スマートスピーカーアプリケーション開発のための日本語パターンマッチング方法の提案

2015SE013 東村祐勢　2015SE052 水野帆奈美　2015SE061 岡山楓子
指導教員：蜂巣吉成

1　はじめに

現在，音声認識の発達により，音楽の再生，ニュースや情報の検索，家電のコントロールなどができるスマートスピーカーが普及しはじめている．アプリケーションを開発することで独自の機能も追加できる [1]．

スマートスピーカーやチャットボットのような対話で操作するシステムを一般に対話システムと呼ぶ [2]．対話システムは，タスク指向型と非タスク指向型に分類できる．タスク指向型システムはユーザとの対話より，特定のタスクを達成することを目的としている．スマートスピーカーアプリケーションの特徴で挙げた機能のほとんどはタスク指向型に分類される．非タスク指向型は雑談そのものが目的の対話であり，自由な話題の会話やゲームなどが例として挙げられる．今後，スマートスピーカーを使った非タスク指向型システムの開発も期待される．

スマートスピーカーアプリケーション開発では，実行したい処理を起動するためのユーザの発話をサンプル発話として登録する必要がある．非タスク指向型対話システム開発の問題点として次の 2 点が挙げられる．

1. 同じことを表す複数の言い回しがあるため，サンプル発話の登録に手間がかかる．
2. 定型文を話すとは限らないので，サンプル発話を考えるのが難しい．

問題点 1 の例として，駅名しりとりゲーム [3] を挙げる．ユーザが負けになる場合，「降参します」と言えば，システムを終了するとする．しかしユーザは必ずしも「降参します」と言うわけではないので，自然な対話のためには，様々な言い回し「降参だ」「降参だよ」や「駅名が思い浮かびません」などをすべてサンプル発話として登録しなくてはならない．問題点 2 は，話題の焦点が決まらないとそもそもサンプル発話を考えることが困難である．問題点 1，2 は機械学習により自然な応答を目指すことができるが，学習用のデータの準備や学習パラメータの調整などが難しい．これが不要で開発できれば，開発しやすくなる．

本研究では，これらの問題点を解決するための日本語パターンマッチング方法を提案する．スマートスピーカーでユーザの発話を自由発話として取得し，パターンマッチングにより解析し，応答を生成する．問題点 1 については，形態素解析結果に対してのパターンマッチングや否定文肯定文判定の方法を提案する．問題点 2 については，問題点 1 の解決方法に加え，係り受け解析を用いた方法を提案する．提案方法によるライブラリを設計・実現し，サンプルアプリケーションを開発して評価をした．本研究では，Amazon Alexa（以降，Alexa とする）を取り上げるが，提案するパターンマッチング方法は他のスマートスピーカーやチャットボットにも適用できる．

2　スマートスピーカーアプリケーション

2.1　実行の仕組み

スマートスピーカーはユーザの発話を受け取り，Alexa サービスに送信する．Alexa サービスはその音声データを解析し，サンプル発話に基づいて実行する処理を決め，アプリケーションに発話テキストを渡す．アプリケーションは応答テキストを Alexa サービスに返すと，音声データにしてスマートスピーカーに送信し，スマートスピーカーが音声としてユーザに伝える（図 1）．

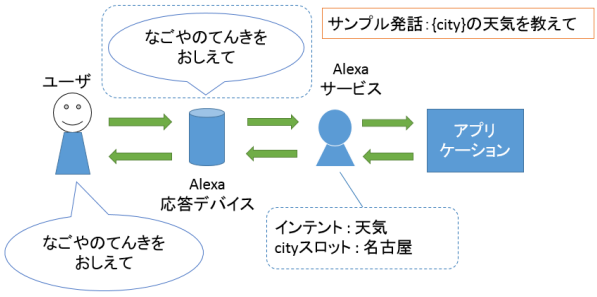

図 1　Alexa アプリケーションの実行の流れ

2.2　アプリケーション開発とサンプル発話

スマートスピーカーは，アプリケーションを開発することで様々な機能を実行できる．タスク指向型システムの多くは，音声認識の結果として得られる自然言語の文を解析して，ドメイン（全体），意図（意図，中心），スロット（処理に必要なデータ，中心の詳細）からなる意味フレームを作成して処理を行う [2]．スマートスピーカーでは，処理を起動するためのサンプル発話を登録することで，意味フレームを定義できる．

