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

v1.0対応

中野 幹生

株式会社C4A研究所

アウトライン

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

- 対話システム構築フレームワーク
- 対話システムを作りながら対話システム技術を学ぶための教材
- 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

Dialogue System Development Framework with Building Blocks

DialBBのドキュメント

C4A
Computing for All

- プロジェクトサイト
- インストール方法とサンプルアプリの動かし方
  - GitHub README
- チュートリアルと仕様詳細

DialBB開発の背景

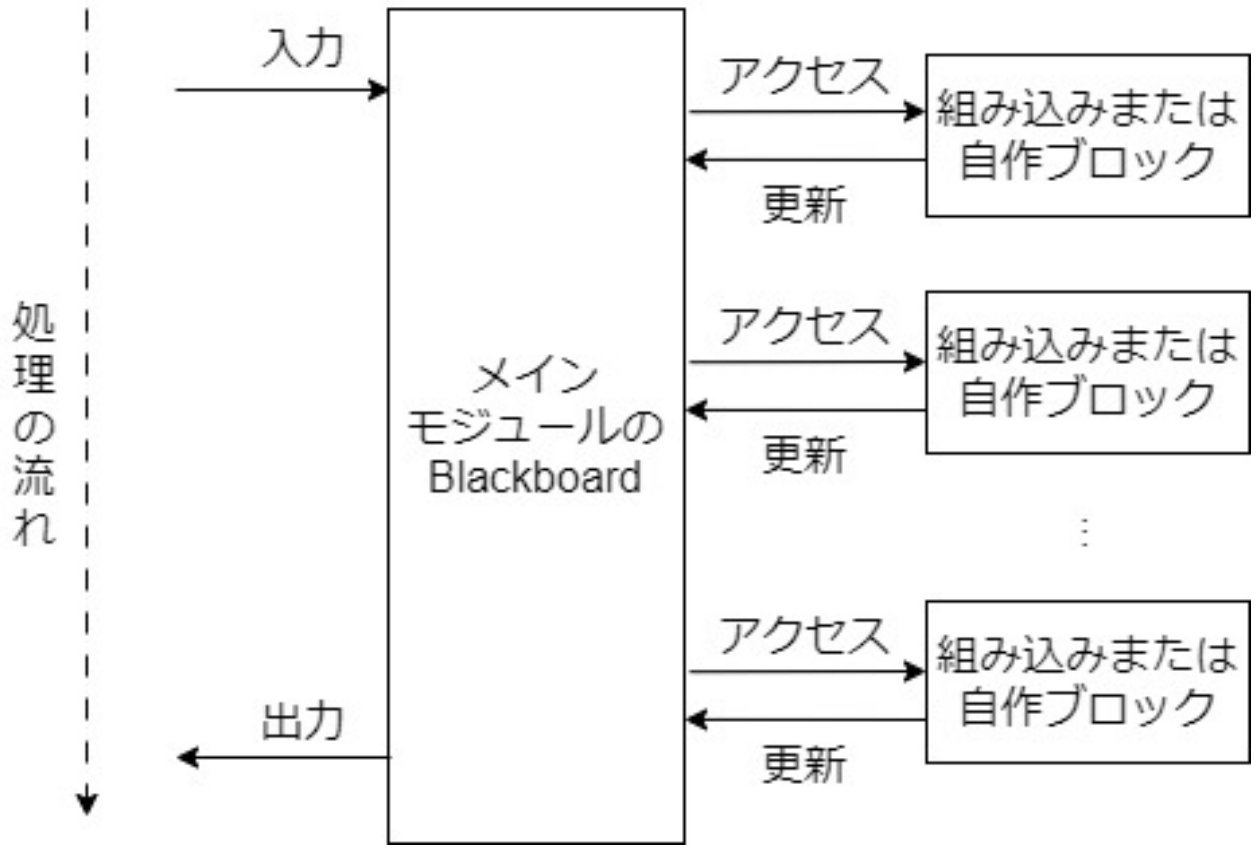
C4A Computing for All

- 対話システム技術を学ぶための教材があるとよい
  - 教科書よりもより実践的なシステム開発の教材
- 学んで欲しい技術
  - 対話システムのアーキテクチャ
  - 対話システムの要素技術
  - 拡張性の高いシステムの構成法
    - アプリケーション依存部分と非依存部分の切り分けなど
  - 多様なユーザ発話に対する頑健性の向上
  - インタラクティブなシステムのデザイン
  - ソフトウエアアプリケーションの構築

