

タイトル

サブタイトル

著者名 著

YYYY 年 MM 月 DD 日

タイトル

hogehoge

執筆：さめ (m e r - c c k)

(x@SaMeGiraffe_VRC,

YouTube@SaMe-Giraffe)

目次

タイトル	2
目次	3
まえがき：さめ	4
1 AI VTuber の個性を RAG で構築する - ソーシャルメディア投稿のナ レッジ DB 化	5
1.1 はじめに	5
1.1.1 本記事の目的	6
1.1.2 プロジェクトの背景	6
1.2 AI VTuber 配信の概要 - 何ができるのか	7
1.2.1 実際の配信画面	7
1.2.2 RAG で検索された知識を用いた応答の実例	8
1.3 システム全体像 - どう動いているか	11
1.3.1 キャラクター設定の課題と解決策	11
1.3.2 システムアーキテクチャ	12
1.3.3 技術スタックと既存インフラの活用	12
1.4 RAG 検索の工夫と課題解決	15
1.4.1 ベクトル検索の限界 - 「犬・トキ・ポケモン問題」	15
1.4.2 解決策: 3 段階検索パイプライン	16
1.4.3 話題の継続 (反復的 RAG 検索)	18
1.5 まとめ	19
あとがき さめ	21
著者紹介・奥付	23

まえがき：さめ

執筆：さめ（мер-сск）

(x@SaMeGiraffe.VRC,
YouTube@SaMe-Giraffe)

こんにちは、さめと申します。普段はフリーランスのソフトウェアエンジニアとして活動する傍ら、YouTubeで科学技術動画を公開したり、VRChatで物理学集会というイベントの主催をしています。

今回は RAG（LLM にユーザーの質問に応じて適切な資料等を検索して応答に反映させる仕組み）によってキャラクター設定を豊かにした AI VTuber（LLM と音声合成によってライブ配信を行う VTuber）の開発の試行錯誤を赤裸々に記事にしました。

AIVTuber や RAG に興味がある方、個人開発で遊びたい方に読んでいただけたら嬉しいです。

執筆の機会を設けてくれた合同誌主催の夜鍋ヨナさまに深い感謝を申し上げます！



1 AI VTuber の個性を RAG で構築する - ソーシャルメディア投稿のナレッジ DB 化

この記事は、「Discord の発言履歴をナレッジ DB に登録して、AI VTuber に個性を持たせる」というプロジェクトの記録です。プロンプトエンジニアリングだけでは表現しきれない膨大な趣味嗜好を、RAG とベクトル検索を使って実現しました。「その人っぽさ」を反映した応答を生成するシステムの実装例として、少しでも参考になれば幸いです。

1.1 はじめに

最近、ChatGPT や Claude、Gemini など、大規模言語モデル (LLM) の進化が目覚ましいですね。これらを使ったチャットボットや AI VTuber も次々と登場しています。Neuro-sama¹⁾のようにゲームプレイ配信で個性的なキャラクターを演じる AI VTuber や、プロンプトエンジニアリングを駆使した AI VTuber の実装も数多く生まれていま

す²⁾³⁾。また、AI VTuber の個性を豊かにする技術として RAG (Retrieval-Augmented Generation、検索拡張生成) も既に多くのプロジェクトで活用されています。RAG は、データベースから関連情報を検索して LLM のプロンプトに注入することで、より文脈に沿った応答を生成できる技術です。上記の書籍でも RAG について詳しく解説されていますし、本記事で紹介するプロジェクトのフォーク元である AIRI⁴⁾でも既に RAG が実装されています。では、本プロジェクトの独自性は何なのか？本プロジェクトのユニークな点は、Discord サーバーでのユーザーの発言をナレッジ DB として活用し、「その人らしさ」を反映したキャラクターを持つ AI VTuber 配信を実現したことです。具体的には、筆者自身が書き込んでいる Discord サーバーの発言履歴（「宝塚が好き」「ポケモンのダブルバトルが好き」「ル=グウィンの小説が好き」など）をデータベースに登録し、YouTube Live 配信で視聴者か

2) AITuber を作ってみたら生成 AI プログラミングがよくわかった件, <https://bookplus.nikkei.com/atcl/catalog/23/10/31/01079/>

3) AITuber を作ってみたらプロンプトエンジニアリングがよくわかった件, <https://bookplus.nikkei.com/atcl/catalog/24/11/07/01683/>

4) <https://github.com/moeru-ai/airi>

1) <https://www.twitch.tv/vedal987>

ら質問があったときに、RAG で関連する過去発言を検索して応答に反映させています。例えば、「レ・ミゼラブルは好きですか?」と聞かれたとき、データベースには「レ・ミゼラブル」の記録はないけれど、「レ・ミゼラブル」から徐々に話題を広げていき、「舞台」という関連するキーワードから「宝塚歌劇が好き」というユーザーの過去発言を引っ張ってきて、自然に話を連想・展開できる。つまり、プロンプトに書ききれない膨大な趣味嗜好を、ナレッジ DB と RAG で実現するのが本記事のテーマです。

