C4A
Computing for All

対話システム構築フレームワーク DialBBチュートリアル

v2.0対応

中野 幹生

株式会社C4A研究所

DialBBの概要

DialBBとは

C4A
Computing for All

- 対話システムを構築するためのツール
- 目標
  - 様々なスキルレベルの人が使える
  - シンプルなシステムから複雑なシステムまで
  - 極力最新の技術を用いることができるように

Dialogue System
Development
Framework with
Building Blocks

ターゲットユーザ

C4A
Computing for All

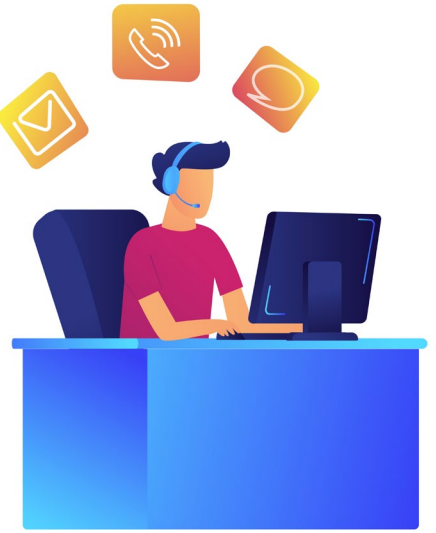
- 対話システム技術の初学者
- 対話システム以外の情報系研究者
- 非情報系の研究者
- 対話システム研究者で楽に論文を書きたい人

対話システム技術の現在

近年の大規模言語モデル（LLM）の発展により、対話システム技術は大きく発展し、様々な実用システムが作られています。

対話システムとは

- 言語によって人間と情報を授受するシステム
- 対話システムの分類
  - タスクの有無・種類（レストラン予約、雑談、etc.）
  - 話題の範囲（野球、食べ物、etc.）
  - 入出力の種類（モダリティ）（テキスト、音声、画像、etc.）
  - 対話参加者の数（1対1、1対多、etc.）



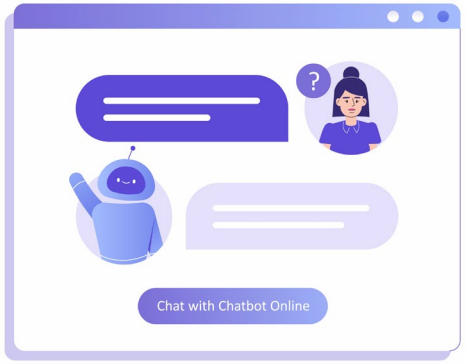
中野 幹生：身近になった対話システム：1．対話システムを知ろう -自然言語による機械と人間とのコミュニケーション-，
情報処理，Vol. 62, No. 10, pp. e1-e6, 2021.

対話システムの例

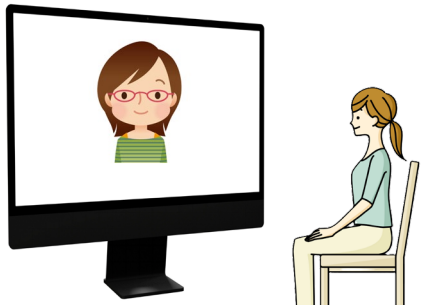
C4A
Computing for All

スマートフォン上
のアシスタント

雑談対話ロボット



FAQチャットボット



カウンセリング
バーチャルエージェント

対話システムの社会的価値

C4A
Computing for All

- 24時間365日いつでも応対できる ⇒ ワークライフバランスの促進
- 人手をかけずに多くの人から情報を引き出せる
- GUI入力では得られない情報が得られる
  - 感情、興味、パーソナリティなど
- 人が応対するよりも本音で話してもらえる*
- 人のコミュニケーションの仲立ちをすることで、人間同士のコミュニケーションを良くする
  - ソーシャルインクルージョン

*Lucas et al, It's only a computer: Virtual humans increase willingness to disclose, Computers in Human Behavior 37, 94-100, 2014

大規模言語モデル（LLM）が変えた対話システム技術

C4A Computing for All

- プロンプトテンプレート一つで対話システムが構築できる
- 従来の対話システムの課題の多くを解決
  - 想定外のユーザ発話への対応
  - 文脈にあった非常に自然な応答が生成される
  - ルールやシナリオを与えなくても対話システムが作れる
- 新しい課題
  - ハルシネーション
  - 遅延
  - プロンプトインジェクションアタックなどの新しいセキュリティリスク

中野： 大規模言語モデルがもたらす対話システム技術の変革, 武蔵大学 AIの社会浸透研究会 第3回公開セミナー

新しい対話アプリケーション

C4A
Computing for All

- 対話の練習
  - 面接（面接する人もされる人も両方）
  - 語学の訓練
- インタビュー・社会調査
  - ユーザから様々な情報を聞き出す
- ファシリテーション

DialBB ノーコードツール

DialBBにはノーコードツールが付属しています。
プログラミング知識がない人でも対話システムを
作ることができます。

ノーコードツール

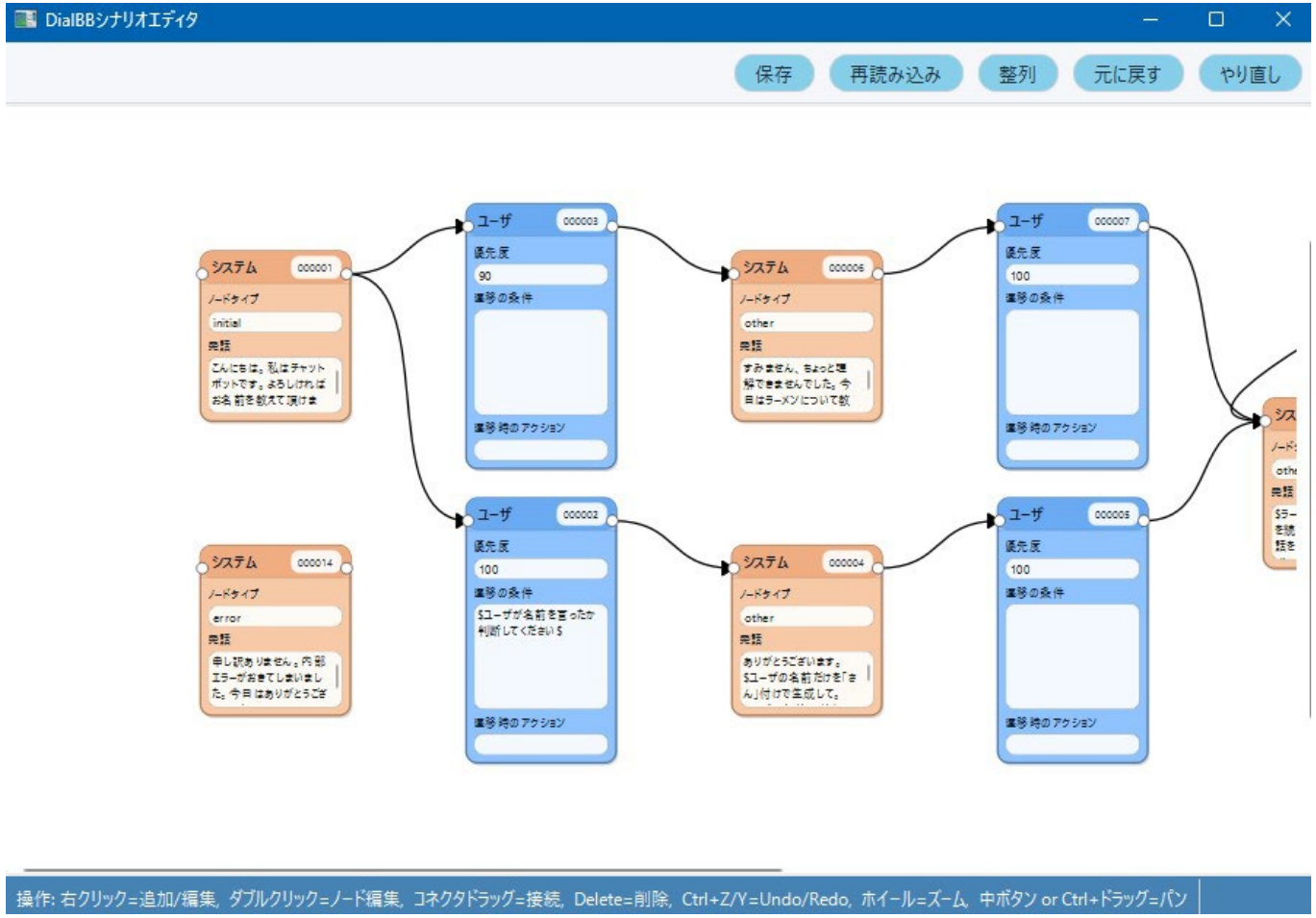
C4A
Computing for All

- GUIで設定、対話シナリオの編集、テストができる
- LLMを使った高度なシステムも作れる

- ユーザシミュレータ（ユーザのように振る舞うソフト）を使った自動テストも可能
- インストールが容易
  - Pythonのインストールは必要
  - Macの場合若干手順が増える

