抽象対話システム ver1.0

安藤光太郎, 鈴木航介, 西田圭嗣
株式会社サイシード

次世代の対話体験を生み出す抽象対話システムを提案する。本システムは対話状況の抽象化を行い、別の対話とマッチングすることでユーザーとボットの対話を実現する。個々のユーザーはシステムと 1 対 1 の会話をしている体験だが、実際には 1 対多、もしくは多対多のやり取りが発生している。これにより、従来の機械対話システムでは実現し得なかったレベルでの自然な対話の実現を目指す。さらに、実効的に現実世界の誰かと対話しているという特徴を活かし、リアルタイムでの情報交換を行うことができる。これにより、チャットをインターフェースとした即時的な情報検索システムに拡張することを計画している。

1 対話システムの概要

本システムは対話状況の抽象化を行い、別の対話とマッチングすることでユーザーとボットの対話を擬似的に実現する。個々のユーザーはシステムと 1 対 1 の会話をしている体験だが、実際には 1 対多、もしくは多対多のやり取りが発生している。

ユーザー間の対話が成立するフローを Fig.1 および Fig.2 で確認しよう。図に記載されているのはユーザーが見ているチャット画面で、右側の緑の吹き出しがユーザーからの発言を表し、左側の白の吹き出しがボットからの発言を表している。上の列をユーザー 1 とし、下の列をユーザー 2 とする。ユーザー 1 の時間軸を t' で表し、ユーザー 2 の時間軸を t で表す。右から左に向かって時間が進んでいるため、チャット画面に相手と自分の発言が追加されている。ユーザー 1 とユーザー 2 は必ずしもお互いに会話しているわけではなく、発言ごとに異なるユーザーとの会話となり得ることに注意されたい。まず対話状況を抽象的なパーツに分解し名前をつける。まず、図中赤枠の部分を「メッセージ」と呼び、その会話全体での最新の発言に相当する。メッセージに対応するものとして、1 時点前以降の発言の連なりを「タイムライン」と呼び、図中青枠部分に相当する。そしてタイムラインとメッセージのペアで構成されるデータを「ログデータ」と呼び、図中緑枠部分に相当する。チャット画面から行う会話は常にタイムラインという状態を持ち、その状態に対して履歴中のログデータからメッセージを作成し、そのメッセージが送信されるというのが、本システムでの会話の描像である。

ユーザー間の対話は Fig.1 に示す「タイムライン類似度判定」と Fig.2 に示す「メッセージの送信」の 2 つに分けて実行される。Fig.1 は、自分のタイムラインと他のタイムラインの類似度を走査する過程を示している。上の列、ユーザー 1 は時刻 $t' - 1$ 時点で「サイコちゃんって買い物が好きなんだ！」と発言しており、時刻 t' 時点でボットからの返信を待っている。ボットは返信を行うために、時刻 t' 時点のタイムラインと似ているタイムラインを検索し、似ているタイムラインとペアで存在するメッセージを返信材料として活用する。これは、タイムラインの状況が似ていれば、次に送られるメッセージも似ているはずという仮定に基づいている。

Fig.1 では時刻 t' 時点のタイムラインと他全てのタイムラインの類似度を計算した結果、時刻 t 時点のユーザー 2 のタイムラインと類似度が高いことを示している。実際、ふたつのタイムラインでは言い回しは違うが、会話の内容が類似していることが確認できる。類似するタイムラインが時刻 t のものとわかると、ボットは時刻 t 時点のログデータに含まれるメッセージで Fig.2 のように返信を作成する。返信材料となるメッセージは「そうですよ!あと、三越の屋上でクレーンゲームするのも好きです!」だが、他のチャットに送信しても問題ないように、メッセージの抽象化を行う。ユーザー 1 のチャットに送信されたメッセージを確認すると「そうだよ!デパートの屋上で遊ぶのも好き!」と文章の大まかな意味を保ちながらも、具体的な情報が抽象化されていることがわかる。

