# 非同期型チャットアプリケーションに対応した雑談システムの構築と評価

# Development and Evaluation of a Chatbot for Asynchronous Chat Applications

若原 基[1]*　ジェプカ ラファウ[1]　荒木 健治[1]
Hajime Wakahara,[1]　Rafał Rzepka,[1]　Kenji Araki[1]

[1] 北海道大学 大学院情報科学研究科
[1] Graduate School of Information Science and Technology, Hokkaido University

**Abstract:** We propose a method for utterance control by which chatbots are able to use asynchronous chat applications. In established methods, chatbots control utterances one after the other. In our method we use a segment model, a segment chain model and a segment relation model between the input and output modules. By using these models, input and output modules can be executed parallelly so that chatbots are able to control utterances asynchronously and output an utterance with their proper timing. We implemented the algorithm and conducted a dialog experiment using 11 users. We compared this asynchronous chat system with a synchronous chat system applying the Semantic Differential Method(5-point scale). As a result, positive feelings toward the asynchronous system were 1.45 points higher than the synchronous system. Moreover, humanity, interestingness and approachablity were 1.28, 1.64 and 1.28 points higher, respectively.

## 1 はじめに

知能ロボット[1]の研究は，ロボットと人間とが協調しあう社会をゴールとする，ひとつの大きな時代の流れであると捉えられる．雑談システム研究は，そうした社会の実現の一端を担うものであると我々は考える．人間の日常会話においては，たとえば感情の昂ぶりなどが生じた際に一方が立て続けにしゃべる，話の流れを断ち切る，間を置くなど，様々なタイミングにおける言語表現が行われる．よって，人間との協調を目指す雑談システムの研究においては，そのような人間らしい言語表現を受容できるユーザビリティを持ち，多様なタイミングでの入力に対して応答が行えるシステムを構築する必要があると我々は考えている．

雑談システムについてはすでに多数の研究が行われているが [1, 2, 3, 4]，十分な成果が出ているとは言い難い．近年の雑談システム研究の一つとして Higuchi らの手法 [5] が挙げられる．Higuchi らはウェブの検索エンジンを用いてユーザ発話に含まれる語の関連語句を抽出し，主語と述語の組み合わせに適切な文末表現を付与することによってユーザ満足度の高いシステム発話の出力を実現している．しかし，この手法は発話処理のスピードが検索エンジンなどの外部要因に依存するため，リアルタイム処理には向いていない．

また，Higuchi らの手法では逐次処理による発話入出力を行っている．すなわち，ユーザの発話入力一回につきシステムの発話が一回出力されるというものである．本稿ではこのようなシステムを同期型のシステムと呼ぶ．それに対し，一般的に用いられるチャットアプリケーションのように各々が任意のタイミングで発言してよいシステムを本稿では非同期型のシステムと呼ぶ．

非同期型のシステムと類似の高い研究として，発話タイミングの研究が挙げられる．西村らの手法 [6, 7] では，音声対話における，ユーザに対する発話タイミングやあいづちの制御を行い，円滑な対話を実現している．また，河添らは発話の内容や相手との関係などの発話要因から発話タイミングを決定するマルチエージェントシステムを提案している [8]．

同期型のチャットアプリケーションを仮定すると，チャットを行うメンバーは物理的に相互協調を行うことが前提であり，雑談システムもまた，交替で発話を行うという協調のルールを外部的に与えられて構築される．一方，非同期型のチャットアプリケーションにおいては

*北海道大学 大学院情報科学研究科 メディアネットワーク専攻 情報メディア学講座 言語メディア学研究室
060-0814 札幌市北区北 14 条西 9 丁目
E-mail: hadzimme@media.eng.hokudai.ac.jp
[1]http://www.ai-gakkai.or.jp/jsai/activity/rloadmap.html