例えば，天気予報を読み上げるアプリケーションを考える．アプリケーションの呼び出し名を「天気予報」とし，「{city} の天気を教えて」というサンプル発話を登録すると，「天気予報」でドメインが決まり，サンプル発話の「天気」がインテントとなり，現在の天気を調べるという処理が決まる．「{city}」がスロットで，Alexa サービスによりユーザが発話した場所（地名）が埋められる．ユーザが「天気予報で名古屋の天気を教えて」と発話すると，Alexa サービスから天気予報アプリケーションに，「天気」のインテントと city スロット「名古屋」が渡される（図 1）．アプリケーションは処理の種類に応じて複数のインテントを

もつ.「降水確率を教えて」というサンプル発話では,「降水確率」がインテントとなり,現在地の降水確率を調べる処理が決まる.

2.3 サンプル発話登録における問題点

自然な対話を行うためには,ユーザの様々な言い回しに対応できるようになるべく多くのサンプル発話を登録しなければならない(1 節に挙げた問題点 1).2.2 節の降水確率の例では,「降水確率は」「雨降る」「雨降るかな」「雨降りそう」「雨は降らないかな」などが挙げられる.「駅名しりとりゲーム」[3] では,ユーザが降参するときのサンプル発話として約 80 種類が挙げられている.

非タスク指向型システムでは,定型文を話すとは限らないのでサンプル発話による意味フレームの定義が難しく,話題の焦点が決まらないとサンプル発話自体を考えることが困難である(1 節で挙げた問題点 2).

2.4 自由発話の取得

本研究では,SearchQuery のスロットでサンプル発話を登録し,他のスロットを使用したサンプル発話に当てはまらなかったユーザの発話を自由発話としてアプリケーションに渡して処理を行う.SearchQuery は,検索クエリーを構成する,予測しにくい入力のときに使用するスロットである.

例えば,「{utterance} でごわす」と登録し,ユーザの発話が「サッカーは好きじゃないけど,野球は好きでごわす」の時,スロット utterance には「サッカーは好きじゃないけど,野球は好き」が入る.ユーザの発話すべてを抽出するので,必要な言葉を抽出する処理をアプリケーション開発者が行わなければならない.

3 日本語パターンマッチングの提案

3.1 概略

本研究では 2.3 節の問題を解決するために次の 3 つの日本語パターンマッチングライブラリ方法を提案する.

1. 形態素パターンマッチング
   形態素解析の結果に対するパターン記述を提案する.形態素とは意味をもつ表現要素の最小単位であり,形態素解析は形態素の品詞や原形を判別することである [4].
2. 否定文肯定文判定機能
   形態素解析を用いて打ち消しの助動詞「ない」「ず」などの有無を判定し,否定文肯定文を判定する
3. 係り受け解析を用いた修飾関係リストの生成
   係り受け解析を用い,修飾関係にある文節のリストを返す.係り受け解析は,文中の文節がお互いにどのような修飾関係になっているかを解析する [4].

機能 1,2 を用いて問題点 1 の同じことを表す複数の言い回しをパターンで表現し,サンプル発話を多数登録する手間を省く.機能 1,2,3 を用いることで問題点 2 の自由発話にもある程度対応できる.

3.2 形態素パターンマッチング

3.2.1 形態素解析の必要性

パターンマッチングの方法については正規表現ではなく,形態素解析を用いたパターン記述を提案する.例えば「思いつく」「思いついた」を正規表現でマッチングする方法を考える.文字列「思いつ」があれば後方にどのような並びの文字列がきても「思いつく」と似た言い回しになると判定する.この場合「思いつめる」などの言葉がマッチしてしまい,正しいマッチングが行えない.