シナリオエディタ

C4A
Computing for All



ノーコードツールの使い方

C4A
Computing for All

- https://c4a-ri.github.io/dialbb/no-code/index-ja.html
  に詳しい説明がある。

ノーコードユーザの皆様へ

C4A
Computing for All

- インストールの仕方や使い方がわからなかったり、何か困ったことがあったら dialbbあっとc4a.jp に連絡ください。
  - "あっと"を@に置き換えてください
- 対話システムを作ってみたいけど、何から始めたら良いかわからない人も是非ご連絡ください。できる限りお手伝いします。

DialBBの概要

- 以降はプログラミングの知識がある人を対象としています。

DialBBの特長 (1)

C4A
Computing for All

- ブロックを組み合わせて対話システムを構築
- コンフィギュレーションファイルで柔軟に構成を変更可能
- クラスAPIとWeb APIでアプリを利用可能
- 組み込みブロックだけでアプリを構築可能
  - 例：スロット抽出＋状態遷移ネットワークによる対話管理
  - 例：LLMのみ
- サンプルアプリが付属

DialBBの特長 (2)

C4A
Computing for All

- Pythonで書かれている
- オープンソースとして公開（Apache License 2.0）
  https://github.com/c4a-ri/dialbb
- 多様なアプリケーション集：https://github.com/c4a-ri/dialbb-apps
- 対話システム構築のお手本となることを目指す
  - コードの質やドキュメンテーションも含む

DialBBのドキュメント

C4A
Computing for All

- プロジェクトサイト
  - https://c4a-ri.github.io/dialbb/
- ノーコードツールのドキュメント
  - https://c4a-ri.github.io/dialbb/no-code/index-ja.html
- インストール方法とサンプルアプリの動かし方
  - GitHub README
    https://github.com/c4a-ri/dialbb/blob/main/README-ja.md
- チュートリアルと仕様詳細
  - https://c4a-ri.github.io/dialbb/document-ja/build/html/

対話システム技術習得のステップ

C4A
Computing for All

1. サンプルアプリケーションを動かして理解し、対話システムの基本的な構成を習得する
2. サンプルアプリケーションが用いている対話知識を変更して、動作がどのように変わるのかを知ることで、要素技術を理解する
3. 組み込みブロックを用いて新しいアプリケーションを作ることで、要素技術の理解を深める
4. 自作ブロックを作成して用いることにより、拡張性の重要性を理解する
5. 構築したシステムを自分以外の人に使ってもらうことで、多様なユーザ発話に対する頑健性やシステムデザインの重要性を理解する

DialBBの基本

サンプルアプリケーションを通じてDialBBの基本を説明します

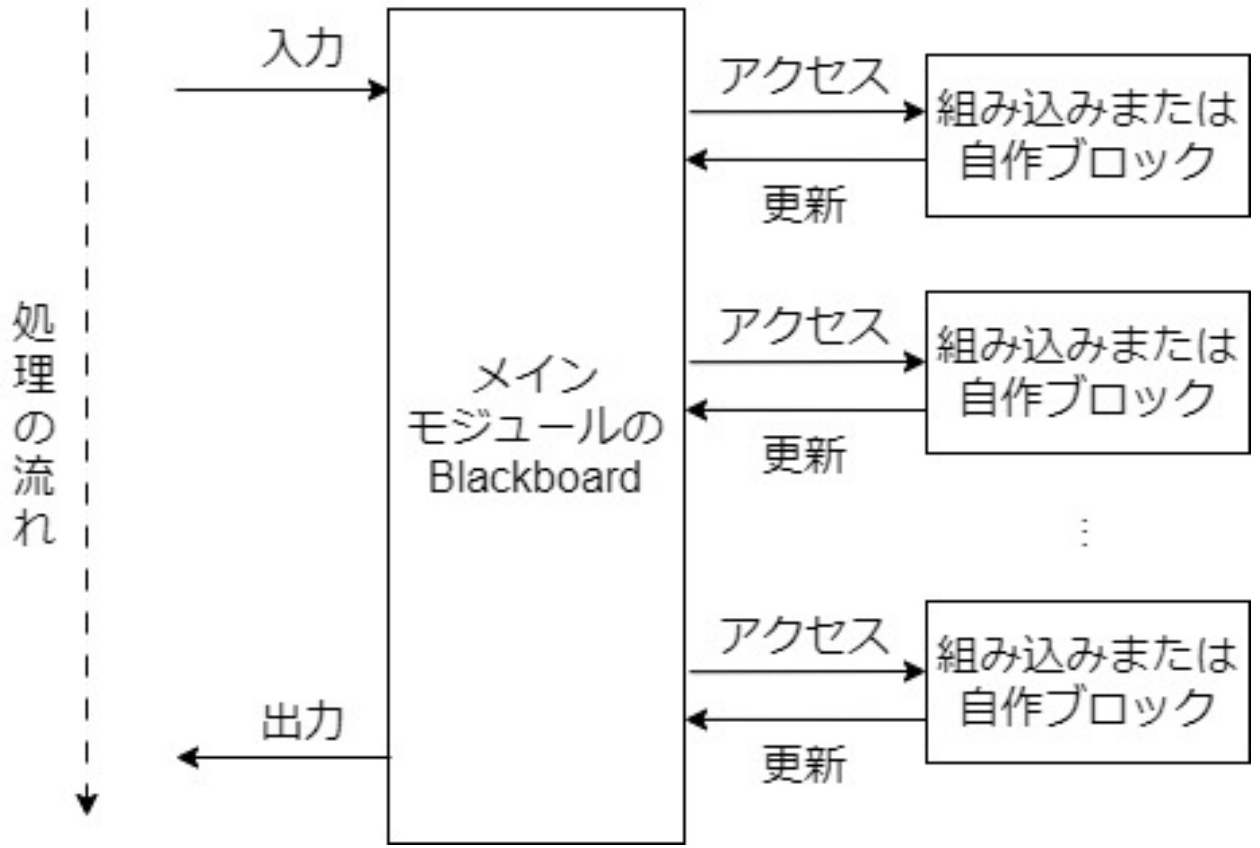
インストールとサンプルアプリの動作確認

C4A
Computing for All

- READMEに手順が書いてある
  - https://github.com/c4a-ri/dialbb/blob/main/README-ja.md
- 普段uvを使っているならuvを使った方が良い

DialBBアプリケーションの構成

C4A
Computing for All



アプリケーションの入出力

C4A
Computing for All

- クラスAPIまたはWebAPI
- クラスAPIの場合の使い方

```
from dialbb.main import DialogueProcessor
app = DialogProcessor(config_file)
response = app.process(request)
```

- 入力がblackboardと呼ぶ辞書データになり、それを各ブロックが順にアップデートする

リクエスト
（対話開始時）

```
{
    “user_id”: <ユーザID：文字列>,
    “aux_data”: <補助データ：オブジェクト>
}
```

リクエスト
（それ以外）

```
{
  "user_id": <ユーザID：文字列>,
  "session_id": <セッションID：文字列>,
  "user_utterance": <ユーザ発話文字列：文字列>,
  "aux_data":<補助データ：オブジェクト>
 }
```

レスポンス

```
{
  "session_id":<セッションID: 文字列>,
  "system_utterance": <システム発話文字列: 文字列>,
  "user_id":<ユーザID: 文字列>,
  "final": <対話終了フラグ：ブール値>
  "aux_data":<補助データ: オブジェクト>
 }
```

コンフィギュレーション

C4A
Computing for All

```
blocks:
  - name: <ブロック名>
    block_class: <モジュール名.クラス名>
    input:
      <ブロック内での参照キー>: <blackboardのキー>
      <ブロック内での参照キー>: <blackboardのキー>
      …
    output:
      <ブロックからの出力のキー>: <blackboardのキー>
      <ブロックからの出力のキー>: <blackboardのキー>
      …
    <このブロック専用のキー>: <ブロック内で用いる情報>
    <このブロック専用のキー>: <ブロック内で用いる情報>
    . . .
 - name: <ブロック名>
    ….
```