東中・稲葉・水上：Pythonでつくる対話システム，オーム社，2020
井上・河原：音声対話システム、基礎から実践まで，オーム社、2022

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
  - 例：ロジスティック回帰とCRF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
  - 例：ChatGPTのみ
- サンプルアプリが付属
- Pythonで書かれている
- オープンソースとして公開（Apache License 2.0）
- 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

対話システム技術習得のステップ

C4A
Computing for All

- 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
- 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
- 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
- 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
- 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

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

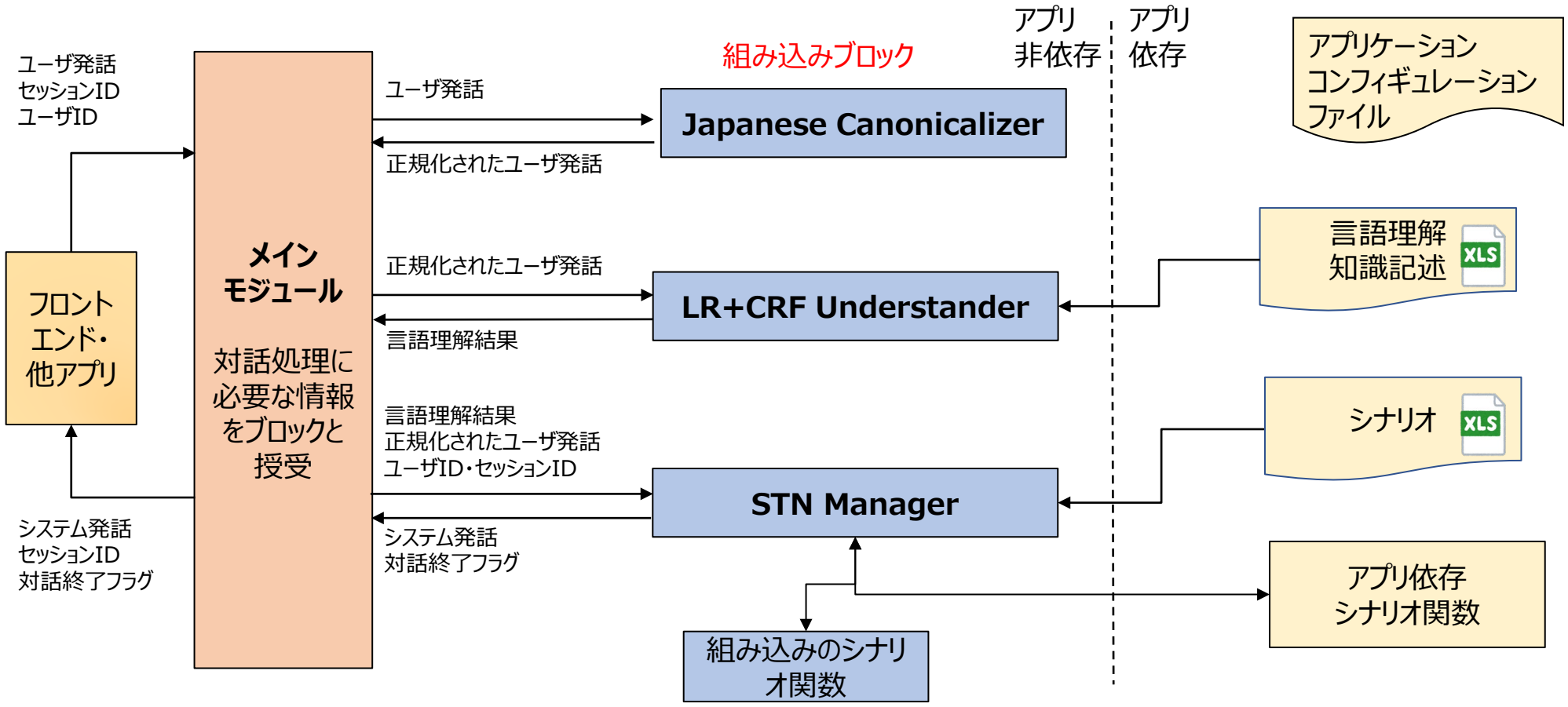
- 組み込みブロックのみを用いたアプリケーション
  - シンプルアプリケーション (sample_apps/simple_ja)
    - ロジスティック回帰とCRF（条件付き確率場）による言語理解
    - 状態遷移ネットワーク（State-Transition Network）による対話管理
  - 実験アプリケーション (sample_apps/lab_app_ja)
    - ChatGPTを用いた言語理解
    - ChatGPTを用いた固有表現抽出
    - 状態遷移ネットワークによる対話管理
      - シンプルアプリケーションに入っていない機能を導入
  - ChatGPT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 = DialogueProcessor(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）言語理解

