リアルタイム情報に対する時事雑談生成

塚原 裕史 [1]　　岩佐 拓哉 [2]
[1] 株式会社デンソーアイティーラボラトリ　[2] 株式会社デンソー
htsukahara@d-itlab.co.jp TAKUYA_IWASA@denso.co.jp

1 はじめに

近年, スマートスピーカーやロボットを介してニュースや天気予報などのリアルタイム情報へアクセスする音声対話サービスが発展している [1, 2, 3]. 音声による情報提供では, 文字と違い, 多くの情報を短時間で提供することが難しく, 簡潔な表現が求められる. ニュース記事をテレビ番組のように提供する [4], 述語項構造を利用して質問応答できるようにする [5], ユーザ主導とシステム主導の対話を行き来できるように対話プランを制御する [6] 統計的な手法や知識ベースによって記事の内容を補う [5, 7], などの研究がなされて来たが, これらの研究で対象とされているのは静的なドメイン知識を対象としたファクトイド的な質問応答であり, 仮に今日本シリーズの期間だとして, その時点において, あと何勝したら優勝できるかなどのニュースが配信されたその時その時で意味があるような非ファクトイド型の応答ができる対話 (以下, 時事雑談と呼ぶ) に対応するものではない.

本研究では, スポーツニュースなどのリアルタイム情報についての時事雑談にも対応できる対話システムとして, ユーザ発話の前後にシステム発話を挟んだ隣接した 3 つの発話 (以下, 隣接トリプルと呼ぶ) を一つの対話の纏まりとした対話プランによる対話制御手法を提案する. これらの対話プランをボトムアップ的に組み合わせることで対話文脈に追従し, 柔軟な応答が可能な対話生成を実現することを目指す.

2 提案システム

システム構成　本研究におけるシステム構成は図 1 のように, ユーザとの音声インタフェースになる対話インタフェース部 (以下, 単にクライアントと呼ぶ) とクライアントと HTTP プロトコルを介して対話制御を行う対話プラットフォーム部 (以下, 単にサーバと呼ぶ) からなる. サーバは, 対話制御部だけではなく, ユーザ情報管理, 外部サービス連携などの機能も持つ.

隣接トリプルによる対話プラン　雑談生成では, ユーザの自由な発話に追従する必要があり, 破綻なくシステム応答を返すことが非常に難しい [8]. そこで, 本研究では対話破綻し難い対話制御の仕組みを作るために, 任意のユーザ発話に応答するシステム発話のルールを設計するのではなく, システム発話の後に予想されるユーザ応答のパターンを設計し, さらに前のシステム発話とそれに対するユーザ発話との文脈を受けてのシステム発話パターンを設計する. これらの 3 つの隣接した発話を以下, それぞれ振り発話, ユーザ応答, 受け発話と呼び, この隣接トリプルを一つの纏まりとして以下, 対話プランと呼ぶ[1] (図 2 参照). 各対話プランは一つの最小単位の目的を有し, 振り発話はその目的が実行でき, かつユーザ応答が予測し易くするように設計し, 受け発話は直前の 2 つの発話の文脈に合うように設定し, かつ別の対話プランへの遷移先を設計する. 対話プランの遷移を通じて, より大きな目的をボトムアップ的に実行できるようにする.

対話プランによる対話制御　図 3 に, 本システムにおける隣接トリプルの対話プランに基づく対話制御の流れの例を示す. 一般の対話システムでは, ユーザ発話を与える度にシステム応答を生成して返すという方式のものが多いが, 本研究では, 対話コンテキストに発話を登録する処理とシステム対話生成を行う処理とを分離している[2]. 対話コンテキストには, システム発話, ユーザ発話は任意の順序で登録できる[3].

クライアントは対話の始めに, サーバから得た最新コンテンツを一つ選択し, サーバに登録する. 登録されたコンテンツは階層的なフレーム形式に構造化に変換され, 対話コンテキストに設定される (図 4 参照). サーバはその構造化データに含まれる情報から対話を実行できる対話プランを対話プランデータベースから検索する. 例えば, 対話プランにおける振り発話パター

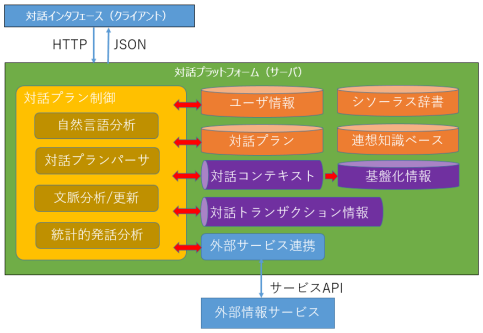

図 1: システム構成の概要

[1]図 2 には, フィラーなどのユーザ応答にレスポンス良くフィードバックするためのシステム応答を繋ぎ発話として示してある.
[2]複数のシステム (エージェント) とユーザが同時に一つの対話に参加し, 多人数会話も可能にすることを意図している.
[3]図 3 の処理の前に, ログイン, セッション初期化, 時事ニュースなどのコンテンツ取得などの処理もあるが省略している.