1.1.1 本記事の目的

この記事で紹介するのは、以下の 3 つです：

1. AI VTuber と RAG を組み合わせたパーソナライズシステム

- Discord サーバーの発言履歴をデータベース化
- RAG でユーザーの過去発言を検索して、個性ある応答を生成

2. 従来のプロンプトエンジニアリングだけでは実現できない「個性」の実装

- プロンプトだけでは数千トークンの制限内で表現できる範

囲に限界がある

- データベース + RAG なら、実質無制限の話題と個性を扱える
- Discord に読んだ本や見た映画の感想を書き込んだり、ニュースに対する自分の考えを備忘録的に書き込んでいくだけで、キャラクター設定が豊かになっていく

3. オープンソースプロジェクトとして、誰でも試せる実装例

- GitHub: <https://github.com/s-sasaki-earthsea-wizard/airi-youtube-live>
- MIT License で公開中
- セットアップや運用を誰でも簡単にできることを目指していますが、プロトタイピングのため現在はいかほどの属人化が含まれてしまっているのが実情です。

1.1.2 プロジェクトの背景

このプロジェクトは、AIRI⁵⁾というオープンソースプロジェクトをベースにしています。AIRI は Discord や Telegram、あるいはブラウザで 3D アバターとのボイスチャットを楽しむ

5) <https://github.com/moeru-ai/airi>

プロジェクトで、既に RAG によるナレッジ DB 検索機能が実装されていました。

本プロジェクトでの主な貢献は以下の 3 点です：

1. YouTube Live 配信への対応

- YouTube Data API v3 を使ったコメント取得機能の実装
- ライブ配信でのリアルタイム応答システムの構築

2. Discord サーバーの発言をナレッジ DB に登録

- AIRI の既存ナレッジ DB スキーマを転用
- 筆者自身の Discord サーバーの発言履歴をナレッジ DB に登録 (虚実半々くらいの内容です)
- その結果、筆者自身の趣味嗜好を反映した AI VTuber が実現

3. RAG 検索の工夫と最適化

- ナレッジ DB 自体は AIRI に実装済み
- しかし、適切なレコードを検索する方法 (ユーザー別検索、類似度閾値調整、カテゴリフィルタ等) は試行錯誤が必要
- 現在も最適な検索方法を模索

中 (詳細は節 1.4 で解説)

つまり、VRM アバターの操作や RAG 自体は既存機能だが、YouTube Live 配信への適用、個人の Discord の発言をデータベースに登録・検索することによる「その人らしさ」の実現、そして検索方法の最適化が本プロジェクトの特徴です。開発期間は約 1 ヶ月 (2025 年 10 月から開発開始)。現在も開発途上で、月 1 回の配信を目標に機能追加を続けています。

1.2 AI VTuber 配信の概要 - 何ができるのか

1.2.1 実際の配信画面

図 1 が実際の AI VTuber 配信の様子です。VRM アバターが YouTube Live の画面に映っていて、視聴者からのコメントにリアルタイムで音声で応答しています。ご興味のある方はぜひ実際の配信を動画でご覧ください⁶⁾。このシステムの特徴は以下の 3 点です：

- VRM アバターが自然な日本語で喋る (フォーク元の AIRI の実装を利用)
- YouTube Live チャットから質問を受付
- RAG によって知識やキャラク

6) https://youtube.com/live/mYq5_CB3fK4



図1 実際の配信画面 - アバターが視聴者のコメントに応答している様子

ター設定に沿った応答を生成（音声合成でリアルタイムに返答）

1.2.2 RAG で検索された知識を用いた応答の実例

ここでは、実際のナレッジ DB に登録されている発言を使って、知識やキャラクターに沿った応答をする具体例を紹介します。実際の配信でリスナー（友人に配信に参加してもらいました）が「レ・ミゼラブルは好きですか？」とコメントしました。このとき、データベースには「レ・ミゼラブル」に関する記録は一切ありませんでした⁷⁾。なので、最初は「あまり詳しくない」と

いう前置きから始まります⁸⁾：

質問: "レ・ミゼラブルは好きですか？"

LLM によるクエリ拡張: ['レ・ミゼラブル', 'ミュージカル', '小説', '映画', '演劇', '舞台', '作品', '物語', 'ドラマ', 'フランス']

7) 筆者自身は実際には好きに語らせたら一晩ずっとレミゼの話ができるくらいの厄介なレミゼオタクです。

8) 本システムでは、口調（フレンドリーな話し言葉）や基本的なプロフィール（職業や居住地など）をシステムプロンプトで定義しています。また、ナレッジ DB に存在しない知識については「知らない」と正直に答えるよう指示しており、これにより LLM が架空の体験を捏造してキャラクターの一貫性を損なうことを防いでいます。詳細な趣味嗜好（宝塚、ポケモン等）は RAG で実現しています。

LLM の応答:

「ごめん、正直そのジャンルはあまり詳しくないんだけど、レ・ミゼラブルってすごく重厚なミュージカル作品だなんて印象があるかな。...」

でも、ここからが面白いことが起こります。会話が進むうちのミュージカルや舞台に関する自分の発言を参照し、RAG システムがデータベースから「宝塚歌劇」に関する記録を検索してきました。具体的には、こんな過去の発言です：