- どのようなブロックを使うかやブロックとメインプロセスの通信を定義

コンフィギュレーションの例

C4A
Computing for All

```
blocks:
  - name: canonicalizer
    block_class: <canonicalizerのクラス>
    input:
      input_text: user_utterance
    output:
      output_text: canonicalized_user_utterance
 - name: <ブロック名>
    ….
```

Canonicalizerブロックは非推奨ですが、処理が単純なのでこれを使って説明します

Blackboardのアップデート

C4A Computing for All

```
{
   "user_id": " user1"
   "session_id": "session1",
   "user_utterance": "Cupラーメン",
   "aux_data": {}
 }
```

input_to_block = {input_text: blackboard['user_utterance']}

{input_text: "Cupラーメン"}

正規化ブロック

{"output_text": "cupラーメン"}

```
{
   "user_id": " user1"
   "session_id": "session1",
   "user_utterance": "Cupラーメン"
   "canonicalized_user_utterance": "cupラーメン"
 ,
   "aux_data": {}
 }
```

blackboard['canonicalized_user_utterance']
=output_from_block['ouput_text']

メインモジュールでの対話履歴の保持

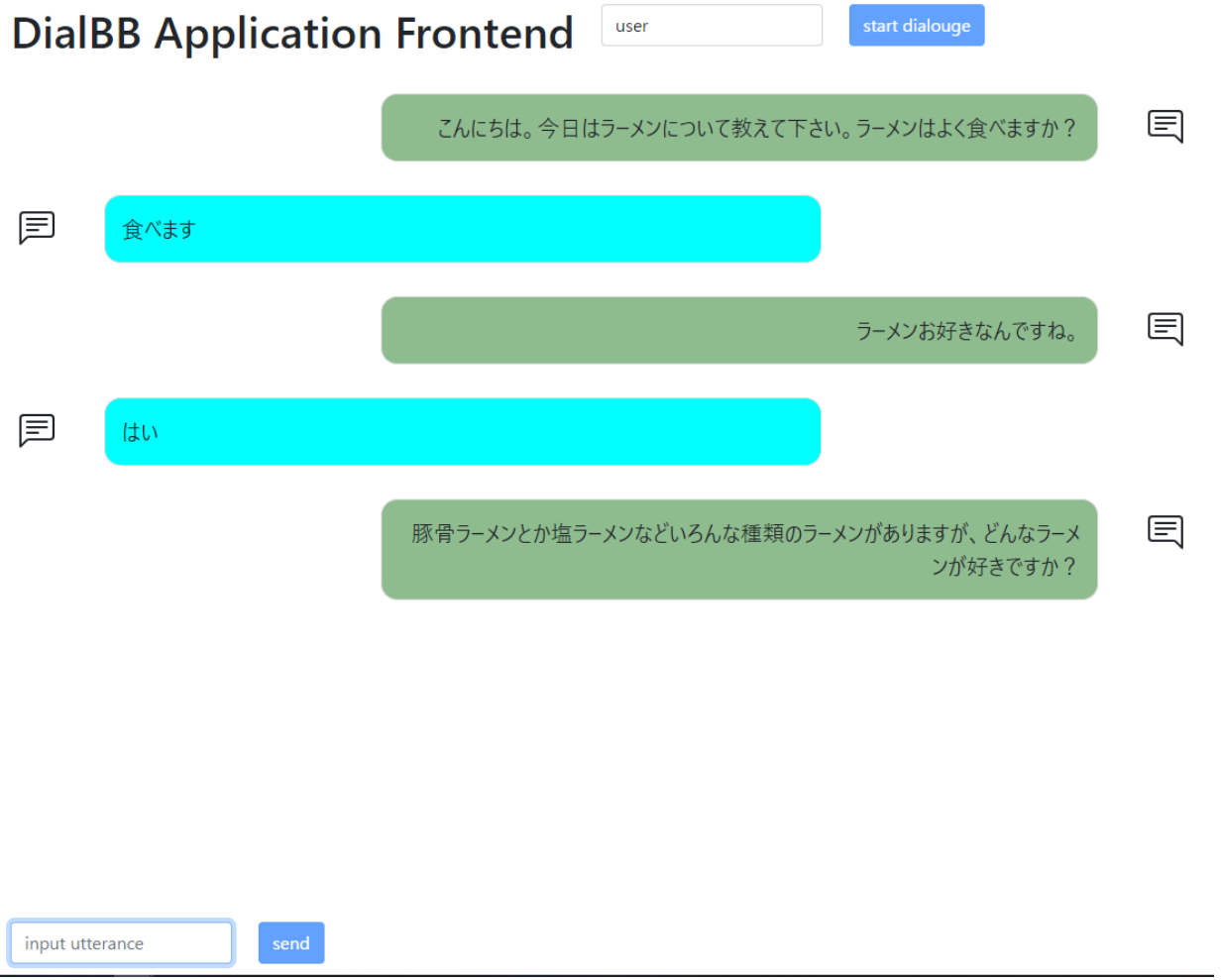
C4A Computing for All

- メインモジュールでblackboardに対話履歴(dialogue_history)を追加する
- ブロックで利用できる

```
[
 {
  "speaker": "system",
  "aux_data": <出力のaux_data. aux_dataが出力にない場合は{}>,
  "utterance": <システム発話文字列>
 },
 {
   "speaker": "user",
   "user_id": <入力のuser_id. user_idが入力にない場合は””>
   "aux_data": <入力のaux_data. aux_dataが入力にない場合は{}>,
   "utterance": <ユーザ発話文字列>
 }
 ...
]
```

フレームワーク付属のフロントエンド

C4A Computing for All



補助データの入出力も行えるフロントエンド

C4A Computing for All

組み込みブロック

C4A
Computing for All

- LLM対話組み込みブロック
  - 単一プロンプトテンプレートを用いて対話する
- DST with LLMブロック
  - LLMを用いてスロット抽出（Dialogue State Tracking: DST)を行う
- STN Managerブロック
  - State-Transition Network (STN)ベースの対話管理と発話生成を行う
- Passage Retrievalブロック
  - RAG（Retrieval-Augmented Generation）用に文書検索を行う

サンプルアプリケーション

C4A
Computing for All

- 組み込みブロックのみを用いたアプリケーションが付属
  - LLM対話アプリケーション（sample_apps/llm_dialogue_ja）
  - DST＋STNアプリケーション (sample_apps/dst_stn_ja)
  - RAGアプリケーション(sample_apps/rag_ja)

サンプルアプリと組み込みブロック（１）

LLM対話アプリケーション

概要

C4A
Computing for All

- sample_appsのllm_dialogue_{ja, en}にある
- LLM対話組み込みブロックを利用
- 単一プロンプトテンプレートを用いてLLMを呼び出す
- LLMのAPIキーが必要で、少額だがお金がかかる

