C4A
Computing for All

対話システム構築フレームワーク DialBB紹介

v0.6対応

中野 幹生

株式会社C4A研究所

Outline

C4A
Computing for All

- DialBBの概要
- サンプルアプリケーションと組み込みブロック
- DialBBを用いた対話システム開発
- 今後の改良予定

DialBBの概要

C4A
Computing for All

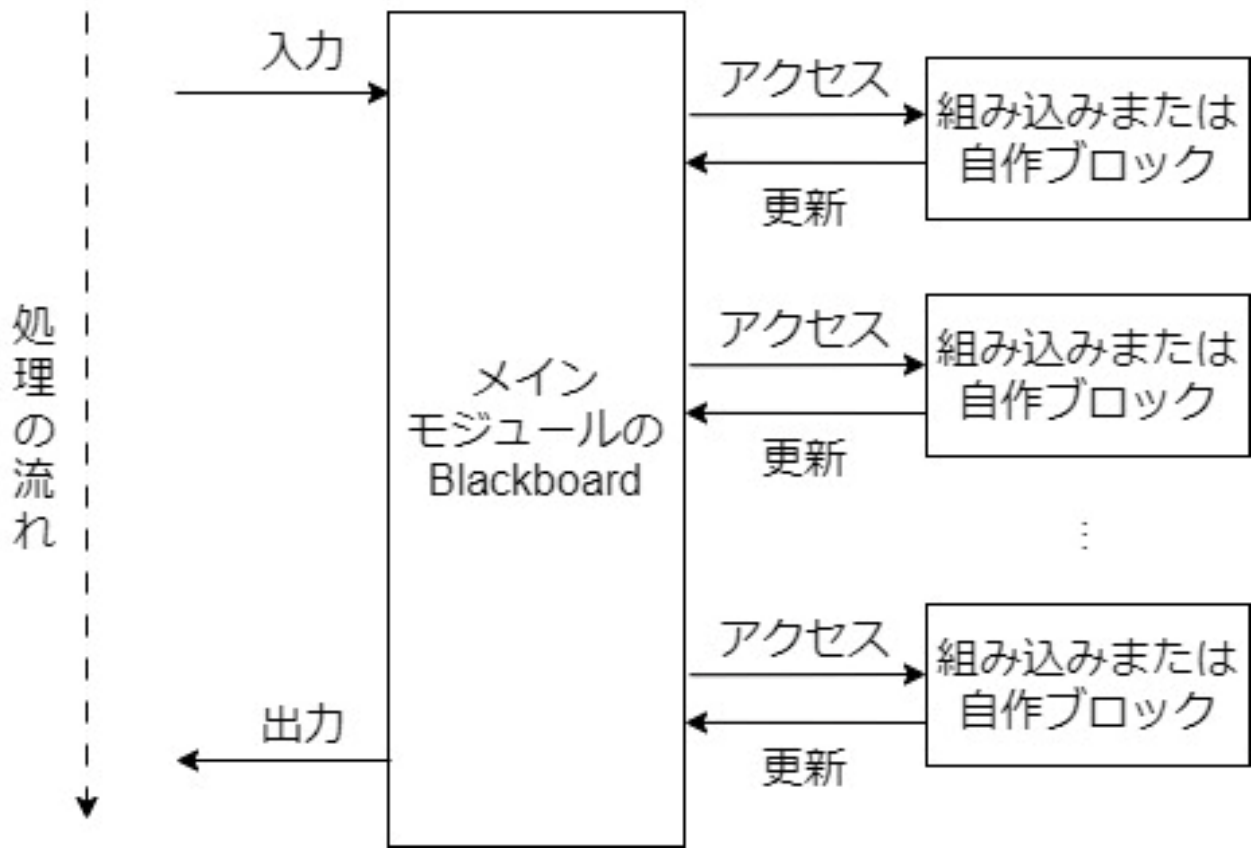
DialBBとは

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

DialBBアプリケーションの構成

C4A
Computing for All



DialBBの特長

C4A
Computing for All

- 組み込みブロックだけでアプリを構築可能
  - Snipsによる言語理解＋状態遷移ネットワークによる対話管理
    - 東中他「Pythonでつくる対話システム」の2.2節、3.2節で説明されている技術に対応
  - ChatGPT
- サンプルアプリが付属
- Pythonで書かれている
- ソースを公開・非商用利用向けにライセンス
  https://github.com/c4a-ri/dialbb
- 対話システムのみならずシステム構築のお手本となることを目指す