2 対話の状態化

本システムでは、ユーザー 1 名とボット 1 名間での対話を実現する。この対話をひとつの状態として捉えるために、「タイムライン」という概念を定義する。Fig.3 にチャット画面とタイムラインの対応関係を示す。タイムラインはユーザーおよびボットからの「メッセージ」の並びから構成されており、それぞれの「メッセージ」は文章、発言時刻、発話者という情報を持っている。したがって、発言内容、その並び順、それぞれの発言時刻などが従属変数となりタイムラインは個別の状態を持つ。まず「メッセージ」を状態ベクトル $\mathbf{m}$ として定義する。

$$\mathbf{m}_i = (t_i, s_i, r_i)^{\mathrm{T}} \tag{1}$$

ここで、添字 i は現在から遡って i 番目に送信されたことを表す。また t_i は発言時刻を、s_i はメッセージの発言文章を、r_i はユーザーの発言かボットの発言かの識別子である、発言者タイプを表す。発言文章は必ずしも発言された文章そのものではなく、抽象化など任意の処理が加えられたものであっても構わない。$\mathbf{m}$ で構

author's e-mail: ando@sciseed.jp

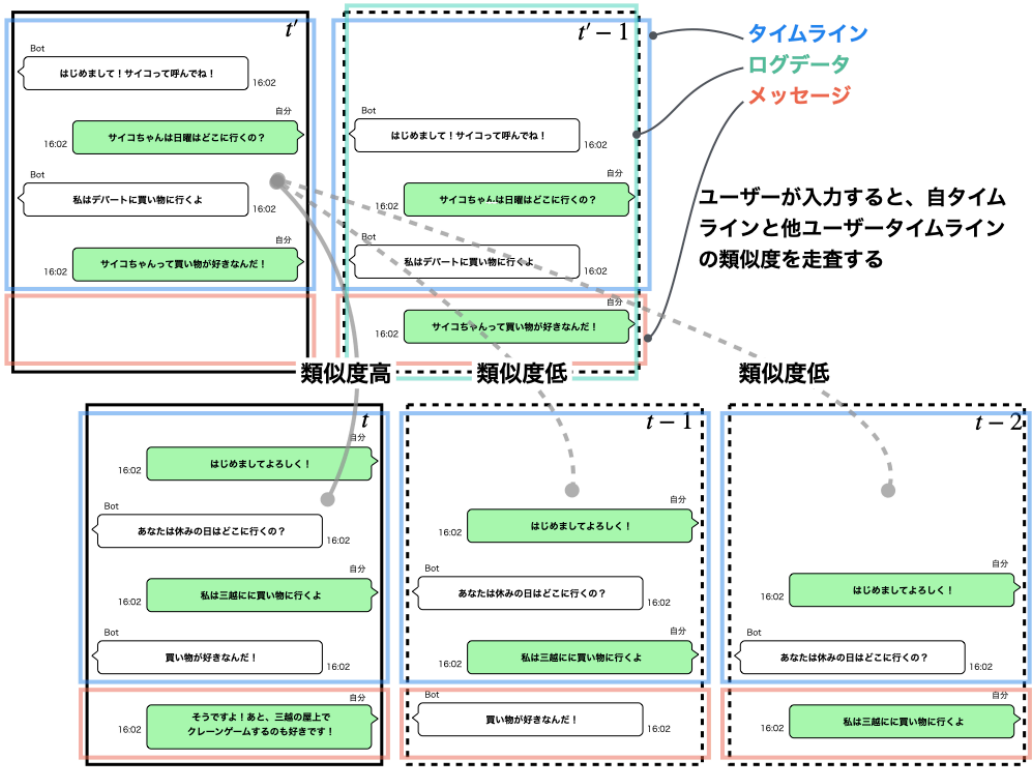

Fig. 1　タイムライン類似度判定

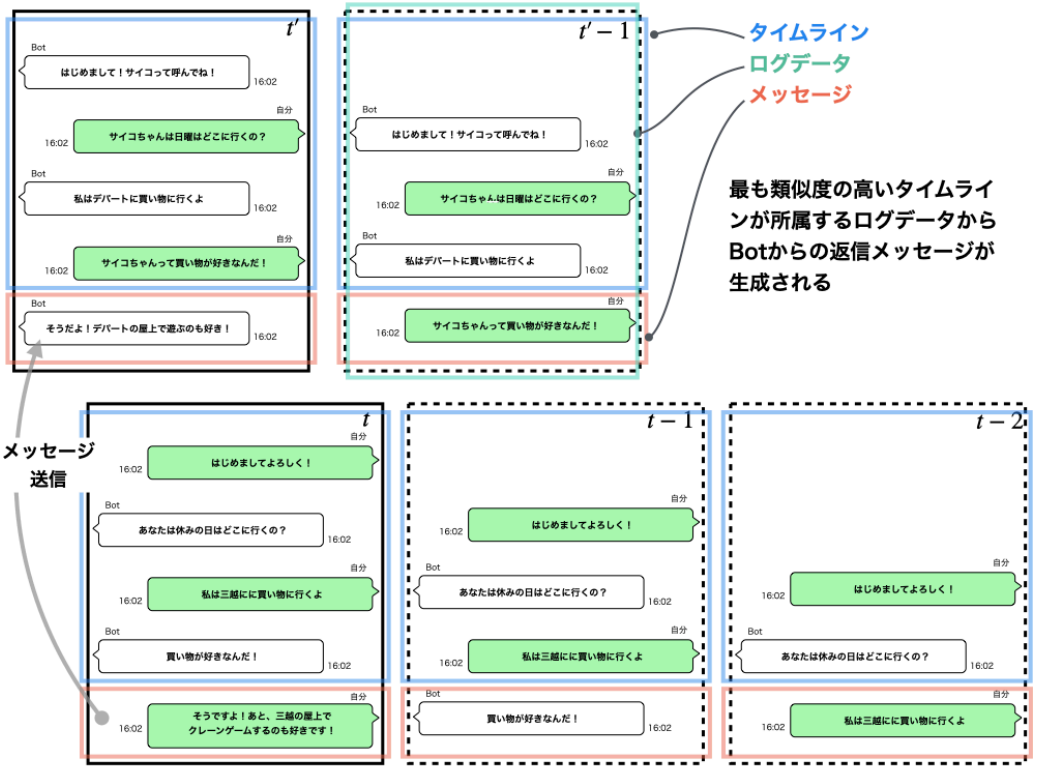

Fig. 2　メッセージの送信

成される状態ベクトル $\mathbf{x}$ を次のように定義する。

$$\mathbf{x} = (\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n)^{\mathrm{T}} \tag{2}$$

