第3回：知識ブロックと外部ナレッジベース

この回で学ぶこと

- ナレッジベースの本質と仕組み
- RAG（Retrieval-Augmented Generation）の理解
- Difyでの知識ベース作成と管理
- 知識検索ノードの使い方
- 外部知識ソースの活用方法
- カスタムツールとMCPツールによる情報取得

> ***前提知識：*** *第2回までの内容（LLMノードとプロンプト設計）を理解していること*

目次

1. ナレッジベースとは

日常でこんな困りごとはありませんか？

ケース1: 新入社員の質問対応

新入社員「有給休暇の申請方法を教えてください」
あなた「えっと、確か社内Wikiのどこかに...（15分探す）」

ケース2: カスタマーサポート

お客様「この製品の保証期間は？」
担当者「少々お待ちください...（マニュアルをめくる）」

ケース3: 個人の知識管理

あなた「あのレシピ、どこにメモしたっけ？」
（ノート、メモアプリ、ブックマーク...あちこち探す）

これらの問題、ナレッジベースで解決できます

ナレッジベースを使うと実現できること：

企業での活用例

- 24時間対応のカスタマーサポートボット
- 社内ヘルプデスク
- 技術サポート

個人での活用例

- パーソナル料理アシスタント
- 学習アシスタント
- 旅行計画アシスタント

活用例①：24時間カスタマーサポートボット

```
お客様「製品の初期設定方法は？」
  ↓
AIボット「製品マニュアルから検索...」
  ↓
「以下の手順で設定できます：
 1. 電源ケーブルを接続
 2. 初期設定画面で言語を選択
 3. Wi-Fi設定を完了
 （参照：製品マニュアル p.12）」
```

効果：

- 深夜や休日でも即座に回答
- 人間のオペレーターは複雑な問い合わせに集中
- 回答の品質が一定（ベテランも新人も同じ回答）

活用例②：社内ヘルプデスク

```
社員「経費精算のルールを教えて」
  ↓
社内AIアシスタント「経費規定から検索...」
  ↓
「領収書は必ず原本を提出してください。
 申請期限は月末から1週間以内です。
 （参照：経費精算規定 第3条）」
```

効果：

- 総務部への問い合わせが減る
- 社員は待たずに回答を得られる
- 最新の規定が常に反映される

活用例③：技術サポート

```
エンジニア「このエラーの解決方法は？」
  ↓
技術AIアシスタント「過去の障害事例から検索...」
  ↓
「似たエラーが2023年11月に発生しています。
 原因：データベース接続タイムアウト
 解決：接続プールのサイズを50→100に変更
 （参照：障害報告書 #2023-1145）」
```

効果：

- 過去の知見を即座に活用
- 同じ問題の再発を防ぐ
- 新人エンジニアでもベテランの知識にアクセス

なぜナレッジベースが必要なのか？

LLMだけでは以下の問題があります：

問題1: 知識の鮮度

- 学習データが古い → 最新情報に対応できない

問題2: 専門知識の不足

- 一般的な知識のみ → 自社独自の情報は持っていない

問題3: ハルシネーション（幻覚）

- もっともらしい嘘を答える → 信頼性に欠ける

問題1: 知識の鮮度

LLMの限界：

```
あなた「2024年の最新の税制改正について教えて」
ChatGPT「私の学習データは2023年4月までです。
         2024年の情報は持っていません」
```

ナレッジベースなら：

```
あなた「2024年の最新の税制改正について教えて」
AIアシスタント「税理士事務所のサイトから最新情報を取得...」
  ↓
「2024年度は住宅ローン控除の要件が変更され...
 （参照：国税庁の最新PDF）」
```

問題2: 専門知識の不足

LLMの限界：

```
新入社員「我が社の勤怠システムの使い方は？」
ChatGPT「一般的な勤怠システムの説明はできますが、
         御社独自のシステムについては分かりません」
```