C4A Computing for All

- LR+CRF Understander
  - ロジスティック回帰と条件付確率場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
- 入力
  - input_text: 正規化されたユーザ発話テキスト（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": {"好きなラーメン": "醤油ラーメン"}}）

言語理解知識

C4A
Computing for All

utterancesシート: 発話とタイプとスロット抽出結果の例

flag	type	utterance	slots
Y	肯定	はい	
Y	否定	そうでもない	
Y	特定のラーメンが好き	豚骨ラーメンが好きです	好きなラーメン=豚骨ラーメン
Y	地方を言う	荻窪	地方=荻窪
Y	ある地方の特定のラーメンが好き	札幌の味噌ラーメンが好きです	地方=札幌, 好きなラーメン=味噌ラーメン

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

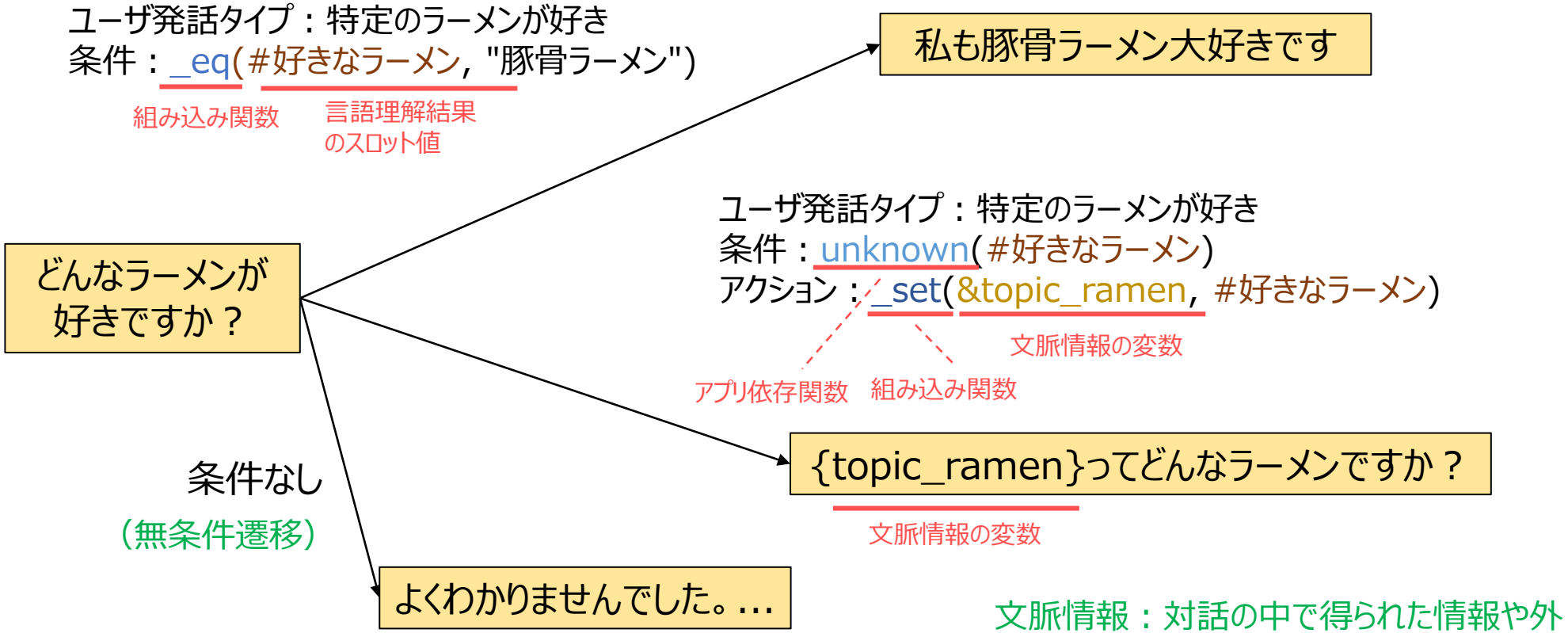
どのflagの行を使うかはコンフィギュレーションで指定可能

組み込みブロック（3）