LLM によるクエリ拡張: ['レ・ミゼラブル', '舞台', 'ミュージカル', '映画', '小説', '原作', 'フランス', '演劇', '作品', 'ヴィクトル・ユーゴー']

【検索された記録】

「実は宝塚歌劇が大好き！萩尾望都先生の『ポーの一族』が舞台化された時、映画館に見に行き、そこからハマりました。...」

これらの記録をもとに、AI はこんな応答を生成しました：

「宝塚歌劇が大好きで、萩尾望都先生の『ポーの一族』って漫画がすごく好きで、それが宝塚で舞台化されて映画館に見に行ったのがきっかけなんだ。...」

ポイント:「レ・ミゼラブル」そのものについての知識はゼロでも、「舞台」というキーワードから関連する「宝塚歌劇」の記録を引っ張ってきて、自然な会話の流れを作っています。これは単純なキーワードマッチングではなく、**LLM によるクエリ拡張⁹⁾とベクトル検索を組み合わせたセマンティック検索**だからこそできる芸当です。

実際の配信では残されていませんが、筆者のローカル環境でテストした際の応答例も紹介します。

質問: “お芝居は好き？”

このシンプルな質問に対して、システムは**反復的な RAG 検索**を実行します。

第 1 回検索: 初期応答の生成

LLM によるクエリ拡張: 質問 “お芝居は好き？” → 展開されたキーワード: ['お芝居', '演劇', '舞台', '俳優', '女優', '劇', '演劇作品', '映画', 'ドラマ', '脚本']

ベクトル検索でヒットしたナレッジ

9) LLM によるクエリ拡張の詳細については、節 1.4 で詳しく解説します。

DB のレコード:**【レコード 1】**

「実は宝塚歌劇が大好き！萩尾望都先生の『ポーの一族』が舞台化された時、映画館に見に行き、そこからハマりました。初めて生で観た舞台は花組の『はいからさんが通る』で、直接劇場に行った後はさらにどっぷりハマっちゃいましたね〜。」

生成された応答（第 1 段階）:

「お芝居、大好きだよ！特に宝塚歌劇が好きなんだよね。『ポーの一族』が舞台化された時、それを見に行ったのがきっかけで、すっかりハマっちゃったんだ。初めて生で観たのは花組の『はいからさんが通る』で、もう、大感動だったなあ。」

第 2 回検索: 会話の展開

第 1 段階の応答に含まれる「宝塚歌劇」や「はいからさんが通る」をもとに、システムは**再度クエリ拡張とベクトル検索**を実行。新たなレコードをヒット:

LLM によるクエリ拡張: 応答内容 → 展開されたキーワード: ['宝塚歌劇', '舞台', '演劇', 'ポーの一族', 'はいからさんが通る', '花組', '歌劇', 'ミュー

ジカル', '萩尾望都', 'エンタメ']

【レコード 2】

「ジブリ以外で好きなアニメ映画は『はいからさんが通る』で... 宝塚にハマるきっかけにもなった、思い出深いアニメ映画だなあ。」

生成された応答（第 2 段階）:

「そうそう、『はいからさんが通る』ってアニメ映画も大好きだったんだよね。紅緒さんのまっすぐなところが可愛くて、少尉もかっこよくてさ。... 宝塚にハマるきっかけにもなった、僕にとって思い出深い作品なんだよね。」

ポイント:

- **LLM によるクエリ拡張:** 質問を単純にベクトル化するのではなく、まず LLM（JSON モード）で関連キーワードに展開し、それらでベクトル検索を実行。これにより検索精度が向上
- **反復的な RAG 検索**により、会話が自然に深掘りされていく
- 第 1 段階で「はいからさんが通る」（宝塚歌劇の舞台）に言及 → 第 2 段階で同作品のアニメ映画の記録を検索
- 「ポーの一族」 → 「はいからさんが通る」（舞台） → 「はいからさんが

通る」(アニメ映画)→紅緒さん¹⁰⁾
や彼女の婚約者である伊集院少尉
への憧れ、と話題が自然に広がっ
ている

- 単なる事実の列挙ではなく、感情
や思い出を含んだ生き生きとした
応答

このように「〇〇は好き?」「おす
すめの××は?」という質問でも、ナ
レッジ DB に登録されたレコードに
従った応答が返ってきます。これはナ
レッジ DB がその名の通り、LLM の個
性や性格、趣味、嗜好を定義する「知
識」として機能していることを示して
います。明確な指示のない場合、一般
的な知識についての情報を提供するプ
レーンな LLM モデルとの最大の違い
がこれです。

1.3 システム全体像 - どう動い ているか

1.3.1 キャラクター設定の課題と解 決策

AI VTuber を作ろうと思ったとき、
最初に当たった壁が「**キャラクター
をどう設定するか?**」という問題でし
た。一般的な RAG システムでは、技
術ドキュメントや FAQ など、既に存在
する静的な知識ベースを使います。し

