



# Azure Cosmos DBのベクトル検索機能で Azure OpenAIとの連携を高度化しよう

～初心者向け機能概要と実装例の紹介

2023/12 [DB Tech Showcase セッション資料ベース]

Global Black Belt OSS-Data, Asia  
マイクロソフトコーポレーション

# 社内でAzure OpenAIを使う場合の課題(例)

Chat session

 Clear chat

 Playground Settings

☒ Show raw JSON

●●商事について教えてほしい





商事とは、企業活動や商取引に関連する法律や経済の分野を指します。商事には、法人や個人が商品やサービスを生産し、市場で販売する過程や、契約や取引の手続き、法的な責任などが含まれます。

商事に関連する法律には、商法や契約法、労働法、消費者保護法などがあります。これらの法律は、企業の設立や運営、契約の締結や履行、労働者の権利保護、消費者の権利保護などを規定しています。



商事における主な活動には、商品の生産や流通、販売、マーケティング、広告、貿易、投資などがあります。企業は、市場で需要のある商品やサービスを提供することで利益を追求します。また、企業は競争力を維持するために、市場調査や競合分析、ブランド戦略などを行うこともあります。

2

# 社内でAzure OpenAIを使う場合の課題(例)

?!

Chat session

 Clear chat  Playground Settings ☒ Show raw JSON

ちょっと裏側に手を入れると・・・

●●商事について教えてほしい



●●商事株式会社は、東京都杉並区に本社を構えるスパイスの卸業者です。創業から100年以上の歴史を持ち、日本全国に50社の支社を持っています。また、インドと台湾にも支社を展開しています。取引開始は2020年10月からであり、2021年4月には1000ユーザーのソフトウェア契約を受け、契約番号AB0001が付けられました。さらに、2022年4月には追加契約として2000ユーザーの契約が行われ、契約番号AB0002が付けられました。営業部の部長や課長とのコンタクトもあり、2022年4月以降は経理部の課長ともコンタクトを取っています。以上が、●●商事株式会社の概要です。何か他に知りたい情報があればお知らせください。