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



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

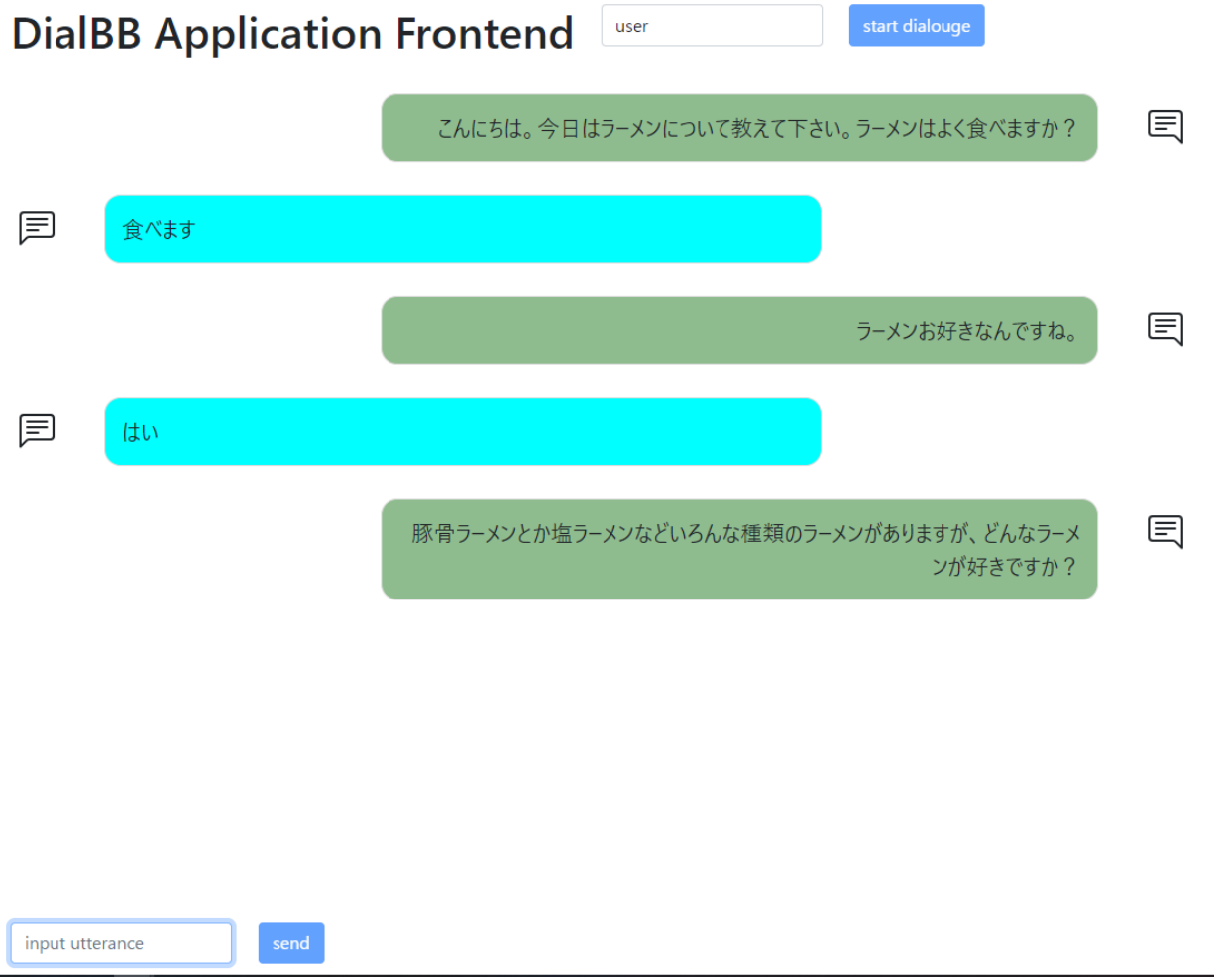
flag	state	system utterance	user utterance example	user utterance type	conditions	actions	next state
Y	好き	豚骨ラーメンとか塩ラーメンなどいろんな種類のラーメンがありますが，どんなラーメンが好きですか？	豚骨ラーメンが好きです.	特定のラーメンが好き	_eq(#好きなラーメン, "豚骨ラーメン")	_set(&topic_ramen, #好きなラーメン)	豚骨ラーメンが好き
Y	好き		豚骨ラーメンが好きです.	特定のラーメンが好き	is_known_ramen(#好きなラーメン)	_set(&topic_ramen, #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			特定のラーメンが好き	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

