベース、API、ドキュメントストアにわたってクエリを連鎖的に実行します。
- **ヒューマンインザループのエスカレーション：**信頼性が安全なしきい値を下回った場合、エージェントは完全な推論過程とともに、そのケースを人間のレビュー担当者に転送します。

次の章では、解析から評価、ファインチューニングまで、これらすべてのコンポーネントが Red Hat AI 内でどのように連携し、さまざまな役割がどのように作業を分担するかについて説明します。

06

Red Hat RAG ワークフローの実践



これまでの章では、RAG システムの各構成要素、つまり 3 つのパイプライン (インデックス作成、取得、メンテナンス)、評価、モデルのカスタマイズについて個別に説明しました。この章では、これらの要素が Red Hat AI でどのように連携し、組織内のさまざまな役割がどのように作業を分担するのかを示します。

エンドツーエンドのワークフロー

Red Hat AI 上のプロダクション RAG ワークフローは、5 つのアクティビティにわたる継続的なループとして実行されます。

1. **インデックス作成パイプラインを構築する：**生のエンタープライズ・ドキュメントを、Docker を使用して解析し、取り込みます。コンテンツは正規化、チャンク化、埋め込みが実行され、選択したベクトルストアに保存されます。メタデータは、ダウンストリームのフィルタリング、引用、監査のために保持されます。

2. **取得パイプラインを立ち上げる：**取得の構成を稼働させます。密検索、疎検索、またはハイブリッド検索を選択し、オプションでリランキングとプロンプト構築の手法を使用できます。推論は vLLM ランタイムで実行されます。規模に対応するために、Red Hat AI は llm-d という分散推論フレームワークを使用します。llm-d は、推論パイプラインをモジュール式のサービスに分割することで、予測可能でスケーラブルなパフォーマンスを提供します。AI ガードレールがモデルの入出力をリアルタイムで監視します。

3. **評価する：**LM-Eval は、評価データセットに対してパイプラインを実行し、コンテキストの再現率、コンテキストの精度、回答の忠実度、回答の関連性をスコアリングします。スコアが精度の目標を下回った場合、その指標はボトルネック (取得、生成、またはその両方) を示しています。EvalHub は、これらの実行のための統合コントロールプレーンと、各変更の前後におけるシステムパフォーマンスに関する監査証跡を提供します。

4. **必要に応じて改善する：**ギャップが取得にある場合は、構成の見直しが必要です。チャンク化、埋め込みモデル、取得戦略、またはリランキングを再評価します。ギャップが生成にある場合は、SDG Hub がエンタープライズ・ドキュメントから合成トレーニングデータを生成し、Training Hub がファインチューニング・ジョブを実行します。ファインチューニングされたモデルを、ベースラインを置き換える前に LM-Eval で再評価します。MLflow はすべての構成と結果を追跡するため、変更は暗黙知ではなく再現可能なものになります。

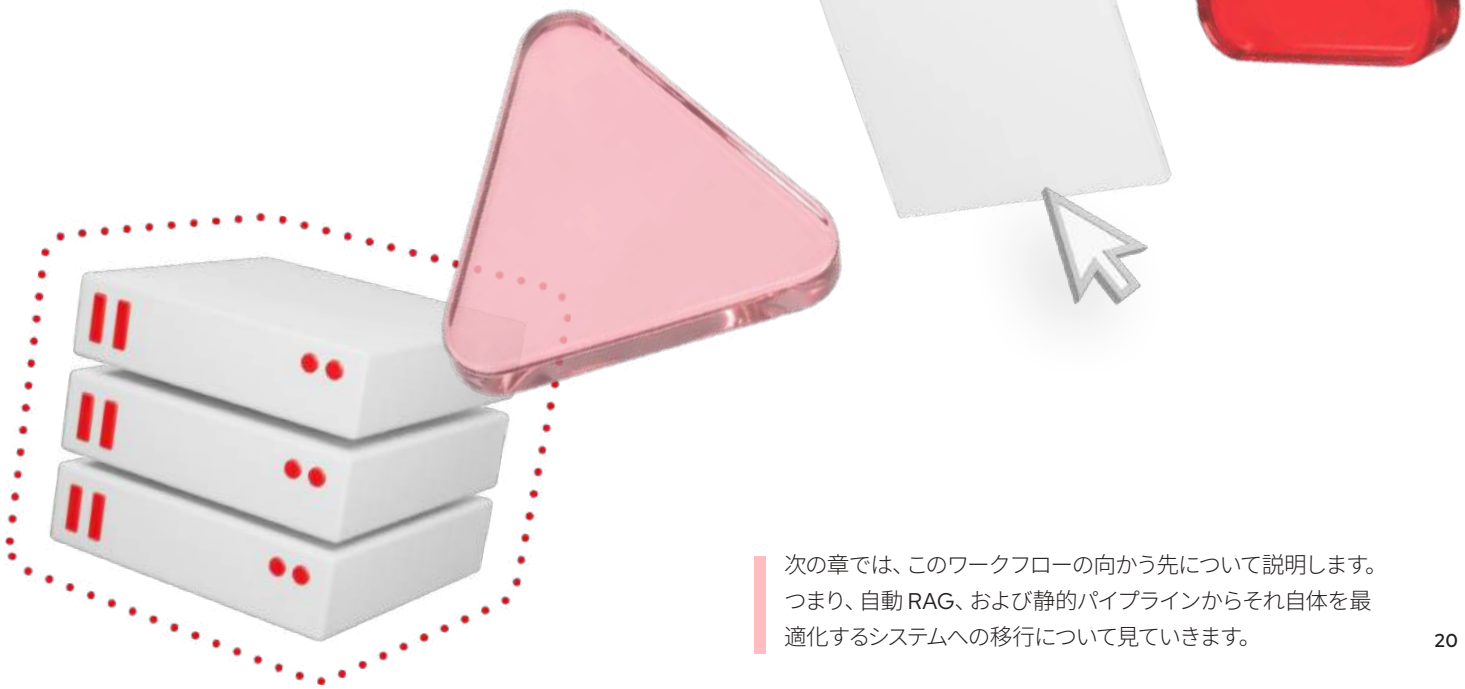
5. **継続的に保守する：**基盤となるデータの変更に応じて、メンテナンスパイプラインがコーパスを正しい状態に維持します。ソースシステムから更新と削除を伝播し、監査証跡を維持しながら忘れられる権利の要求を実行し、モデルやチャンク化戦略が変更されたときに再埋め込みを行い、アクセス制御の更新を同期し、コーパスドリフトを検出します。ドリフトが検出されると、新たな評価サイクルがトリガーされ、ループが閉じてステップ 3 に戻ります。

