Amazon Bedrock、Amazon Auroraを組み合わせたRAGで回答精度の向上に取り組んでみた！

セゾン情報システムズ　石原直樹

自己紹介

- **普段の業務**
  - グループ企業のCCoE
  - 社内LLM研究会にも所属
- **好きなAWSサービス**
  - Amazon Bedrock
- **トピック**
  - 来週AWS MLS受験します！

aws PARTNER NETWORK

2023 Japan AWS Jr. Champions

@Ishihara_Naok1 X最近始めました！

宣伝！

https://minorun365.connpass.com/event/309029/

お助けメンバーとして皆さんを全力でサポートします！
興味ある方是非ご応募ください！

今日話す内容

- AWSにおけるRAG実装方法3選
- ↑のうち1つの回答精度向上手法
- 実際に精度向上したか評価（3つのRAG回答と比較）
- 今回の検証で苦戦したこと

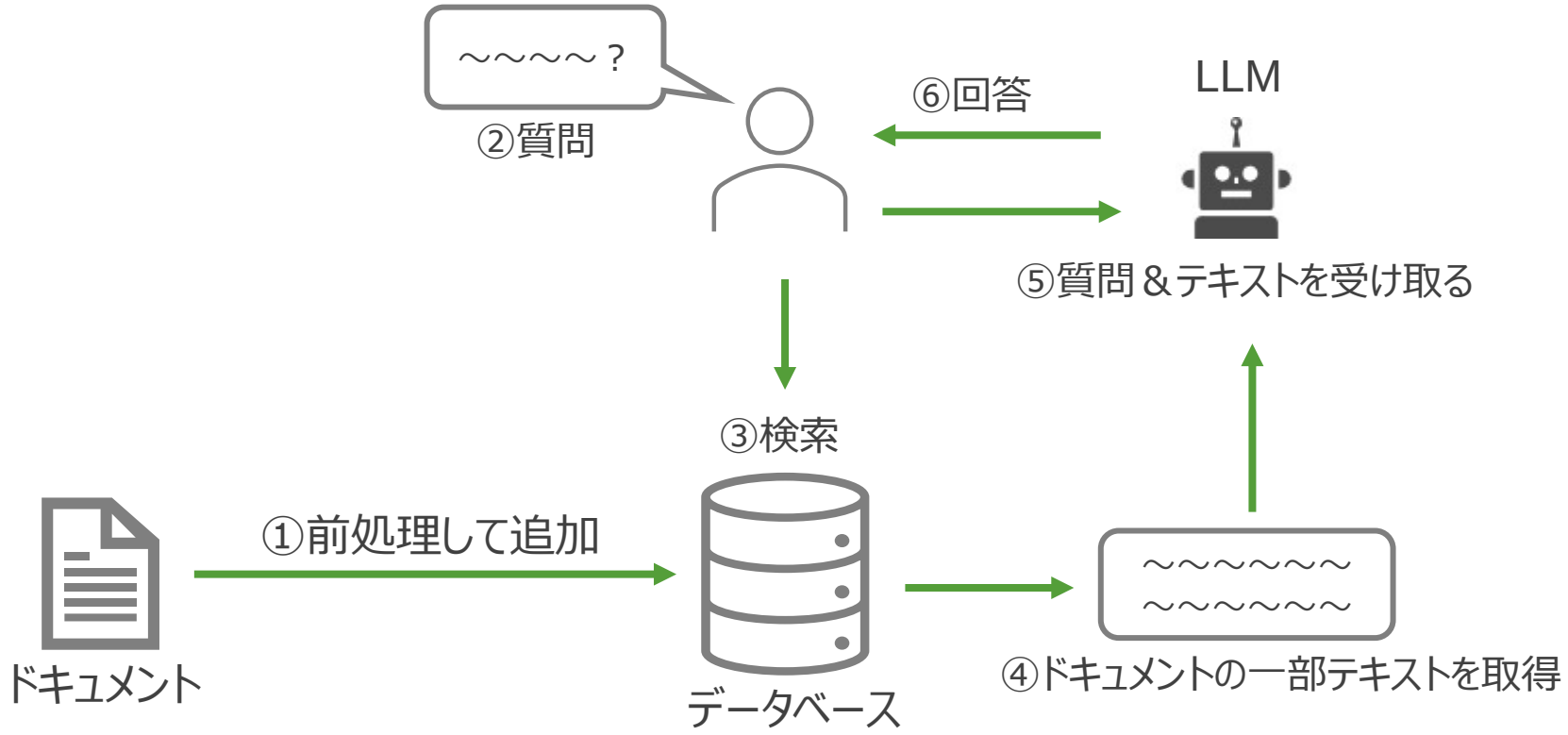
お気軽に質問ください！

はじめに

RAGとは

・LLMが**外部のDB**から情報を取得し回答を生成する仕組み

用途例：社内ナレッジに関する質問に回答できるAIチャットボット



RAG回答生成までの流れ

AWS上でRAGを実装する方法3選

※ AWSが提供している全文検索エンジン　　※ 2023.11.29 一般利用開始

RAGでの実装 アプローチ	1. Bedrock & Kendra※	2. Knowledge Base for Amazon Bedrock	3. Bedrock & ベクター DB
RAGのフロー処理	**要開発**	AWSマネージドで提供 ＝開発不要	**要開発**
ドキュメントの前処理	AWSマネージドで提供 ＝開発不要	AWSマネージドで提供 ＝開発不要	**要開発**
検索処理	AWSマネージドで提供 ＝開発不要	AWSマネージドで提供 ＝開発不要	**要開発**
DBの選択肢	Kendra	OpenSearch Serveless Aurora PostgreSQL Pinecone Redis Enterprise Cloud	OpenSearch Service/Serveless Document DB MemoryDB for Redis Aurora/RDS PostgreSQL Pinecone Redis Enterprise Cloud その他 3rd Party SaaS

※ Amazon Q For Business Useは省略しています

AWS上でRAGを実装する方法3選

※ AWSが提供している全文検索エンジン　　※ 2023.11.29 一般利用開始

RAGでの実装 アプローチ	1. Bedrock & Kendra※	2. Knowledge Base for Amazon Bedrock	3. Bedrock & ベクター DB
RAGのフロー処理	**要開発**	AWSマネージドで提供 ＝開発不要	**要開発**
ドキュメントの前処理	AWSマネージドで提供 ＝開発不要	AWSマネージドで提供 ＝開発不要	**要開発**
検索処理	AWSマネージドで提供 ＝開発不要	AWSマネージドで提供 ＝開発不要	**要開発**
DBの選択肢	Kendra	OpenSearch Serveless Aurora PostgreSQL Pinecone Redis Enterprise Cloud	OpenSearch Service/Serveless Document DB MemoryDB for Redis Aurora/RDS PostgreSQL Pinecone Redis Enterprise Cloud その他 3rd Party SaaS