サンプルアプリケーションと組み込みブロック

C4A
Computing for All

サンプルアプリケーション

C4A
Computing for All

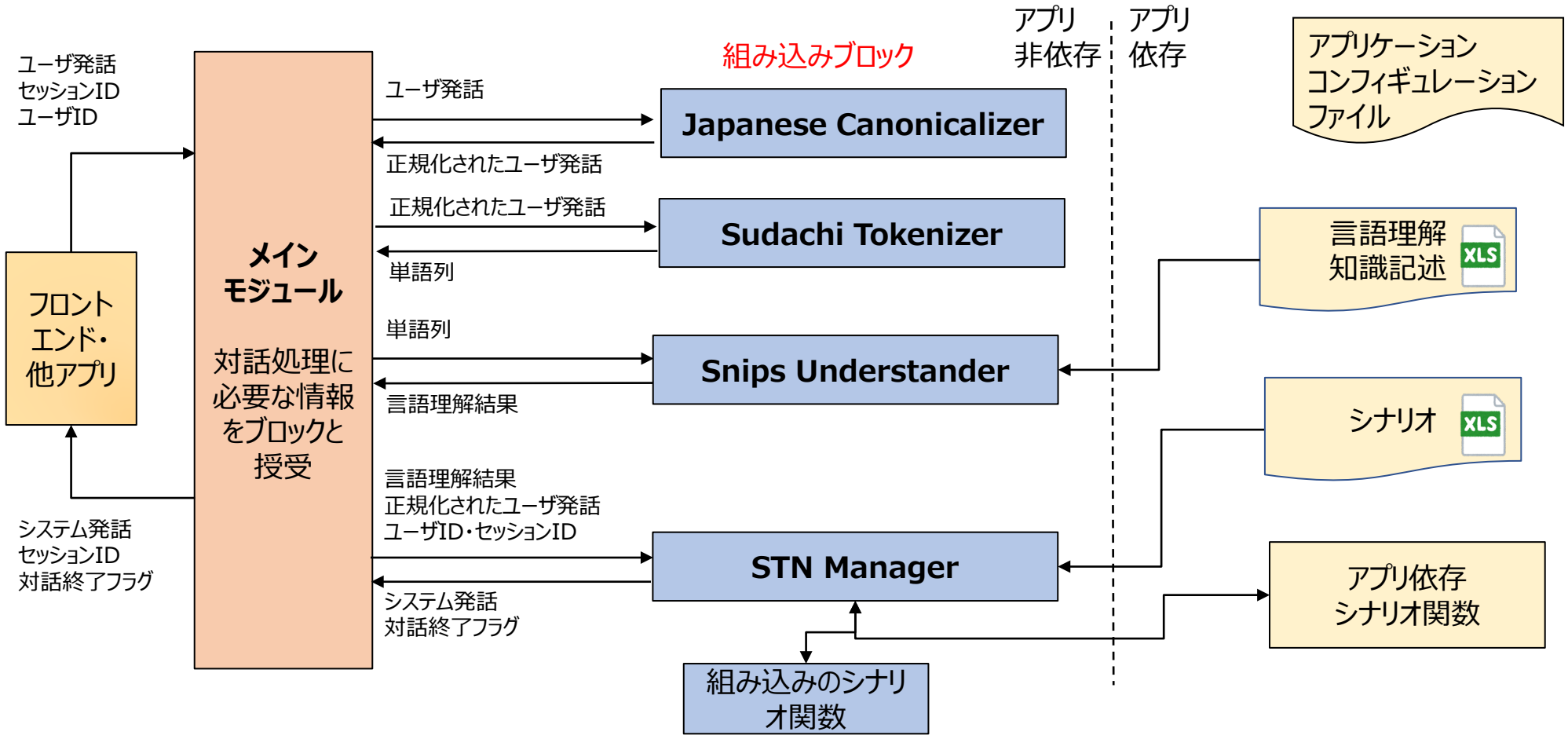
- 組み込みブロックのみを用いたアプリケーション
  - シンプルアプリケーション (sample_apps/network_ja)
    - Sudachiによる言語理解
    - Snipsによる言語理解
    - 状態遷移ネットワークによる対話管理
  - 実験アプリケーション(sample_apps/lab_app_ja)
    - シンプルアプリケーション＋GiNZAを用いた固有表現抽出
    - シンプルアプリケーションに入っていない機能を導入
  - ChatGPTのみのアプリケーション