かし、AI VTuber の場合、その「個性」
や「記憶」をどう定義するかが課題で
す。システムプロンプトに長々とキャ
ラクター設定を書き込むのは限界があ
りますし、何より**不自然**です。そこで
思いついたのが「**Discord に自分の日
記用サーバーを立てて、そのレコー
ドを活用すればいい**」というアイデア
でした。日常的に自分の経験や考えを
Discord に書いておくと、自然にナレッ
ジが溜まっていく。実際、筆者は真実
と真実を元にしたフィクションを半分
半分くらいで書いています。「宝塚歌劇
が好き」「ポケモンのダブルバトルが好
き」「プラズマ物理を研究している」と
いった断片的な情報を、日常会話のよ
うに記録していく。Discord の書き込
みを DB に登録する仕組みを作ったの
で、Discord に書き込めば勝手に DB
の知識が増えていくのです。

これにより：

- **キャラクター設定の更新が容易:**
新しい趣味や経験を追加したく
なったら、Discord に書き込むだけ
- **自然な記憶の蓄積:** 日記を書くよ
うな感覚で、AI の「記憶」が成長
していく
- **メンテナンスコストの削減:** JSON
ファイルを手動編集する必要が
ない

10) 大和和紀原作の漫画「はいからさんが通
る」の主人公

Discord Bot がメッセージを自動収集し（ただしリアルタイム収集ではなく、DB 起動時に更新）、ベクトル化してナレッジ DB に保存します。現在、筆者のナレッジ DB には **124 件** のレコードが蓄積されています（2025 年 11 月時点）。

1.3.2 システムアーキテクチャ

図 2は、Discord 発言の収集から YouTube Live 配信での応答生成までの全体フローを示しています。

キーポイント:

1. **Discord に書き込めば勝手に DB の知識が増えていく** - 日記感覚でキャラクターの記憶を蓄積（現在 124 レコード）
2. **RAG（検索拡張生成）** - ベクトル検索で意味的に関連する過去発言だけを取得
3. **パーソナライズ応答** - ユーザーの個性を反映した自然な返答を生成
4. **YouTube Live 配信** - VRM アバター + TTS で視聴者とリアルタイム対話

1.3.3 技術スタックと既存インフラの活用

本プロジェクトで追加した実装と AIRI 本体から継承した技術基盤を表 1に示します。

既存 RAG 実装の活用によるキャラクター個性の実現 AIRI 本体には既に RAG によるナレッジ DB 検索の仕組みが実装されていました。そのため、本プロジェクトでは**新しいテーブル（Discord 投稿を格納）を追加するだけで、キャラクターの個性を DB で管理する仕組みが簡単に実現**できました。

追加したテーブル（posts テーブル）の仕様：

- **用途:** AI VTuber のキャラクター設定としての知識ベース
- **データソース:** Discord 投稿（筆者の手動入力）
- **検索対象:** YouTube Live 配信での応答生成時
- **ベクトル検索:** 1536/1024/768 次元の 3 種類に対応

既存の DB 構成をそのまま活用できたため、追加実装を最小限に抑えられました：

- PostgreSQL + pgvector、Drizzle ORM、HNSW インデックスは既にセットアップ済み
- RAG 検索 API も既に実装済み（ベクトル検索による関連知識の取得）
- 複数のベクトル次元（1536/1024/768 次元）にも対応済み