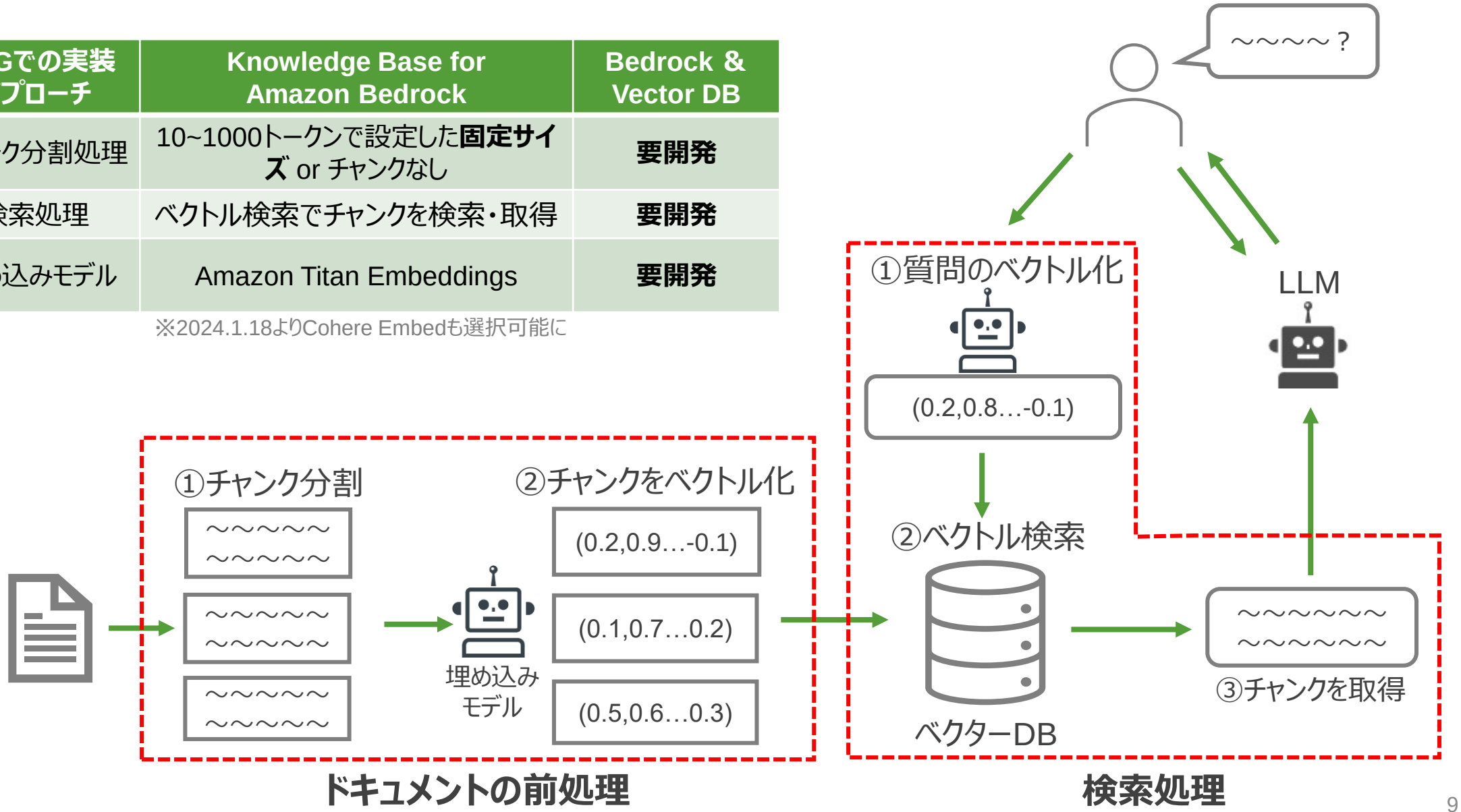
それぞれの処理をカスタム実装することで**精度の向上**に挑戦
最小コストが低く※管理がしやすい**Aurora Serveless**を選択

※ Aurora Serverless v2 0.5 ACU **$43.8/月**
OpenSearch Serveless 2 OCU **＄350.4/月**
Kendra Developer Edition **＄810/月**

補足：ベクターDBを用いたRAGの処理

RAGでの実装アプローチ	Knowledge Base for Amazon Bedrock	Bedrock & Vector DB
チャンク分割処理	10~1000トークンで設定した**固定サイズ** or チャンクなし	**要開発**
検索処理	ベクトル検索でチャンクを検索・取得	**要開発**
埋め込みモデル	Amazon Titan Embeddings	**要開発**