C4A
Computing for All

シンプルアプリケーション

シンプル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>
    ….
```

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']

組み込みブロック（1）正規化

C4A Computing for All

- Japanese Canonicalizer/Simple Canonicalizer
  - ユーザ入力文の正規化（大文字→小文字，全角⇒半角の変換など）
- 入力
  - input_text: 入力文字列
- 出力
  - output_text: 入力文字列を正規化したもの

*https://github.com/WorksApplications/Sudachi

組み込みブロック（2）単語分割

C4A Computing for All

- Sudachi tokenizer/Whitespace tokenizer
  - Sudachi*や空白文字ベースで単語分割
  - Sudachi正規化も行える
- 入力:
  - input-text: 入力文字列　（例："私はラーメンが食べたい"）
- 出力
  - tokens: トークンのリスト　（例：['私','は','ラーメン','が','食べ','たい']）

*https://github.com/WorksApplications/Sudachi

組み込みブロック（3）言語理解

C4A Computing for All

- Snips Understander
  - Snips NLU **を利用した統計的言語理解
  - Excel/Google Sheetで書かれた知識を読み込む
  - 起動時にモデルを訓練
  - 辞書関数を用いることで、外部知識から辞書項目を作れる
- 入力
  - tokens: トークンのリスト（例：['好き','な','の','は','醤油']）
- 出力
  - nlu_result:言語理解結果または言語理解結果のリスト
    （例： {"type": "特定のラーメンが好き", "slots": {"favorite_ramen": "醤油ラーメン"}}）

**https://snips-nlu.readthedocs.io/en/latest/

シンプルアプリケーションの言語理解

C4A Computing for All

好きなのは醤油