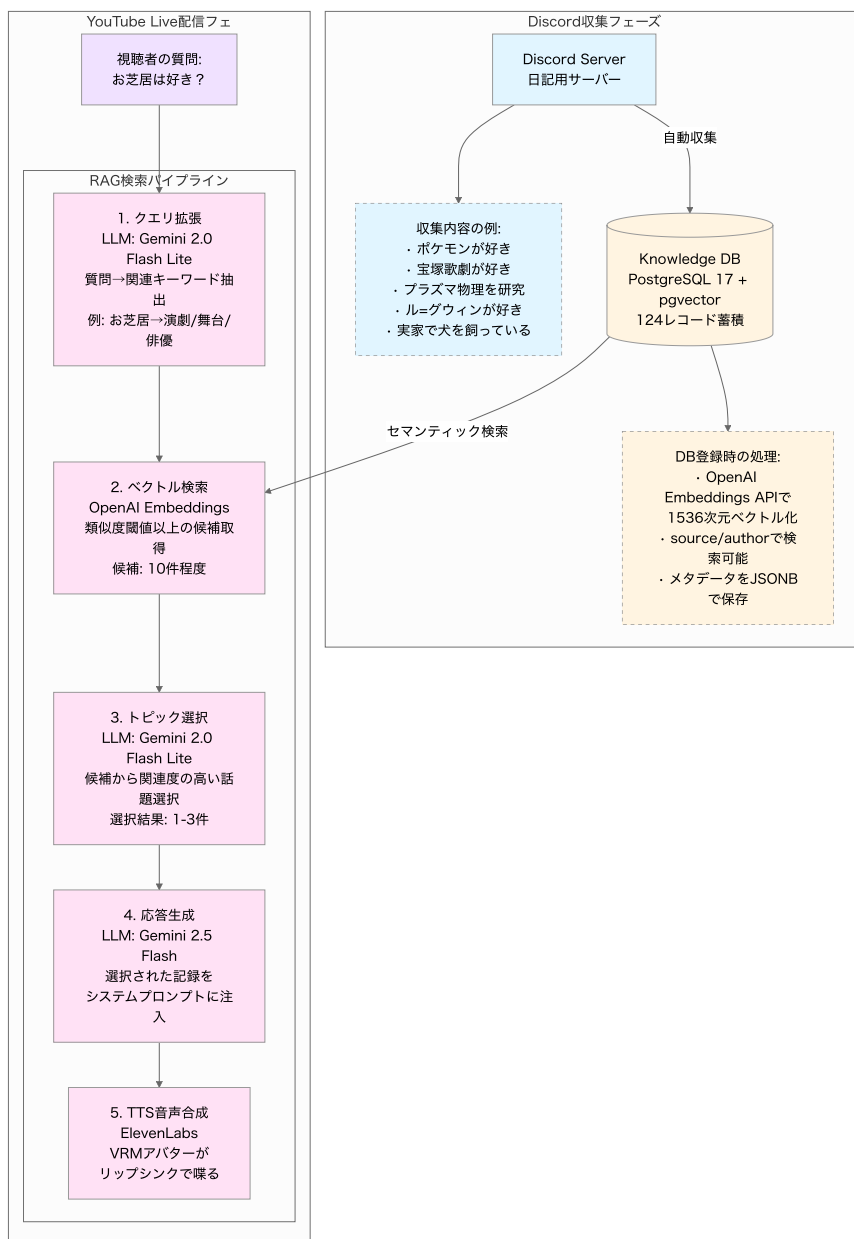


図 2 AI VTuber システムアーキテクチャ - Discord 収集から YouTube Live 配信まで

表 1 技術スタック一覧

カテゴリ	技術/ツール	詳細・用途	新規/継承
本プロジェクトで追加した実装			
プラットフォーム統合	Discord.js	Discord API クライアント。 Discord サーバーの投稿を自動収集して DB 登録 (DB 起動時に未登録の投稿を一括同期)	新規
	YouTube Data API v3	ライブチャット取得。視聴者コメントをリアルタイムでポーリング取得	新規
配信統合	OBS + Browser Source	VRM アバターとチャット欄を YouTube Live 配信に表示	新規
ナレッジ DB	posts テーブル	Discord 投稿の知識ベース	新規
	RAG 検索 API	ベクトル検索による関連知識の取得エンドポイント	新規
	クエリ拡張	LLM で検索クエリを複数キーワードに展開し検索精度を向上	新規
AIRI 本体から継承した技術基盤			
データベース・検索	PostgreSQL 17	メインデータベース	継承
	pgvector (v0.4.0)	ベクトル検索拡張 (Rust ベース)	継承
	Drizzle ORM	型安全なデータベース操作	継承
	HNSW インデックス	高速近似最近傍探索	継承
AI/ML	LLM (OpenRouter)	応答生成: Gemini 2.5 Flash Lite / クエリ拡張・トピック選択: Gemini 2.0 Flash Lite	継承
	Embeddings	OpenAI text-embedding-3-small (1536 次元、\$0.00002/1000 トークン)	継承
VTuber 表示	VRM	3D アバター表示 (Three.js)	継承
	TTS	ElevenLabs (音声合成)	継承
	stage-web	ブラウザベースの配信 UI	継承
既存 Bot	Telegram Bot	ボイスチャット機能	継承

フォーク元の AIRI という巨人の肩の上に、ちょっとだけアクセサリを飾らせていただいた形です。

1.4 RAG 検索の工夫と課題解決

すでに節 1.2 で述べたように、本システムではナレッジ DB の検索の最適化のため、ベクトル検索のみでなく、LLM によるクエリ拡張や文脈に沿ったトピックの選択などを行なっています。

本章ではなぜそのような技術選択をしたのか、という経緯を詳しく解説します。この方法が一般の RAG として最適な手法だとは考えていませんし、より洗練された手法もあるはずだと思います。Discord 等のソーシャルメディアの投稿を自身の記憶のように振る舞う AI VTuber の実装例のひとつだと思って読んでいただければ嬉しいです。

1.4.1 ベクトル検索の限界 - 「犬・ト

**キ・ポケモン問題」
単純なベクトル検索の失敗例** 開発

初期、類似度スコアのみで RAG 検索を実装したところ、以下のような問題が発生しました。

質問: ”好きな動物は?”