$$= (t_1, t_n, \cdots, s_1, \cdots, s_n, \cdots, r_1, \cdots, r_n)^{\mathrm{T}} \tag{3}$$

最後にタイムライン状態ベクトルとするために、特殊なメッセージを追加する必要がある。タイムラインの中には会話が活発に交わされる (活性な) ものもあれば、長期間にわたって会話が全く交わされない (非活性な) ものも存在する。会話が長期間ない状態を類似度として反映するために、タイムラインには必ず仮想的に現在時刻を発言時刻とする null メッセージ $\mathbf{m}_*$ が入っているものとする。以上より、タイムラインを $\mathbf{x}_*$ として以下のように定義する。

$$\mathbf{x}_* = (\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_i, \mathbf{m}_*)^{\mathrm{T}} \tag{4}$$

$$\mathbf{m}_* = (t_*, s_*, r_*)^{\mathrm{T}} \tag{5}$$

ただし、t_* は現在時刻、s_* は null 文、r_* は必ず自分を指し示す。これにより、あらゆるタイムラインにおいて null メッセージ $\mathbf{m}_*$ 同士の類似度は 1 となる。よって、非活性なタイムラインでは以前の会話の類似度は、null メッセージの類似度に対して無視できるほど小さくなり、結果的に非活性なタイムライン同士でマッチングする。一方、活性なタイムラインでは null メッセージの類似度の影響が希薄化するため他のメッセージの類似度でマッチングすることができる。