LLM対話ブロック

C4A
Computing for All

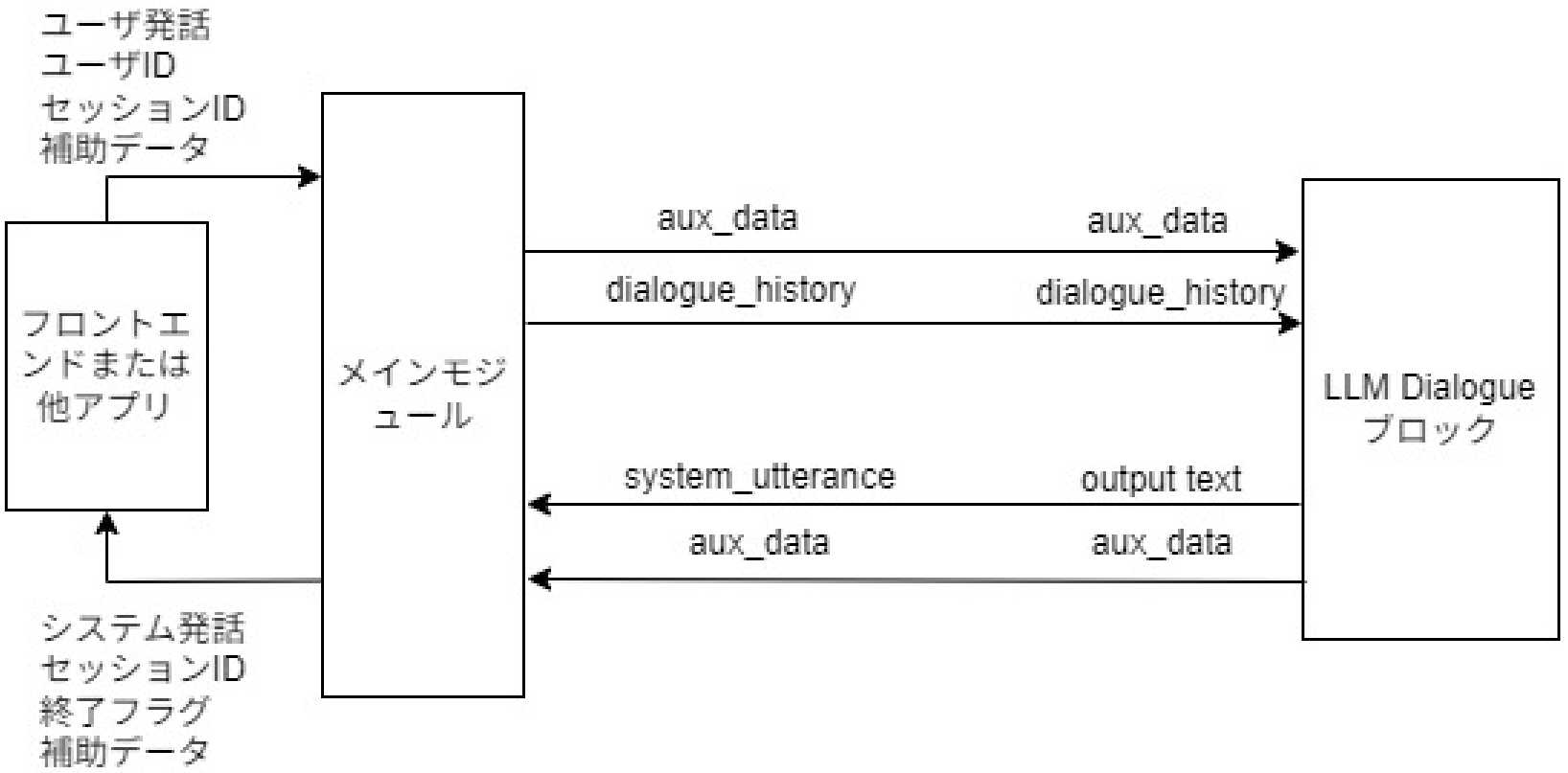
- 入力
  - 対話履歴
  - 補助データ (aux_data)
- 出力
  - システム発話
  - 補助データ (aux_data)
- 処理
  - プロンプトテンプレートに対話履歴を追加してLLMにシステム発話を生成させる

コンフィギュレーション

C4A
Computing for All

```
blocks:
  - name: llm_dialogue
    block_class: dialbb.builtin_blocks.llm_dialogue.llm_dialogue.LLMDialogue
    input:
      dialogue_history: dialogue_history
      aux_data: aux_data
    output:
      system_utterance: system_utterance
      aux_data: aux_data
      final: final
    user_name: ユーザ
    system_name: システム
    first_system_utterance: "こんにちは。私の名前は由衣。少しお話させてね。スイーツって好き？"
    prompt_template: prompt_template.txt
    model: gpt-4o-mini
    temperature: 0.7
```

アーキテクチャ



プロンプトテンプレート

C4A
Computing for All

```
# タスク説明

- あなたは対話システムで、ユーザと食べ物に関して雑談をしています。
あなたの次の発話を50文字以内で生成してください。

# あなたのペルソナ

- 名前は由衣
- 28歳
- 女性
- スイーツ全般が好き
- お酒は飲まない
- IT会社のwebデザイナー
- 独身
- 非常にフレンドリーに話す
- 外交的で陽気

#  状況

- ユーザとは初対面
- ユーザは同年代
- ユーザとは親しい感じで話す
```

```
# 対話の流れ

- 自己紹介する
- 自分がスイーツが好きと伝える
- スイーツが好きかどうか聞く
- ユーザがスイーツが好きな場合、どんなスイーツが好きか聞く
- ユーザがスイーツが好きでない場合、なんで好きじゃないのか聞く

# 注意事項

- あなたの名前や「ユーザ」を発話の先頭に入れないでください。
- 「」はつけないでください。
[[[
{notes}
]]]
```

[[[
{notes}
]]]
の部分は、入力のaux_dataのnotesの値で置き換えられる。
notesがない場合は、この部分は消去される

対話例

C4A
Computing for All

System: こんにちは。私の名前は由衣。少しお話させてね。スイーツって好き？
User: こんにちは、由衣さん！スイーツ大好きです！特に素朴なお菓子が好きなんですよ。あなたは？
System: 私もスイーツ大好き！素朴なお菓子、具体的には何が好き？
User: おからドーナツやかき餅が大好きです！手作りするのも楽しいですよ。由衣さんは何か特別なお菓子ありますか？
System: 手作りいいですね！私はチーズケーキが特別好きです。クリーミーで幸せな気持ちになります！
User: チーズケーキ、いいですね！クリーミーで贅沢感がありますよね。家族で作ることもありますか？
System: 家族で作るのも楽しそうですね！私はいつも一人で作ってますよ。
User: 一人で作るのも良いですね！自分のペースで楽しめますし、特別な時間ですよね。どんなレシピ使っていますか？
System: 私はシンプルなレシピが好きで、クリームチーズと砂糖を混ぜるだけ！あなたは？
User: シンプルなレシピ、最高ですね！私は、少しアレンジしてフルーツをトッピングすることが多いです。美味しさが増しますよ！
System: フルーツトッピング、いいアイデアですね！どんなフルーツが好きですか？
User: 私はりんごやブルーベリーが好きです！爽やかさが加わって、とても美味しくなりますよ。由衣さんはどんなフルーツが好きですか？
System: 私はいちごやマンゴーが好き！甘酸っぱくて、スイーツにぴったりです！
User: いちごやマンゴー、最高ですね！甘酸っぱさがアクセントになって美味しいですよね。特にいちごは見た目も華やかで素敵です！
System: いちごは本当に可愛くて、見てるだけでも幸せになりますよね！他に好きなスイーツはありますか？
User: はい、やっぱり青森の「やっこいサブレ」が好きです！キャラメルがサンドされていて、素朴な味わいが最高なんですよ。試してみたいですか？
System: やっこいサブレ、美味しそう！キャラメルの組み合わせ、絶対好きになりそう！どこで買えるの？
User: ベニーマートで買えるんです！青森の特産品なので、見かけたらぜひ試してみてください。きっと気に入りますよ！
System: ベニーマート、チェックしてみますね！青森の特産品、楽しみです！他にもオススメはありますか？
User: 青森の特産品では、リンゴのお菓子もおすすめです！リンゴのパイや、ジャムも絶品ですよ。自然な甘さが楽しめます！
System: リンゴのお菓子、いいですね！ナチュラルな甘さが大好きです。特にどれが一番好きですか？