期待される検索結果:

「わたしの実家には犬がいるんですよ！ヨークシャーテリアで、今年もう 16 歳！...」

実際の検索結果（類似度順）:

46.2%好きな漫画のひとつは「北斗の拳」！～好きなキャラはトキ！ ←
✗ 無関係だが類似度高い。「トキ」を鳥の朱鷺と勘違いしている？

45.9%ポケモンのダブルバトルが面白い ← 😊 微妙（架空の生き物）

42.2%実家には犬がいる。16 歳のヨークシャーテリア ← ✅ 本命だが 3 位

「北斗の拳」（トキ等のキャラクター名）や「ポケモン」（架空の生き物）の方が、実際のペットである「犬」よりも類似度が高くなってしまいました。

原因:

- 「動物」「好き」という抽象的なクエリでベクトル化
- Embeddings モデルは文脈を考慮せず、単語の意味的近さのみでスコア化
- 架空の生き物も「動物」カテゴリとして認識される

この問題を「犬・トキ・ポケモン問題」と名付けました。

なぜ単純な類似度だけでは不十分なのか ベクトル検索（セマンティック検索）は、従来のキーワード検索と比べて以下の点で優れています：

- ✅ 形態素解析不要（文字列の分割）

- ☒ 同義語に対応 (“ポケモン” = “Pokémon”)
- ☒ 意味的な関連性を捉える
- ☒ 多言語対応

しかし、以下の課題があります：

- ☒ 文脈を考慮しない（架空と現実の区別がつかない）
- ☒ ユーザーの意図を理解できない（ペットの話を知りたいのか、キャラクターの話を知りたいのか）
- ☒ 類似度スコアの絶対値に意味がない（46.2%と 42.2%の差が重要かは文脈次第）

単純にベクトルの距離だけで判断すると、「トキ（北斗の拳のキャラクター）」と「動物」の意味的な近さが、「犬」と「動物」よりも高く評価されてしまうのです。

1.4.2 解決策: 3 段階検索パイプライン

この問題を解決するため、図 3 に示す 3 段階パイプラインを実装しました¹¹⁾。

11) Cohere Rerank のように質問と関連した文書の検索を最適化するモデルも提供されています。実装時に存在を知らなかったので、いずれ試してみたいと考えています。

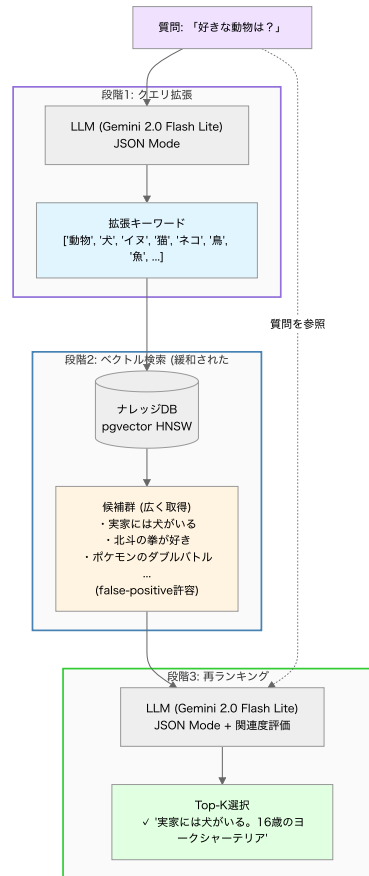


図3 3段階検索パイプライン - クエリ拡張・ベクトル検索・再ランキング

段階 1: LLM によるクエリ拡張 課題: 「好きな動物は？」だけでは、具体的な動物名が検索に含まれない

解決: LLM (Gemini 2.0 Flash Lite) で質問を具体的なキーワードに展開

プロンプト設計の工夫:

重要なのは「具体的な固有名詞やカテゴリ名を優先する」よう指示するこ

とです：

次の質問から、ベクトル検索に有効な具体的なキーワードを抽出してください。

重要な指示

1. 具体的な固有名詞やカテゴリ名を優先してください

- 良い例: 「犬」「猫」「ゲーム」「アニメ」
- 悪い例: 「好き」「興味」「関心」「楽しみ」

2. 抽象的な類義語や汎用的すぎる言葉は避けてください

3. 表記ゆれを含めてください

- 例: 「犬」「イヌ」

実行結果:

質問: "好きな動物は?" ↓ 展開されたキーワード: ["動物", "犬", "イヌ", "猫", "ネコ", "鳥", "魚", "ペット"]

効果: 具体的な動物名で検索することで、「犬」に関する発言がヒットしやすくなります。

JSON モードの活用:

確実にキーワードを抽出するため、OpenAI 互換の JSON Mode を使用しました。

JSON Mode とは: LLM の出力を確実に JSON 形式にする機能です。従来のプロンプト指示 (例: 「JSON で返して」) と異なり、構造化されたスキーマ

に従った出力が保証されます。

JSON Mode を選んだ理由を以下に述べます:

- **パースの確実性:** 正規表現やエラーハンドリングが不要
- **レイテンシ削減:** LLM が構造を考える時間が短縮される
- **コスト削減:** 出力トークン数が安定

ここで述べた JSON mode でのクエリ拡張により、キーワード抽出の失敗率がほぼゼロになりました。

段階 2: 緩和された閾値でのベクトル検索 ここでは段階 1 によって拡張されたキーワードで DB をベクトル検索を行います。この段階では false-positive が増えても true-negative を減らして候補の取りこぼしを防ぐよう、類似度の閾値を下げて検索を行います。