※2024.1.18よりCohere Embedも選択可能に



今回検証した精度向上手法

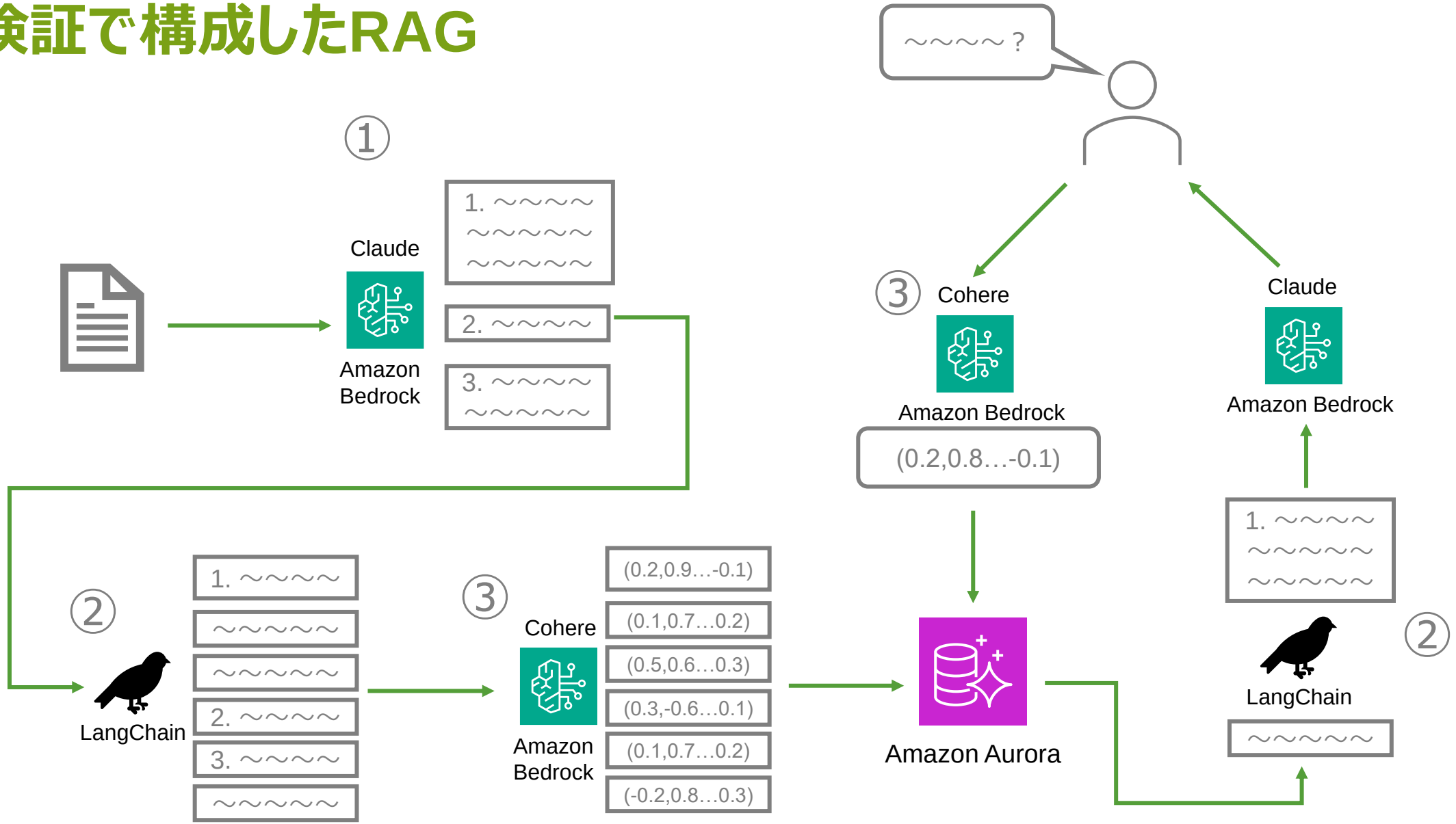
検証した精度向上手法3つ

① Claudeによるチャンク分割

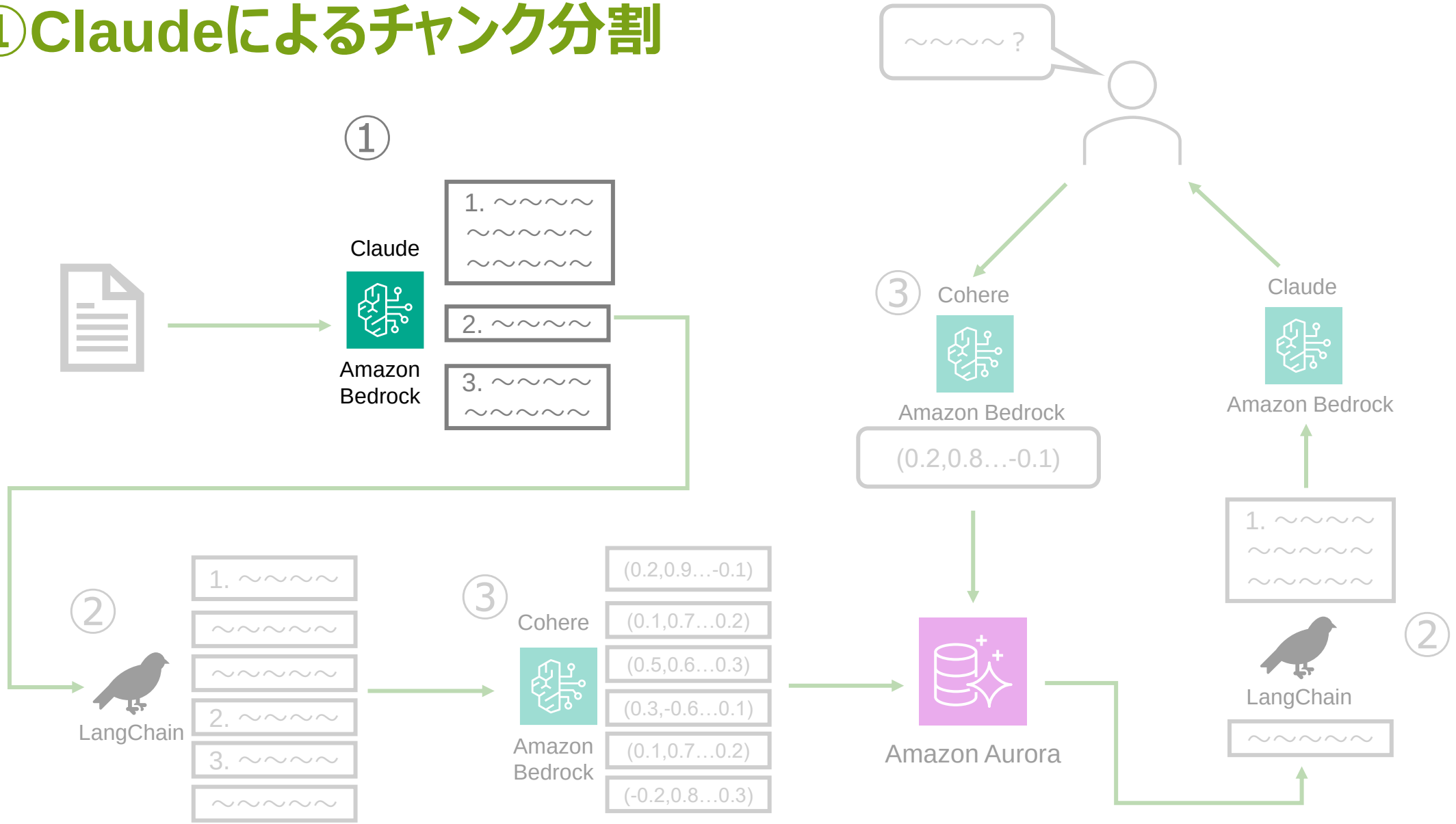
② LangChainの高度な検索機能

③ 高性能な埋め込みモデル

検証で構成したRAG


~~~~~ ?
①
Claude
Amazon Bedrock
1. ~~~~~ ~~~~~~ ~~~~~~
2. ~~~~~
3. ~~~~~ ~~~~~~
③
Cohere
Amazon Bedrock
(0.2,0.8…-0.1)
Claude
Amazon Bedrock
②
LangChain
1. ~~~~~
~~~~~~
~~~~~~
2. ~~~~~
3. ~~~~~
~~~~~~
③
Cohere
Amazon Bedrock
(0.2,0.9…-0.1)
(0.1,0.7…0.2)
(0.5,0.6…0.3)
(0.3,-0.6…0.1)
(0.1,0.7…0.2)
(-0.2,0.8…0.3)
Amazon Aurora
1. ~~~~~ ~~~~~~ ~~~~~~
②
LangChain
~~~~~~
~~~~~~