# 社内でAzure OpenAIを使う場合の課題(例)

## Assistant setup

System message

Add your data

⇒ Apply changes

↶ Reset to default

System message ⓘ

あなたは営業支援のデータ提供者です。ユーザーの質問に対し、以下の社内システムからの情報を勘案して適切な返答を行ってください。

社内の情報についてはmarkdown形式で記述してあります。ここに記載されていない情報については絶対に回答しないでください。

#会社概要

●●商事株式会社

東京都杉並区に本社がある創業100年のスパイスの卸業者。

日本全国に50社、インドと台湾にそれぞれ支社がある。

あなたは営業支援のデータ提供者です。ユーザーの質問に対し、以下の社内システムからの情報を勘案して適切な返答を行ってください。

社内の情報についてはmarkdown形式で記述してあります。ここに記載されていない情報については絶対に回答しないでください。

#会社概要

●●商事株式会社

東京都杉並区に本社がある創業100年のスパイスの卸業者。

日本全国に50社、インドと台湾にそれぞれ支社がある。

#取引情報

2020/10 取引開始

2021/04 弊社ソフトウェア契約受諾(1000ユーザー) 契約番号AB0001

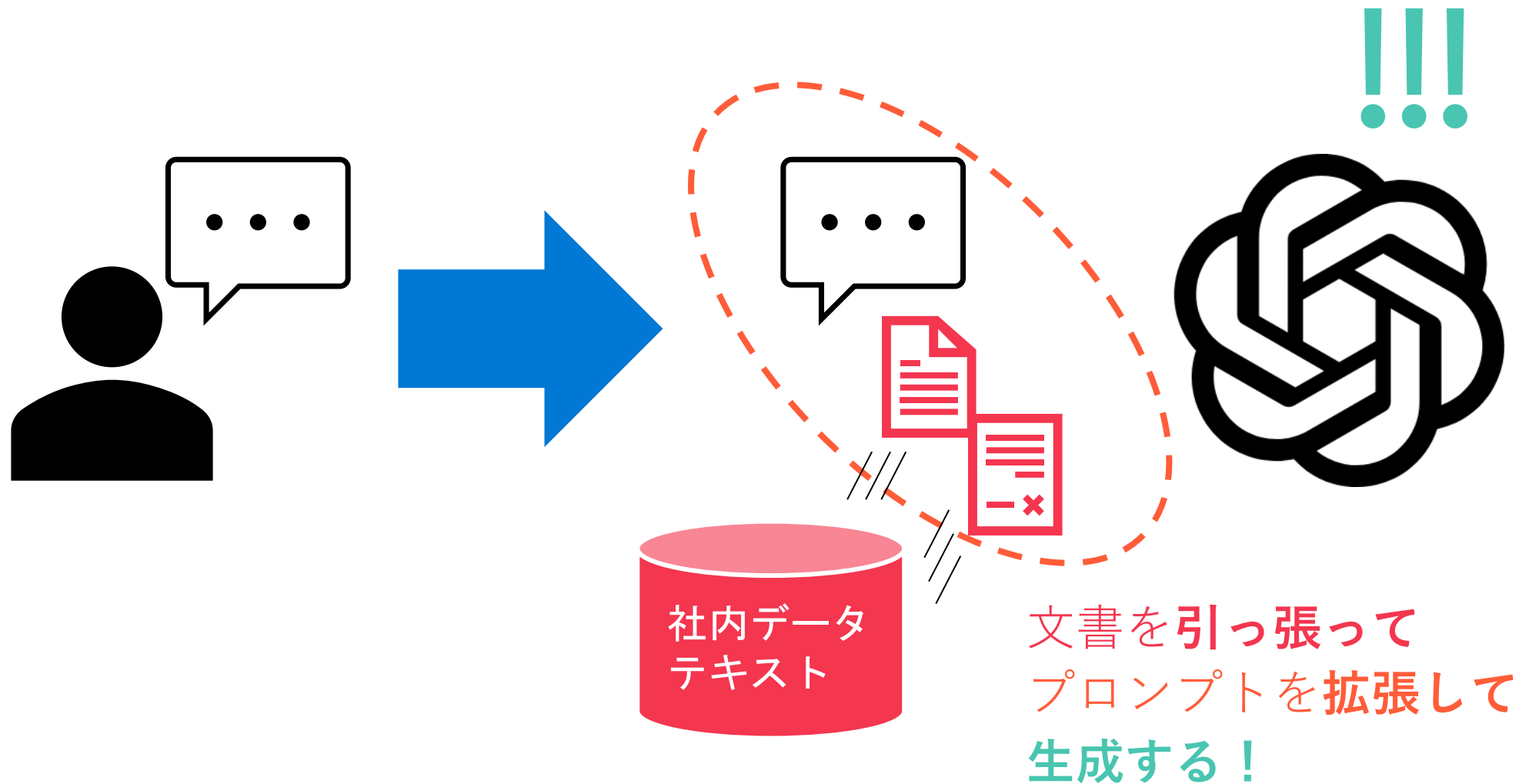
2022/04 追加契約(2000ユーザー) 契約番号AB0002

#取引先コンタクト

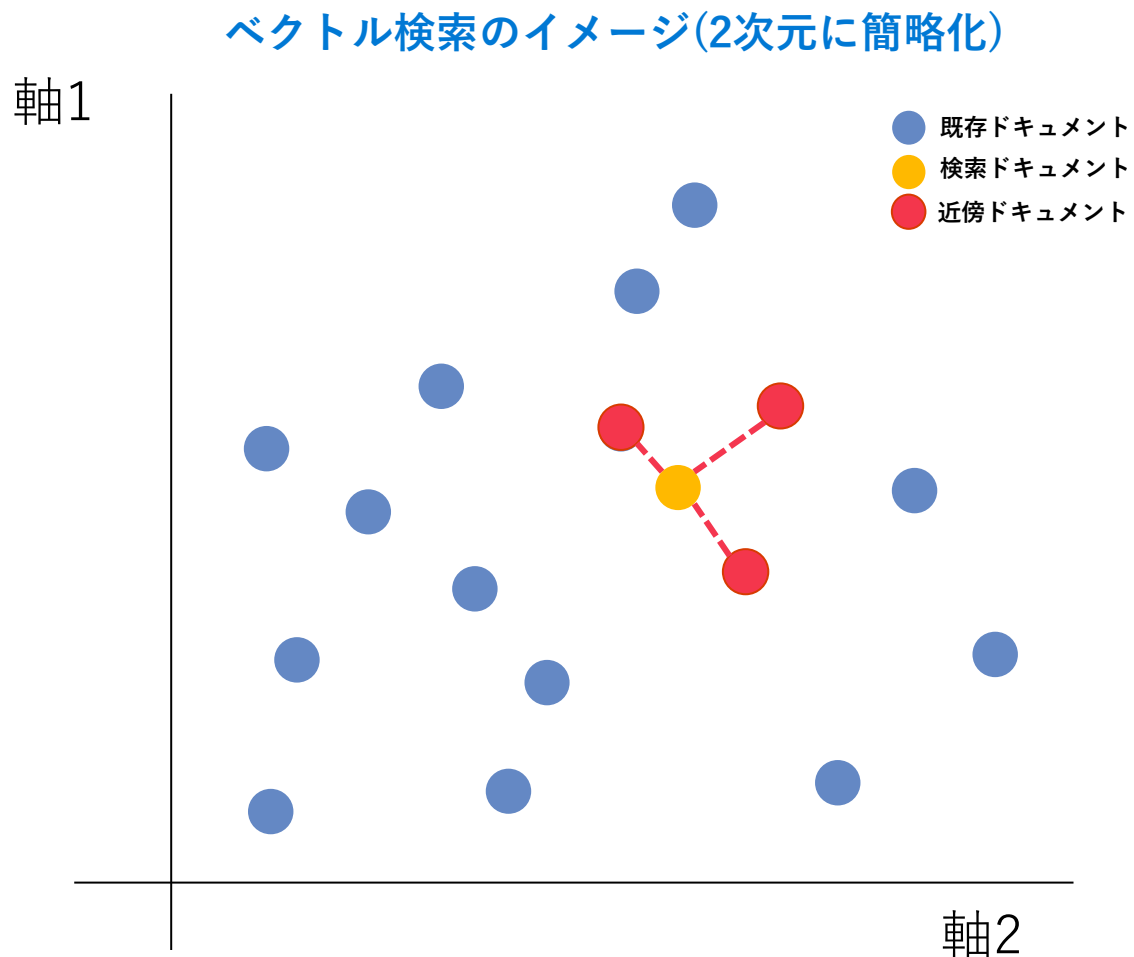
営業部 部長、課長とのコンタクトあり(2022/04契約より)