形態素解析を用いたパターン記述の場合,形態素の原形「思いつく」の有無を判定してマッチングを行う.「思いつめる」の形態素の原形は「思いつめる」になり,「思いつく」の似た言い回しにマッチしない.

3.2.2 提案するパターン記述

形態素に対するパターンマッチングを提案する.提案するパターンの文法を BNF で表すと次のようにする.

<Type>::=表層|品詞|原形|読み
<Value>::= 任意の日本語文字列
<KT>::=<Type>:<Value>
<KTS>::=<KT>|<KTS> ␣&&␣ <KT>
<Elem>::=<KTS>|(␣<KTS>␣)|*|@
<Pattern>::=<Elem>|<Elem>␣<Pattern>

パターンの意味は次の通りである.

1. 1 つの形態素を *type*:*value* として表す
2. 形態素の 2 つの属性を同時にマッチさせる場合は &&でつなぐ
3. *は任意の 0 個以上の形態素を表す
4. 抽出したい形態素はパターンを括弧で囲う
5. @は任意の 0 個以上の形態素を抽出する

1 の *type* には,「表層語」「品詞」「原形」「読み」の形態素解析で得られる属性名を記述し,*value* にはその検索したい値を記述する.

2 は形態素の 2 つ以上の属性を調べたいときに使用する.例えば,「助動詞」の「ない」をマッチさせたい場合は「品詞:助動詞 && 表層:ない」と記述する.

3 はワイルドカードを参考にした.ワイルドカードとはすべて文字列にマッチする特殊な記号である.ここではすべての形態素にマッチする.例えば「私はなにも思いつきません」であれば,「* 原形:思いつく *」で,形態素「私」「は」「なに」「も」が前方の*にマッチし,「ない」が後方の*にマッチする.

4 はパターンにマッチした形態素を抽出する記述である.

5 は任意の 0 個以上の形態素を抽出する記述である.

「(*)」という記述も考えられるが,パターン記述を容易にするために「@」とした.

なお，パターンにマッチする箇所が複数個ある場合は最初に出現した形態素がマッチングされる．

3.3 否定文肯定文判定機能

3.3.1 否定文肯定文判定の必要性

日本語のパターンマッチングのみではテキストの否定肯定を判断することが難しい．例えば，駅名しりとりゲーム [3] で降参を意味する言い回しのサンプル発話の登録の数は約 80 個あった．例の 1 部を以下に示す．

- 降参，降参する，降参だね，降参ですね，降参します，降参ですよ，降参だよ…
- 思い浮かばない，思い浮かばないよ，思い浮かばないです…
- 思いつかない，思いつかないです，思いつかないよ，思いつくわけない…

この無数にある降参を意味する言い回しを 3.2.2 節で提案したパターン記述を行うと，「* 原形:思いつく *」といったパターンで表すことで，「もう思いつかない」「駅名が全く思いつかん」などと言った様々な言い回しを網羅することができるが，「思いついた」や「今思いついたよ」などといった降参とは逆の意味の言い回しもマッチする．3.2.2 節のパターン表記で否定文肯定文を判定する方法を考えた．否定文の判定のために打ち消しの助動詞を判定するが，打ち消しの助動詞の数だけパターンを記述しなくてはならない．また二重否定も考えられるので記述パターンが多くなる．さらに肯定文の判定において，打ち消しの助動詞が出現しないこと表現することはできない．このような理由から，否定文肯定文判定機能を作成し，パターンマッチングと組み合わせるほうが効率的だと考えた．

3.3.2 否定文肯定文判定の方法

否定文肯定文を判別する仕組みについて述べる．パターンマッチング同様，形態素解析を用いた判定を行う．打ち消しの語である「ない」「ん」「ず」「ぬ」などは助動詞であるので，品詞が助動詞である打ち消しの語の有無を判定し，ある場合は否定文，ない場合は肯定文と判定する．また二重否定を考慮し，打ち消しの語が偶数回出現したら肯定と判定し，奇数回の場合は否定とする．