```
{"type": "特定のラーメンが好き",
 "slots": {
     "favorite_ramen": "醤油ラーメン"
   }
}
```

言語理解知識

type	utterance
肯定	はい
肯定	そう
否定	そうでもない
否定	違う
好き	好きです
好き	好きだよ
好きじゃない	あまり好きじゃない
好きじゃない	そんなに好きじゃない
食べない	あまり食べない
食べない	食べません
食べる	食べすぎるくらい食べる
食べる	食べます
特定のラーメンが好き	(豚骨ラーメン)[favorite_ramen]が好きです
特定のラーメンが好き	好きなのは(味噌ラーメン)[favorite_ramen]
特定のラーメンが好き	(醤油)[favorite_ramen]かな

組み込みブロック（4）
対話管理＋言語生成

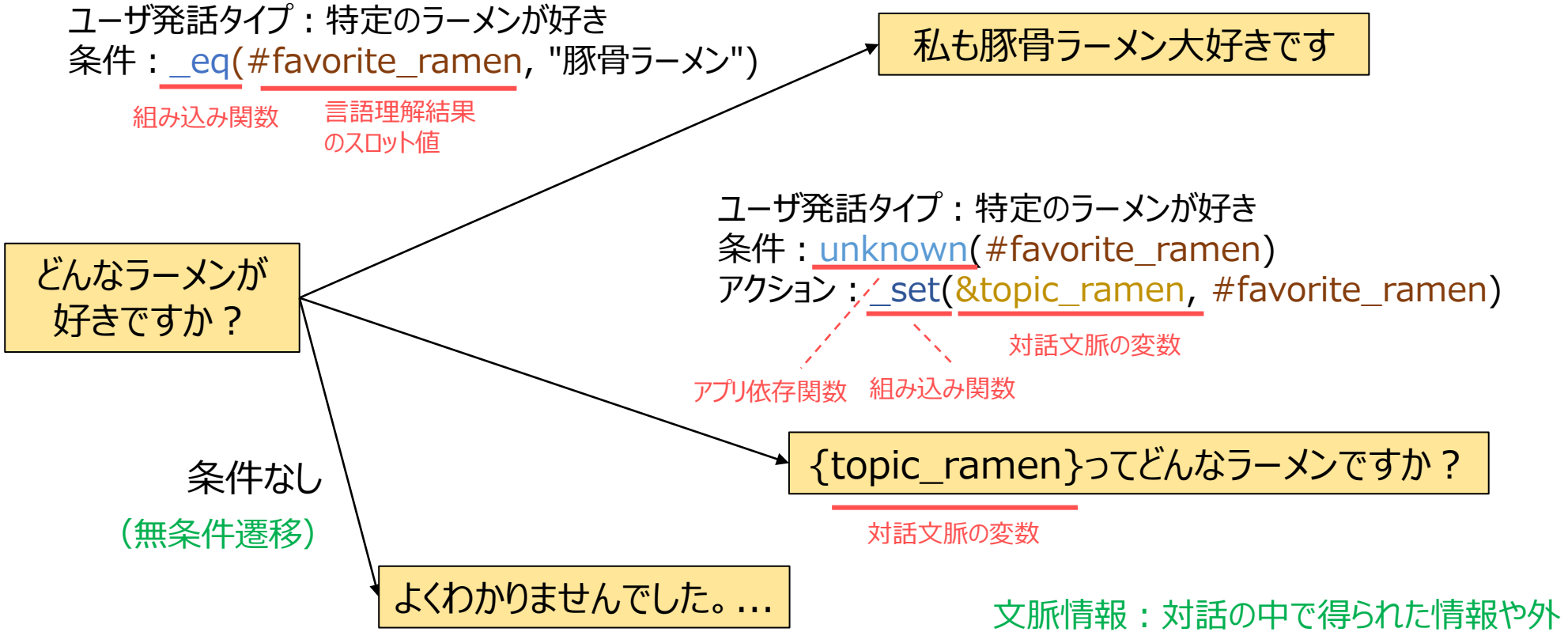
C4A
Computing for All

- 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
  - sentence: 正規化後のユーザ発話
  - nlu_result: 言語理解結果（のリスト）
  - user_id: ユーザID
  - aux_data: 補助データ
- 出力
  - output_text: 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）
  - aux_data: 入力された補助データに対話状態のIDを含めたもの

状態遷移ネットワークによる対話管理

C4A Computing for All