経理部 課長とコンタクト中

# 解決法：RAG(Retrieval Augmented Generation)



# RAGを支える技術：「近いものを探せ」（ベクトル近傍検索）



1

既存のドキュメントを「軸」で評価して、配置・蓄積する

2

検索したい内容を同じ「軸」で評価して置いてみる

3

検索したい内容に「近い」ものを探す

# RAGを支える技術：「近いものを探せ」（ベクトル近傍検索）

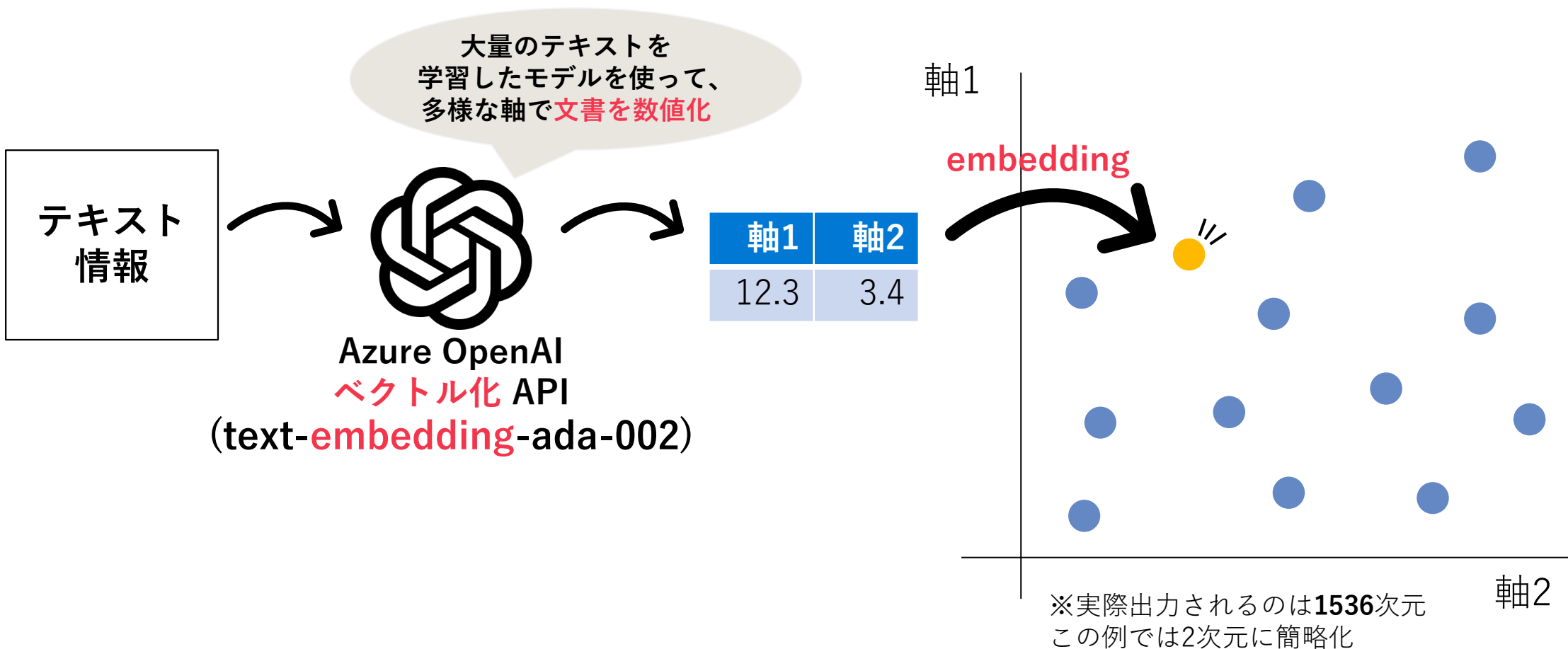
## ポイント1. ベクトル化

①

既存のドキュメントを「軸」で  
評価して、配置・蓄積する

②

検索したい内容を同じ「軸」で  
評価して置いてみる



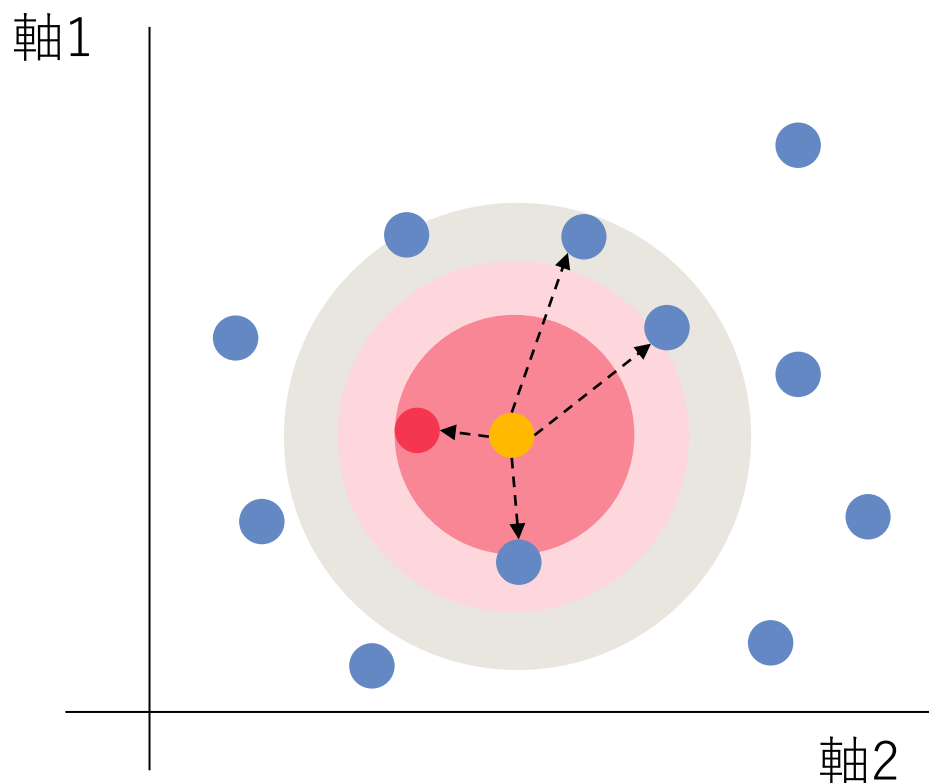
# RAGを支える技術：「近いものを探せ」（ベクトル近傍検索）

## ポイント2. ベクトル検索

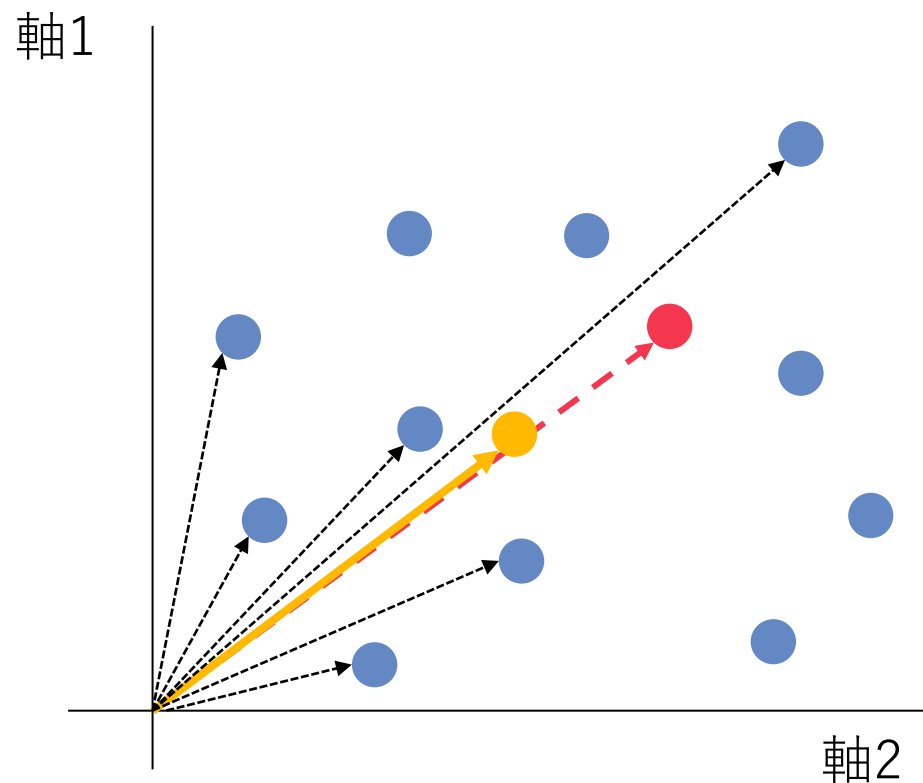