このセットアップは、1 回だけ行うものではありません。これはサイクルです。エンタープライズデータの変更やユーザーのニーズの進化に応じて、システムはプロダクションで知らない間に劣化するのではなく、継続的に再評価と改善を行います。

ワークフロー全体における役割

Red Hat AI は、エンタープライズのチームが AI プロジェクトに関して実際にどのように組織されているかに対応した、役割ベースのインタフェースを提供します。各役割が 3 つのパイプラインすべてに関わりますが、それぞれの重点は異なります。

- **プラットフォームエンジニア**：AI hub を通じて作業を行います。基盤となるインフラストラクチャ (モデル、MCP サーバー、API エンドポイント、グラフィックス・プロセッシング・ユニット (GPU) リソース割り当て、アクセス制御) のデプロイと管理を行います。llm-d により、プラットフォームエンジニアは大規模なクラスタのリソースを予測どおりに管理するツールを手に入れることができます。また、この役割はコンプライアンスおよびガバナンスレイヤーも担当し、ロールベースのアクセス制御 (RBAC) の構成、データプライバシーの制御を行い、機密性の高い内部データがパブリックモデルやサードパーティのライブラリに公開されないようにします。メンテナンスパイプラインでは、プラットフォームエンジニアが保持、削除、監査のポリシーと、それらを実行するインフラストラクチャを管理します。
- **AI エンジニア**：gen AI studio を通じて作業します。利用可能なモデルを見つけ、取得および生成の構成を試し、RAG ワークフローのプロトタイプを作成し、検証された構成をプロダクション環境に展開します。この役割の日常的な課題は、顧客関係管理 (CRM) システム、データウェアハウス、PDF、メール、従来のシステムなど、データが実際に存在するシステムから使用可能なデータを取得することです。gen AI studio は、インフラストラクチャの変更を待つことなく、これらの統合を反復するためのワークスペースを提供します。また、AI エンジニアは、メンテナンスパイプラインを駆動するトリガーの定義も行い、ソースシステムの変更がインフラストラクチャの再インデックス作成に値するかどうかを判断します。
- **機械学習 (ML) エンジニア**：実験とプロダクションをつなぐ役割を担います。AI エンジニアがプロトタイプとして作成したワークフローを取り入れて、適切なエラー処理、ロギング、自動化を備えたプロダクション対応のパイプラインに産業化します。3 つのパイプラインすべてにわたって可観測性と監視を担当し、パフォーマンスの低下が気づかれないまま放置されるようなギャップが、プロダクションのアプリケーションから生じないようにします。ML エンジニアは、コーパスのドリフト検出、モデルのドリフト検出、およびそれらがトリガーする評価サイクルを管理します。



次の章では、このワークフローの向かう先について説明します。つまり、自動 RAG、および静的パイプラインからそれ自体を最適化するシステムへの移行について見ていきます。

07

自動 RAG と今後の道のり

これまでの章で説明したものはすべて、手動で構成されたパイプラインを前提としています。ユーザーが質問すると、システムがチャンクを取得し、モデルが応答を生成します。取得戦略、チャンク化のアプローチ、埋め込みモデルは事前に選択され、手動で再設定しない限り固定されたままになります。