ナレッジベースなら：

```
新入社員「我が社の勤怠システムの使い方は？」
社内AIアシスタント「社内マニュアルから検索...」
  ↓
「当社の勤怠システム『SmartTime』の使い方：
 1. 社員IDでログイン
 2. 出勤ボタンをタップ
 （参照：新人研修マニュアル p.25）」
```

問題3: ハルシネーション（幻覚）

LLMの限界：

```
あなた「○○製品の型番ABC-123の仕様は？」
ChatGPT「ABC-123は以下の仕様です：
        - サイズ：30cm × 20cm
        - 重量：500g ...」
（実際には存在しない製品だが、もっともらしく答える）
```

ナレッジベースなら：

```
あなた「○○製品の型番ABC-123の仕様は？」
AIアシスタント「製品データベースから検索...」
  ↓
「申し訳ございません。型番ABC-123は
 データベースに見つかりませんでした。
 類似する型番：ABC-124, ABC-122」
```

ナレッジベースの仕組み

ナレッジベース（知識ベース） とは、LLMが推論・生成時に参照するための、**整理・保存・検索可能な外部知識の集合**です。

```
[外部知識] → [整理・保存] → [検索可能な状態] → [LLMが参照] → [正確な回答]
```

重要：RAGは手段であって本質ではない

ナレッジベースの本質：

- LLMが利用できる外部知識の集合
- 信頼できる情報源
- 検索・取得可能な構造化された知識

RAGの役割：

- ナレッジベースから知識を取得する**手段の一つ**

外部知識を取得する様々な方法

方法	説明	使いどころ
RAG	ベクトル検索で類似文書を取得	マニュアル、FAQ
カスタムツール	独自APIやDBから取得	在庫、顧客データ
MCPツール	Model Context Protocol連携	Notion、Google Drive
Webスクレイピング	Webサイトから取得	ニュース、価格情報
DBクエリ	SQLで直接アクセス	商品情報、取引履歴

RAGだけがナレッジベースではない！

ナレッジベースを使うメリット

正確性の向上

- 信頼できる情報源から回答を生成
- 根拠のある回答（ソース付き）
- ハルシネーション（幻覚）の大幅な削減

最新情報の活用

- リアルタイムで更新される情報にアクセス
- ドキュメントを更新すれば即座に反映

専門性の実現

- 企業独自の知識を活用
- 自社のルールや文化に沿った対応

業務効率化

- 問い合わせ対応時間の削減
- 24時間365日対応可能

まとめ：ナレッジベースの本質

ナレッジベースは、**「AIに正しい知識を使って答えてもらうための仕組み」**

```
【従来】
AI単体 → 一般的な知識のみ → 不正確な回答の可能性

【ナレッジベース活用】
AI + 専門知識 → 根拠のある正確な回答 → 信頼性の高いサービス
```

次のセクションでは「RAG」について詳しく学びます

2. RAGの仕組み

RAGとは

RAG（Retrieval-Augmented Generation：検索拡張生成） は、外部知識検索と言語生成を組み合わせた技術です。

RAGの3つのステップ

```
ステップ1: インデキシング（事前準備）
[文書] → [分割] → [埋め込み変換] → [ベクトルDB保存]

ステップ2: 検索（Retrieval）
[質問] → [埋め込み変換] → [類似検索] → [関連文書取得]

ステップ3: 生成（Generation）
[関連文書] + [質問] → [LLM] → [根拠のある回答]
```

ステップ1: インデキシング（事前準備）

1-1. チャンク化（文書の分割）

長い文書を小さな単位（チャンク）に分割します。

```
【元の文書】（1000文字の製品マニュアル）
↓ チャンク化
【チャンク1】製品の概要と特徴について...（200文字）
【チャンク2】セットアップ手順について...（200文字）
【チャンク3】トラブルシューティング...（200文字）
```