フレームワーク付属のフロントエンド

C4A Computing for All



対話例

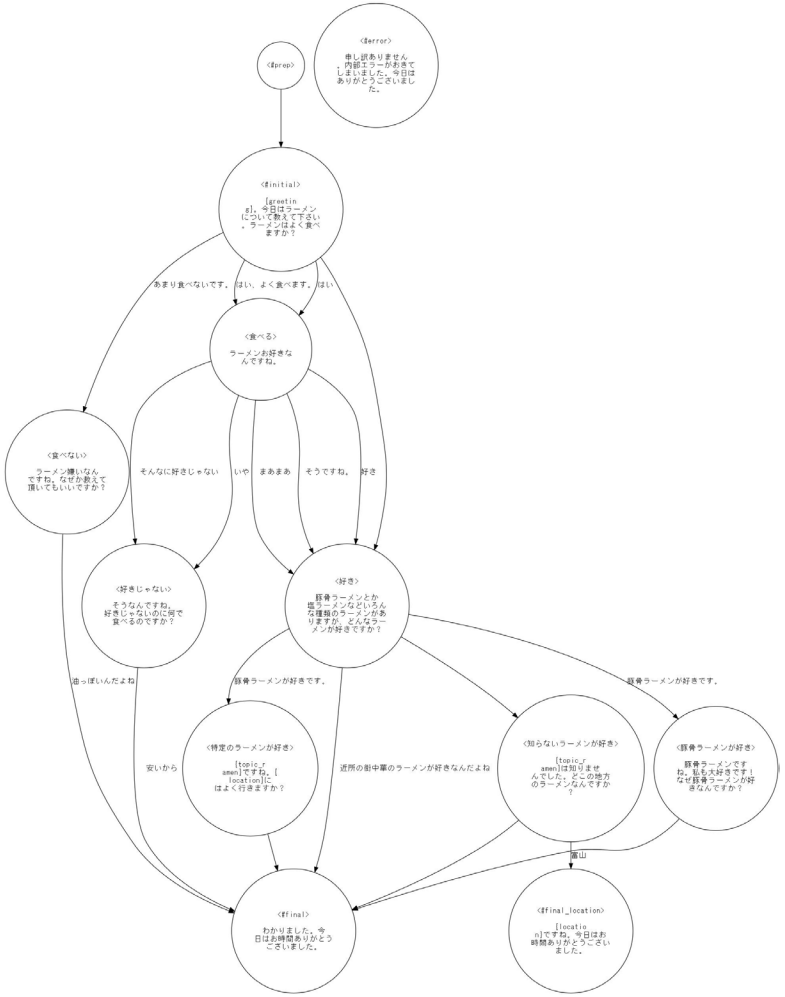
C4A
Computing for All

•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シナリオグラフ

C4A
Computing for All

- 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

- 利用する組み込みブロック
  - Japanese Canonicalizer
  - ChatGPT言語理解
  - ChatGPT固有表現抽出
  - STN Manager
- STN Managerの先進機能
  - システム発話中の関数呼び出し・特殊変数参照
  - ChatGPTを用いた発話生成
  - ChatGPTを用いた遷移条件判定
  - 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

ChatGPTベースの言語理解

C4A
Computing for All

- 入力
  - 正規化されたユーザ発話文字列
- 出力
  - LR＋CRF言語理解と同じ
- LR＋CRF言語理解と同じ形式の知識記述
  - 発話例をFew-shotに用いる

ChatGPTを用いた固有表現抽出の組み込みブロック

C4A Computing for All

- ChatGPTを用いて固有表現を抽出
  - 人名、組織名、場所名など
- 固有表現の定義と例をExcelに記述
- 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_人名