サンプルアプリと組み込みブロック（2）

DST+STNアプリケーション

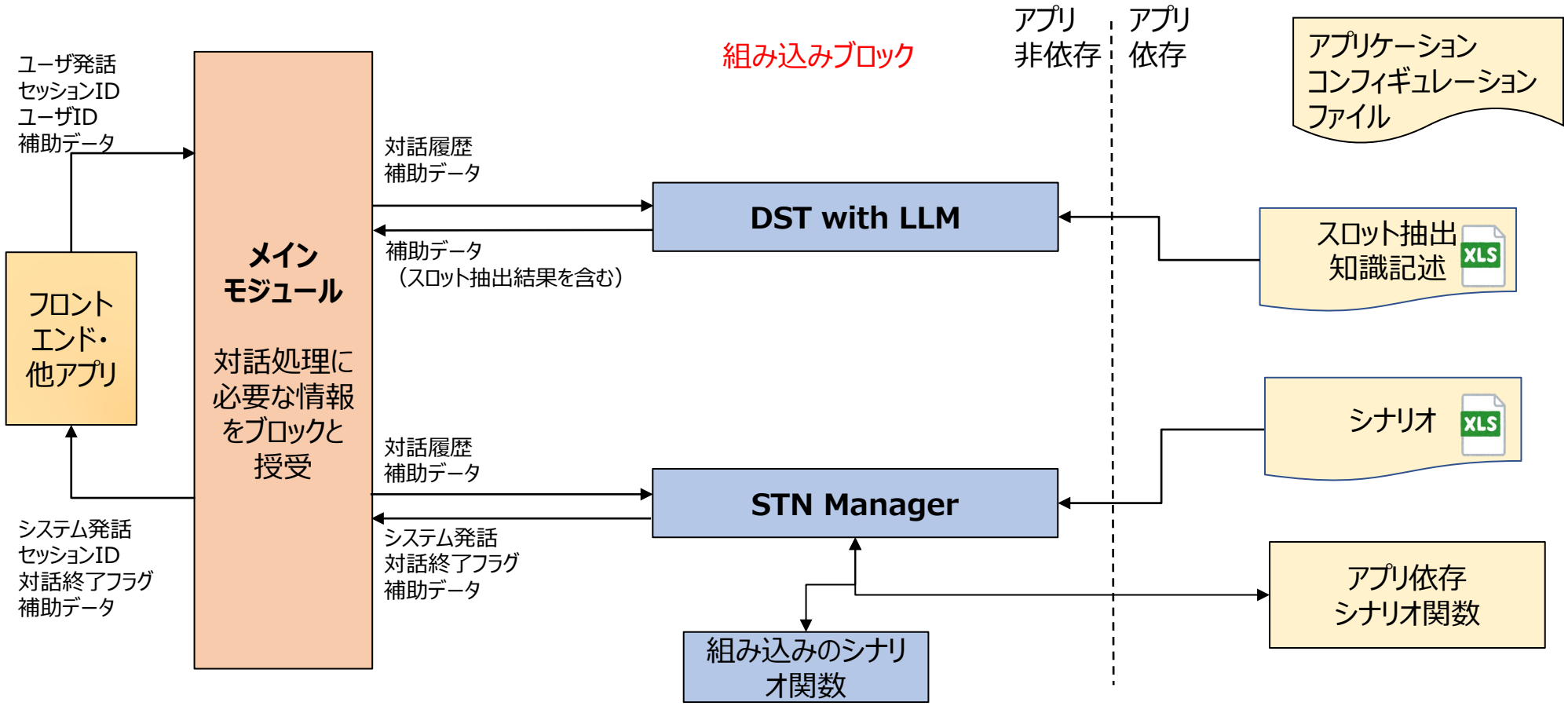
概要

C4A
Computing for All

- sample_appsのdst_stn_{ja, en}にある
- DST with LLMブロックとSTN Managerブロックを利用
- 状態遷移モデルベースの対話管理
- スロット抽出結果を利用する
- 状態遷移の条件判定やシステム発話の生成にLLMを利用する場合がある

DST+STNアプリケーションの構成

C4A
Computing for All



DST with LLM組み込みブロック

C4A Computing for All

- 対話履歴からスロットを抽出
- 入力
  - 対話履歴
  - 補助データ
- 出力
  - 補助データ（スロット抽出結果を含む）

システム：好きなラーメン教えて
ユーザ：豚骨ラーメン
システム：博多の？
ユーザ：そう

aux_data['好きなラーメン']='豚骨ラーメン'
aux_data['地方']='博多'

スロット抽出知識

C4A Computing for All

dialoguesシート: 対話とスロット抽出結果の例

flag	dialogue	slots
Y	システム：好きなラーメンは何ですか？ ユーザ：豚骨ラーメンです	好きなラーメン=豚骨ラーメン
Y	システム：好きなラーメン教えて ユーザ：豚骨ラーメン システム：博多の？ ユーザ：そう	地方=博多，好きなラーメン=豚骨ラーメン

slotsシート：各スロットの値の例とその同義語

flag	slot name	entity	synonyms
Y	好きなラーメン	豚骨ラーメン	とんこつラーメン，豚骨スープのラーメン，豚骨
Y	好きなラーメン	味噌ラーメン	みそらーめん，みそ味のラーメン，味噌

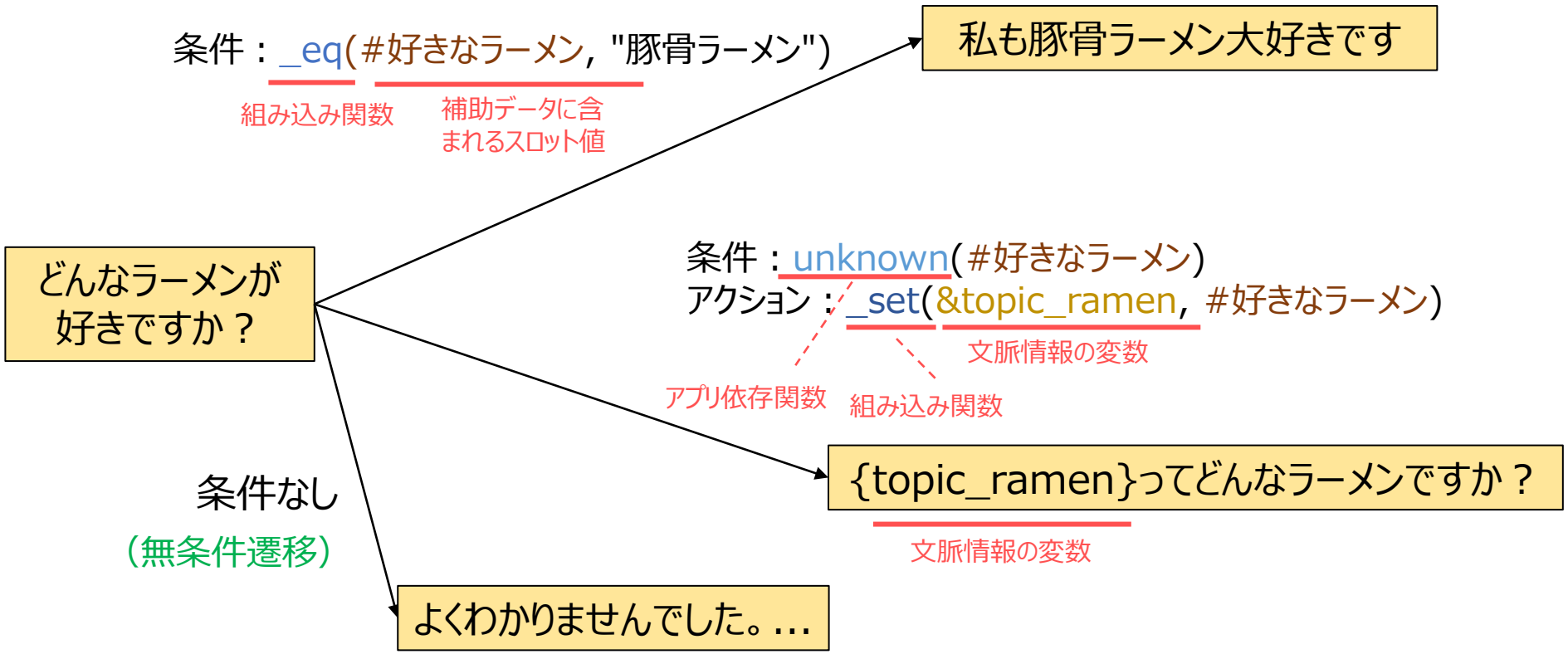
どのflagの行を使うかはコンフィギュレーションで指定可能

STN Manager組み込みブロック

C4A
Computing for All

- STN manager
  - 状態遷移ネットワーク(State-Transition Network)を用いた対話管理＋言語生成
    - 遷移の条件や遷移時のアクションを関数呼び出しで記述（シナリオ関数）
    - 組み込み関数を用意
  - Excel/Google Sheetでシナリオを記述
- 入力
  - 対話履歴
  - 補助データ
- 出力
  - システム発話文字列
  - final: 対話終了かどうかのフラグ（ブール値）
  - 補助データ

状態遷移ネットワークによる対話管理



文脈情報：対話の中で得られた情報や外部知識から得られた情報を保持するもの

シンタクスシュガー

C4A
Computing for All

- 組み込み関数を簡単に書ける
  - 条件関数
    - _eq(y, "aa") → y=="aa"
    - _ne(y, "aa") → y!="aa"
  - アクション関数
    - _set(y, "aa") → y="aa"

スプレッドシートによるシナリオ記述

C4A Computing for All

遷移

flag	state	system utterance	user utterance example	conditions	actions	next state
Y	好き	豚骨ラーメンとか塩ラーメンなどいろんな種類のラーメンがありますが，どんなラーメンが好きですか？	豚骨ラーメンが好きです.	_eq(#好きなラーメン, "豚骨ラーメン")	_set(&topic_ramen, #好きなラーメン)	豚骨ラーメンが好き
Y	好き		豚骨ラーメンが好きです.	is_known_ramen(#好きなラーメン)	_set(&topic_ramen, #好きなラーメン); get_ramen_location (*topic_ramen, &location)	特定のラーメンが好き
Y	好き			is_novel_ramen(#好きなラーメン)	_set(&topic_ramen, #好きなラーメン)	知らないラーメンが好き
Y	好き		近所の街中華のラーメンが好きなんだよね			#final