なぜ分割するのか：

- LLMには入力できる文字数に制限がある
- 関連する部分だけを効率的に取得するため

チャンクサイズの設定

サイズ	目安	使いどころ
小さい（100-200トークン）	1-2段落	短い質問、キーワード検索
中程度（300-500トークン）	数段落	一般的なドキュメント
大きい（800-1000トークン）	複数段落	詳細な説明が必要な場合

ポイント

- 小さすぎると文脈が失われる
- 大きすぎると検索精度が下がる
- 用途に応じて調整が必要

1-2. 埋め込み変換（Embedding）

テキストを数値ベクトルに変換します。

```
テキスト「猫は可愛い動物です」
↓ 埋め込みモデル
ベクトル [0.2, -0.5, 0.8, 0.1, ...]（1536次元など）
```

埋め込みの性質：

- 意味が似た文章は、ベクトル空間で**近い位置**になる
- 「犬は可愛い」と「猫は可愛い」は近い
- 「猫は可愛い」と「数学の公式」は遠い

これにより**意味的な検索**が可能になる

主要な埋め込みモデル

モデル	提供元	次元数	特徴
text-embedding-ada-002	OpenAI	1536	高性能、英語に強い
text-embedding-3-small	OpenAI	1536	コスト効率が良い
text-embedding-3-large	OpenAI	3072	最高性能
embedding-001	Google	768	多言語対応

日本語向けの選択肢：

- multilingual-e5-large-instruct
- intfloat/multilingual-e5-large

1-3. ベクトルデータベース保存

生成されたベクトルを専用のデータベースに保存します。

ベクトルデータベースの役割：

- 高速な類似検索
- 大量のベクトルを効率的に管理

主要なベクトルDB：

- **Pinecone** - クラウド型、簡単に使える
- **Weaviate** - オープンソース、多機能
- **Qdrant** - 高速、Rust製
- **Dify内蔵DB** - Difyに組み込み済み

Difyを使う場合は内蔵DBで十分

ステップ2: 検索（Retrieval）

ユーザーの質問から関連する文書を取得します。

```
ユーザーの質問「製品のセットアップ方法は?」
↓ 同じ埋め込みモデル
ベクトル [0.3, -0.4, 0.7, 0.2, ...]
↓ ベクトルDBで類似検索
↓ コサイン類似度計算
スコア: 0.95（チャンク2：セットアップ手順） ← 最も関連
スコア: 0.72（チャンク1：製品概要）
スコア: 0.45（チャンク3：トラブルシューティング）
```

検索パラメータ：

- **Top K**：取得する文書の数（例：上位3件）
- **スコア閾値**：関連度の最低基準（例：0.7以上）

ステップ3: 生成（Generation）

取得した文書を使ってLLMが回答を生成します。

```
【システムプロンプト】
あなたは親切なアシスタントです。
以下の情報を参考に、ユーザーの質問に答えてください。

【参考情報】（検索で取得した文書）
チャンク2（スコア: 0.95）
セットアップ手順：
1. 電源ケーブルを接続
2. 初期設定画面で言語を選択
3. Wi-Fi設定を完了

【ユーザーの質問】
製品のセットアップ方法は?
```

RAGのメリットとデメリット

メリット

- **正確性の向上**：実際の文書に基づいた回答
- **ソースの提示**：どの文書から情報を得たか明示
- **知識の更新が容易**：文書を追加・更新すれば即座に反映

デメリット

- **初期設定のコスト**：文書の準備とインデキシング
- **検索精度の課題**：関連文書が見つからない場合がある
- **処理時間**：検索 + 生成で通常のLLMより遅い

3. Difyでの知識ベース作成

ステップ1: 知識ベースの作成

1. Difyにログイン
2. 左メニューから「ナレッジ」をクリック
3. 「ナレッジベースを作成」ボタンをクリック
4. 以下を設定：
    - **名前**：わかりやすい名前（例：「製品マニュアル」）
    - **説明**：このナレッジベースの用途
    - **権限**：「自分のみ」または「チーム全体」