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

スプレッドシートによるシナリオ記述

C4A
Computing for All

遷移

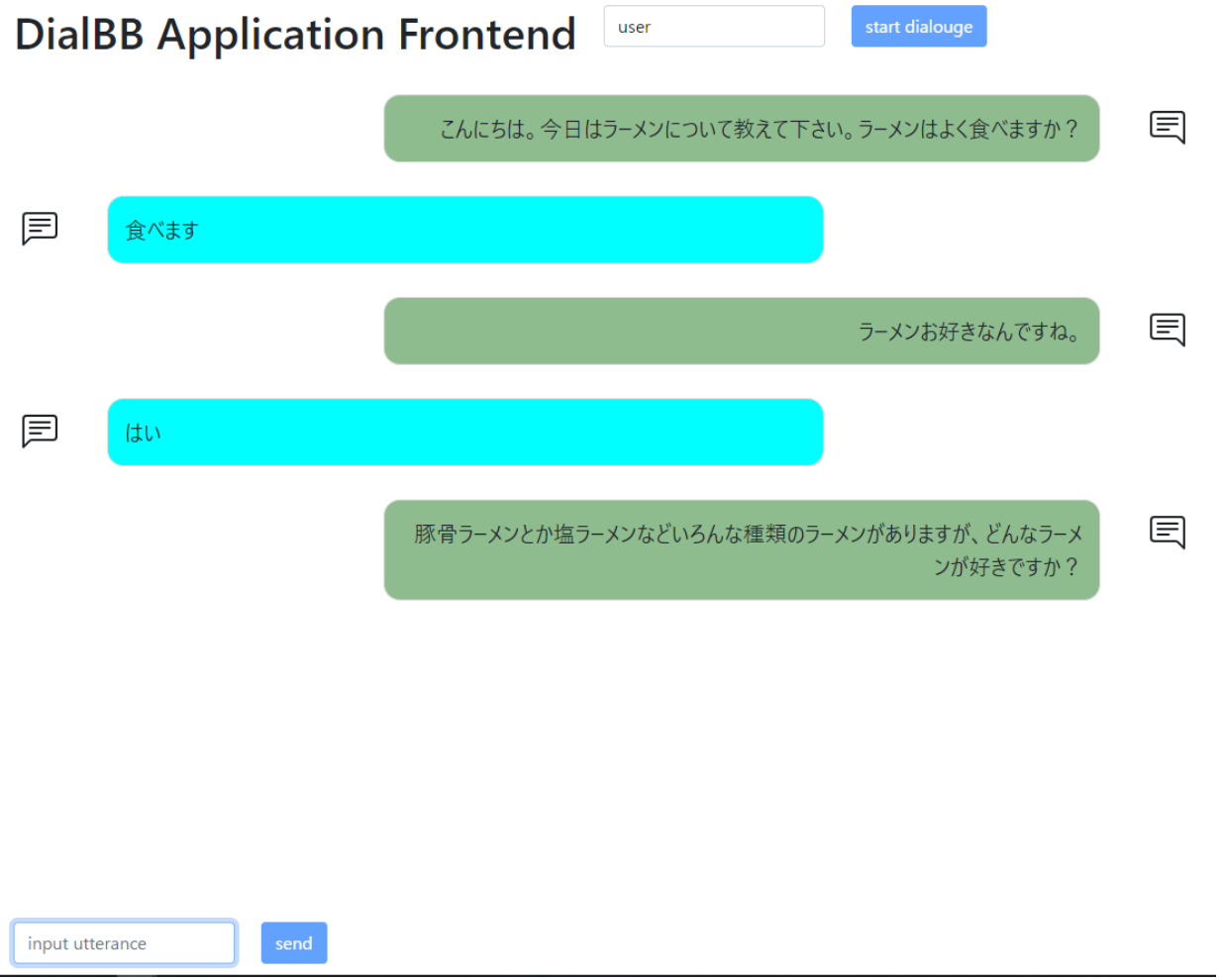
state	system utterance	user utterance type	conditions	actions	next state
#prep				decide_greeting(&greeting)	#initial
#initial	{greeting}．今日はラーメンについて教えて下さい．ラーメンはよく食べますか？	食べる			食べる
#initial		食べない			食べない
#initial		肯定			食べる
#initial					好き
食べる	ラーメンお好きなんですね．	肯定			好き
食べる		好き			好き
食べる		否定			好きじゃない
食べる		好きじゃない			好きじゃない
食べる					好き
食べない	ラーメン嫌いなんですね．なぜか教えて頂いてもいいですか？				#final
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？	特定のラーメンが好き	_eq(#favorite_ramen, "豚骨ラーメン")	_set(&topic_ramen, #favorite_ramen)	豚骨ラーメンが好き
好き		特定のラーメンが好き	is_known_ramen(#favorite_ramen)	_set(&topic_ramen, #favorite_ramen); get_ramen_location(*topic_ramen, &location)	特定のラーメンが好き