同じstateの行は上から順に条件判定を行う

system utteranceカラムは、stateカラムとだけ結びついていて、遷移とは関係ない

STN Manager: システム発話中の関数呼び出し・特殊変数参照

C4A Computing for All

- 関数呼び出しを埋め込める
  - 生成関数と呼ぶ

私は{get_system_name()}です。

- 特殊変数を埋め込める
  - スロット値
  - 固有表現

{#地方}ですね。

こんにちは {#NE_人名}さん。

STN Manager: LLMを用いた発話生成（１）

C4A Computing for All

- 組み込み生成関数_generate_with_llm(<タスク文字列>)を使う
- シンタクスシュガー $<タスク文字列>$ （前後の{}は不要）
- システム発話の中に関数呼び出しを埋め込む
- LLMのプロンプトに含まれるもの
  - システムペルソナ（コンフィギュレーションで指定）
  - 対話の状況（コンフィギュレーションで指定）
  - 対話の履歴
  - タスク（関数の引数）

利用例：システム発話

わかりました。{_generate_with_llm("それまでの会話につづけて、対話を終わらせる発話を50文字以内で生成してください。")}今日はお時間ありがとうございました。

わかりました。$ それまでの会話につづけて、対話を終わらせる発話を50文字以内で生成してください。$ 今日はお時間ありがとうございました。

STN Manager: LLMを用いた発話生成（２）

C4A
Computing for All

コンフィギュレーションの記述

```
llm:
    model: gpt-4o-mini
    temperature: 0.7
    situation:
      - あなたは対話システムで、ユーザと食べ物に関して雑談をしています。
      - ユーザとは初対面です
      - ユーザとは同年代です
      - ユーザとは親しい感じで話します
    persona:
      - 名前は由衣
      - 28歳
      - 女性
      - ラーメン全般が好き
      - お酒は飲まない
      - IT会社のwebデザイナー
```

生成発話例

わかりました。ラーメン好きなんですね。次は函館の塩ラーメンをおすすめします。美味しいですよ。楽しいおしゃべりありがとうございました。また話しましょうね。今日はお時間ありがとうございました。

STN Manager: LLMを用いた条件判定

C4A
Computing for All

- 組み込み条件関数_check_with_llm(<タスク文字列>)を使う
- シンタクスシュガー $<タスク文字列>$
- LLMのプロンプトに含まれるもの
  - 対話の状況、システムペルソナ、注意事項（コンフィギュレーションで指定）
  - 対話の履歴
  - タスク（関数の引数）

利用例

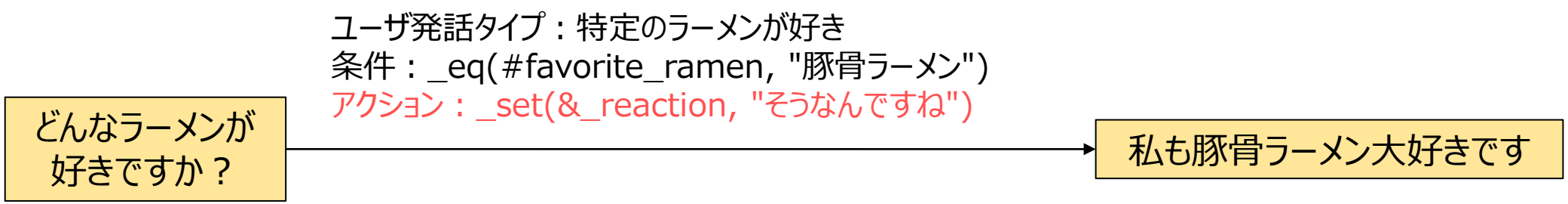
```
_check_with_llm("ユーザが理由を言ったかどうか判断してください。")
```

```
$ユーザが理由を言ったかどうか判断してください。$
```

STN Manager:リアクション発話生成

C4A Computing for All

- 変数"_reaction"に設定されたシステム発話を次の状態のシステム発話の前に発話



システム：どんなラーメンがすきですか？
ユーザ：豚骨ラーメンが好きです
システム：そうなんですね 私も豚骨ラーメン大好きです。

システム質問⇒ユーザ応答⇒システムリアクション という連鎖を書くのに便利

その他のSTN Managerの機能

C4A Computing for All

- 音声入力に対処するための機能
  - 入力の音声認識確信度(入力のaux_dataのconfidenceの値）が閾値より低い場合に確認要求発話を行う
  - システム発話終了後、一定時間ユーザが話さなかった場合（入力のaux_dataのlong_silenceの値がTrueの場合）、前の発話を繰り返す
    - 具体的な動作はconfigurationで変更可能
- スキップ遷移：システム発話を行わずに次の遷移を行う
  - アクション関数の実行結果に応じてさらに分岐させたいときに使う
- サブダイアローグ
  - 確認対話など、よく使う対話シーケンスを別のネットワークとして記述しておき再利用する（サブルーチンのようなもの）

詳細はドキュメントを参照

対話例

C4A
Computing for All

System: こんにちは。私はラーメンボットです。よろしければお名前を教えて頂けますか?
User: 健太です。よろしくお願いします。ラーメンボットさんは、どんなラーメンが好きですか？
System: ありがとうございます。健太さん、今日はラーメンについて教えて下さい。ラーメンはよく食べますか?
User: はい、よく食べます。特にあっさりした醤油ラーメンが好きです。
System: ラーメンお好きなんですね。
User: はい、好きです。あっさりした味わいがたまりません。
System: そうなんですね。 豚骨ラーメンとか塩ラーメンなどいろんな種類のラーメンがありますが、どんなラーメンが好きですか?
User: あっさりした醤油ラーメンが一番好きです。豚骨はちょっと苦手です。
System: 醤油ラーメンですね。東京にはよく行きますか?
User: あまり行きません。東京では醤油ラーメンが有名ですよね。
System: わかりました。
そうですね！東京の醤油ラーメンは美味しいですよね。機会があればぜひ！
今日はお時間ありがとうございました。