タイムライン $\mathbf{x}_*$ は最新のメッセージに $\mathbf{m}_*$ が存在するため、今この瞬間に固有の状態ベクトルである。タイムラインはメッセージが送信されたタイミングで更新され、古いタイムラインは送信されたメッセージと共にログデータとして履歴に保存される (Fig.4)。ログデータは送信されたメッセージ $\mathbf{m}_s$ と送信時のタイムライン $\mathbf{x}_*$ を用いて、以下のように定義される。

$$\mathbf{x}_n = (\mathbf{x}_*, \mathbf{m}_s)^{\mathrm{T}} \tag{6}$$

$$= (\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_i, \mathbf{m}_*, \mathbf{m}_s)^{\mathrm{T}} \tag{7}$$

ここで、添字 n は n 番目のログデータであることを示している。ログデータは複数存在するのに対して、タイムラインはひとつしか存在しないことに注意する。なお便宜上、$\mathbf{x}_n$ に含まれる null メッセージ $\mathbf{m}_*$ の時刻 t_* は送信されたメッセージ $\mathbf{m}_s$ の時刻 t_s と同じものとする。

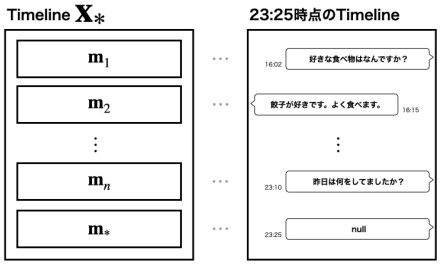

Fig. 3　対話の状態ベクトル化

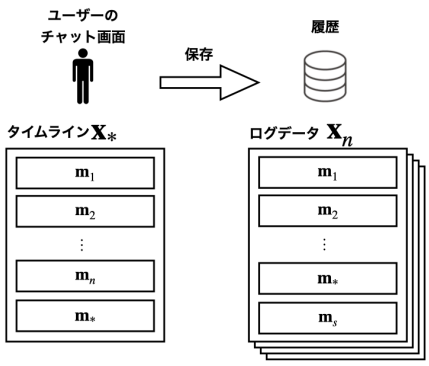

Fig. 4　タイムラインはログデータとして履歴に保存される

3　タイムライン類似度

本節では、ユーザー A のタイムラインとユーザー B のタイムラインの類似度の計算方法を示す。ユーザー A のタイムライン $\mathbf{x}_a$ には N 個のメッセージが存在し、ユーザー B のタイムライン $\mathbf{x}_b$ には M 個のメッセージが存在すると仮定する。タイムライン $\mathbf{x}_a$ とタイムライン $\mathbf{x}_b$ の類似度 L は関数 f を用いて

$$L = f(\mathbf{x}_a, \mathbf{x}_b) \tag{8}$$

と計算される。ユーザー A の i 番目のメッセージ $\mathbf{m}_i$ と、ユーザー B の j 番目のメッセージ $\mathbf{m}_j$ の類似度 g_{ij} を関

数 g を用いて

$$g_{ij} = g(\mathbf{m}_i, \mathbf{m}_j) \tag{9}$$

と定義し、タイムラインの類似度関数 f を

$$f(\mathbf{x}_a, \mathbf{x}_b) = \frac{2}{N+M} \sum_{i}^{N} \max\{g_{i1}, g_{i2}, \cdots, g_{iM}\} \tag{10}$$

と定義する。ただし $0.0 \leq f \leq 1.0$ である。メッセージ類似度 g_{ij} は時間類似度 w_{ij} と文章類似度 h_{ij} の積で以下のように定義する。

$$\begin{aligned} g_{ij} &= g(\mathbf{m}_i, \mathbf{m}_j) \\ &= w_{ij} h_{ij} \quad (0 \leq w_{ij} \leq 1, \quad 0 \leq h_{ij} \leq 1) \end{aligned} \tag{11}$$

時間類似度 w_{ij} はメッセージ間の発言時刻の時間的な近さを表し

$$\begin{aligned} w_{ij} &= w(t_i, t_j) \\ &= \exp(-a|(t_{i,base} - t_i) - (t_{j,base} - t_j)|) \end{aligned} \tag{12}$$