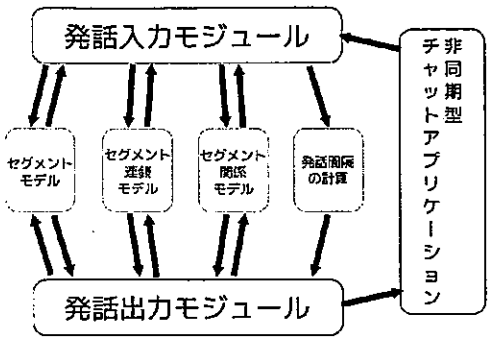


図 1: システム全体図

そのような物理的協調が前提とされていないため，雑談システムは，人間が行うように，発話タイミングのような相互協調のルールを内部に持って発話を行う必要がある．こうした雑談システムの研究は，未だ十分に行われているとは言えない．

そこで本稿では，モデルを介して入力系と出力系とが並行動作し，非同期型チャットアプリケーションにおける相互作用を可能とする雑談システムを提案する．

## 2 提案手法

本稿で扱う雑談システムとは，ユーザ発話に対し何も応答のできない状態から，ユーザとの対話からの応答パターンの獲得を行い，少しずつ雑談ができるようになっていくものである．

雑談システムにおいても，対話を実現するための効率などの面から，ユーザ発話への応答パターンなどをあらかじめインプットしておくことが考えられる．しかしながら，我々の研究は白紙状態からの言語発達メカニズムの構築を目的としており，現時点ではシステムに対しこのような操作を行わないことを前提とする．

### 2.1 システムの概要

システムの全体像を図1に示す．このシステムは，これまでに我々が提案した手法 [9] を非同期型チャットアプリケーションに対応させたものである．発話の入出力系では，システムによる発話の非同期的な入出力を可能とするために，2.2で述べる統計モデルを介し，発話入力モジュールと発話出力モジュールとが独立して動作する．これら二つのモジュールは，連携を取りつつ並行して動作するプロセスとして，同時に起動される．

### 2.2 発話のための統計モデル

本手法では，システムによる発話の認識を定義するために，以下の三つの統計モデルを定める．

- セグメントモデル
- セグメント連鎖モデル
- セグメント関係モデル

システムによる発話の入出力は，これらのモデルに基づいて処理される．以下に，各モデルの定義について述べる．

#### 2.2.1 セグメントモデル

本手法で定めるセグメントモデルとは，言葉に単位を与え，入力発話を認識するためのモデルである．

言葉において意味を持つ最小の単位は一般的に単語であると言われ，これは形態素解析を用いることにより得ることが可能である．しかし，文字チャットなどで用いられる口語文を，既存の形態素解析器を用いて正しく単語に分割することは困難である．

そこで，このセグメントモデルでは，連続する 4 文字のセグメントを言葉の単位であると定義し，一連の処理を行うこととする．セグメントの文字数は，2.2.2 で述べる N-gram モデルにおいて，精度を落とさず，発話生成の際に文法が崩れない文字数として，本手法においてあらかじめ定めたものである．

まず，ユーザによる入力発話がすべて，$k$ 文字（4 文字以上）からなる文字列

$$s = c_1,\ c_2,\ \cdots,\ c_k\ (k \geq 4)$$

であると仮定する．ここで，$s$ が入力された時刻 $t$ を得ておく．$s$ から，連続する 4 文字のセグメント

$$\begin{aligned} seg_1 &= c_1, c_2, c_3, c_4 \\ seg_2 &= c_2, c_3, c_4, c_5 \\ &\vdots \\ seg_{k-3} &= c_{k-3}, c_{k-2}, c_{k-1}, c_k \end{aligned}$$

をすべて抽出する．そして，$t$ をセグメントの更新時刻とし，セグメントとともにデータベースのセグメントテーブルに保存する．既にセグメントがテーブルに存在する場合は，更新時刻を $t$ により上書きする．セグメントとその更新時刻は，発話入力モジュールと発話出力モジュールの両方から参照することができる．また，本手法で用いるデータベースは，複数の並行プロセスからのアクセスがあっても問題なく更新処理が行えるものでなければならない．