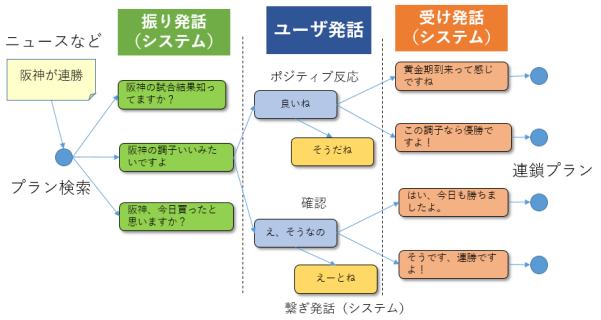

図 2: 隣接トリプルによる対話プラン

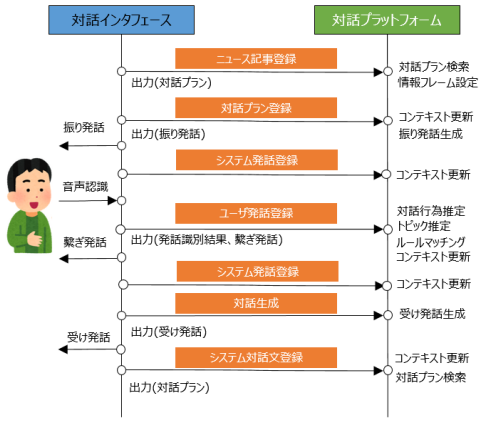

図 3: 基本的な対話制御の流れ

ンは, 図 5 のようになっており, 後述するように, 対話コンテキストに設定されている文脈情報が振り発話パターンのクエリ条件とマッチするかがチェックされ, マッチする振り発話を含む対話プランが実行可能と判断され, 適合度合いを表すスコアとともにクライアントへ返される. 次にクライアントは対話プランを一つ選択し, サーバに設定すると, その対話プランが対話コンテキストに現在の実行中の対話プランとして設定され, 文脈にマッチする振り発話が生成され, クライアントに返される. この際, サーバではその発話生成に利用された情報は, 生成した発話ごとに, 対話トランザクションに一時的に保持され, システム発話が設定された時点で, 対応する情報が文脈情報として基盤化され, 対話トランザクションはクリアーされる. (このようなトランザクション処理は受け発話生成時も同様.)

イベント名	プロパティ名	値
試合結果	試合種別	日本シリーズ
	日付	20181103
	試合区分	ナイター
	チーム1	広島
	チーム2	ソフトバンク
	ホームチーム	広島
	ビジターチーム	ソフトバンク
	スタジアム	マツダスタジアム
	チーム1総得点	0
	チーム2総得点	2

図 4: コンテンツの構造化データの例

システムの振り発話に対するユーザ応答が入力されると, 対話コンテキストに設定されている対話プランにおけるユーザ応答パターンとのマッチングが行われる. ユーザ応答とのマッチングでは, クエリ条件とのマッチングの前に, 発話内容について自然言語分析を行い, 形態素, 述語項構造, 拡張固有表現などの情報を抽出した後にクエリ条件とのマッチングが行われる[4](図 8). マッチするパターンに応じて, ユーザの発話行為, 話題, 発話内のキーワードなどを決定し, さらにそのパターンからの遷移先として定義されている受け発話のクエリ条件ともマッチングし, 適合する受け発話がある場合に, そのユーザ応答パターンが適合したと判断される (受け発話パターンの記述は図 5 と同様だが, 遷移先対話プランを登録する部分が追加される.). もし, 適合するユーザ発話パターンがない場合には, 任意応答パターンとして, 対話行為や話題は統計的識別器を用いて識別され, 任意応答用の汎用的な受け発話あるいは対話プランにあるユーザ応答パターンに適合し易いユーザ入力が得られるような質問 (選択肢を選ばせるなど) などが受け発話として対応付けられる.