好き		特定のラーメンが好き	is_novel_ramen(#favorite_ramen)	_set(&topic_ramen, #favorite_ramen)	知らないラーメンが好き
好き					#final
好きじゃない	そうなんですね．好きじゃないのに何で食べるのですか？				#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

フレームワーク付属のフロントエンド

C4A Computing for All



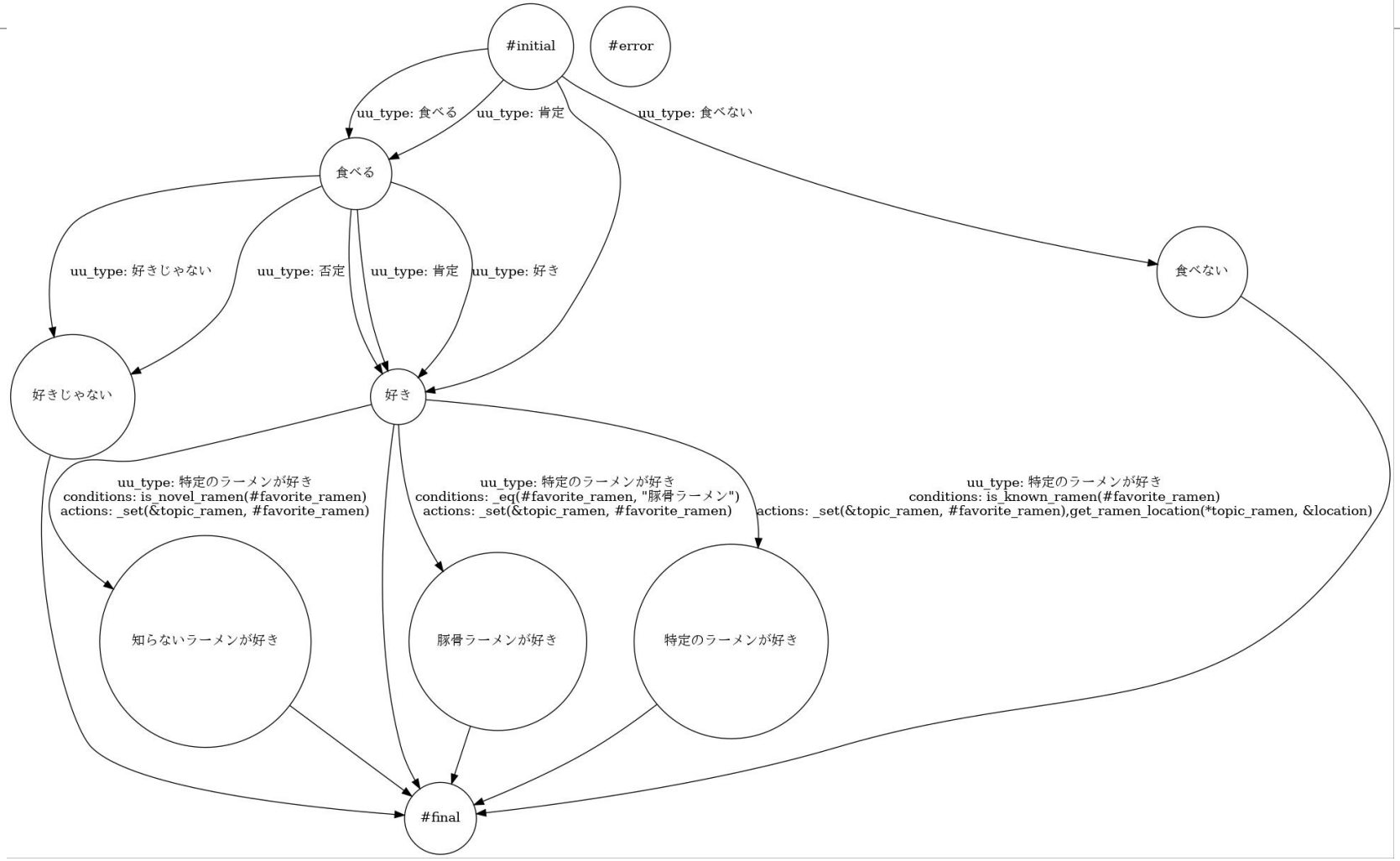
対話例

C4A
Computing for All

•System: こんにちは。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: 食べます
•System: ラーメンお好きなんですね。
•User: 好きです
•System: 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: 味噌ラーメンが好きです
•System: 味噌ラーメンですね。札幌にはよく行きますか？
•User: はい！
•System: わかりました。今日はお時間ありがとうございました。

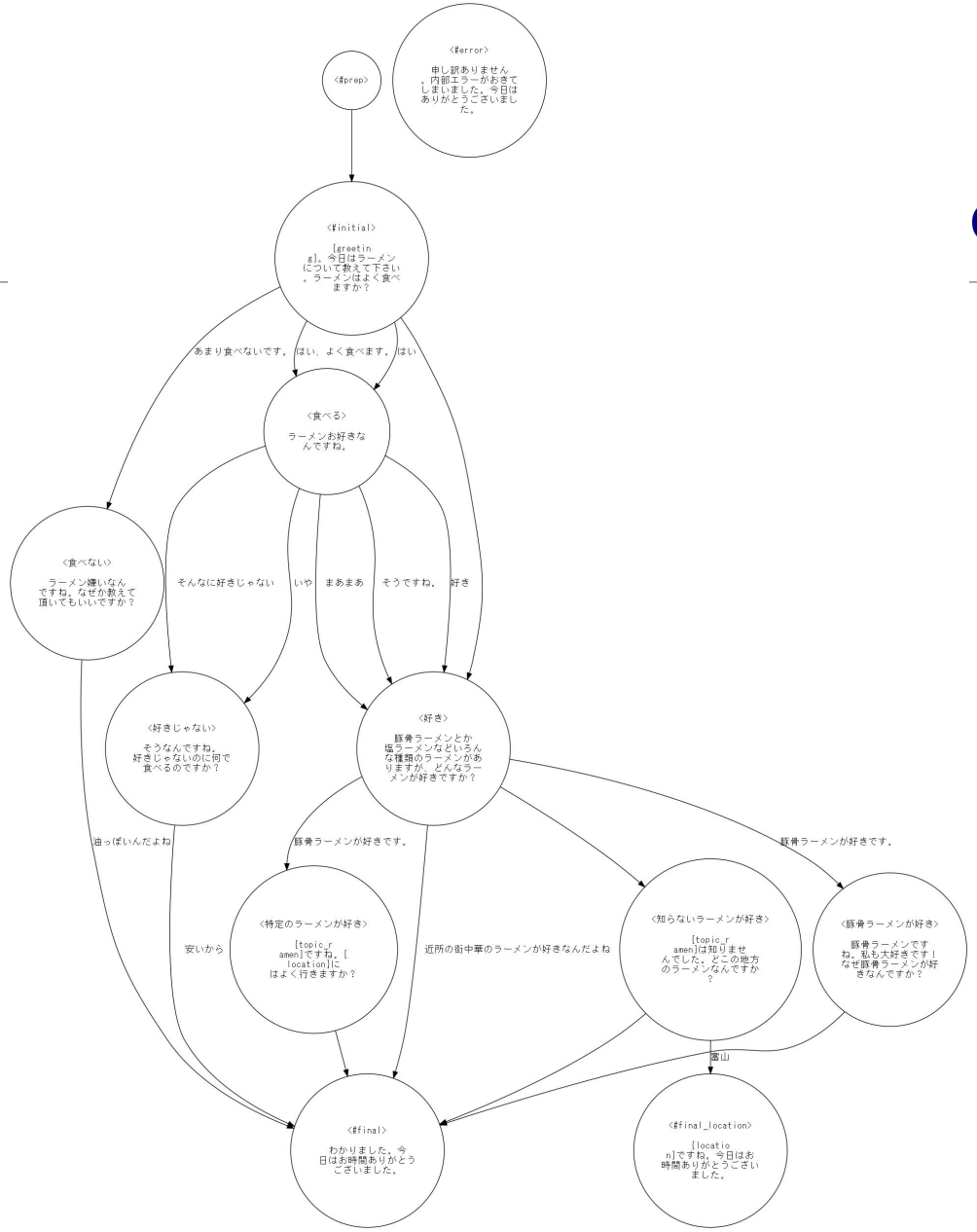
シンプルアプリケーションの状態遷移ネットワーク

C4A
Computing for All



シンプルアプリの シナリオグラフ

- ユーザ発話例を 用いたグラフ



自動テスト

C4A
Computing for All

- テキストファイルのシナリオのユーザ発話だけを使う
- 出力も同じ形式