3

検索したい内容に  
「近い」ものを探す

A) 位置・距離が「近い」



B) 向きが「似てる」 = 「近い」





# RAGを支える技術：「近いものを探せ」（ベクトル近傍検索）

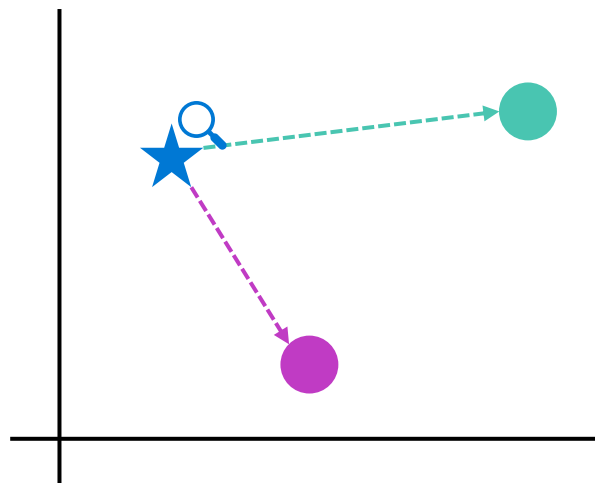
## ポイント2. ベクトル検索

3

検索したい内容に  
「近い」ものを探す

A) 位置・距離が「近い」

ユークリッド距離

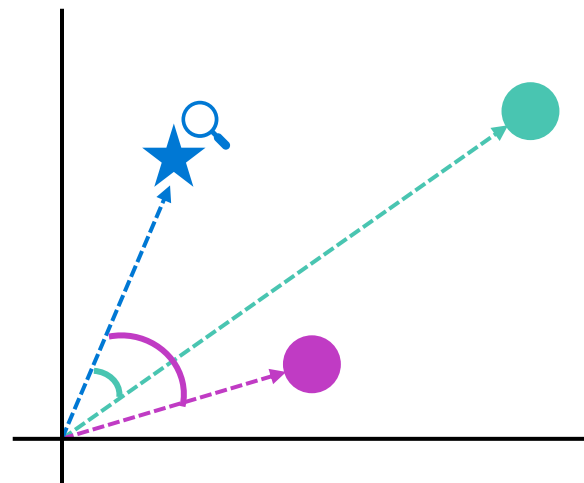


いわゆる  
直線距離

B) 向きが「似てる」 = 「近い」

コサイン類似度

推奨

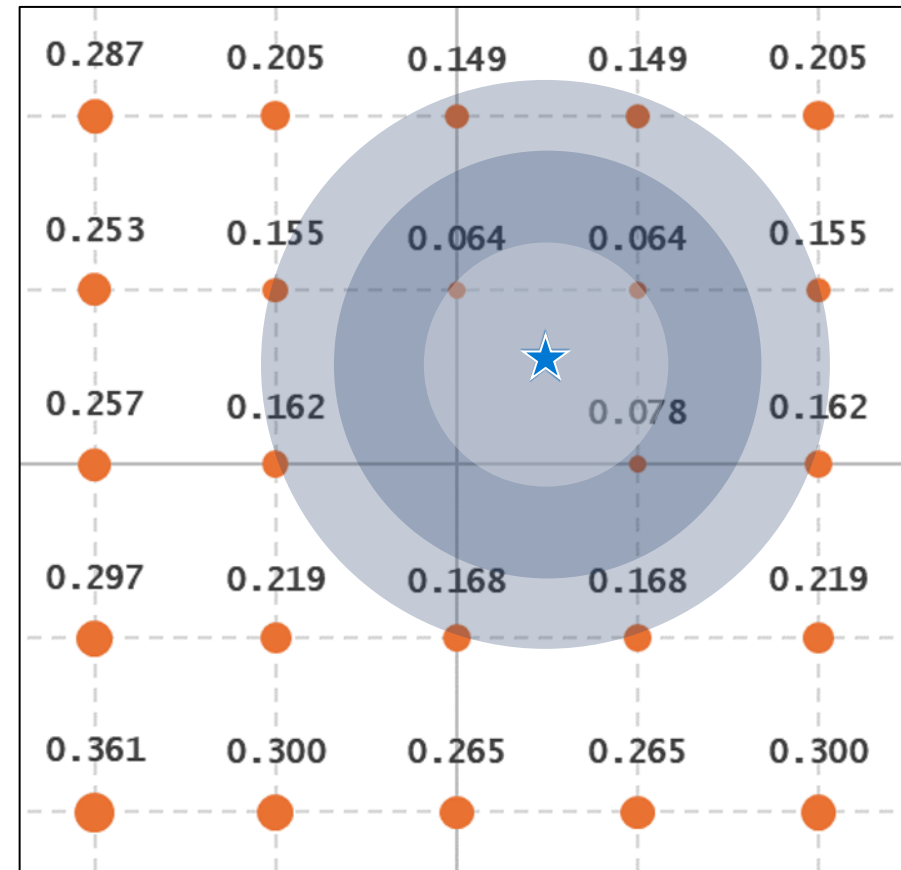
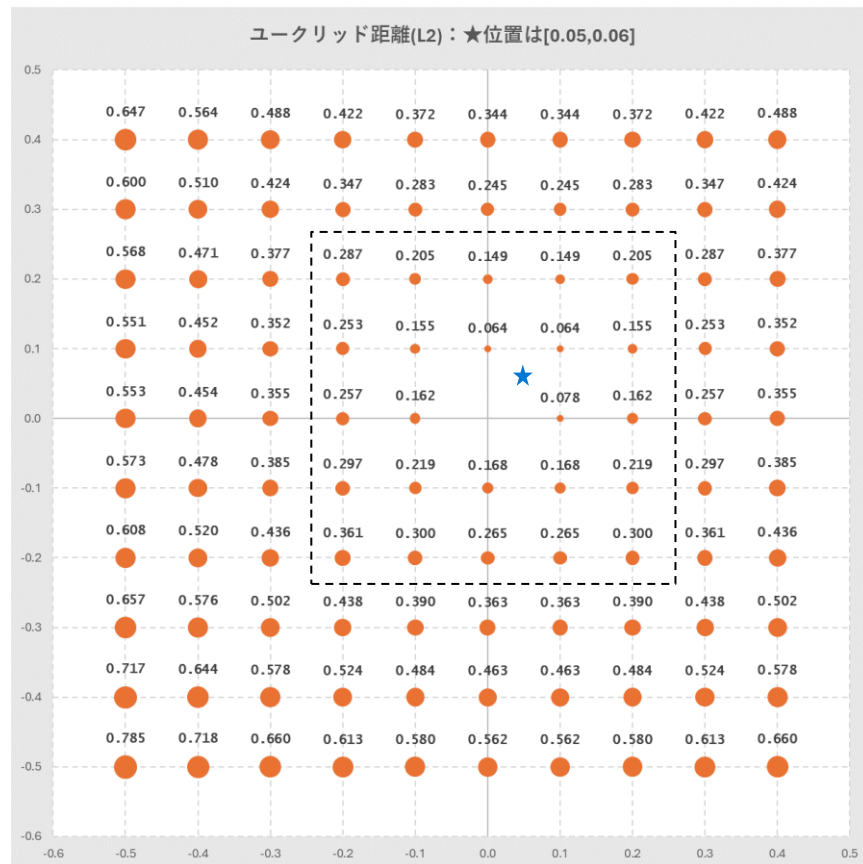