3.4 係り受け解析を用いた修飾関係リスト生成

係り受け解析を用いた修飾関係リスト生成の必要性について述べる．好きなスポーツを抽出する場合で，ユーザが「サッカーは好きじゃないけど，野球は好き」と答えたとき，「野球」を抽出したい．3.2 節の「* (品詞:名詞) * 表層:好き *」でパターンマッチングを行い，抽出すると 1 番最初の名詞「サッカー」が抽出され，適切な形態素「野球」を抽出することができない．

長文であることが問題なので，短い文をパターンマッチングすることで適切な形態素を抽出することができる．ユーザの発話を係り受け解析し，修飾関係にある 2 つの文節を 1 つの文にしたリストを作成する．日本語の文は最後の文節が重要と考えてリストは文節の出現の逆順とする．図 2 の例からは「野球は好き」「好きじゃないけど，好き」「サッカーは好きじゃないけど,」のリストが作られる．3 つの文を「* (品詞:名詞) * 表層:好き *」でパターンマッチングを行い，名詞を抽出した結果「野球」を抽出することができる．否定な発話に対しては，否定文肯定文判定機能を使用する．

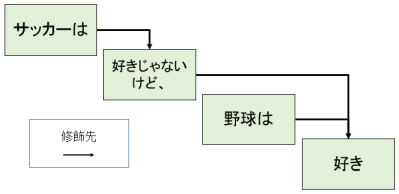

図 2 修飾関係の例

4 設計・実現

4.1 ライブラリの設計・実現

形態素解析器 MeCab[*1] と係り受け解析器 CaboCha[*2] を使って Python でライブラリを設計・実現した（約 340 行）．3 節の 3 つの方法は関数として実現した．

4.2 本研究における実行の仕組み

アプリケーションの開発者がサーバを用意し，図 3 のようにアプリケーションを実行する．Alexa Interface Adapter は，Alexa サービスから送られてきた JSON 形式のデータからユーザ発話のテキストを抽出してアプリケーションに渡し，アプリケーションからの応答テキストを JSON 形式にして Alexa サービスに返す．アプリケーションは発話テキストをコマンドライン引数で受け取り，これに対する応答発話を標準出力に出力する．GoogleHome などの他のスマートスピーカーでも，Alexa Interface Adapter に該当するプログラムを作成すれば適用可能である．

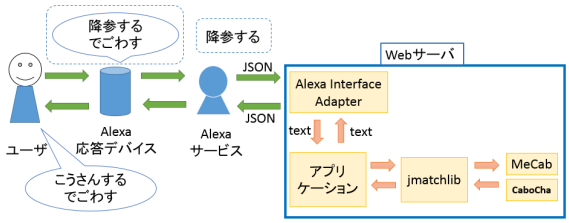

図 3 本研究の Alexa アプリケーションの実行の流れ

5 サンプルアプリケーションの開発

提案した日本語パターンマッチングライブラリを用いて名城線駅名記憶ゲームという独自ルールのゲームと雑談アプリケーションを作成した．

*1 http://taku910.github.io/mecab/
*2 https://taku910.github.io/cabocha/

5.1 名城線駅名記憶ゲーム

簡単なゲームの例として「名城線駅名記憶ゲーム」を作成した．システムとユーザで名古屋市市営地下鉄名城線にある駅名をそれまでに言われた順番通りに言い，最後に新しい駅名を言う．順番を間違えたり，存在しない駅名を言うと負けというルールである．

このゲームをサンプル発話を用いてスマートスピーカーで実現するのは難しいのでライブラリを用いて作成した．3.3.1 節で挙げた降参を意味する言い回しについては「* 原形:思いつく *」のパターン記述を行い，さらに getStype を呼び出して，その文が否定文であることを判定した．「思い浮かばない」「降参」といった言い回しについても同様のことを行い，サンプル発話を多数登録することなく，名城線駅名記憶ゲームを実現することができた．

5.2 雑談アプリケーション

スポーツにテーマに絞り，昔のことを思い出すような簡単な会話モデルに基づいて雑談アプリケーションを作成した．

1.「名前」2.「好きなスポーツ」3.「観るのが好きか，やるのが好きか」4.「いつやっていたか (観ていたか)」5.「当時の思い出」の 5 つの質問をしてユーザに答えてもらい，その発話から質問に対する答えを抽出して，応答生成を行う．