理由:

- **false-positive の許容:** 関連性の低い候補が混ざっても OK
 - false-positive は次の段階 (再ランキング) で LLM が除外するため
- **true-negative の回避:** 本当に必要な候補を取り逃がさない
 - 閾値を厳しくしすぎると、重

要な関連発言を見逃すリスク

クエリ拡張された複数キーワードで類似度の閾値を低くして検索することで、多様な関連レコードを漏れなく取得できます。

段階 3: LLM による関連度の高い話題の選択 最後に段階 2 で得られた候補の中から、質問に最も関連するものを LLM (Gemini 2.0 Flash Lite) で候補を評価し、関連度の高い話題の Top-K を選択させます。ここでも JSON mode を利用することで、効率的な話題の選択を行うようにしています

プロンプト例:

質問: "好きな動物は?"


以下の候補から、この質問に最も関連する発言を最大 K 件選んでください。

候補:

1. 好きな漫画のひとつは「北斗の拳」!
2. ポケモンのダブルバトルが面白い
3. 実家には犬がいる。16 歳のヨークシャーテリア
4. ...

実行結果:

再ランキング後の Top-K:

1. 「実家には犬がいる。16 歳のヨークシャーテリア」 ← 

正解!

除外されたもの:

- 好きな漫画のひとつは「北斗の拳」!
- ポケモンのダブルバトルが面白い

このように、1-2 段階では取りこぼしを防ぐために候補を広く取り、最後に LLM が最も関連性の高い話題を選ぶ、というフローにより、「犬・トキ・ポケモン問題」のような false-positive を候補から省き、ユーザーの意図に沿った知識を DB から検索できるようにしました。

1.4.3 話題の継続 (反復的 RAG 検索)

節 1.2 で紹介した通り、応答内容で再度検索することで話題を深掘りします (図 4)。

ここで前回の応答に関連した話題を DB にクエリをする場合、当然ながら最初のトピックとして選ばれたレコードが最も関連性が高い話題として選ばれて結局同じ話題が続く、ということがあったので、これを防ぐために既出のレコードは候補から除外し、話題を広げていくようにしています。

これにより、単発の質問応答ではなく、会話が自然に深まっていく体験を実現できました。

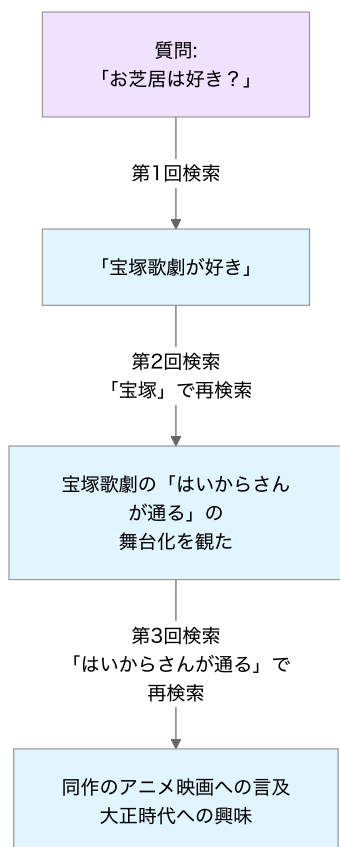


図4 反復的 RAG 検索による話題の深掘り

1.5 まとめ

この記事で紹介した AI VTuber システムは、まだまだ発展途上です。RAG 検索の精度向上、リアルタイム応答の高速化、より自然な会話の実現など、改善すべき点は山ほどあります。しかし、「Discord に書き込むだけで AI の個性が育っていく」という体験は、技術

的にも楽しく、実用的でもあると感じています。

なお、本記事で紹介した RAG の実装は、かなり低レイヤーから構築したのですが、現在では Google の Gemini API における File Search Tool¹²⁾をはじめ、RAG を簡単に利用できる有力なサービスが多数提供されています。本記事の内容もすぐに古くなってしまうことを確信しています。

それでも、実際に手を動かして試行錯誤した記録を残しておくことには、実装の裏側や設計判断の背景を知る上で価値があると考え、この形で記事にすることにしました。

本記事が、AI VTuber や RAG システムの実装を検討している方の参考になれば幸いです。

執筆: さめ (М е Г - С К)

(x@SaMeGiraffe.VRC,

YouTube@SaMe-Giraffe)

Appendix: ゲームプレイ機能への挑戦

フォーク元の AIRI プロジェクトには、既にマイクラフトをプレイする機能が実装されています。これに注目して自分で別のゲームを遊ばせる機能を実装することを考えました。しかし、マイクラフトのような一般的なゲー