2つのベクトルの  
なす角度に基づいた  
類似度

-1(真逆)~1(同じ向き)  
の値を取る

# ユークリッド距離のイメージ

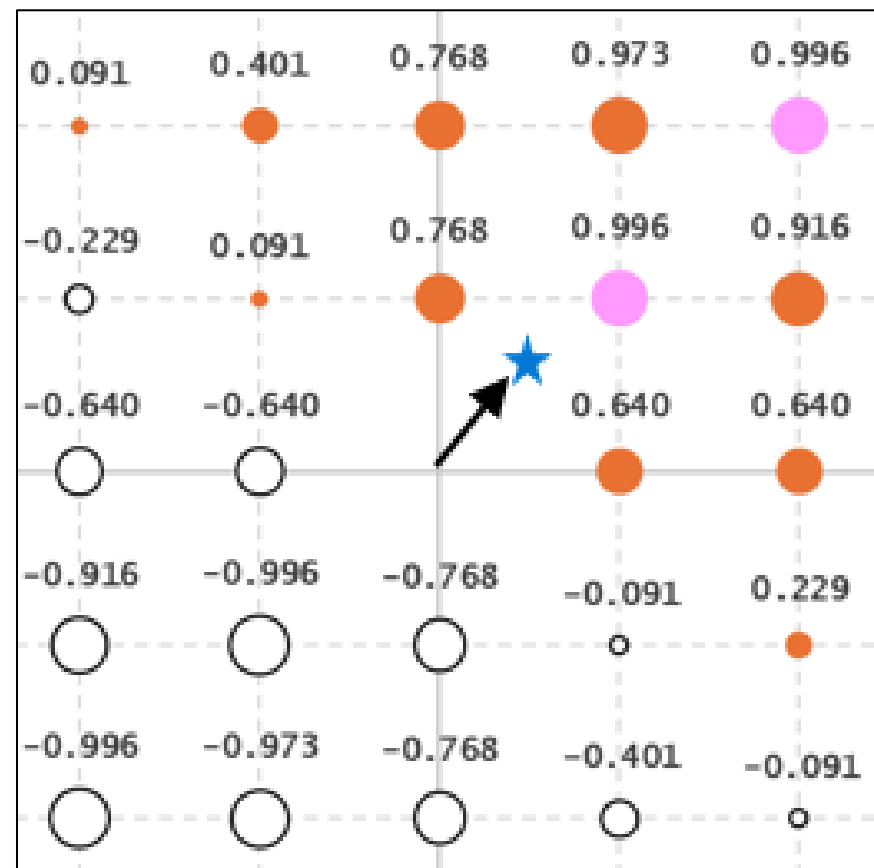
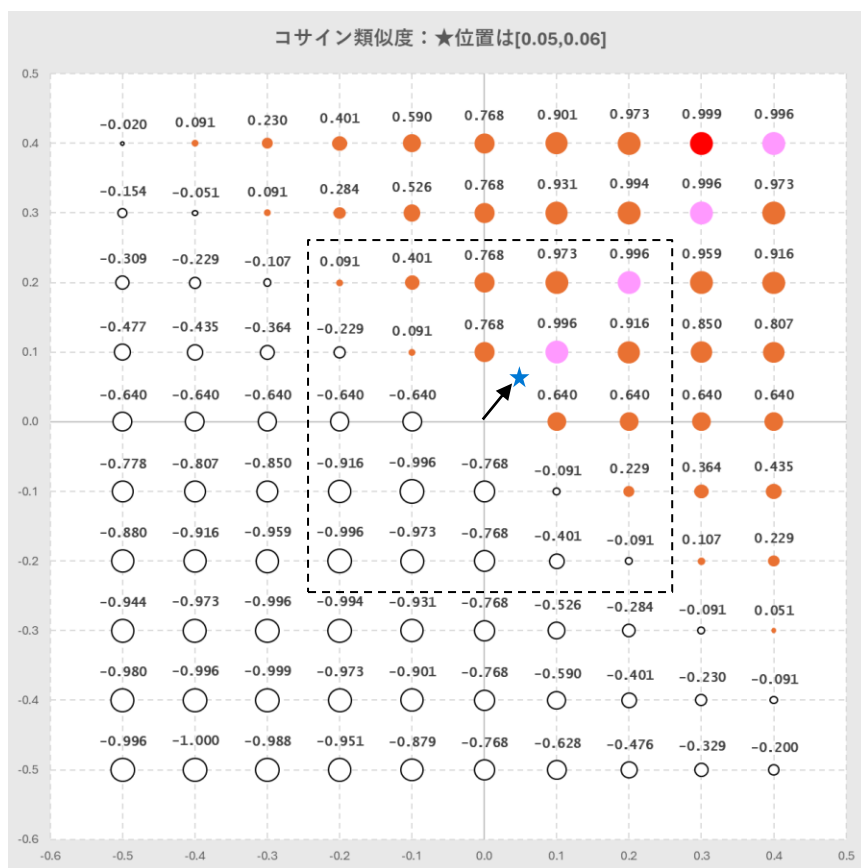
X,Y -0.5~0.5までにすべてに点がある空間にX0.05,Y0.06の★を置いて、各点のユークリッド距離を出力。**一般的な距離感覚に基づく解釈**ができる。



※類似度・距離計算はpgvectorを利用

# コサイン類似度のイメージ

X,Y -0.5~0.5までにすべてに点がある空間にX0.05,Y0.06の★を置いて、各点のコサイン類似度を出力。0,0から★に引かれた線↗の延長方向に類似度高が多い。



※類似度・距離計算はpgvectorを利用

# RAGを支える技術：「近いものを探せ」（ベクトル近傍検索）

## ポイント3. 大量データからの近似ベクトル検索手法

検索対象と既存のベクトルの  
**1vsN x 同時接続数の処理**  
が必要  
(+1536次元の計算は大変)

課題

データ量が  
多くなると  
計算量が増える

完全

**BruteForce**

**全探索!!**

最高精度  
検索速度最低

近似:精度を犠牲に高速化

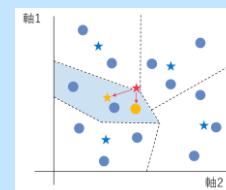
**IVFFlat** (反転リスト/インデックス検索)

データをクラスタに分けて、クラスタの重心を検索  
→そのクラスタのデータを全検索する

比較的高精度

低メモリ利用

検索速度はそこそこ



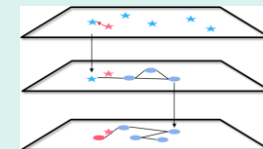
**HNSW**(階層化ナビ可能な小世界)