ユーザ応答に対してシステム応答する場合には, サーバから受け発話の候補を取得し (これらの情報は対話トランザクション上で発話候補毎に保持されている), その内の一つを選択し, ユーザに出力後に, サーバへ登録すると, 選択された受け発話の情報が対話コンテキストに反映され, 対話トランザクション上の情報はクリアーされる. そしてクライアントへは, 次に遷移させるべき対話プランの情報が返され, 次の対話プランへと継続されていく.

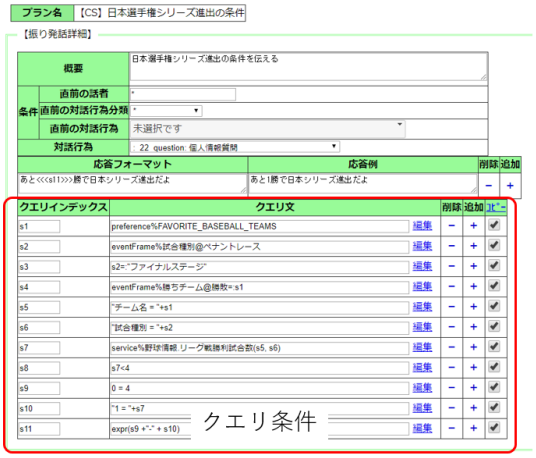

図 5: 振り発話パターン

簡易スクリプトによる文脈情報とのマッチング パターンと文脈情報とのマッチングに用いるクエリ条件には図 6,7 に示すような簡易的な構文に従うスクリプトを記述できる. 特に, 図 7 の動的パラメタを設定で

[4]他の発話パターンにおいても同様であるが, 文脈情報とのパターンのマッチングにおいて, 直前の話者やその発話行為も条件として加えることができ, これらが設定されている場合には一番最初にチェックされる.

きるクエリ条件により, リアルタイム情報との時間的かつ対話的な文脈に沿った対話生成が可能となっている. クエリ条件は上のものからチェックされ, その結果はクエリ変数として保持し, 以降のクエリ条件とのマッチングに利用できる. クエリ条件は AND 条件で結合され, 途中でマッチに失敗すると, その発話パターンは文脈とマッチしないと判断される. 同じクエリ変数名を複数のクエリ条件に設定することができ, これらは OR 条件で結合される. また, クエリ条件はユーザ応答パターン, 受け発話パターンでも同様に設定でき, かつクエリ変数は対話プラン内で共有できる.

文脈との一致をセマンティックウェブなどで利用される推論式を記述する方式もあるが [9], 本研究では, 対話プランを構築するためのコストを下げるために, 対話プランのクエリ条件においてそのような推論式を直接記述せずに, 図 6,7 のような簡易的記述に留め, これらの簡易的な構文で記述された条件をサーバ内でパースし, 推論式へ変換することにした. 具体的には, サーバ内において対話文脈情報は RDF グラフとして対話コンテキストに保持しておき, パターンに記述されたクエリ条件から SPARQL の検索式を生成し, RDF グラフへの検索を行うことで発話パターンと文脈情報とのマッチングを行うようにした[5].

[＜上位クエリ変数＞$]＜リソース＞%＜パラメータ名＞[演算子＜上位クエリ変数＞$＜リソース＞%＜パラメータ名＞]

リソース	リソースの意味	戻りのデータ型	パラメータ名
preference	ユーザプリファレンス情報	リスト（文字列）	プリファレンス情報ID
slot	スロット情報	文字列	スロット番号(0以上の整数値)
eventFrame	情報フレーム	文字列	“イベント名@プロパティ名” 形式
regexGroup	正規表現のグループ	文字列	マッチングのグループ（0以上の整数値）
grounded	ユーザ／システム間の基盤化情報	リスト（文字列）	＜名前空間＞:＜プロパティ名＞
associated	システムが連想できる知識体系情報	リスト（文字列）	プロパティ名（述語）
nowTime	現在時刻（HH:mm）	文字列	なし
nowDayOfWeek	現在曜日（1～7）	整数値	なし
nowDateOfYear	現在日付（MMdd）	整数値	なし