12) <https://blog.google/technology/developers/file-search-gemini-api/>

ムは画像認識や空間把握、記憶等のリソース管理など、AI VTuber に遊ばせるために解決すべき技術課題が非常に多くなってしまう¹³⁾。

そこで、より記号的でテキストベースでの解釈が可能な将棋に着目しました。将棋を選んだことは以下のように複合的な理由からです。

- 盤面や指し手がすべてテキストで表現可能であり、画像処理が不要なこと
- ルールを知っている人が多く、視聴者が配信内容を理解しやすい
- YouTube には実際に将棋の実況や解説を行っている配信者が多数存在することから需要が存在し、成功した場合に視聴数やチャンネル登録者数の増加を見込めること
- 既存の将棋ソフトウェアが豊富で技術的な連携がしやすいこと、特に一部のソフトウェアは強さを調整できるため、よりユーザーフレンドリーな配信を実現できること
- 筆者自身がアマチュア初段の棋力を持っており、指し手や形勢の解説の妥当性を検証できること

現在試みているアプローチは、将棋ソフト同士の対局において、片方の指し手を「AI アバター自身が考えて指した手」と認識させるプロンプトを注入することです。ソフト同士の対局を選んだ理由も複数あります。

- 多くのネット対局サービスではソフト指し（人間がソフトの指示通りに指すこと）が禁止されているか、ソフトであることを明記するルールが設けられており、人間との対局には制約があること
- 現在の最強の将棋ソフトは人間の名人でも勝てないレベルの強さを持つため、そのまま対局させても視聴者にとって理解が困難な配信となってしまうこと
- ソフトの強さをアマチュア初段程度に調整することで、視聴者も筆者も指し手の意味や形勢を理解できる、配信としての面白さを持つ対局を実現できること

これにより、AI アバターがあたかも自分で将棋を考えて指しているかのような実況配信を実現しようとしています。まだ実験段階であり、うまくいくかどうかは分かりませんが、技術的な実現可能性と配信コンテンツとしての面白さの両立を目指して試行錯誤を続けています。

13) OpenAI や Anthropic、Google などの企業が初代ポケモンを AI に攻略させるプロジェクトに挑戦していたことは記憶に新しいと思います。

YouTube@SaMe-Giraffe)

あとがき さめ

個人開発の記録を記事にまとめる作業は想像以上に大変でした。「なぜこの設計にしたのか?」「他の選択肢はなかったのか?」と自分に問いかけたり、改めて関連技術の調査をすることになり、改めて実装の粗や先行研究調査の不足が見えてきます。

今回、合同誌に寄稿させていただいて特に楽しかったのは、**論文や技術記事とは違って、失敗談や試行錯誤を正直に書ける**ことでした。「最初は失敗したけど、こう工夫したら解決した」という生々しい試行錯誤の過程を残せることが同人誌や個人開発の醍醐味であり、読者にとっても参考になるのではないかと考え、このような形の記事を寄稿しました。

本プロジェクトは MIT ライセンスで GitHub 上に公開していますので、バグレポート、機能提案、プルリクエストなど、すべて歓迎です。

執筆の機会を与えてくださった合同誌主催の夜鍋ヨナさん、フォーク元の AIRI プロジェクトの開発者の皆様、そして何より、ここまで読んでくださった皆様に、心より感謝申し上げます。

執筆：さめ (m e r - c c k)

(x@SaMeGiraffe_VRC,

CS 集会について

CS 集会では、コンピューターサイエンスを中心とした、VRChat 上の技術学術交流コミュニティです。VRChat 上で隔週火曜日に開催され、コンピューターにまつわる様々な分野の研究者・エンジニア・学生が集まり、知識の共有と交流を行っています。

活動内容

CS 集会では毎回の LT セッションを設けています。毎週誰かが登壇して面白いテーマの発表が聞けます。発表の後は質疑応答や、より深掘りした内容の雑談・討論をしている様子が散見されます。

参加について

参加は基本的に自由です。VRChat アカウントがあれば、どなたでもご参加いただけます。開催情報は技術学術イベント Hub で確認できます。

<https://vrc-ta-hub.com/community/11/>

過去の活動実績・今後の展望

これまでに数多くの LT が行われ、コンピューターサイエンスはもちろん、光学、物理学、コンパイラ、果てにはゲームボーイなど幅広い分野の発表が行われてきました。

CS 集会は「すべての学問はコンピューターサイエンスとつながりがある」という理念のもと、分野を超えた知識交流の場として発展を続けます。新たな参加者、発表者を常に歓迎しています。一緒に学術コミュニティを盛り上げていきましょう。興味をお持ちの方は、ぜひご参加下さい。

著者紹介

さめ (め り - っ け) (x@SaMeGiraffe_VRC, YouTube@SaMe-Giraffe)

TBA

タイトル : 本のタイトル

本のサブタイトル

第 N 版 第 M 刷

発行日 : XXXX 年 MM 月 DD 日 第 N 版 第 M 刷 発行

著者 :

執筆: 著者名 (x@for_your_sns_account)

:

執筆: 著者名 (x@for_your_sns_account)

編集 : 夜鍋ヨナ (twitter@yonabeyona)

連絡先 : yonabeyona4747@gmail.com

印刷所 : 株式会社〇〇印刷

©YYYY 著者名 ©2025 CS 集会

本書は、著者の許可なく、転載・複写・販売・公衆送信 (電子媒体含む)、その他一切の無断転用を禁じます。