```
----init
System: こんにちは。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
System: ラーメンお好きなんですね。
User: はい
System: 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: 豚骨ラーメンが好きです
System: 豚骨ラーメンですね。私も大好きです！なぜ豚骨ラーメンが好きなんですか？
User: 九州出身なので
System: わかりました。今日はお時間ありがとうございました。
----init
System: こんにちは。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
….
```

シンプルアプリケーションで使われている その他の機能

C4A
Computing for All

- N-best言語理解結果の利用
  - 言語理解ブロックは複数個の言語理解結果を出力
  - その中で遷移の条件に合う結果を利用する

C4A
Computing for All

実験アプリケーション

実験アプリケーションの概要

C4A
Computing for All

- シンプルアプリケーション＋新機能の実装例
  - spaCy/GiNZAを用いた固有表現抽出の組み込みブロックの利用
  - STN Managerの機能
    - ChatGPTを用いた発話生成
    - リアクション発話生成
    - 音声入力に対処するための機能
    - サブダイアローグ
    - スキップ遷移

spaCy: https://spacy.io
GiNZA: https://megagonlabs.github.io/ginza/

spaCy/GiNZAを用いた固有表現抽出の組み込みブロック

C4A Computing for All

- spaCy/GiNZAを用いて固有表現を抽出
  - 人名、組織名、場所名など
- 追加のルールもコンフィギュレーションファイルに記述可能
- 抽出結果は補助データ(aux_data)に入れて出力
- STN Managerのシナリオシートで#NE_<クラス名>で参照可能
  - 例：#NE_Person

ChatGPTを用いた発話生成

C4A Computing for All

- アクション関数の中で、システムが感想を言う発話を生成する
  - プロンプト：対話の履歴に続けて依頼文をつける
  - ChatGPTの返答からシステム発話を抽出

プロンプト

システム「こんにちは。私はチャットボットです。よろしければお名前を教えて頂けますか？」
....
システム「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？」
ユーザ「豚骨ラーメンが好きです」
システム「豚骨ラーメンですね。私も大好きです！なぜ豚骨ラーメンが好きなんですか？」
ユーザ「こってりしてるからです」
の後、システムが感想を短く言う発話を生成してください。

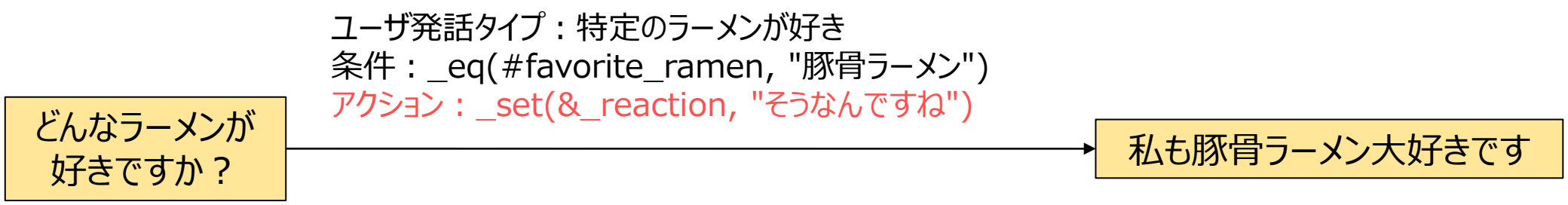
ChatGPTの応答

システム「豚骨ラーメン、こってりしているのが好きなんですね。私もその濃厚な味わいが大好きです」

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

C4A
Computing for All

ChatGPTを用いたアプリケーション

概要

C4A
Computing for All

- ChatGPTベースの対話の組み込みブロックを利用