固有表現抽出知識

C4A
Computing for All

utterancesシート：
発話と固有表現抽出結果の例

flag	utterance	entities
Y	札幌の味噌ラーメンが好きです	地名=札幌
Y	田中です	人名=田中
Y	太郎は東京に住んでます	人名=太郎，地名=東京

<クラス名>=<固有表現>

classesシート：
固有表現クラスの説明と固有表現の例

flag	class	explanation	examples
Y	人名	人の名前	田中，山田，太郎，花子，ジョン
Y	地名	場所の名前	札幌，東京，大阪，日本，アメリカ，北海道

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: ChatGPT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
- システム発話の中に関数呼び出しを埋め込む
- ChatGPT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。

STN Manager: ChatGPT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

```
chatgpt:
    gpt_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: ChatGPT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
- ChatGPT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）

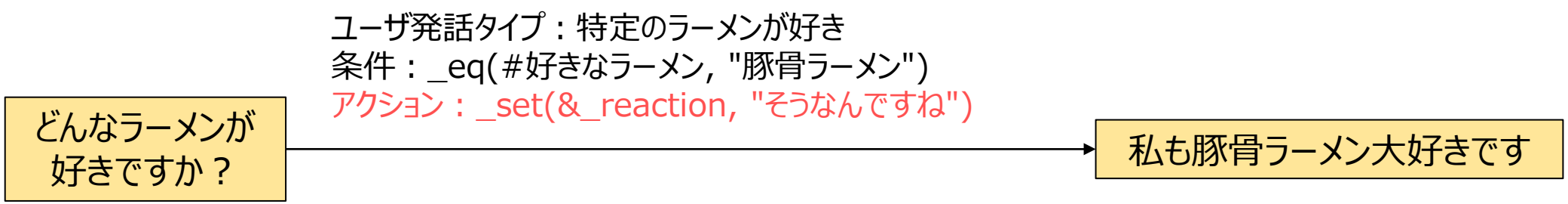
利用例

```
_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。")
```

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 (1)

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

詳細はドキュメントを参照

その他のSTN Managerの機能 (2)

C4A Computing for All

- スキップ遷移：システム発話を行わずに次の遷移を行う
  - システム発話のかわりに"$skip"と書く
  - 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
- 条件を満たす遷移がない時に同じ状態に戻る
  - コンフィギュレーションで repeat_when_no_available_transitions に true を設定
  - システム発話に{_generate_with_llm("それまでの会話を踏まえて雑談してください")}と書いて何ターンか雑談させたりできる
  - 遷移の条件に_num_turns_in_state_exceeds("5")と書くと、5ターンで抜ける
- 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

STN Manager: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
    - y=="aa"　→　_eq(y, "aa")
    - y!="aa"　→　_ne(y, "aa")
    - $"<タスク>"　→　_check_with_llm("<タスク>")
  - アクション関数
    - y="aa"　→　_set(y, "aa")
  - 生成関数
    - $"<タスク>"　→　_generate_with_llm("<タスク>")

C4A
Computing for All

ChatGPTを用いたアプリケーション

概要

C4A
Computing for All

- ChatGPT対話の組み込みブロックを利用
  - 次ページのプロンプトを使って発話を生成させる
  - {dialogue_history}はそれまでの対話で置き換えられる

発話生成のためのプロンプト

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

# 現在までの対話

{dialogue_history}
```

それまでの対話で置き換える

```
system: こんにちは。好きな食べ物は何ですか？
user: 甘いものが好きです
system: 私もです！パンケーキとか好きですか？
user: パンケーキはあまり好みじゃないです
```