演算子	右辺値	真条件
==	スカラー値	値が一致
	範囲（“[最小値,最大値]”）	左辺値が最小値以上、かつ最大値以下
	リスト（“{要素１,要素２,…}”）	要素数および各要素の値がすべて一致
+= または =	リテラル値、リソースおよびクエリ変数	• += の場合は追加 • =の場合は書き換え
	リスト（{“文字列１”,”文字列２”}）	
:=	両辺ともスカラー値	左辺文字列が右辺文字列の一部
+	両辺のリソースの連結（データ型に応じた）	"1" + "2", slot%0 + s1$text(0)
? [:]	三項演算子	expr{s1) < 3 ? "3未満" : "３以上"

図 6: クエリ構文と利用可能なリソースや演算子の抜粋

[＜上位クエリ変数＞$]＜リソース＞[%＜パラメータ名＞](<動的パラメータ>,])

リソース	リソースの意味	戻りのデータ型	パラメータ名
utterance	発話履歴を取得	リスト	system：システム発話、user：ユーザ発話、なし:両方
service	外部サービス	オブジェクト（Map）	＜サービス名＞:＜メソッド名＞
text	文字列へ変換	文字列	なし
regex	正規表現による語句の抽出	文字列	なし
expr	四則演算文字列の計算結果	整数値 or 実数値	なし
join	リストを１つのテキストに連結	リスト（文字列）	なし
split	テキストをリストに分割	リスト（文字列）	なし

図 7: 動的パラメタを含むクエリ構文の抜粋

クエリ条件においては, いくつかのリソースと呼ぶ変数を利用できる. 例えば, 図 6 における preference はユーザの好みなどの情報にアクセスするためのリソースで, 応援している野球チーム名などを取得できる. また, slot はユーザ応答 (図 8) における発話スロットパターンにおいて定義されているスロットとマッチした情報を取得できるリソースで, チーム名や人名を質問しているなどのユーザ発話の内容を理解するために利用できる. 図 8 における発話スロットパターンは, 関根の拡張固有表現 [10] とのマッチングを行うことができ, かつその拡張固有表現の周辺にある文脈とのマッチングを正規表現で指定することもできるようになっている. また, eventFrame は, 設定されているコンテンツを構造化した情報にアクセスするためのリソースで, 例えば図 4 のような情報が対話コンテキストに設定されている場合, 「eventFrame%試合結果:試合種別」で「日本シリーズ」という情報を取得できる. もし対話コンテキストに設定されているコンテンツ情報に, 「試合結果:試合種別」で表されるような情報が存在しない場合, このリソースに対するクエリが失敗し, このリソースが記述されたクエリ条件はアンマッチとなる. regexGroup というリソースでは, 発話文と図 4 の発話正規表現パターンとのマッチング結果を取得できるリソースで, 発話文と正規表現とがマッチした場合に, その値を取得することができる. この正規表現によるマッチングは拡張固有表現によるキーワードが含まれていない発話や拡張固有表現が出現した文脈を絞り混むことを可能にする.

クエリ条件には, 対話コンテキストへの対話状態の設定も行うことができる. grounded は, 対話基盤化情報を対話状態を更新 (図 6 の演算子「=」あるいは「+=」などを利用) あるいは取得するためのリソースである. 基盤化情報へのアクセスでは, 「grounded%野球:応援チーム」などのように RDF のおける名前空間と述語名をパラメタとして指定する.

図 8: ユーザ応答パターン

実世界における時間文脈と対応した雑談を行うためには, これらのリソースだけではなく, nowTime のように現在時刻を取得するためのリソースに加えて, 図 7 に示すような動的パラメタに応じた値を返すリソースが必要である. 特に, ドメインに特化した知識が必要な場合には, service のように外部サービスと連携して, 値を取得するリソースが必要となる. 例えば, 日本シリーズにおいて優勝するにはあと何勝すれば良いかなどの対話を生成するには, 「service%野球情報:日本シリーズ勝利試合数 (⟨ チーム名を表すクエリ変数 ⟩)」のようにニュース記事のみでは情報を取得困難な場合が多く, 外部サービスからその時点での対戦成績情報やドメイン知識を取得できるようにする必要がある.

[5]RDF グラフの構築, SPARQL による検索処理には Apache Jena を利用した (https://jena.apache.org/index.html)

3　野球雑談への適用

実際に, 提案システムによって, 時事ニュースの一つとして野球の対話を生成できることを示す. 野球情報については, データスタジアム社から試合結果についての最新情報をウェブ経由で提供を受けて利用した. 試合結果情報から試合における勝敗, ペナントレースの順位, 次の試合予定などのイベント情報を抽出し (図 4 参照), これらのイベント情報に基づいた対話プランを作成した. また球団, 野球選手の所属などの基本情報及び連想される情報を外部知識ベースとして用意し, 対話プランから利用できるようにした.