質問 1，2 のような質問は次のユーザの発話にその答えとなる名詞が含まれることが予測されるので「* (品詞:名詞) *」といったパターン記述を行い，質問に対する答えを発話から抽出する．好きなスポーツの質問については「〜は好きじゃないけど，〜は好き」といった長文の発話も考えられるので，3.4 節の修飾関係のリストを用いて，すべてパターンマッチングさせ，さらに 3.2.2 節の否定文肯定文の判定も行った．

質問 4 については，次のユーザの発話に「〜の時」「〜の頃」といった発話が予想される．その直前の形態素が複数になることも予測されるので「@ 表層:の 表層:時 *」といったパターン記述をした．

質問 5 についてはユーザが答えたことに対して，反復相槌を生成するために係り受け解析を行い，最後の文節と修飾関係にある文節を抽出することで短文化して簡潔にまとめた．

5.3 評価

Alexa アプリケーション「駅名しりとりゲーム」[3] で登録された，降参という意味の約 80 個のサンプル発話を，本研究で提案したパターンマッチング方法では 12 パターンで書くことができた．

本研究で作成した雑談アプリケーションを我々の研究室の 3 年生 5 名に使用してもらい，最後まで会話ができるか確認した．最後まで独自で会話が終了した人 2 名，我々が補助を行い最後まで会話が終了した人 1 名，補助を行ったが最後まで会話できなかった人 2 名という結果になった．

会話がうまくいかなかった原因は Alexa の性能による問題と，提案したパターンによる問題に分けられる．最後まで会話ができなかった学生はいずれも発話に時間がかかり，Alexa がシステムをタイムアウトさせてしまうことが原因であった．音声を正しく認識しないこともあり自然な会話が行えないこともあった．

我々が補助を行った学生はパターンにマッチしない発話をしたので，補助が必要となった．時期を問う質問は「〜の時」や「〜の頃」といった発話でマッチするように設計したが「高校時代」や「昨日」といった発話があり，パターンにマッチしない状態が続いた．この解決案としてパターン記述を増やすことがあげられる．また文献 [5] で提案されている深層格を推定する方法を用いれば，時間格としてより適切に時期の抽出ができると考える．

6 考察

発話を文字に起こす際に，適切な漢字に変換されるかはスマートスピーカーの性能による．提案したパターン記述では「読み」でのマッチングは可能であるが，MeCab の形態素解析では，表層語の「読み」しか得ることができない．例えば，「写す」と「映す」をマッチングさせたいとき，パターンを「* 読み：ウツス *」と書けば両方にマッチングできる．しかし，「写した」と「映した」の読みは「ウツシ」なのでマッチしない． 動詞の原形の「読み」でのマッチングを行えるようにパターンを拡張することが今後の課題になる．

7 おわりに

本研究では，非タスク指向型対話システムを開発する際に同じことを表す複数の言い回しがある点，定型文を話すとは限らないので，サンプル発話の登録が難しいという点の 2 つの問題点に対して，日本語パターンマッチング方法を提案した．これを用いて 2 つのサンプルアプリケーションを開発し，5 名の学生に使用してもらい評価した．スマートスピーカーで実現することを考慮してパターンを拡張することや様々なアプリケーションを作成し，機能の有効性を確認することが今後の課題である．

参考文献

[1] amazon developer，https://developer.amazon.com.

[2] 狩野芳伸：コンピューターに話が通じるか対話システムの現在，J-STAGE 情報管理，59 巻，10 号，pp.658-665(2017).

[3] Alexa(アレクサ) に駅しりとりスキルをローンチするまで，https://qiita.com/t_ryusuke/items.

[4] 奥野 陽 (著)， グラム・ニュービッグ (著)，萩原 正人 (著)，小町 守 (監修)，イノウ (編集) ：自然言語処理の基本と技術，翔泳社 (2016).

[5] 奥村学：自然言語処理の基礎，コロナ社 (2010).