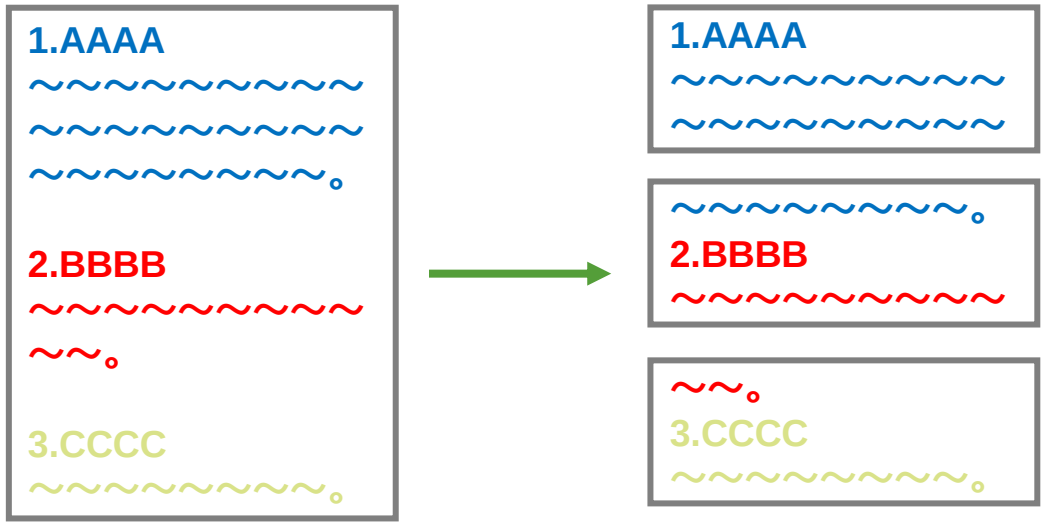
①Claudeによるチャンク分割



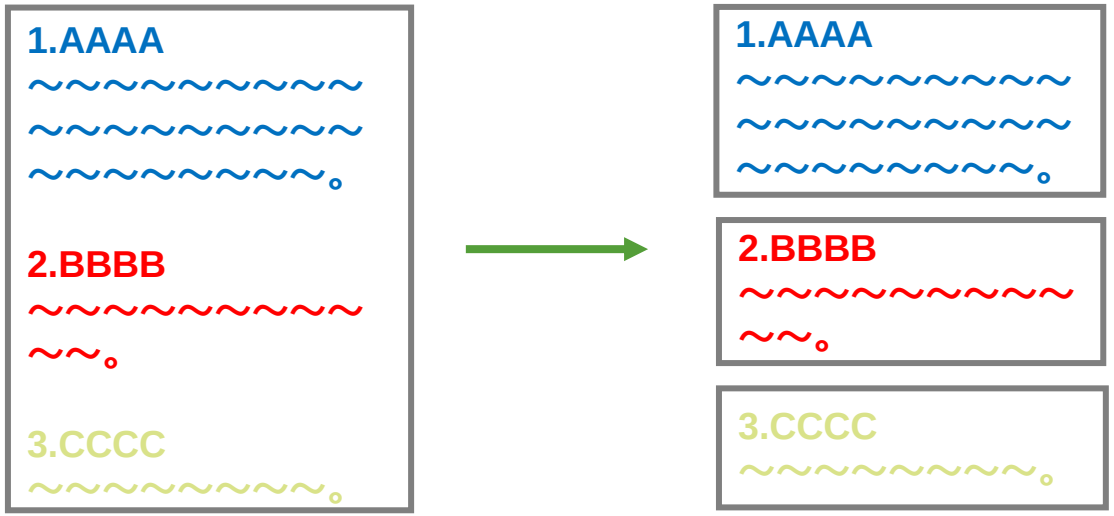
チャンク分割による回答精度への影響

※例：Knowledge base bedrockでのチャンク分割

・固定サイズでのチャンク分割※



・セクションごとのチャンク分割

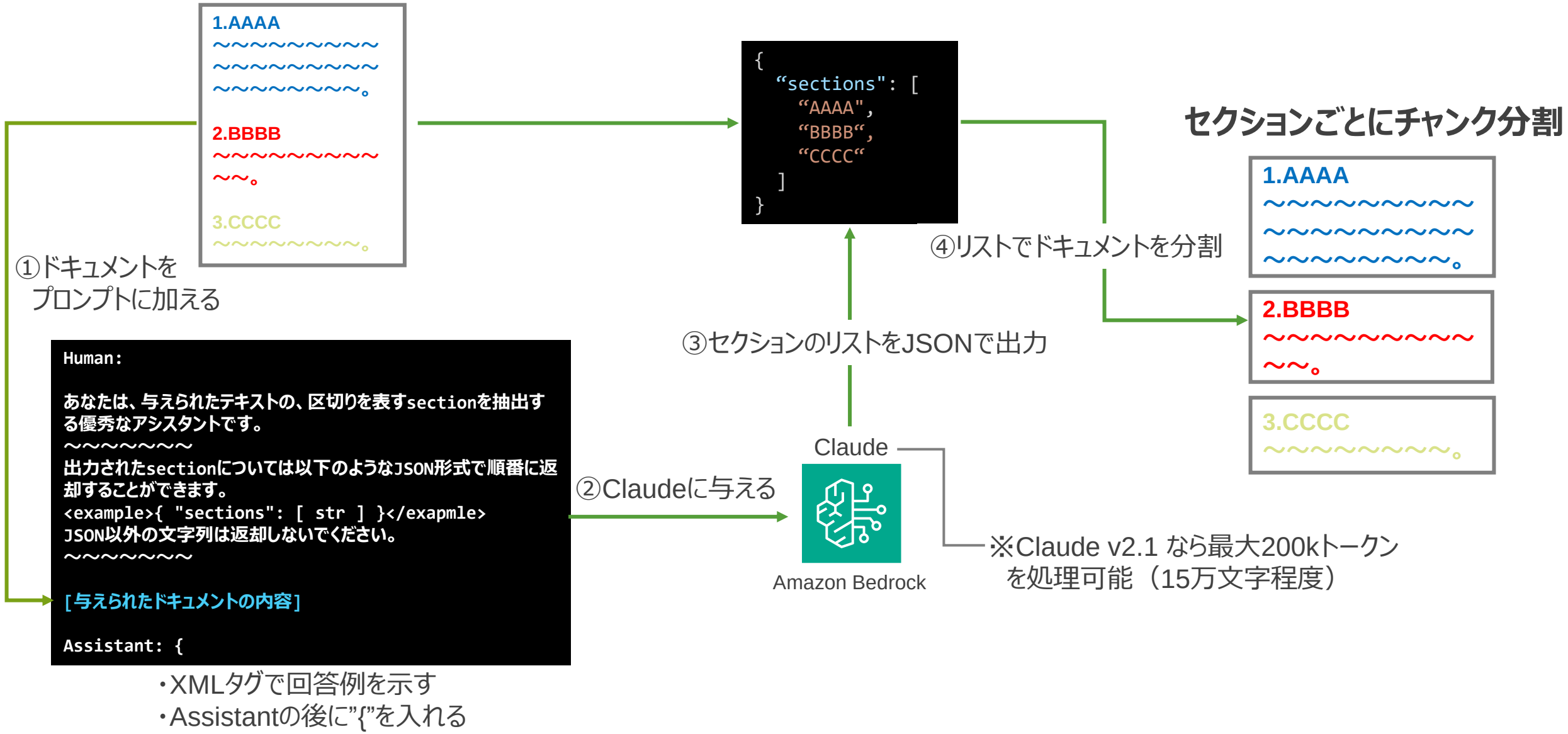


- 1チャンクに異なるセクションの内容が含まれる
  - LLMに**不要な**情報や**断片的**な情報を渡す
    - 回答精度が**悪化**

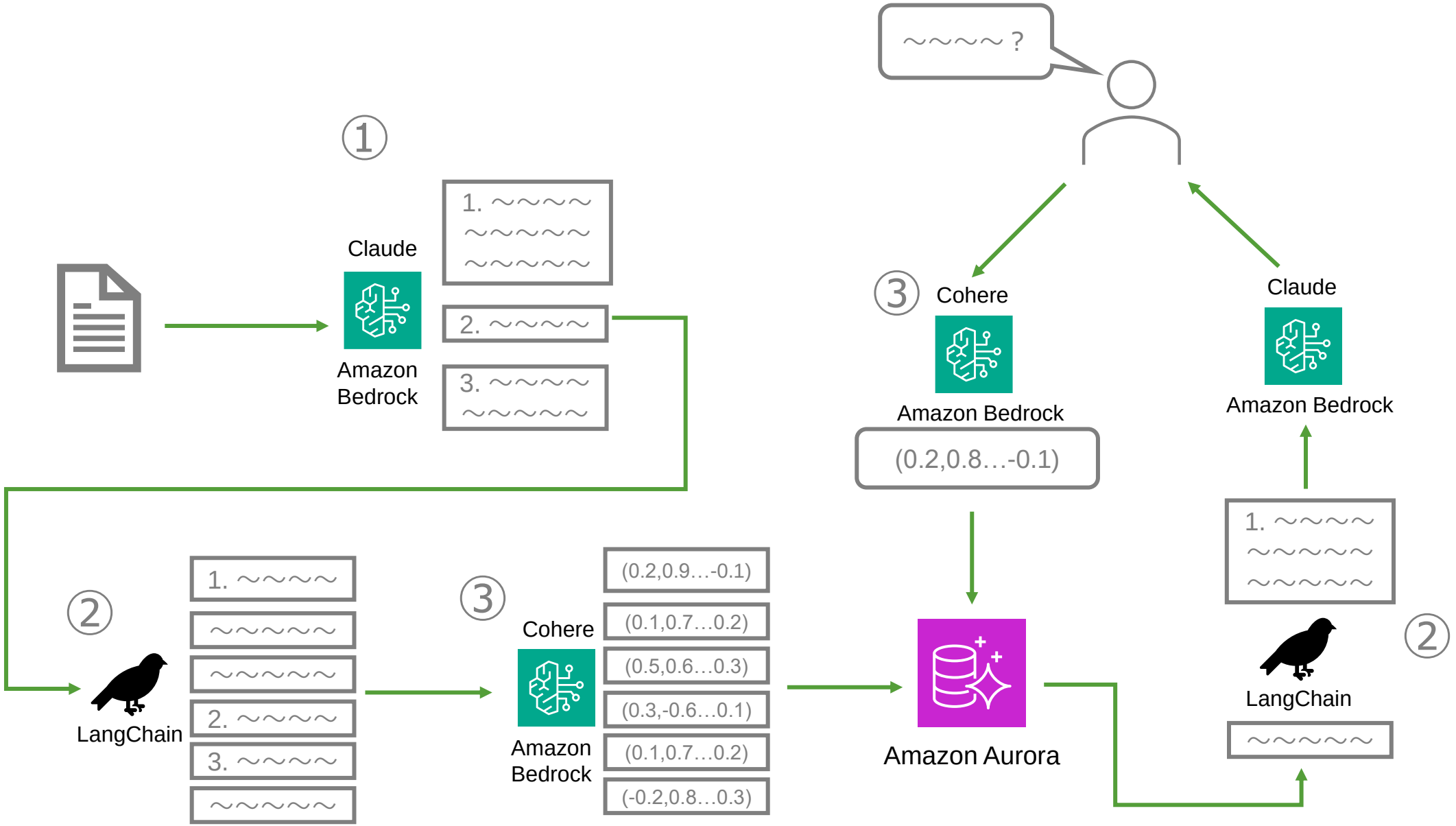
- 1チャンクに1セクションの内容がまとまっている
  - LLMに**まとまった情報**を渡すことができる
    - 回答精度の**向上**につながる