サンプルアプリと組み込みブロック（3）

RAGアプリケーション

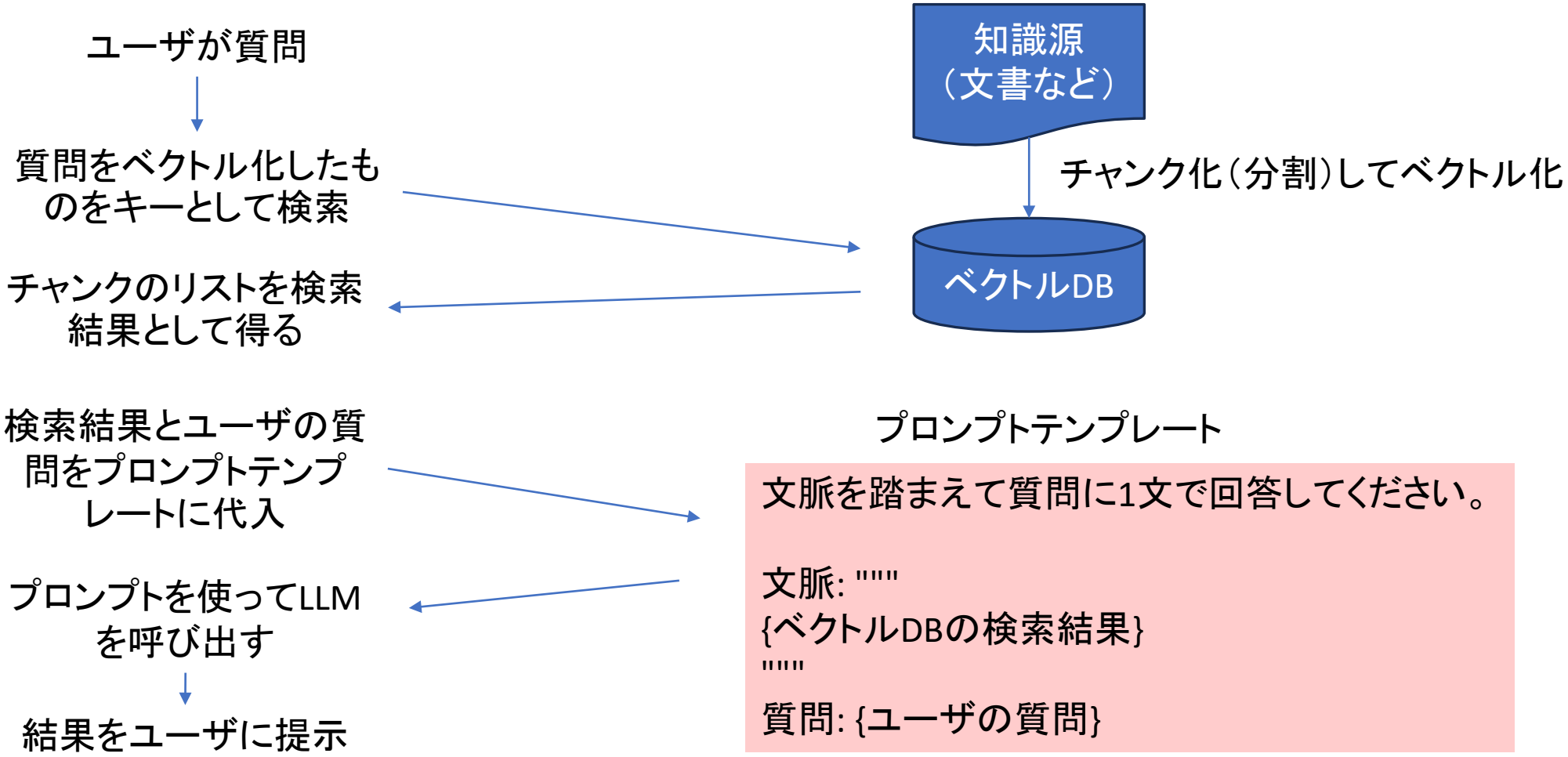
RAGとは？

- Retrieval Augmented Generation
  - 検索拡張生成
- RAGが出てきた理由
  - ChatGPTのようなLLMは訓練データにある知識を内在しているが、それだけでは不十分
    - 最新の情報がない
    - 企業内の情報などオープンになっていない情報がない
    - 正確性に欠ける

  → 外部知識を与える必要

RAGの基本的な方法

C4A
Computing for All



RAGアプリケーションの概要

C4A
Computing for All

- sample_appsのrag_ja, rag_enにある
- Passage検索ブロック、LLM対話組み込みブロックを利用
- Passage検索ブロックは、検索結果を補助データ(aux_data)の、"passages"の値に入れる

コンフィギュレーション

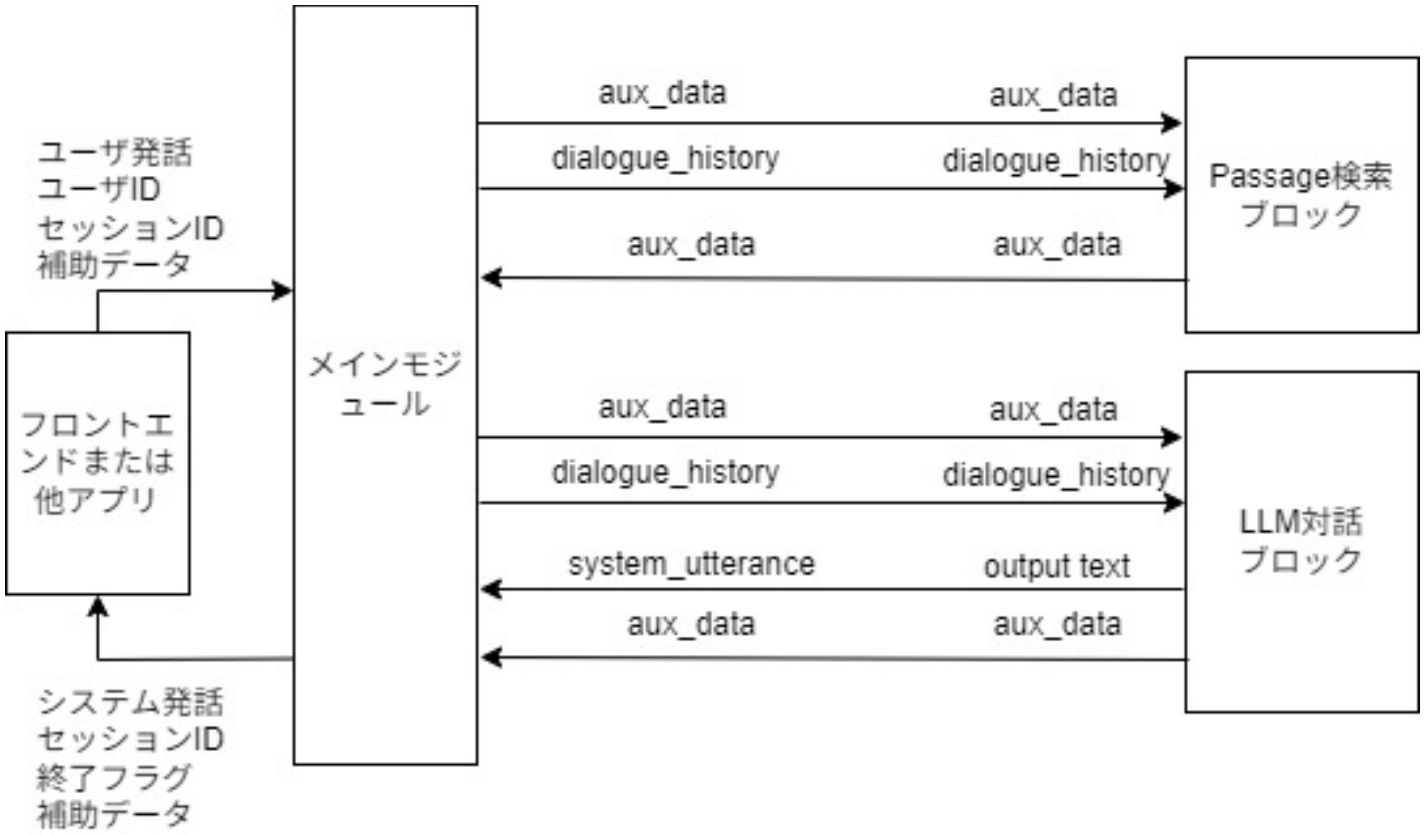
C4A
Computing for All

```
language: ja

blocks:
  - name: passage_retrieval
    block_class:
dialbb.builtin_blocks.passage_retrieval.Retriever
    input:
      dialogue_history: dialogue_history
      aux_data: aux_data
    output:
      aux_data: aux_data
    # vector_db_dir: vector_db コメントはデフォルト値
    # collection: rag_docs
    # chunk_size: 800
    # chunk_overlap: 100
    # top_k: 5
    # clear_before_ingest: True
    # seperator: "\n\n---\n\n"
    extensions:
      - ".md"
    sources:
      - docs
```

```
- name: llm_dialogue
    block_class:
dialbb.builtin_blocks.llm_dialogue.LLMDialogue
    input:
      dialogue_history: dialogue_history
      aux_data: aux_data
    output:
      system_utterance: system_utterance
      aux_data: aux_data
      final: final
    user_name: ユーザ
    system_name: システム
    first_system_utterance: "こちらはブルーオーシャンレン
タカーです。どのようなご用件でしょうか？"
    prompt_template: prompt_template.txt
    model: gpt-4o-mini
    temperature: 0.7
```

アーキテクチャ



検索対象文書

C4A
Computing for All

ブルーオーシャンレンタカー FAQ

予約について

Q. 予約はどこからできますか？
A. 公式Webサイト、電話、営業所窓口から予約できます。

Q. 当日予約はできますか？
A. 空き車両がある場合は当日でも予約できます。

Q. 何日前から予約できますか？
A. 最大6か月前から予約できます。

Q. 予約なしで営業所に行っても借りられますか？
A. 空き車両があれば可能ですが、事前予約をおすすめします。

Q. 予約内容は変更できますか？
A. 利用開始前であれば、車種・日時・営業所などを変更できます。

Q. 予約の変更に手数料はかかりますか？
A. 利用開始24時間前までの変更は無料です。

Q. 予約をキャンセルできますか？
A. はい、Webサイトまたは営業所へ連絡することでキャンセルできます。

Q. キャンセル料はかかりますか？
A. 利用開始24時間前以降のキャンセルにはキャンセル料が発生します。

Q. 複数台をまとめて予約できますか？
A. はい、法人・団体利用として複数台予約が可能です。

Q. 予約確認メールが届きません。どうすればよいですか？
A. 迷惑メールフォルダを確認のうえ、届いていない場合は営業所へお問い合わせください。

利用資格について

Q. レンタカーを借りるには何が必要ですか？
A. 有効な運転免許証、本人確認書類、支払い手段が必要です。

Q. 免許を取得したばかりでも借りられますか？
A. はい、借りられます。ただし一部車種は免許取得後1年以上が条件です。

Q. 20歳未満でも利用できますか？
A. 18歳以上で有効な免許証があれば利用できます。

.....