で計算される。ここで a は $a > 0$ を満たす任意定数であり、$t_{i,base}$ は相対時刻に変換するための基準時刻で、i 番目のメッセージが所属するタイムラインの最新の発言時刻である。また、文章類似度は Levenshtein 距離によって計算されるが、発話者タイプの扱いに注意が必要である。発話者タイプ r_i を自分とボットに関して反転させたものを r_i^* と表すことにすると、文章類似度 w_{ij} は

$$\begin{aligned} h_{ij} &= h(s_i, r_i, s_j, r_j) \\ &= \begin{cases} Levenshtein(s_i, s_j) & (\text{if} \quad r_i = r_j^*) \\ 0 & (\text{otherwise}) \end{cases} \end{aligned} \tag{13}$$

で計算される。

続いて、タイムライン類似度を計算する際に、タイムライン自体の古さを考慮できるようにメッセージ類似度を拡張する。式 (12) で示した通り、メッセージ類似度は相対的な時間差で決定される。これに加えて、メッセージ自体の絶対時間による類似度への忘却係数をかけることにする。これは人間の記憶が 1 年前のことより、昨日のことのほうが思い出しやすい傾向を持つことへのアナロジーである。そこで、i 番目のメッセージに対する忘却係数 k_i を関数 k を用いて

$$k_i = k(t_i) = \exp(-b|t_{base} - t_i|) \tag{14}$$

と定義する。ここで b は $b > 0$ を満たす任意定数である。式 (9) で計算したユーザー A の i 番目のメッセージ $\mathbf{m}_i$ と、ユーザー B の j 番目のメッセージ $\mathbf{m}_j$ 間の類似度は、平均化された忘却係数 $(k_i + k_j)/2$ をかけて

$$g(\mathbf{m}_i, \mathbf{m}_j) = \frac{k_i + k_j}{2} w_{ij} h_{ij} \tag{15}$$

となる。忘却効果を無視する場合は $k_i = k_j = 1$ とすればよく、この時、式 (15) は式 (11) に一致する。

4　類似タイムライン検索

ユーザーからボットに対して発言が行われた時、ユーザーに対するボットからの返答は類似するタイムラインを検索することで実現される。タイムラインは質問と返答の履歴データであるので、現在のタイムライン状態に極めて近い別のタイムラインを見つけることができれば、そのタイムラインからユーザーの発言 (質問) に近い質問に対する返答を引き出し、ボットからの返答として使うことができる。したがって、ボットからの返答作成作業は、類似するタイムラインを検索する作業と同値である。本節では、まずタイムライン類似度判定を通じて返答計画を設定する方法を示す。続いてタイムライン類似度判定の計算量を削減する枝刈り戦略を示す。返答計画の設定方法を示すにあたりログデータという量を定義する。n 番目の発言時点でのログデータを状態ベクトル $\mathbf{y}_n$ として

$$\mathbf{y}_n = (\mathbf{m}_n, \mathbf{x}_{n-1}, \delta)^T \tag{16}$$

と定義する。ここで $\mathbf{m}_n$ は最新時刻 n 番目の発言、$\mathbf{x}_{n-1}$ は 1 期前でのタイムライン、δ は最新メッセージと 1 期前のメッセージの時間差である。返答計画の基本方針を示す。今、ユーザーから新規入力されたメッセージによって新しく作成されたタイムラインを $\mathbf{x}'$ とする。まずタイムライン類似度が最大となるタイムライン $\tilde{\mathbf{x}}$ を見つける：

$$\tilde{\mathbf{x}} = \arg\max_{\mathbf{x}} f(\mathbf{x}', \mathbf{x}) \tag{17}$$

$\tilde{\mathbf{x}}$ が所属するログデータ $\tilde{\mathbf{y}}_n = \left(\tilde{\mathbf{m}}_n, \tilde{\mathbf{x}}_{n-1}, \tilde{\delta}\right)^T$ を特定する。これによりユーザーは $\tilde{\delta}$ 秒後にメッセージ $\tilde{\mathbf{m}}_n$ を受け取ることになる。

タイムラインは

$$\mathbf{x} = (\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n)^{\mathrm{T}} \tag{18}$$