データをグラフ(つながりのあるデータ)に分割し、  
階層化してたどれるようにする

検索速度高速化

メモリ利用量増大

検索精度そこそこ低下

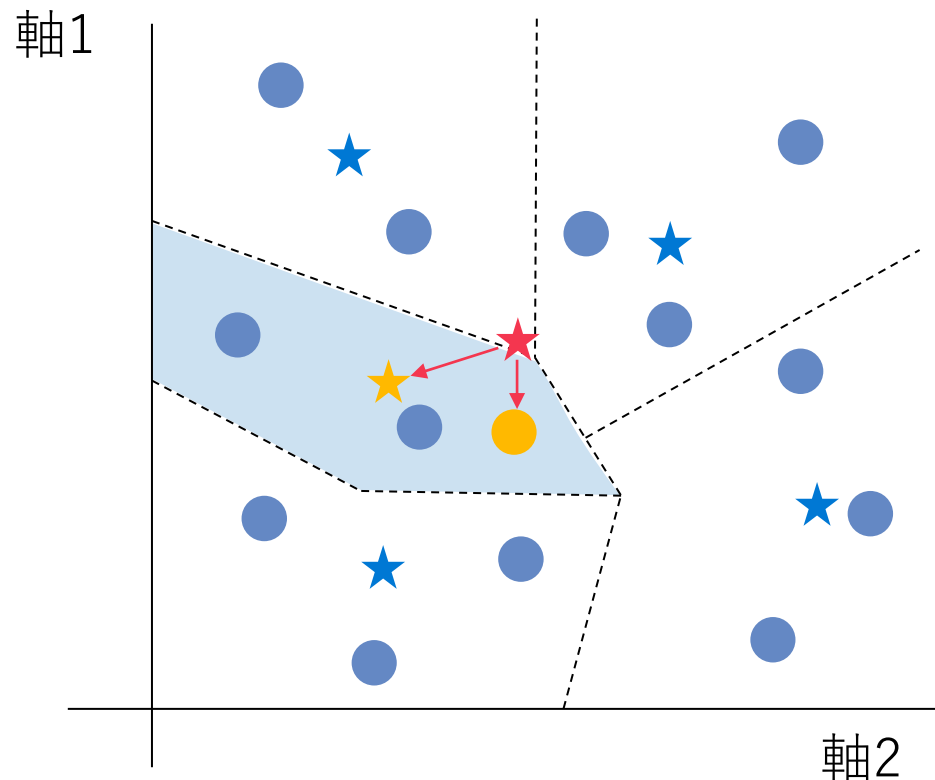


※その他ベンダー独自の近似手法も随時開発されている

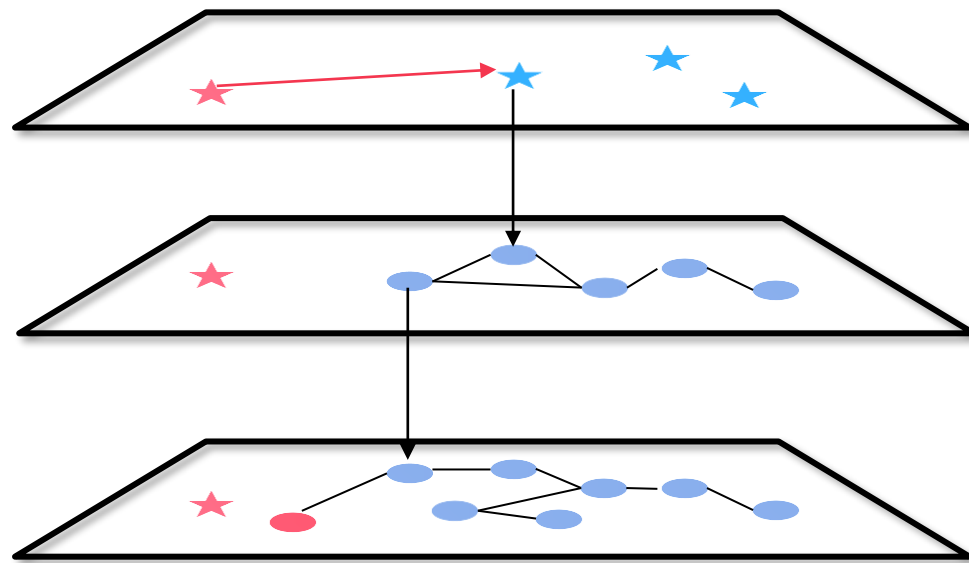
# RAGを支える技術：「近いものを探せ」（ベクトル近傍検索）

## ポイント3. 大量データのベクトル検索効率

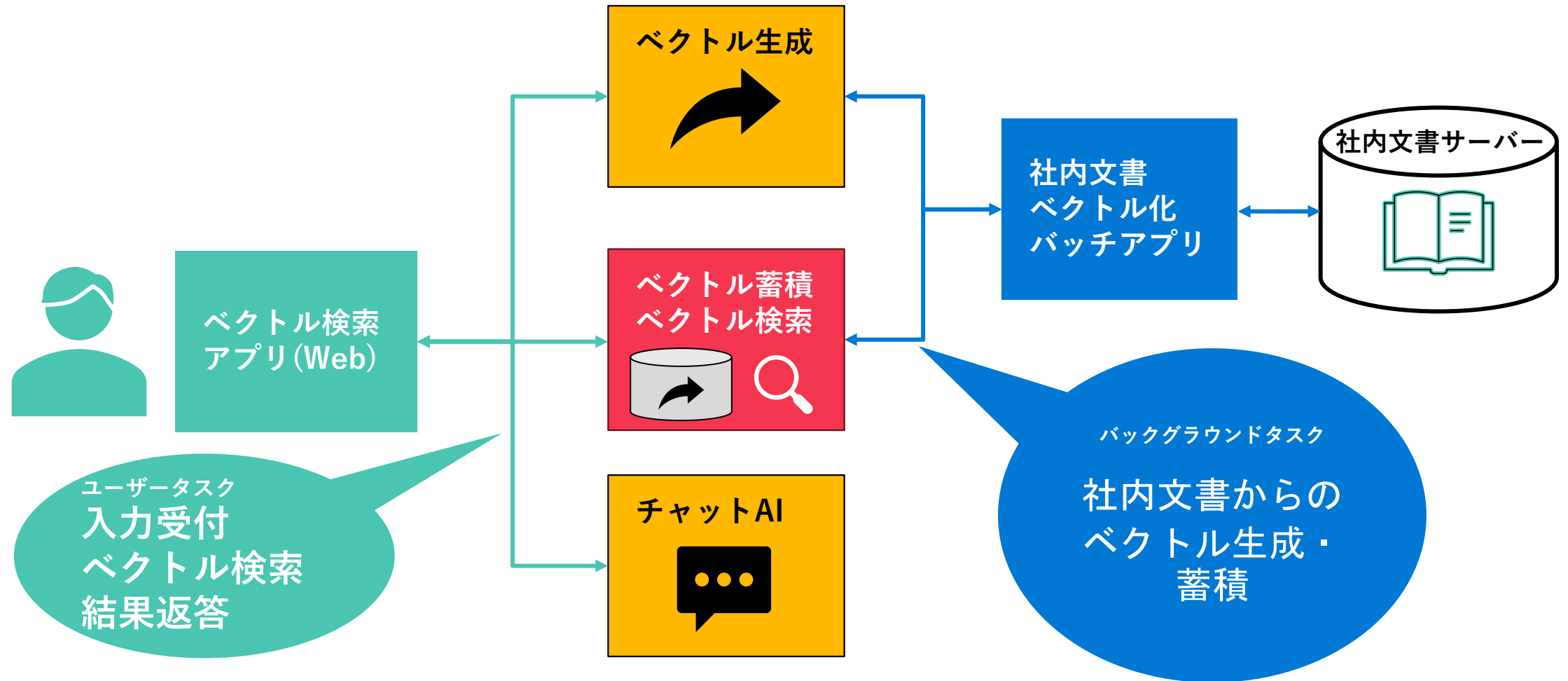
IVFFlat … 近い★を検索→そのクラスタを検索



HNSW … 近い★を検索→階層化グラフを検索



# RAGを利用したチャットシステム機能関係図



# Azureのベクトル検索機能

サービス	概要	ベクトル実装	近傍検索方法・備考
Cosmos DB For MongoDB API vCore 	MongoDB互換のNoSQL 環境。VM性能ベースの展開	aggregate関数の\$search 演算子”cosmosSearch”でのアクセス	IVFFlat/HSNWが利用可能  ※MongoDB vCoreのHSNW検索 はプレビュー段階
Cosmos DB For PostgreSQL (+Azure Database for PostgreSQL) 	PostgreSQL互換の分散 RDBMS環境。VM性能 ベースでの展開	pgvector拡張で実装。 vectorデータ型とベクトル 演算子の組み合わせ SQLでアクセス	
Azure Managed Instance for Apache Cassandra 	カラムファミリー系 NoSQL基盤である Apache Cassandraの マネージドサービス	追加機能として実装。 CQLによるアクセス	Microsoft独自の近似近傍検索 ロジック(DiskANN)を搭載
その他	Azure AI Searchや Azure Data Explorer等	REST等	HNSWだったり独自だったり