可能であれば**セクション単位で分割**したい

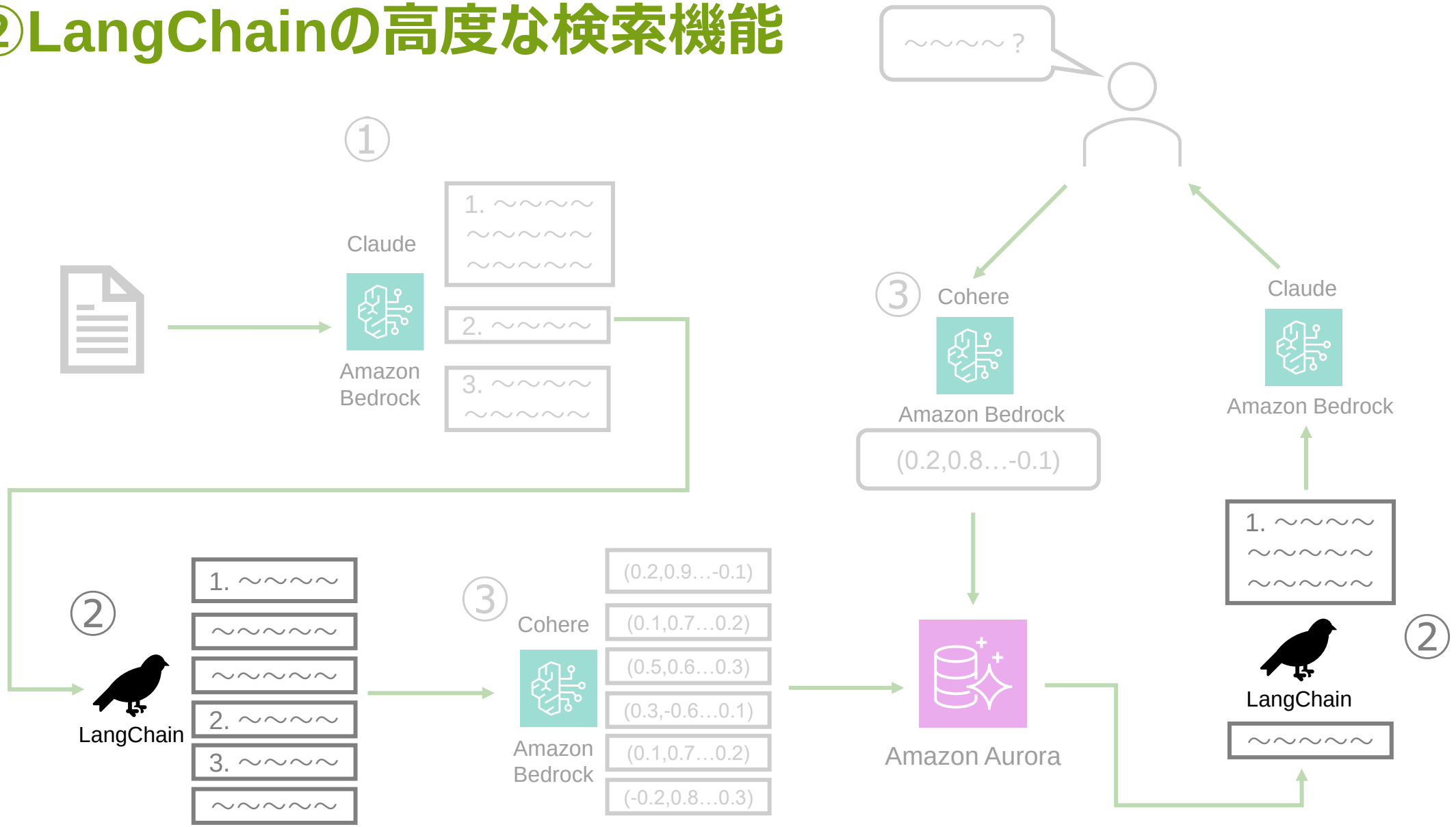
Claudeによるチャンク分割※



※参考：GPT-4 Turboにドキュメントのチャンク分けを任せてみる(https://tech.explaza.jp/entry/2023/12/04/162659)

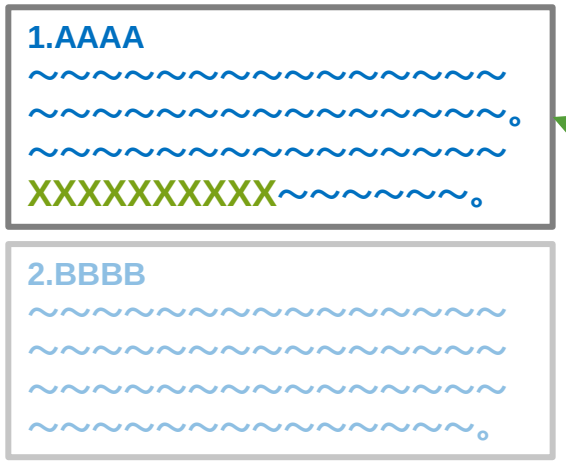

~~~~ ?
①
Claude
1. ~~~~
~~~~~
~~~~~
2. ~~~~
3. ~~~~
~~~~~
Amazon
Bedrock
③
Cohere
Amazon Bedrock
(0.2,0.8…-0.1)
Claude
Amazon Bedrock
1. ~~~~
~~~~~
~~~~~
②
LangChain
1. ~~~~
~~~~~
~~~~~
2. ~~~~
3. ~~~~
~~~~~
③
Cohere
Amazon
Bedrock
(0.2,0.9…-0.1)
(0.1,0.7…0.2)
(0.5,0.6…0.3)
(0.3,-0.6…0.1)
(0.1,0.7…0.2)
(-0.2,0.8…0.3)
Amazon Aurora
LangChain
②
~~~~~