なお本手法では，このセグメントモデルによる一元的な発話処理を行うこととする．そのために 4 文字未満の入力発話は捨象され，処理が行われない．

### 2.2.2 セグメント連鎖モデル

本手法で定めるセグメント連鎖モデルとは，セグメントモデルにおけるセグメントを言葉の単位とする，システムの発話生成に用いるための言語モデルである．

2.2.1 でユーザ発話から得られたセグメント列

$$seg_1,\ seg_2,\ \cdots,\ seg_{k-3}$$

の先頭と末尾とに未定義値を付与し，連続する二つのセグメントの対

$$\begin{aligned} chain_1 &= null,\ seg_1 \\ chain_2 &= seg_1,\ seg_2 \\ &\vdots \\ chain_{k-2} &= seg_{k-3},\ null \end{aligned}$$

をすべて抽出する．そして，出現頻度 $freq_{chain}$ とともにデータベースのセグメント連鎖テーブルに保存する．既にセグメント対がテーブルに存在する場合は，$freq_{chain}$ に 1を加算する．存在しない場合は，$freq_{chain}$ の初期値を 1 とし，新たにカラムを追加する．

このように連続する 4 文字からなるセグメントの連鎖対を与えることは，5-gram の N-gram モデルを考えることに等しい．

発話出力モジュールにおいては，まず，このモデルにシードとなるセグメントが与えられる．そして，テーブル上の $freq_{chain}$ を参照し，文字列の可読方向とその逆方向に，未定義値に至るまでマルコフ連鎖を行う．さらに，得られたセグメント列の先頭と末尾から未定義値を除去し，文字列への変換を行う．

### 2.2.3 セグメント関係モデル

本手法で定めるセグメント関係モデルとは，システムによるユーザ発話からの応答パターン獲得に用いるモデルである．

ユーザとシステムとの両者による発話の集合において，ある時刻 $t$ のユーザ発話を $u_{next}$ とし，$u_{next}$ からの近傍範囲の中にある，$t$ よりも前の発話のひとつを $u_{prev}$ とする．近傍範囲はチャットアプリケーションの実装方法に依存して決定されるが，本手法では，発話の入出力系において得られる発話間隔の秒数とする．発話間隔の決定については，2.3.2 で述べる．

ここで，$u_{prev}$ からは

$$prev_1,\ prev_2,\ \cdots,\ prev_m$$

また，$u_{next}$ からは

$$next_1,\ next_2,\ \cdots,\ next_n$$

というセグメント列が抽出されるとする．

ここで，$prev$ と $next$ の取りうるすべての組み合わせ

$$\begin{aligned} ref_1 &= prev_1,\ next_1 \\ ref_2 &= prev_1,\ next_2 \\ &\vdots \\ ref_{m\cdot n} &= prev_m,\ next_n \end{aligned}$$

を関連セグメント対として共起頻度 $freq_{rel}$ とともにデータベースのセグメント関係テーブルに保存する．$freq_{rel}$ については 2.2.2 における $freq_{chain}$ と同様，初期化と更新を行う．

発話出力モジュールにおいては，あるセグメント $seg_i$ が入力発話から得られているとき，テーブルに保存された $prev$ セグメントに対し $seg_i$ をクエリとする検索を行い，対応するすべての $next$ セグメントを得る．このようにして得られるセグメント群に含まれるセグメント $seg_{rel}$ を，本稿では $seg_i$ に対する関連セグメントと呼ぶ．また，テーブル上で $seg_{rel}$ に対応する出現頻度 $freq_{rel}$，セグメント関係テーブルのカラム総数 $T$，そして $seg_i$ に対する関連セグメントの総数 $count$ から次式で得られる $deg_{rel}$ を，本稿では $seg_{rel}$ の $seg_i$ に対する関連度と呼ぶ．