図 9 にユーザの応援チームが勝利した翌日における対話事例を示す. ユーザから入力された投手名から, 勝ち投手になったという応答により, 応援チームが勝利したことを伝達し, かつ勝利打を誰が打ったかも伝達している. また, 勝利投手のその時点における勝利数のように, その時点において意味ある対話が生成できている.

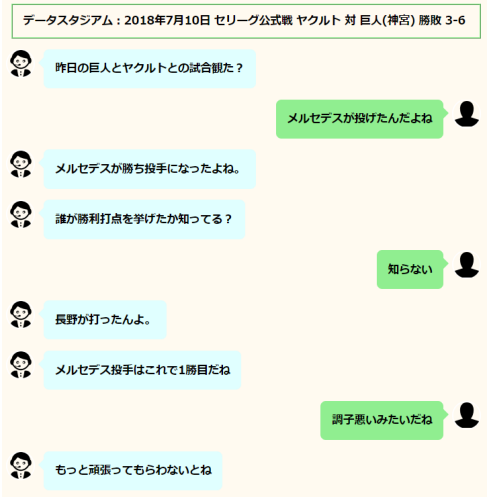

図 9: 対話事例 (左:システム, 右:ユーザ)

4　まとめ

リアルタイム情報に関する時事雑談を行えるようにするために, 隣接トリプル発話からなる対話プランに基づく対話システムを提案し, 実際に野球雑談において, 前日の野球結果についてその時点の状況に応じた対話が行えることを示した. しかし, 試合中の実況を元に対話を行うようなよりリアルタイム性が高い対話については, 未対応であるが, コンテンツの動的なアップデートなどの仕組みを導入し, 対応していくことが考えられる.

また今回, 対話プランを作成する際に, 事前に対話事例を収集し, それらの事例を元に対話プランとして一般化する作業は, クラウドソーシングにより人手で行なった. 今後は, 対話事例から対話プランを自動生成できるできるようにして行きたい. さらに, 空間的にも実世界にグラウンディングした対話への適用についても進めて行きたい.

謝辞

本研究において, データスタジアム株式会社様によりご提供した頂いたプロ野球の試合データを利用させて頂きました. ここに感謝の意を表します.

参考文献

[1] 渡邉亮裕, 田中剛, 藤本拓, 吉村健. 自然対話プラットフォームオープン化への取り組み. *SIG-SLUD*, Vol. B5, No. 02, pp. 56–57, oct 2018.

[2] Ashwin Ram et al. Conversational AI: the science behind the alexa prize. *CoRR*, Vol. abs/1801.03604, , 2018.

[3] 四倉晋平, 佐藤敏紀, 東山昌彦, 橋本泰一. Ai アシスタントプラットフォーム clova とそのスキル開発事例の紹介. *SIG-SLUD*, Vol. B5, No. 02, pp. 66–67, oct 2018.

[4] 灘本明代, 林正樹, 道家守, 浜口斉周, 田中克己. 係り受け構造及びシソーラスによる対話文生成と簡易演出技法を用いた web コンテンツの受動的視聴. In *DEWS2005*, 2005.

[5] 吉野幸一郎, 森信介, 河原達也. 述語項の類似度に基づく情報抽出・推薦を行う音声対話システム. 情報処理学会論文誌, Vol. 52, No. 12, pp. 3386–3397, dec 2011.

[6] 高津弘明, 福岡維新, 藤江真也, 林良彦, 小林哲則. 意図性の異なる多様な情報行動を可能とする音声対話システム. 人工知能学会論文誌, Vol. 33, No. 1, pp. 1–24, 2018.

[7] 目黒豊美, 杉山弘晃, 東中竜一郎. ルールベース発話生成と統計的発話生成の融合に基づく対話システムの構築. 人工知能学会全国大会論文集, Vol. 28, pp. 1–4, 2014.

[8] 東中竜一郎, 船越孝太郎, 稲葉通将, 角森唯子, 高橋哲朗, 赤間怜奈. 対話システムライブコンペティション. *SIG-SLUD*, Vol. B5, No. 02, pp. 106–111, oct 2018.

[9] 森田武史, 山口高平. Printeps アーキテクチャの構築と実践. 人工知能学会全国大会論文集, Vol. 29, pp. 1–4, 2015.

[10] Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. Extended named entity hierarchy. In *LREC 2002*, 2002.