②LangChainの高度な検索機能

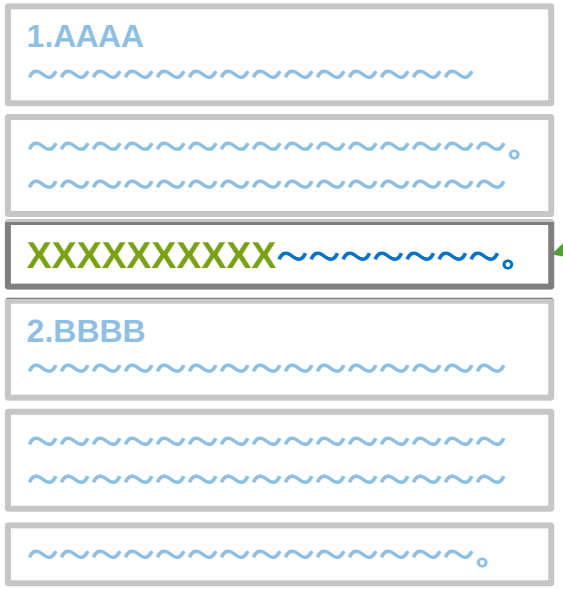


チャンクサイズとベクトル検索の精度

・チャンクサイズが大きい



・チャンクサイズが小さい

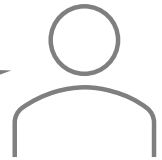


・ベクトル（意味）は遠い
・検索でヒットしにくい
・情報は**まとまっている**

・ベクトル（意味）は**近い**
・検索で**ヒットしやすい**
・情報は断片的

質問(クエリ)