ステップ2: ドキュメントのアップロード

対応ファイル形式

カテゴリ	形式
文書	TXT, MD, PDF, DOCX
表計算	CSV, XLSX
コード	HTML, JSON, XML

アップロード方法

1. **ファイルアップロード** - ドラッグ&ドロップで簡単
2. **テキスト入力** - 直接入力
3. **Webサイトから取得** - URLを指定
4. **Notion連携** - Notionページを同期

ステップ3: チャンク化設定

自動モード（推奨）

- **チャンクサイズ**：500トークン（デフォルト）
- **チャンク重複**：50トークン（前後のチャンクとの重複）

重複（Overlap）とは

```
【チャンク1】
...文脈が途切れないように、次の章では...

【チャンク2】（50トークン重複）
...次の章では詳しく説明します...
```

→ 文脈が繋がり、検索精度が向上

推奨チャンク設定

用途	チャンクサイズ	重複	区切り
一般的なドキュメント	500トークン	50	段落区切り
FAQ形式	300トークン	30	Q&A単位
技術マニュアル	800トークン	100	見出し区切り

ポイント

- 重複を設けることで文脈の断絶を防ぐ
- カスタムモードで細かく調整可能

ステップ4-5: モデル選択とインデキシング

埋め込みモデルの選択

用途	推奨モデル
英語中心	text-embedding-3-small
日本語中心	multilingual-e5-large-instruct
高精度重視	text-embedding-3-large
コスト重視	text-embedding-ada-002

インデキシング開始

1. 設定を確認
2. 「保存してインデキシング」をクリック
3. ステータス：「処理中」→「完了」を待つ

4. 知識検索ノードの使い方

知識検索ノードとは

知識検索ノード は、ワークフロー内でナレッジベースを検索するためのノードです。

```
[開始ノード] → [知識検索ノード] → [LLMノード] → [終了ノード]
      ↓                ↓                ↓
  ユーザー質問     関連文書を検索     文書を参考に回答生成
```

ノードの設定

1. ナレッジベースの選択

- 作成済みのナレッジベースを選択

2. クエリ変数の設定

```
クエリ変数：{{#start.question#}}
```

→ ユーザーの質問をそのまま検索

3. 検索パラメータ

項目	推奨値
Top K	3-5件
スコア閾値	0.7
リランキング	有効

検索パラメータの詳細

Top K（取得件数）

設定値	使いどころ
1-2件	簡潔な回答、FAQ
3-5件	一般的なドキュメント検索
10件以上	複雑な質問、詳細な回答

スコア閾値（Score Threshold）

設定値	説明
0.7以上	高い関連性のみ（厳密）
0.5-0.7	標準的な関連性
0.5未満	緩い基準（広範囲に取得）

LLMノードでの使用

```
以下の情報を参考に、ユーザーの質問に答えてください。

【参考情報】
{{#知識検索ノード.result#}}

【質問】
{{#start.question#}}

【回答】
```

出力変数：

- **result** - 検索結果のテキスト（結合済み）
- **records** - 検索結果の配列（個別のチャンク）

5. 実践：RAGアプリケーション

演習1: 製品FAQボット

目標： 製品マニュアルを使った質問応答ボット

ワークフロー構成

```
[開始ノード]
  ↓ 入力変数：question（ユーザーの質問）
[知識検索ノード]
  ↓ ナレッジベース：製品FAQ
  ↓ Top K: 3、スコア閾値: 0.7
[LLMノード]
  ↓ FAQ情報を参考に回答生成
[終了ノード]
```

FAQボット：LLMノードのプロンプト

```
あなたは製品サポート担当者です。
FAQを参考に、ユーザーの質問に丁寧に答えてください。

【FAQ情報】
{{#知識検索ノード.result#}}

【お客様からの質問】
{{#start.question#}}

【回答のルール】
- FAQに記載されている情報を正確に伝える
- 丁寧で親切な言葉遣いを使う
- FAQに情報がない場合は
  「申し訳ございません、その情報はFAQに記載がございません」
  と伝える
```