# ベクトル検索基盤の選定観点の例

## ベクトル 検索機能

- 格納できるベクトル数の上限はOpenAI-Embeddingをカバーできるか(1,536次元)
- BruteForce以外の効率的な検索方法があるか

## 容量・拡張性

- ベクトル格納・インデックスの蓄積に上限がないか、あっても十分足りているか
- 容量を追加・拡張できるか

## パフォーマンス チューニングの しやすさ

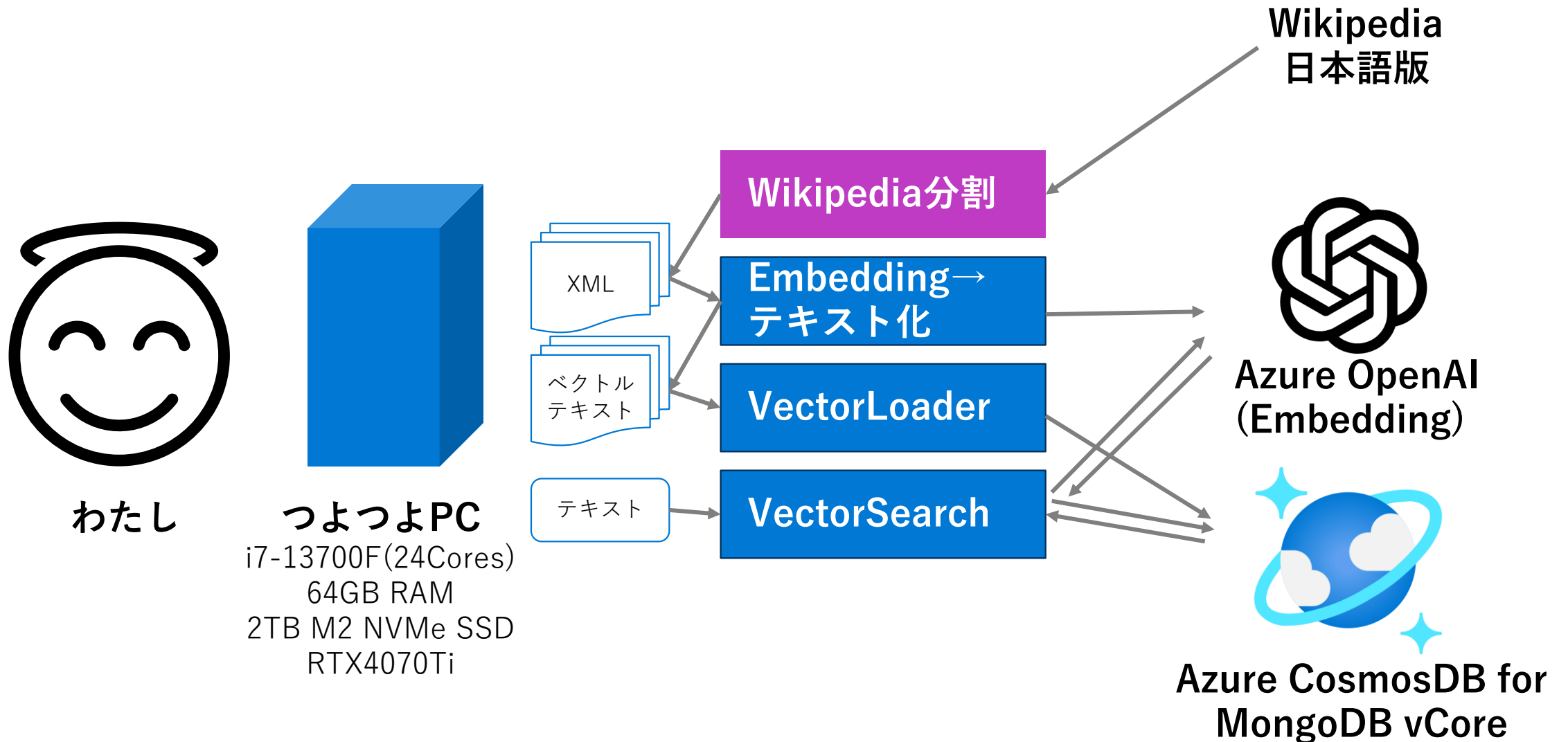
- 速度改善に効果のあるパラメーターを持つか
- 分散などで速度改善が可能なアーキテクチャか
- 全体を検索する必要がない場合に部分的に検索が可能か
  - パーティション・部分インデックスなど

## 開発生産性

- エンジニアのスキルに合った言語・環境であるか
  - 開発言語・オブジェクトモデル
  - APIアクセスの構成など



# ベクトル検索だけを試してみる



# ベクトル検索だけを試してみる

処理	言語	処理概要	備考(サイズ感等)
Wikipedia分割	PowerShell	◆ WikipediaからダウンロードしたXMLをページごとに切り出す	■ <page></page>で分割 280万ファイルぐらい
Embedding ↓ テキスト化	TypeScript (Node.js)	◆ ページごとにデータを分割(チャンク化)してEmbedding。元テキストとセットでjsonにして連番付きファイルとして保管 #様々な基盤でロードできるように	■ チャンク化基準:5,000Tokens ■ 出力380万ファイル(114GB) ■ 出力に約10日程度
VectorLoader		◆ ファイルを <b>10万件</b> 読み込んでCosmosDBへロード ◆ 非同期処理にて最大1,000並列ロードとなるように調整	■ <b>Cosmos DB for MongoDB vCore M30</b> (2 vCore / 8GB) ■ IFVFlatインデックス 100クラスタ ■ 10万件ロードに約10分
VectorSearch		◆ 入力されたテキストをEmbedding ◆ CosmosDBでベクトル検索 ◆ 検索されたドキュメントを表示するコンソールアプリ	■ 10万件データからの検索 1秒以内(800ms前後) ■ 検索文言と検索されたドキュメントは次のページで

# Azureのベクトル検索機能

```
Connected to MongoDB...
Enter text: アメリカ合衆国の大統領選挙に関する考察をしたいんだけど
Chunks: 1
Vectors: 1536
Searching Vectors from 100000 documents...
Search took 795 ms
Result: [
  {
    _id: new ObjectId('6563d749b8760778a587cd9c'),
    similarityScore: 0.8821439199151924,
    document: {
      _id: new ObjectId('6563d749b8760778a587cd9c'),
      name: '0003518_1.txt',
      content: '== 選出 ==\n' +
        '\n' +
        '{{See also アメリカ合衆国大統領選挙}}\n' +
        '\n' +
        '\n' +
        '\n' +
        '大統領はアメリカ合衆国憲法I憲法第2条第1節の規定により、4年に1度国民の投票によって新しく選出、
        または、再任されるため、任期は1期につき4年である。アメリカ合衆国憲法修正第22条I修正第22条の規定
        により、2度を超えて選出されることは認められていない（3選禁止）。すなわち、原則として同一人物が最
        長で務められるのは連続・返り咲きを問わず2期8年である。前大統領の辞任・死去などに伴い昇格した場合には例
        外がありうるが<ref name="n" group="注釈">任期途中で大統領に昇格した場合は、その任期が残り2年以内であれ
        ば、その後の大統領選挙に2度挑戦できる（修正第22条）。この場合、最高で10年間在任出来ることになる。第36
        代大統領のリンドン・ジョンソンにはこの資格があったがベトナム戦争の責任を取り3期目をかけた選挙に出馬し
        なかったため2期で退任した。</ref>、修正22条の下で<ref group="注釈">修正22条の制定前に、フランクリン・
        ルーズベルトが4回大統領に選出されている。</ref>通算3期以上を務めた大統領経験者はいない。
        \n' +
```