$$deg_{rel} = freq_{rel} \cdot \log \frac{T}{count}$$

## 2.3 発話の入出力

システムは，2.2 で述べたモデルに基づき，発話の入出力を行う．以下に，発話入力モジュール，システムの発話間隔の決定，そして発話出力モジュールについて述べる．

### 2.3.1 発話入力モジュール

発話入力のプロセスは，ユーザの各発話ごとに別スレッドで行い，前のユーザ発話の処理が終了するのを待つことなく，ユーザ発話の入力があった時点でその都度開始する．

まず，2.2.1 のセグメントモデルにおいてユーザ発話を分割し，セグメント列とそれぞれの更新時間を得る．そして，2.2.2 のセグメント連鎖モデルにおいて発話生成に用いるマルコフ連鎖の統計量を得る．さらに，2.2.3 のセグメント関係モデルにおいてシステムの応答パターンを獲得する．

### 2.3.2 システムの発話間隔の決定

同期型の雑談システムでは，システムの発話タイミングは常に，1つのユーザ発話の直後に1回，と固定されている．しかし，我々の提案する非同期型の雑談システムでは，そのタイミングはユーザ発話に直接依存せず，システムの内部によって決定する．

人間による発話タイミング決定の根拠としては，自身の性格や対話における主導権などが考えられているが [8, 10]，本手法ではまず，それらの要因のうち最も基本的なものの一つであると考えられる，相手の発話ペースへの同調を雑談システムに導入する．

$n$ 回目のユーザ発話が入力された時刻 $t_n$ から，$n-1$ 回目のユーザ発話が入力された時刻 $t_{n-1}$ を差し引いた秒数の集合 $A$ を

$$A = \{sec_1,\ sec_2,\ \cdots,\ sec_{n-1}\}$$

とする．ここで，これらの秒数の平均値を $\mu$，標準偏差を $\sigma$ とおき，$A$ の部分集合 $B$ を

$$B = \{x \mid \mu - \sigma < x < \mu + \sigma,\ x \in A\}$$

とする．さらに，確率変数 $Y$ について

$$y \in B,\ P(Y = y) = \frac{1}{|B|}$$

とし，システムの発話間隔を

$$\begin{cases} Y, & |B| > 0 \\ \mu, & |B| < 1 \end{cases}$$

として得る．

### 2.3.3 発話出力モジュール

発話出力のプロセスは，2.3.2で得られる発話間隔の秒数ごとに開始される．

まず，セグメントモデルにおいてセグメントテーブルを参照し，前回のシステム発話が出力された時刻よりあとに更新された全てのセグメントを得る．ここで得られたセグメント群を $S_{input}$ とする．

つぎに，セグメント関係モデルにおいてセグメント関係テーブルを参照し，$S_{input}$ に含まれる各セグメントに対する全ての関連セグメントを得る．ここで得られた関連セグメント群を $S_{rel}$ とする．

そして，$S_{rel}$ に含まれるセグメントのうち最も関連度の高いセグメントをシードのセグメントとし，セグメント連鎖モデルにおいてシステムの出力発話を得る．なお，ここでシードのセグメントが得られなかった場合，システムは発話を行わない．

表 1: 対話例 1（U: ユーザ，S: システム）

| | |
|---|---|
| U | お酒飲みたいね |
| S | 居酒屋でグイッと飲みたいね。 |
| U | 居酒屋いいですね |
| S | ビール飲みたいね。 |
| U | ビールが好きですか。 |
| U | 僕はビールが好きですよ |
| U | こないだ飲んだビールはおいしかったよ |
| S | ビール飲みたいなあ |
| U | ラーメンとか堂ですか |
| U | どうですか |
| U | ラーメン食べませんか |
| S | ラーメンとか食べましょう |
| U | これからお風呂に入ってきます。 |
| U | あたたまりますよー |
| S | あたたまるなあ。 |
| U | そうなんですよ。 |
| U | 冬は寒いので |
| U | お風呂は最高です |
| U | そして、お風呂のあとに、ビールです。 |
| S | そして、お風呂のあとに、ビールね。 |
| S | そして、お風呂のあとに、ビールね。 |
| U | グイッとね。 |
| S | お酒飲みたいね。 |
| U | そろそろお風呂に入ります。 |
| S | またあとでね。 |
| U | あたたまってきます。 |
| U | あとでね。 |
| S | あとでね。 |
| S | あたたまるなあ。 |