演習2: 社内文書検索システム

目標： 複数の社内文書から情報を検索

```
[開始ノード]
  ↓
[LLMノード1：質問の分類]（構造化出力）
  ↓
[条件分岐ノード]
  ├─ カテゴリ「就業規則」 → [知識検索：就業規則DB]
  ├─ カテゴリ「休暇」 → [知識検索：休暇DB]
  └─ カテゴリ「経費」 → [知識検索：経費DB]
  ↓
[LLMノード2：回答生成]
  ↓
[終了ノード]
```

質問分類のJSON Schema

```
{
  "type": "object",
  "properties": {
    "category": {
      "type": "string",
      "description": "質問のカテゴリ",
      "enum": ["就業規則", "休暇", "経費", "その他"]
    },
    "optimized_query": {
      "type": "string",
      "description": "検索に最適化された質問文"
    }
  },
  "required": ["category", "optimized_query"]
}
```

演習3: 引用元を表示するRAGシステム

目標： 回答に引用元（ソース）を明示

```
以下の文書を参考に、ユーザーの質問に答えてください。
回答の最後に、参考にした文書のタイトルとスコアを記載。

【参考文書】
{{#知識検索ノード.result#}}

【質問】
{{#start.question#}}

【回答形式】
（回答内容）
---
参考文書：
- 文書名1（関連度: XX%）
- 文書名2（関連度: XX%）
```

6. 外部ナレッジベースの活用

外部ナレッジベースとは

外部ナレッジベース は、Difyの外部で管理されている独自のナレッジベースと連携する機能です。

使用するケース

ケース	説明
既存システムとの統合	すでに運用中の検索システムを活用
独自のベクトルDB	Pinecone、Weaviateなど専用DBを使用
リアルタイム更新	常に最新データを取得したい場合
高度なフィルタリング	メタデータによる複雑な検索条件

外部ナレッジベースAPI仕様

エンドポイント： `POST <your-endpoint>/retrieval`

リクエスト：

```
{
  "knowledge_id": "your-knowledge-id",
  "query": "ユーザーの質問",
  "retrieval_setting": {
    "top_k": 3,
    "score_threshold": 0.7
  }
}
```

Difyでの設定：

1. 「外部ナレッジベース」を選択
2. エンドポイントURL、APIキー、Knowledge IDを設定

7. カスタムツールとMCPツール

なぜツールが必要なのか？

RAGでは対応できないケース：

```
お客様「商品Aの在庫はありますか？」
  ↓
RAG検索「商品Aの情報を検索...」
  ↓
「商品Aは高品質な素材で作られた人気商品です。」
  ↓
（在庫数という「今この瞬間のデータ」は取得できない！）
```

カスタムツールなら：

```
お客様「商品Aの在庫はありますか？」
  ↓
カスタムツール「在庫DBにリアルタイム照会...」
  ↓
「商品Aの在庫は現在15個です。」
```

ナレッジベースとツールの違い

項目	ナレッジベース（RAG）	カスタムツール
用途	静的な文書検索	動的な情報取得・処理
データソース	事前登録された文書	リアルタイムAPI、DB
更新頻度	手動更新	常に最新
得意なこと	マニュアル、FAQ	在庫照会、計算処理
セットアップ	文書をアップロード	APIエンドポイント設定

カスタムツールでできること

カテゴリ	具体例
データ取得	在庫照会、顧客情報検索、注文状況確認
外部サービス連携	天気情報、為替レート、ニュース取得
処理・計算	見積計算、税額計算、配送料計算
アクション実行	メール送信、チケット作成、データ登録

カスタムツールの仕組み

```
[Difyワークフロー] → [外部API] → [レスポンス] → [LLMで回答生成]
```

Difyでのカスタムツール作成

1. 「ツール」→「カスタムツールを作成」
2. 基本情報の設定（ツール名、説明、アイコン）
3. **OpenAPI（Swagger）形式**でAPIスキーマを定義
4. 認証設定（必要な場合）
5. テスト実行