検索結果の例

C4A
Computing for All

ユーザ発話

運転免許証は必須ですよね？他に必要なものはありますか？

検索結果

'Q. 追加運転者にも免許証は必要ですか? A. はい、運転される方全員の有効な免許証が必要です。

Q. レンタカーを借りるには何が必要ですか? A. 有効な運転免許証、本人確認書類、支払い手段が必要です。

Q. 出発時に必要な手続きは何ですか? A. 免許証確認、契約内容確認、車両状態確認を行います。

Q. 学生でも借りられますか? A. はい、運転免許証と本人確認書類があれば利用できます。

Q. 外国の運転免許証だけで借りられますか? A. 一部の国・地域の免許証は日本語翻訳文と合わせて利用できます。

LLM対話ブロックで使うプロンプトテンプレート

C4A Computing for All

```
# タスク説明

- あなたはブルーオーシャンレンタカーの電話応対係です。以下の情報をもとに、お客様の質問に簡潔に答えてください。以下の情報に書いてないことは、「申し訳ありません、今はお答えできません」と言って下さい。

# 元となる情報

{passages}
```

ここに検索結果が入る

対話例

C4A
Computing for All

System: こちらはブルーオーシャンレンタカーです。どのようなご用件でしょうか？
User: レンタカーの料金と保険について教えてください。
System: 申し訳ありませんが、具体的な料金や保険の詳細についてはお答えできません。料金は車両クラス、利用時間、オプション、補償内容によって決まりますので、詳しい情報については直接お問い合わせいただければと思います。他に何かお手伝いできることがあれば教えてください。
User: 具体的な車両の種類やオプションは教えていただけますか？
System: システム: はい、当社の車種には軽自動車、コンパクトカー、セダン、ミニバン、SUV、商用バンなどがあります。オプションについては申し訳ありませんが、今はお答えできません。他に何かお手伝いできることがあれば教えてください。
User: ミニバンのサイズや人数制限はありますか？
System: 申し訳ありませんが、ミニバンのサイズや人数制限についてはお答えできません。他に何かお手伝いできることがあれば教えてください。
User: 予約の際に必要な書類は何ですか？
System: システム: 予約の際に必要な書類については、申し訳ありませんが、今はお答えできません。他に何かお手伝いできることがあれば教えてください。
User: 運転免許証は必須ですよね？他に必要なものはありますか？
System: はい、運転免許証は必須です。他には本人確認書類と支払い手段が必要です。何か他にお手伝いできることがあれば教えてください。

補足

DialBBで行えるテスト

C4A
Computing for All

- テストシナリオを用いたテスト
  - ユーザ発話文字列を順に入力
- テストリクエストを用いたテスト
  - 補助データも含めたJSONを順に入力
- ユーザシミュレータを用いたテスト
  - LLMを用いて文脈にあったユーザ発話を生成

ドキュメント・README参照

他のアプリケーション

C4A
Computing for All

- DialBBのアプリケーションを集めたリポジトリも公開
  - https://github.com/c4a-ri/dialbb-apps

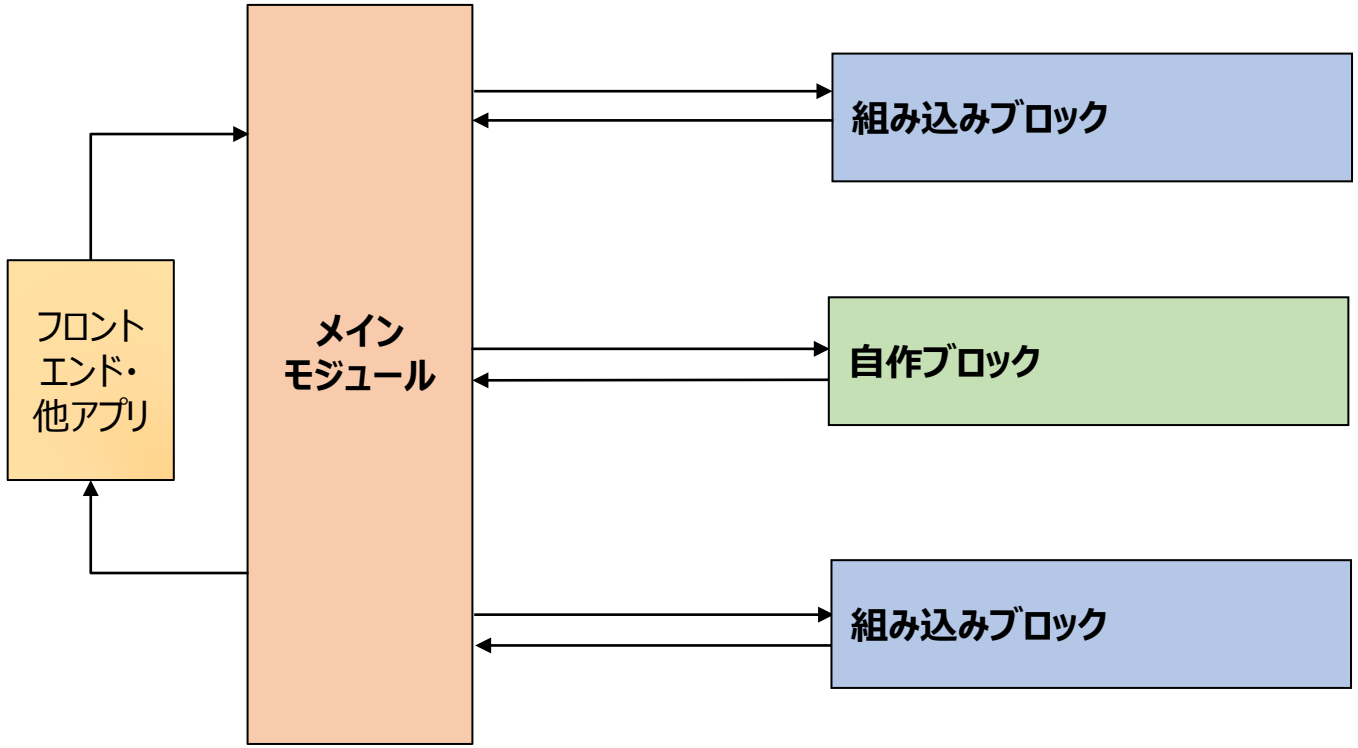
新規アプリケーション開発

一般的なアプリ作成ステップ

C4A
Computing for All

- 組み込みブロックのみを用いたシステム
  - サンプルアプリをディレクトリごとコピーして編集
  - 知識記述を変更
    - 対話シナリオ
    - スロット抽出知識
    - プロンプト
    - シナリオ関数
  - 必要に応じてコンフィギュレーションを変更
- 自作ブロックを組み込んだシステム

自作ブロックの利用



自作ブロックのクラス名をコンフィギュレーションファイルで指定する

Tips (1)

C4A
Computing for All

- シナリオ関数を使うことで外部知識にアクセスできる
  - 外部API
  - データベースへのアクセス
- 入力の補助データ（aux_data）を利用することでマルチモーダル入力などにも対応できる
  - 音声認識の確信度
  - 感情・態度認識結果
  - 年齢・性別推定結果

Tips (2)

C4A
Computing for All

- 出力のaux_dataを活用することで音声以外の情報を出力できる
  - ロボットのジェスチャーなど

- 極力記述をわかりやすくするのが良い
  - 例：シナリオ関数を使えばどんな複雑な処理も書けるが、できる限りSpreadsheetやコンフィギュレーションに書いた方がよい

DialBBのソースコードの変更や、動作の確認を行う方法

C4A
Computing for All

- uvをインストール
- GitHubからDialBBをクローン
  `git clone git@github.com:c4a-ri/dialbb.git <DialBBのディレクトリ>`
- DialBBのディレクトリに移動
  `cd <DialBBのディレクトリ>`
- 環境変数を<DialBBのディレクトリ>/.envに記述
- 必要ならソースコードを変更
- 実行
  `uv run dialbb-server <コンフィギュレーションファイル>`

今後の
改良予定

短期プラン

C4A
Computing for All

- 組み込みブロックの追加
  - 相づち生成ブロックなど
- 音声・マルチモーダル入力対応
- デプロイ・運用のためのツール・ドキュメントの整備
- ノーコードツールでRAGをつかえるようにする
- アプリ例の増加
  - tool callingやMCPの利用例

長期プラン

C4A
Computing for All

- 使ってくれる人の希望に合わせて改良
  - 初期アプリを作ることも可能
  - 研究アドバイスも可能
- Discordサーバ、GitHubのDiscussionsで要望・提案を募集
  - Discordサーバ招待リンク: https://discord.gg/spk2FCe2ub