# ベクトル検索だけを試してみる

Cosmos DB for Mongo DB vCore でのベクトル検索呼び出しコード

```
// Submit vector search query to MongoDB
const result = await collection.aggregate([
  {
    $search: {
      "cosmosSearch": {
        "vector": vectors,
        "path": "embedding",
        "k": 2
      },
      "returnStoredSource": true
    },
    "$project": { "similarityScore": {
      "$meta": "searchScore" },
      "document": "$$ROOT"
    }
  }
]).toArray();
```

MongoDB API aggregate()…集計パイプライン

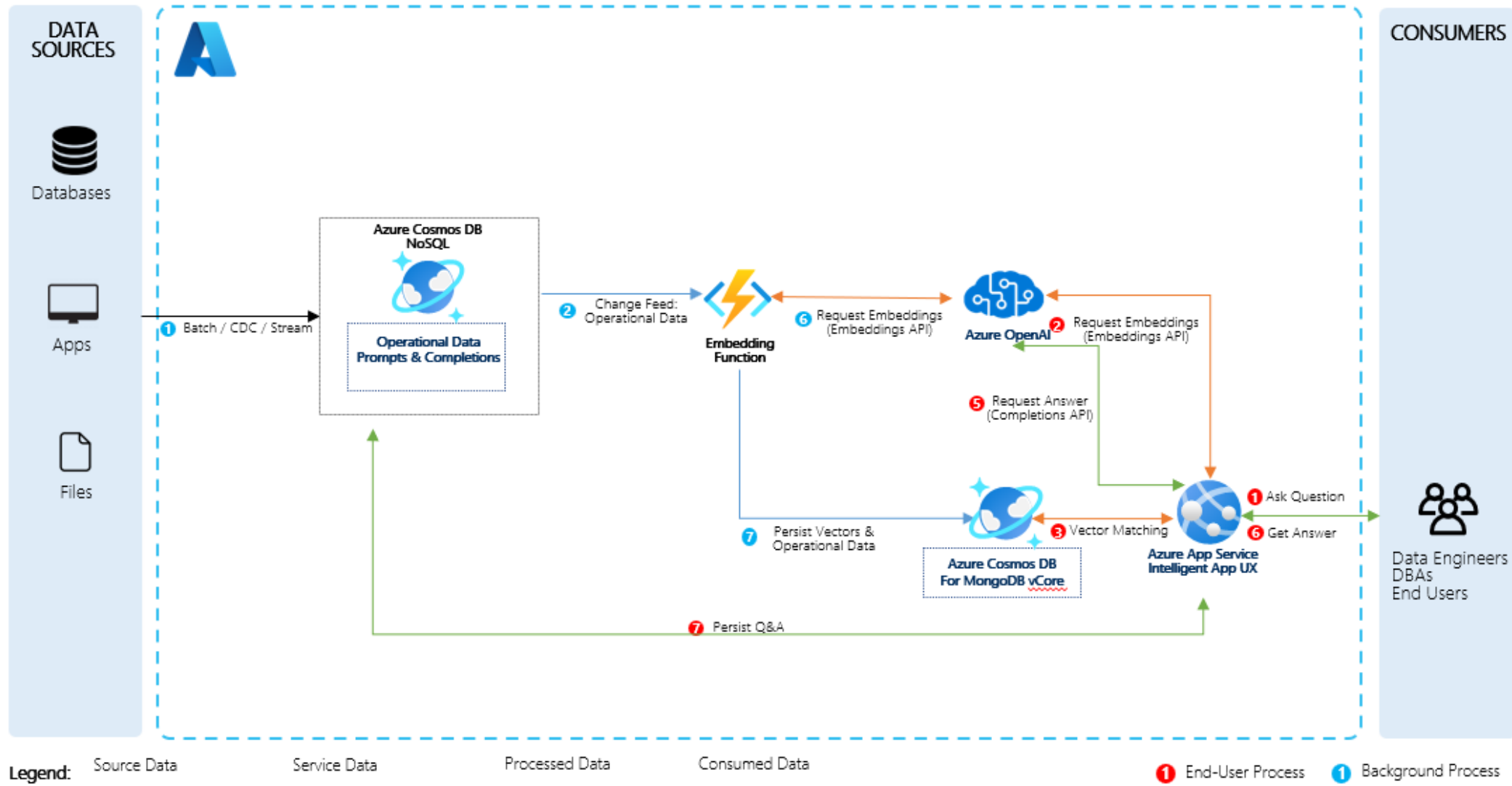
\$searchで“CosmosSearch”を呼び出す

vectorsに検索対象のベクトル(1,536次元)を入力

Wikipediaのベクトル(embedding)と比較

\$projectで類似度やデータを出力する

# Azure Cosmos DB - Vector Search & AI Assistant



[Azure/Vector-Search-AI-Assistant at MongovCore \(github.com\)](https://github.com/Azure/Vector-Search-AI-Assistant-at-MongovCore)

# 実用化にあたっての検討事項(案)

## ベクトル化

生データを  
テキスト化する  
だけでいいの？  
画像とかは？

ベクトルはいつ、  
どのモデルで  
生成した？

ベクトル化  
元データの  
クレンジング  
どうする？

単語の揺らぎ  
修正や名寄せを  
行うべき？

要約だけで別の  
ベクトル空間を  
持ってもよいの  
では？

## プラットフォーム

ベクトル検索って  
他の業務でも使え  
るから共通化すべ  
きでは？

IFVFlatやHNSW  
のパラメータは  
いくつが適当？

Topいくつまで  
検索するべき  
だろう？

## ベクトル検索

チャンク分割し  
たテキストに  
いくつ当たれば  
良い？

類似度の閾値は  
いくつが適当？

## アプリ

検索した結果を  
プロンプトに入  
れるだけだと単  
純すぎない？

社内情報なのか  
そうでないのか  
わかるべき？

回答の精度を  
ユーザーはどう  
考える？



# まとめ

- **社内の文書をテキスト化して、OpenAIのプロンプトに付与するとより業務に役立つ結果を返してくれるようになります。これをRAGアーキテクチャと呼びます**
- **RAGでプロンプトに付与する社内の文書を検索するためには、入力内容と社内の文書の類似度を測る必要があります。そのためにベクトル検索技術が活用できます**
- **ベクトル検索技術には、ベクトル化(Embedding)とベクトル検索の2つのステップがあります。前者はAzureOpenAIのEmbeddingで、後者は各種ベクトル検索ができるAzureサービスが対応します**
- **この仕組みを組み込んだAzure RAGアーキテクチャーのサンプルがあります**
- **実用化に向けた検討事項を解決して、Azure OpenAIを用いた社内データの活用を進めましょう！**