認証タイプ

タイプ	説明
なし	認証不要のAPI
APIキー	ヘッダーまたはクエリにキーを追加
Basic認証	ユーザー名とパスワード
Bearer Token	JWTなどのトークン認証

MCPツールとは

MCP（Model Context Protocol）ツールは、外部サービスとの統合を簡単にするプロトコル

項目	カスタムツール	MCPツール
設定方法	OpenAPI形式で定義	MCPサーバーに接続
対象	任意のREST API	MCP対応サービス
セットアップ	API仕様を手動で記述	接続設定のみ

対応サービス例

- **Notion** - ページの検索、作成、更新
- **Google Drive** - ファイル検索、取得
- **Slack** - メッセージ送信、検索
- **GitHub** - リポジトリ、Issue検索

演習: RAG + カスタムツール統合ボット

目標： 問い合わせ内容に応じて最適なデータソースを選択

```
[開始ノード：顧客からの問い合わせ]
  ↓
[LLMノード：問い合わせ分類]
  ↓
[条件分岐]
  ├─ 「製品の使い方」→ [知識検索：製品マニュアル]（RAG）
  ├─ 「注文状況」→ [カスタムツール：注文DB照会]
  ├─ 「在庫確認」→ [カスタムツール：在庫API]
  └─ 「社内情報」→ [MCPツール：Notion検索]
  ↓
[LLMノード：回答生成] → [終了ノード]
```

よくある質問（FAQ）

Q1: チャンクサイズはどう決める？

サイズ	用途
小さい（100-300トークン）	FAQ形式、短い段落中心
中程度（300-600トークン）	一般的なドキュメント
大きい（800-1000トークン）	技術文書、文脈の連続性が重要

Q2: Top Kとスコア閾値は？

ケース	Top K	スコア閾値
精度重視	2-3	0.75-0.8
バランス	3-5	0.6-0.7
網羅性重視	5-10	0.5-0.6

Q3: 検索結果が期待と違う場合は？

1. **チャンクサイズの調整**
    - 小さくする：より細かく検索
    - 大きくする：より広い文脈を取得
2. **質問の最適化**
    - LLMノードで質問を言い換える
    - キーワードを明確にする
3. **埋め込みモデルの変更**
    - 日本語に強いモデルに変更
4. **スコア閾値を下げる**
    - より多くの候補を取得

Q4: RAGとカスタムツールの使い分け

RAGを使う場合：

- 文書が静的（頻繁に変わらない）
- テキスト検索が中心
- 事前に文書を準備できる

カスタムツールを使う場合：

- リアルタイムデータが必要
- データベースから動的に取得
- 計算や処理が必要

多くの場合、両方を組み合わせるのがベスト

まとめ

この回で学んだこと

ナレッジベースの本質

- LLMが参照する外部知識の集合
- RAGは情報取得手段の一つ

RAGの仕組み

- インデキシング → 検索 → 生成

Difyでの実装

- ナレッジベースの作成と管理
- 知識検索ノードの使い方

ツールの活用

- カスタムツールによる動的データ取得
- MCPツールによる外部サービス統合

次のステップ

第4回：エージェントとツール連携

- AIエージェントの設計と実装
- 複数のツールを組み合わせた複雑なワークフロー
- 自律的なタスク実行

練習課題

1. レシピ検索ボット（RAG基礎）
2. 技術文書Q&Aシステム（引用元表示）
3. 天気予報アシスタント（カスタムツール）
4. 顧客サポート統合ボット（RAG + ツール）

お疲れ様でした！

参考リンク

- Dify公式ドキュメント - ナレッジベース
- RAG（検索拡張生成）とは
- 外部ナレッジベースAPI仕様

次回はAIエージェントについて学びます