## 3 予備実験

2章で述べたアルゴリズムを実装し，自然言語による対話を行った．そして，システムとの対話にどのような特徴が見られるか観測を行った．

実験に用いるチャットアプリケーションの構築には，Ruby on Rails[2]のライブラリである ShootingStar[3]を使用した．ShootingStar は，サーバーと非同期通信を行うウェブアプリケーションを構築するためのライブラリである．

### 3.1 対話の成功例

システムとの対話例を表1に示す．システムは，初めのうちは的確な受け答えがほとんどできないが，ユーザがシステム発話に呼応し，話題を広げるように話し

[2]http://rubyonrails.org/
[3]http://rubyforge.org/projects/shooting-star/

表 2: 対話例 2 (U: ユーザ, S: システム)

| | |
|---|---|
| U | 返事ができるかどうかは難しいよね |
| S | ラーメンが食べたいよ |

かけてあげることで，少しずつ表 1 のような受け答えができるようになる．

表 1 の例では，前半が飲食の話題であるが，ユーザが途中で入浴に関する発話に切り替えるところで，ユーザの「あたたまりますよ」という発話があったあとに「あたたまるなあ」という応答を行っている．対話ログを遡ると，過去に入浴の話を行ってはおらず，「晩ごはんに鍋を食べるとあたたまる」という話題の雑談から例のような応答パターンを獲得していた．

また，それに続いて，「お風呂は最高です」に至るまでユーザによる連続的な発話が存在する．システムは，このようなユーザによる一方的な発話からも応答パターンの獲得を行っている．このようなケースは，ユーザによる一種の教示であると考えられる．このとき，ユーザが対話の主導権を強制的に奪う必要があるが，これはシステムが非同期型チャットに対応することの有効性の一つであると捉えられる．

### 3.2 対話の失敗例

まだ応答パターンの獲得が十分に行われていない段階で，表 2 に示すようなやりとりが見られた．ここでは，ユーザ発話の内容とシステムによる応答とに齟齬が生じている．

このような齟齬が生じる条件として考えられるのは，以下のようなことである．

- 「返事ができるかどうかは難しいよね」という文から抽出される「しいよね」というセグメントについて，高い関連度を持った関連セグメントが得られていた
- 抽出されたそれ以外のセグメントについては，高い関連度を持った関連セグメントが得られていなかった
- 「しいよね」というセグメントは，過去に行われた対話における「おいしいよね」という文字列から得られたものであった
- 「しいよね」の関連セグメントとして得られている「ラーメン」といったようなセグメントをもとに，出力発話が生成された

本来「しいよね」というセグメントは「うれしいよね」「怪しいよね」といった様々な形容表現の語尾として現れるものであるため，多くのセグメントとの関連を持ちうるセグメントであると考えられる．我々の先行研究 [9] における調査で，そのような特徴を持つセグメントには活用語尾や助詞・助動詞句が多く見られる傾向にあることがわかっている．そのため，**2.2.3** における関連度の算出においては正規化を行い，そのようなセグメントの関連度を低く算出するようにしている．

しかし，得られている関連セグメントが少ない場合は正規化が正常に行われず，誤った関連度に基づいた関連セグメントを発話生成のシードとしてしまうことが考えられる．ただし，応答パターンの獲得が進むことによってこのような誤りは改善されることが予想される．