となっており、最新のメッセージ $\mathbf{m}_n$ の発言時刻 t_n を基準として、過去のメッセージの発言時刻の相対時刻を計算する。しかし、t_n を基準とした類似度は一般に極大値とならず、類似度が極大値を特定するには t_n に対してオフセット δt_i を与えて、類似度が最大となる基準時刻を探索する必要がある。オフセット δt_i を加えたタイムラインを $\mathbf{x}'_i$ は

$$\mathbf{x}'_i = \mathbf{x}_i + \delta t_i \tag{19}$$

で定義される。δt_i を変化させることで類似度 $f(\mathbf{x}_i, \mathbf{x}'_i)$ が極大となる値をタイムライン間の類似度とする。

5　メッセージの送受信

前節までででタイムライン類似度の計算方法を示した。本節では類似タイムラインが見つかった場合に、メッセージの送受信がどのように行われるかを示す。

まず、Fig.5 でメッセージ受信の流れを確認する。図中、人のアイコンが一人のユーザーを表し、各ユー

ザーはタイムライン $\mathbf{x}_*$ とログデータ $\mathbf{x}_1, \cdots, \mathbf{x}_n$ を持っている。図中、左側に居るユーザーをユーザー A とする。ユーザー A がメッセージを送信すると、ボットからの返信を待つ状態になる。ユーザー A の更新後のタイムラインと、全ログデータのタイムラインとの類似度を計算し、類似度が最大となるタイムラインが所属するログデータからメッセージが生成され、送信される。

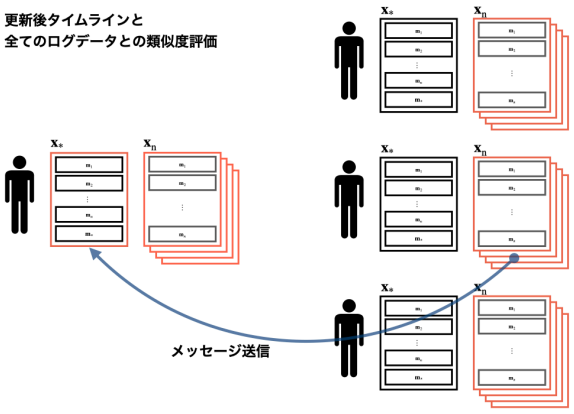

Fig. 5　メッセージの受信

次に、Fig.6 でメッセージ送信の流れを確認する。基本的には Fig.5 の逆の流れである。他のユーザーが発言するとタイムラインが更新され、ユーザー A のログデータのタイムラインとの類似度計算が行われる。他ユーザーのタイムラインにとって、ユーザー A のログデータの中にタイムライン類似度が最大となるものがあった場合、そのログデータからメッセージが送信されることになる。

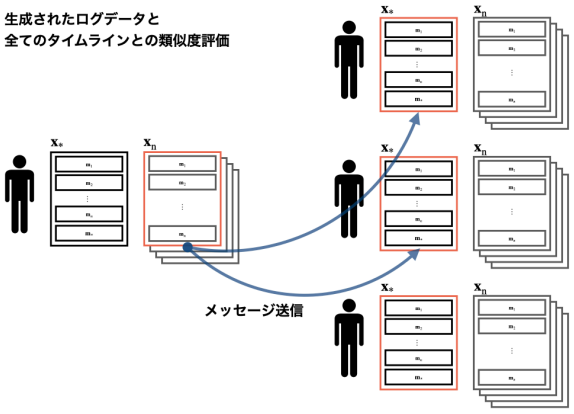

Fig. 6　メッセージの送信

6　まとめ

次世代の対話体験を生み出す抽象対話システムを提案した。本システムではユーザーとボットの対話状況を抽象化し、別の対話とマッチングすることでユーザーとボットの対話を仮想的に実現している。ユーザーはボットと 1 対 1 の対話を体験するが、実際には 1 対多、もしくは多対多のやり取りが発生している。対話同士のマッチングを実現する核は、対話状況をタイムライン状態ベクトルとしてすることにある。状態ベクトルとすることで、タイムライン同士の類似度が計算可能となり、ユーザーからボットに対する発言をキーとして、発言に対する最適な応答を他のタイムラインから検索することが可能となる。これにより、ユーザーはボットとの対話を体験するが、実効的には現実世界の誰かと対話していることになる。この特徴を活かし、将来的にはチャットをインターフェースとした即時的な情報検索システムに拡張することを計画している。