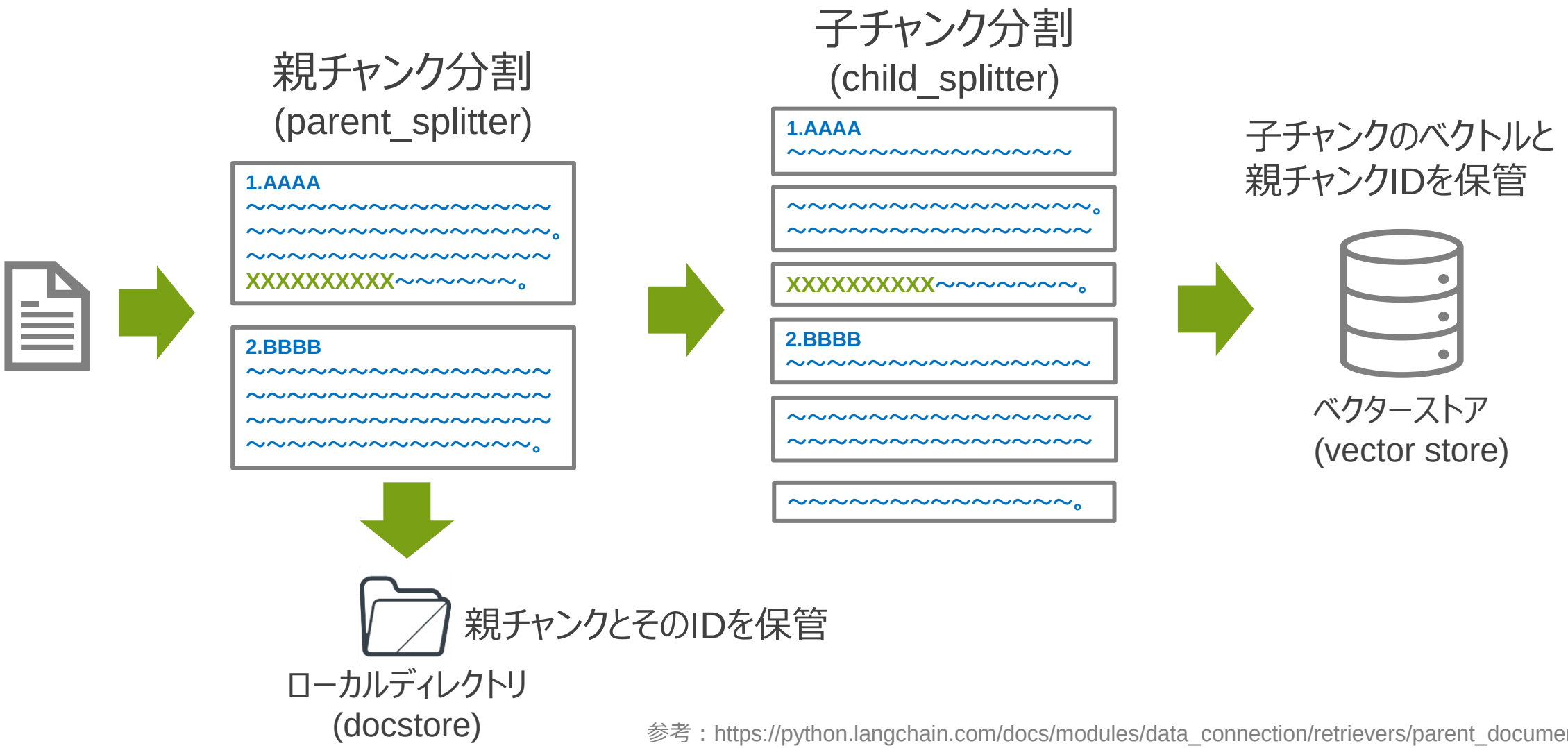
XXXXXXXXXX？



検索対象としてのチャンクは**小さく**
LLMに渡すチャンクは**大きく**したい

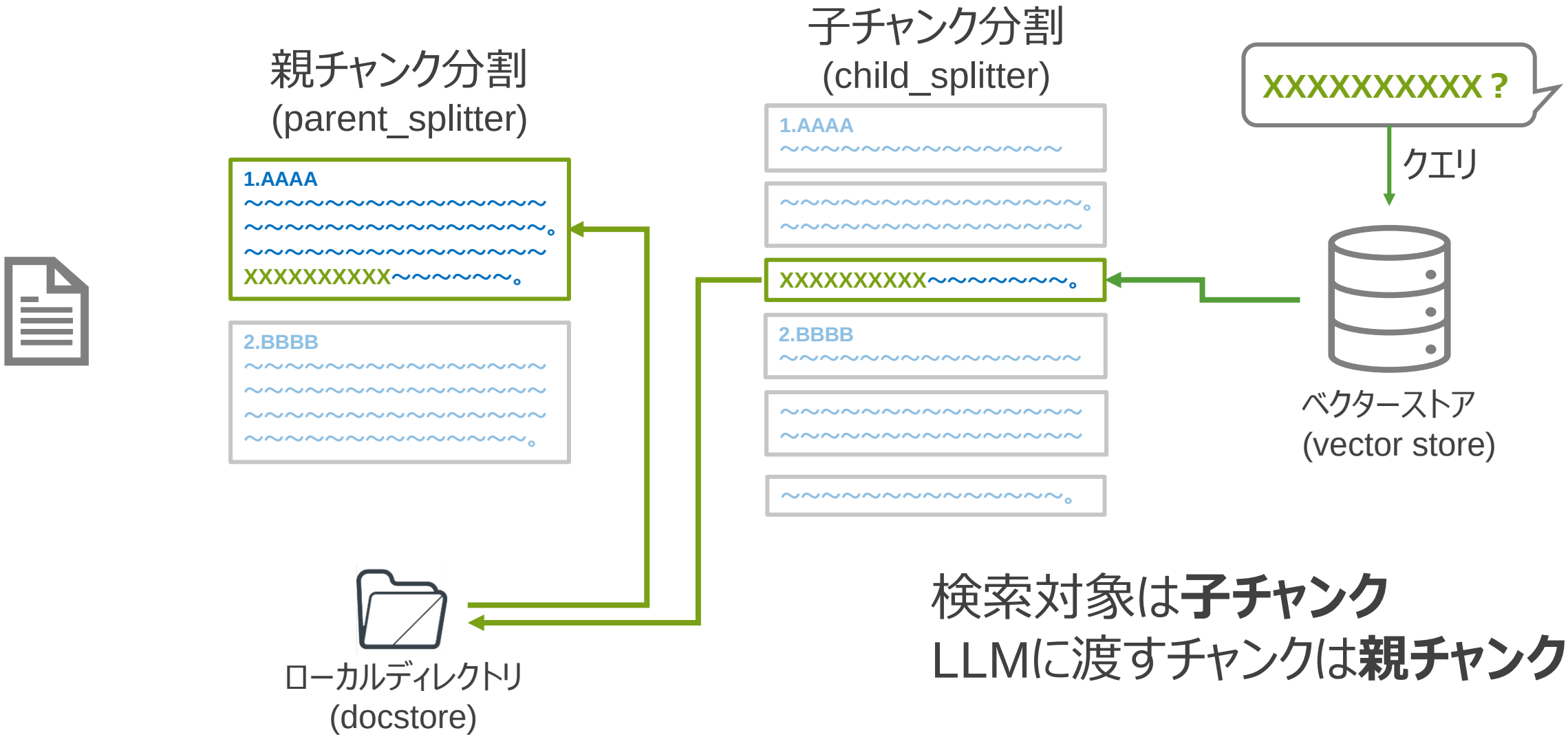
LangChain※のParent Document Retrieverを使用

※LLMを使ったアプリケーション開発のOSSフレームワーク

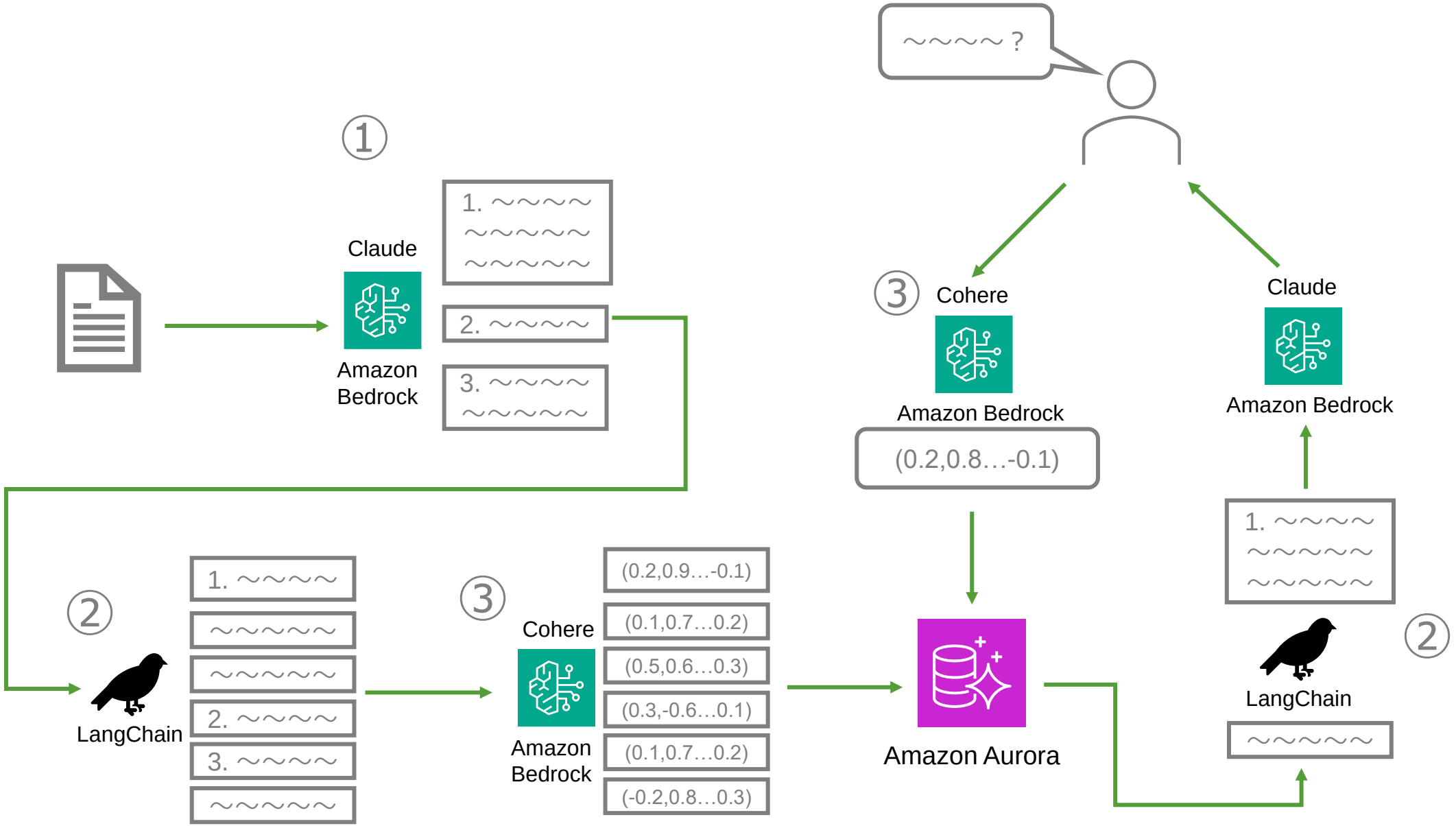


参考：https://python.langchain.com/docs/modules/data_connection/retrievers/parent_document_retriever