DialBBを用いた対話システム開発

C4A
Computing for All

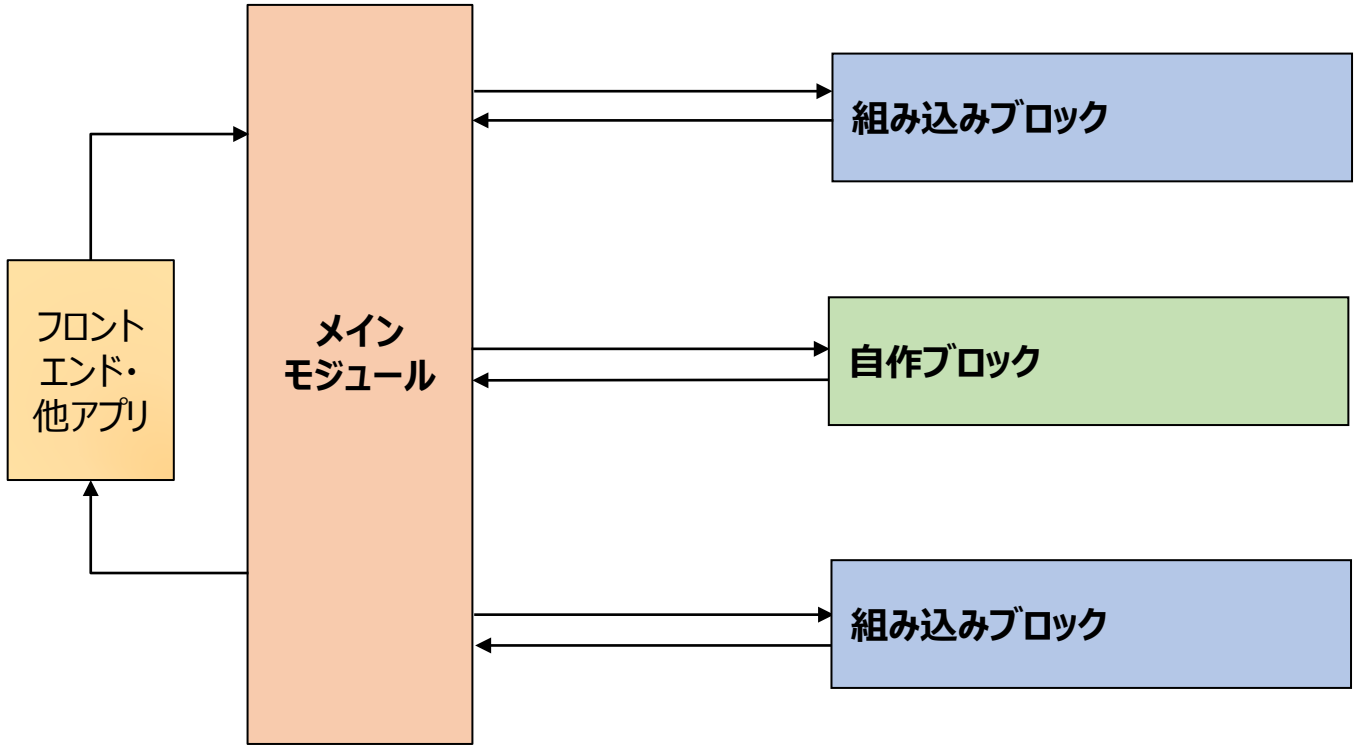
一般的なアプリ作成ステップ

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
  - 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
  - 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) シナリオ関数

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

Tips (2) マルチモーダル

C4A
Computing for All

- 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
- 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
  - クライアントプログラム次第
  - 将来的には出力の補助データ（aux_data）に入れる情報をSTN Managerのシナリオで書けるようにする

Tips (3) ノーコードツール

C4A
Computing for All

- 最初はノーコードツールを利用するのが良いかもしれない
  - DialBBにはノーコードでアプリケーションを作るためのツールがある（ドキュメント）
    - STN Manager, ChatGPT言語理解、ChatGPT固有表現抽出ブロックを使う
    - GUIで編集できる
  - ステップ
    - まずノーコードツールでアプリケーションを作って、動作確認をする
    - アプリケーションファイルを展開（unzip）し、高度な編集をする
  - 注意
    - ノーコードのアプリケーションは、STNマネージャのコンフィギュレーションの repeat_when_no_available_transitionsがtrueになっているので、遷移がなければ同じstateでループする

Tips (4) その他

C4A
Computing for All

- LLMを用いたユーザシミュレータを使ってテストできる
  - リポジトリ

詳細はレポジトリの
README参照

- 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

短期プラン

C4A
Computing for All

- STN ManagerのChatGPT組み込み関数でプロンプトテンプレートを自由に書けるようにする
- サンプルアプリの追加
  - サンプルアプリ用のリポジトリ
  - 検討中のアプリ
    - フレームベース対話管理のカスタムブロックを使うアプリ
    - 用例ベース応答生成（sentence-BERTなど）
    - Retrieval Augmented Generation (RAG) を利用したアプリケーション
    - 音声・マルチモーダル対話システム
      - 音声・マルチモーダルフロントエンドとの接続例
    - RDBの利用例を入れる

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