- ブロックのクラスは容易に拡張（サブクラスの作成）が可能

ChatGPTを用いた応答生成

C4A
Computing for All

- プロンプト

<コンフィギュレーションファイルのprompt_prefixの値>
システム「<システムの1番目の発話>」
ユーザ「<ユーザの1番目の発話>」
システム「<システムの2番目の発話>」
ユーザ「<ユーザの2番目の発話>」
…
システム「<システムの最新の発話>」
ユーザ「<ユーザの最新の発話>」
<コンフィギュレーションファイルのprompt_postfixの値>

プロンプト例

システムはユーザと親しく話せます．
システム「こんにちは．楽しくお話しましょう．何についてお話しますか？」
ユーザー「映画について話したいです」
システム「素晴らしいですね．映画について大好きです．最近見た映画やお気に入りのジャンルがあれば教えてください．」
ユーザー「宮崎駿監督の最新作を見ました」
に続くシステムの発話を最大100文字で生成してください．

ChatGPTの応答

システム「それは素晴らしいですね．宮崎駿監督の作品は常に素晴らしいです．その映画についてどんな感想を持ちましたか？」

サブクラスの作成による拡張

C4A
Computing for All

- 応答生成関数の中で以下の情報を利用可能
  - 対話の履歴（ユーザとシステムの発話シーケンス）
  - ユーザID
  - 補助データ
- プロンプトの作成の仕方を変更することで、より目的に即した応答が可能

DialBBを用いた対話システム開発

C4A
Computing for All

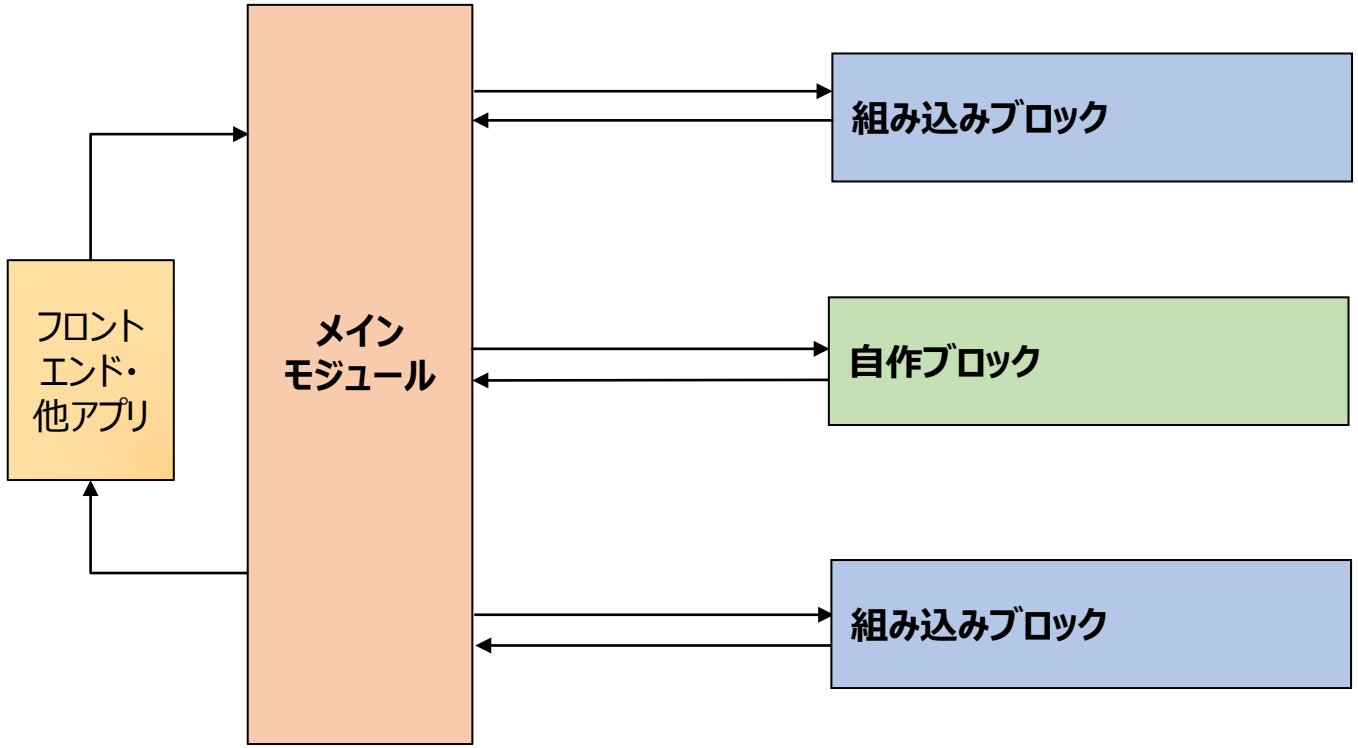
一般的なアプリ作成ステップ

C4A
Computing for All

- 組み込みブロックのみを用いたシステム
  - サンプルアプリのディレクトリをコピー
    - DialBBのディレクトリの外が良い (gitのレポジトリ外)
  - 知識記述を変更
    - 対話シナリオ
    - 言語理解知識
    - シナリオ関数
    - 辞書関数
  - 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

C4A
Computing for All



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

- 出力の発話文字列には発話以外の情報も含められる
  - ロボットのジェスチャーなど
  - 将来的には出力の補助データ（aux_data）に入れる情報をシナリオで書けるようにする
- 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

今後の
改良予定

現在進めている改良

C4A
Computing for All

- 組み込みブロックの追加
  - ChatGPT JSON modeを用いた言語理解
- インストールを容易にする
  - pipでインストールできるようにする
  - Poetry, Dockerの導入
- チュートリアルの作成
- 状態遷移ネットワークのGUIエディタ

今後の改良予定

C4A
Computing for All

- 組み込みブロックの追加
  - sentiment analysis
  - フレームベース対話管理（RDBの利用例を入れる）
  - 用例ベース応答生成（sentence-BERTなど）
- クラウドで動作させる例

長期プラン

C4A
Computing for All

- 様々な人に使ってもらってフィードバックを得る
- GitHubのDiscussionsで要望・提案を募集