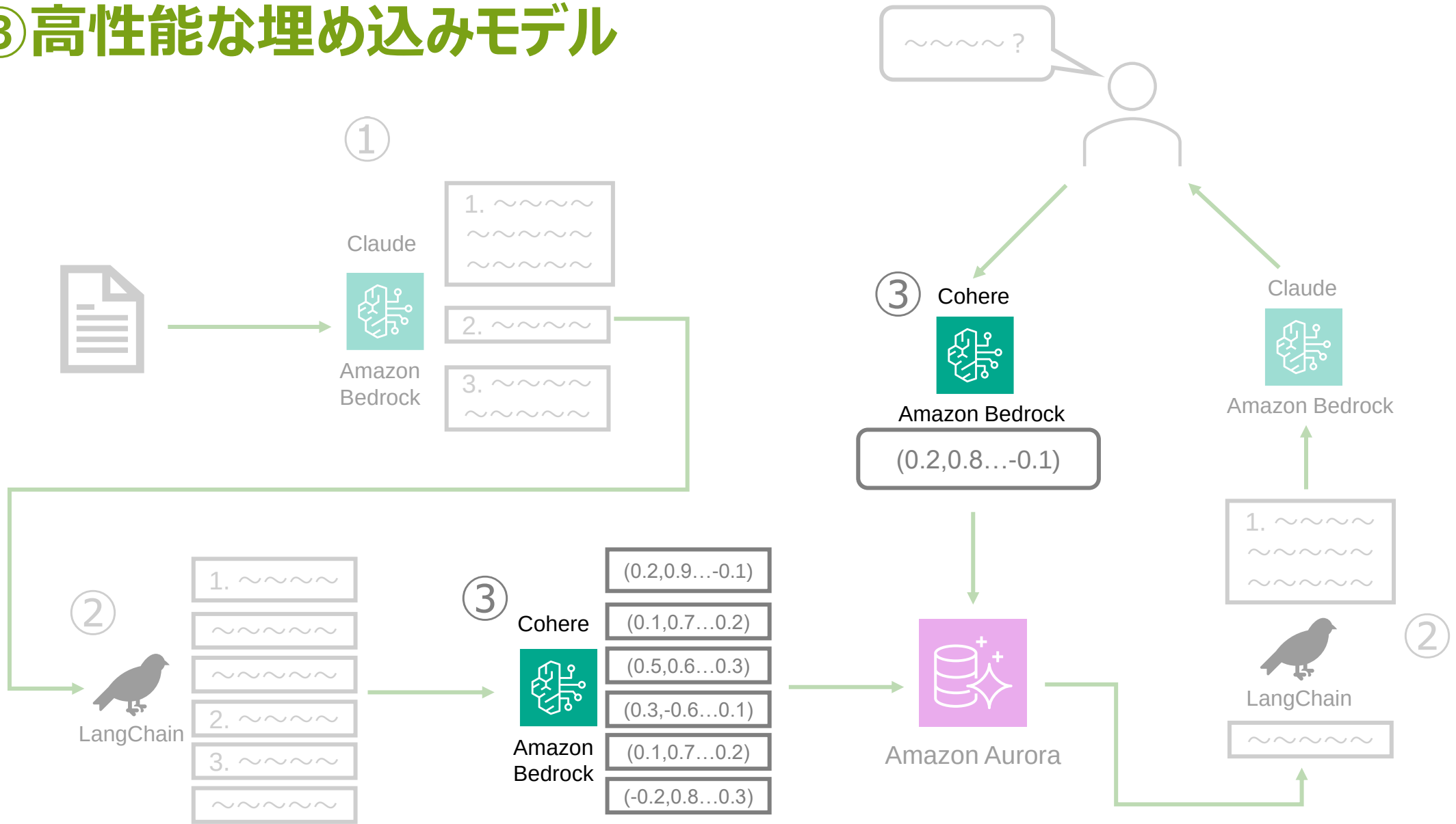
LangChainのParent Document Retrieverを使用



検索対象は**子チャンク**
LLMに渡すチャンクは**親チャンク**


~~~~ ?
①
Claude
1. ~~~~
~~~~~
~~~~~
2. ~~~~
3. ~~~~
~~~~~
Amazon
Bedrock
③
Cohere
Amazon Bedrock
(0.2,0.8…-0.1)
Claude
Amazon Bedrock
1. ~~~~
~~~~~
~~~~~
②
LangChain
1. ~~~~
~~~~~
~~~~~
2. ~~~~
3. ~~~~
~~~~~
③
Cohere
Amazon
Bedrock
(0.2,0.9…-0.1)
(0.1,0.7…0.2)
(0.5,0.6…0.3)
(0.3,-0.6…0.1)
(0.1,0.7…0.2)
(-0.2,0.8…0.3)
Amazon Aurora
LangChain
②
~~~~~

③高性能な埋め込みモデル



埋め込みモデルにCohere Embed Multilingualを採用

理由：埋め込みモデルのベンチマークMTEBの評価が高い※※

Cohere Embed Multilingual > text-embedding-ada-002 > Amazon Titan Embeddings

↑Amazon Bedrock 利用可　↑Azure OpenAI Service利用可　↑Amazon Bedrock 利用可

特徴：一度に埋め込みできる最大サイズは**512トークン**（Titanは8192トークン）

- Parent Document Retrieverと組み合わせることで、埋め込むチャンクが小さくても、大きなチャンクをLLMに渡すことができる

※「OpenAI の Embeddings API はイケてるのか、定量的に調べてみる [AWS の Embeddings を追加]」
https://qiita.com/akeyhero/items/ce371bfed64399027c23

※ Cohereの多言語用の埋め込みモデルを日本語で評価してみる
https://hironsan.hatenablog.com/entry/2023/11/06/133504

評価

今回検証したRAGの評価について

- databricks社の記事※を参考にアレンジ
  - 回答の**正しさ**、**包括性**、**読みやすさ**を0~3の4**段階**でGPT-4に評価させる
  - それぞれに重み付け（正しさ**60%**、包括性**20%**、読みやすさ**20%**）し総合評価点を算出
- 以下条件でそれぞれの回答と比較

※Best Practices for LLM Evaluation of RAG Applications
https://www.databricks.com/blog/LLM-auto-eval-best-practices-RAG

RAGの実装方法	1. Bedrock ＆ Kendra	2. Knowledge Base for Amazon Bedrock	3. 今回検証したRAG
チャンクサイズ	設定なし	300トークン	親チャンク=セクションごとのチャンク 子チャンク=100トークン
検索処理	全文検索＆セマンティック検索	ベクトル検索でチャンクを検索・取得	Parent Document Retriever
埋め込みモデル	設定なし	Amazon Titan Embeddings	Cohere Embed Multilingual
DB	Kendra	Aurora Serveless v2	Aurora Serveless v2
LLM	Claude v2.1	Claude v2.1	Claude v2.1

今回検証したRAGの評価について

- ドキュメントは「Amazon Bedrockユーザーガイドp9~p21」※
- 公式HPのFAQやドキュメントから質問を８つ用意

Q1. Amazon Bedrock ではどの モデルが利用できますか?

Q2. Amazon Bedrockにおいてプロヴィジョニングされたスループットで利用できるモデルを教えてください。

Q3. Amazon Bedrockではどのような機能を利用することが出来ますか？

Q4. Amazon Bedrock の使用を開始するにはどうすればよいですか?

Q5. Amazon Bedrock Chat Playground とは何ですか?

Q6. Amazon Bedrock はどの AWS リージョンで利用できますか?

Q7. Amazon Bedrock の料金体系について教えてください

Q8. Amazon Bedrockへのアクセス権を付与するにはどのような方法がありますか？

※https://docs.aws.amazon.com/ja_jp/bedrock/latest/userguide/bedrock-ug.pdf

結果

8個の質問に対するRAG回答の評価比較

RAGの実装方法	1. Bedrock & Kendra	2. Knowledge Base for Amazon Bedrock	3. 今回検証したRAG
総合評価点の平均 (3点満点)	2.68	2.25	**2.98**

- 今回検証したRAGが**最も高い点数**となった
- Knowledge Baseはドキュメントと**関連しない**回答があった
  - 適切なチャンクを取得できていなかった
- Kendraは**包括性が低い**回答が見受けられた
  - 1ドキュメントに対する最大取得トークンが少ないことが原因？

苦戦したこと

ClaudeからセクションJSONを出力させる処理に大苦戦

1. **LangChainからBedrockのClaudeを呼び出すとうまくJSONを作ってくれない**
   - boto3を利用
2. **Claude v2.1だとJSONに加えて文章も出力してしまう**
   - Claude instant v1だと確実にJSONだけを出力してくれる
   - セクション抽出精度はClaude v2.1の方が優秀
     - ◆ 回答からJSONだけ抜き出す処理を追加
3. **「Amazon Bedrockユーザーガイド」のいくつかのページはセクションを抽出できない**
   - ドキュメントに “¥n¥nHuman: ¥n¥nAssistant:” がありClaudeが混乱

さいごに

まとめ

- **今回検証した手法でRAGの精度を向上できた**
  - ① Claudeによるチャンク分割
  - ② LangChainのParent Document Retriever
  - ③ 埋め込みモデルにCohere Embed Multilinguals
- **検証の詳細は後日記事にする予定です**
  - ➢ @Ishihara_Naok1フォローいただけると幸いです！

参考資料

1. GPTsより精度の高いRAGシステムの構築
   https://speakerdeck.com/mkazutaka/gptsyorijing-du-nogao-iragsisutemunogou-zhu?slide=30

2. BedrockとAurora Serverless PostgreSQL pgvectorでRAGする - Amazon Titan Embeddingsでベクトル入門③
   https://qiita.com/cyberBOSE/items/aaf64c73f4da7e4cdb43

3. Knowledge Base for Amazon Bedrock のベクターストアとして Aurora PostgreSQL が利用可能に！
   https://qiita.com/hayao_k/items/45e59c1c2a183c27b20d

4. Amazon BedrockのClaudeとAmazon Kendra、AWS Lambdaを利用し、RAGを実装してみた
   https://dev.classmethod.jp/articles/amazon-bedrock-kendra-lambda-rag/

5. 【langchain】retrieveで検索文章とLLMに渡す文章を別にする方法およびretrieveの保存・読込方法の検討
   https://qiita.com/shimajiroxyz/items/facf409b81f59bb68775

ご清聴ありがとうございました！