自動 RAG により、この構成プロセスから当て推量をなくすことができます。自動 RAG はパラメーターのさまざまな組み合わせを体系的に評価し、データやユースケースに最適なパフォーマンスを発揮する構成を推奨します。適切なパイプライン構成を選択するために直感や試行錯誤に頼る必要はありません。

手動チューニングから自動評価へ

従来の RAG から自動 RAG への移行は、本質的には、手作業で調整されたパイプラインから経験的に最適化されたパイプラインへの移行を意味します。自動 RAG システムは、実際のデータに対するさまざまなチャンク化戦略、埋め込みモデル、取得方法、パイプライン設定をテストします。それぞれの組み合わせのパフォーマンスを測定し、最良の結果が得られる構成を明らかにします。

実際のエンタープライズデータは多種多様であるため、この点は重要です。FAQ 形式の短いドキュメント用に最適化されたパイプラインは、長い規制関連ファイルの場合はパフォーマンスが低下する可能性があります。コードドキュメントに有効なチャンク化戦略は、法的契約書については重要なコンテキストを見逃す場合があります。体系的な評価が行われなければ、チームは推測で作業することになり、その推測はパイプラインのあらゆる段階で積み重なっていきます。

自動 RAG が最適化するもの

RAG パイプラインには多くの要素が含まれており、それぞれの選択が精度、レイテンシー、コストに影響します。自動 RAG は、パイプラインをモジュール式の最適化問題として扱い、以下の段階にわたる構成を体系的に評価します。

- **ドキュメントの解析とチャンク化：**ドキュメントの種類によって、適切な分割戦略は異なります。自動 RAG は、再帰的、階層的、およびハイブリッドのチャンク化手法を、さまざまなチャンクサイズとオーバーラップ設定で評価し、取得に最も有用なシグナルを保持する構成を見つけ出します。
- **埋め込みモデルの選択：**埋め込みモデルの選択により、クエリが関連するドキュメントにどの程度一致するかが決まります。自動 RAG は、評価データセットに対して複数の埋め込みモデルのベンチマークテストを行い、特定のコンテンツに対して最適な検索結果を生成するモデルを特定します。
- **取得戦略：**自動 RAG は、単純なベクトル類似性検索、ハイブリッド検索 (密ベクトルと疎キーワードベースの検索の組み合わせ)、さまざまなリランキングのアプローチをテストします。オプティマイザーは、どの取得方法がユースケースに対して最も高い精度と再現率をもたらすかを調べます。
- **生成パラメーター：**自動 RAG はさまざまな基盤モデルと生成設定を評価して、取得したコンテキストから最も忠実度の高い正しい回答を生成する組み合わせを見つけます。

Red Hat AI 上の AutoRAG

プロダクショングレードの RAG パイプラインを手動で構築するには、チャンク化戦略のテスト、埋め込みモデルの比較、取得のチューニング、回答品質の評価など、何週間も試行錯誤する必要があります。詳細な技術ドキュメントに最適な構成は、短い製品説明や法的文書の場合とは異なります。AutoRAG は、この手動によるチューニングを、構造化および自動化されたプロセスに置き換えます。

Red Hat OpenShift® AI 3.4 のテクノロジープレビューとして利用できる AutoRAG は、gen AI studio を通じてガイド付きのエクスペリエンスを提供します。AI エンジニアがドキュメントと評価データセットをアップロードし、候補となる基盤モデルと埋め込みモデルを選択して、最適化の実行を開始します。次に、AutoRAG が RAG パイプライン構成を体系的にテストし、回答の忠実度、回答の正確性、コンテキストの正確性などの指標を使用して、評価データセットの組み合わせをそれぞれ評価します。

次のステップとリソース

どこから始めるべきか

初めて RAG パイプラインを構築する場合は、第 2 章のパイプライン・アーキテクチャから始めて、LM-Eval をセットアップしてから他の作業を進めてください。初日から評価を実施しておくことで、後で何週間も当て推量で作業する必要がなくなります。

すでにパイプラインがあり、パフォーマンスが低い場合は、第 3 章の評価フレームワークから始めてください。処理を実行し、指標を確認し、第 4 章の診断ガイダンスを使用して、具体的なボトルネックを特定します。

評価の結果、ファインチューニングが必要だとわかった場合は、SDG Hub と Training Hub を確認して、手動でラベル付けされたデータを待つことなく開始してください。

手動による構成チューニングを完全に省略したい場合は、パイプラインの最適化を最初から自動化する自動 RAG を検討してください。

- Red Hat AI に含まれる、RAG パイプライン構築のためのツールの詳細をご覧ください。
- Red Hat AI の詳細をご確認ください。
- Red Hat の AI/ML サービスの詳細をご確認ください。
- 無料のディスカバリー・セッションを予約しましょう。
- RAG の大規模なデータ処理について詳細をご確認ください。