## 4 実験

提案した非同期型の雑談システムと同期型の雑談システムとの比較実験を行った．11 名の被験者による実験の概要を以下に述べる．被験者の年代は，20 代が 4 名，30 代が 6 名，40 代が 1 名であり，性別については男性が 6 名，女性が 5 名であった．また，学生は 1 名で，ほかの 10 名は社会人であった．日常生活におけるコンピュータの使用頻度については，主としてウェブサイト閲覧を行っている人が 1 名，文書作成等を行う人が 4 名，仕事・学業等で専門的に扱っている人が 3 名，仕事・学業以外にも携行して使用している人が 3 名であった．キーボードによる文字入力の熟練度については，時々手元を見るという人が 4 名，タッチタイピングを行っている人が 7 名であった．

### 4.1 実験条件

被験者にはインターネット上のチャットアプリケーションにアクセスし，同期型・非同期型両方のチャットアプリにおいて雑談システムとの自由対話を行ってもらった．発話のターン数や制限時間などは設けず，任意のタイミングでシステムとの対話をやめるよう被験者に指示した．

また，被験者には「まだシステムは言葉を覚え始めたばかりであり，人間との対話から少しずつ言葉を覚えていく」ということをあらかじめ説明し，両方のシステムに言葉を教えることを試みるよう促した．

比較対象としての同期型システムには，提案手法における発話の入出力系を同期型に改造したものを用いた．このとき，**2.2.3** で述べた近傍範囲を提案手法と同様の方法で決定することができないため，比較対象のシステムにおいては，この近傍範囲を発話 5 ターン分と定めた．また，セグメント関係モデルにおいて関連

表 3: 各評価の平均値（括弧内は標準偏差）

| 評価基準 | 非同期型 | 同期型 |
|---|---|---|
| 好感度 | **4.00(0.43)** | 2.55(0.50) |
| 積極性 | 3.73(0.86) | 2.91(1.08) |
| 朗らかさ | 3.64(0.98) | 3.00(0.95) |
| 人間らしさ | **3.55(1.23)** | 2.27(1.21) |
| 面白さ | **4.09(1.16)** | 2.45(0.89) |
| 社交性 | 3.45(1.30) | 2.82(1.27) |
| 気分の良さ | 3.55(0.89) | 2.64(1.07) |
| わかりやすさ | 2.91(1.00) | **3.73(1.05)** |
| 賢さ | 2.90(1.22) | 2.20(1.17) |
| 優しさ | 3.00(0.85) | 3.00(1.21) |
| 付き合いやすさ | **3.64(0.88)** | 2.36(0.98) |
| 行儀の良さ | 3.18(0.83) | **3.64(0.77)** |

セグメントが得られず発話生成が行えない場合，被験者の発話をそのまま応答として返す．

なお，2つのシステムの試行順序が評価に影響を及ぼす可能性があるため，被験者募集の際どちらのシステムから先に試すかを指示し，2通りの試行順序に被験者を均等に割り振るようにした．

### 4.2 評価方法

被験者に自由対話を行ってもらったあと，Semantic Differential Method（以下，SD 法）と自由記述によるアンケートを行った．SD 法においては，単極性の評価基準 12 個（好感度，積極性，朗らかさ，人間らしさ，面白さ，社交性，気分の良さ，わかりやすさ，賢さ，優しさ，付き合いやすさ，行儀の良さ）を用意し，1 から 5 の整数による 5 段階評価を行ってもらった．また，自由記述においては，まず 2 つのシステムに違いを感じたかどうかを尋ねた上で，

- 違いを感じた（感じなかった）理由
- システムについての意見・感想

を述べてもらった．

## 5 考察

### 5.1 SD 法に基づく定量的評価

実験の結果，提案手法に対する評価は好感度において 1.45 ポイント，人間らしさにおいて 1.28 ポイント，面白さにおいて 1.64 ポイント，付き合いやすさにおいて 1.28 ポイント，それぞれ従来手法を上回った．このことは，これら 4 項目が提案手法の優位性であることを示している．SD 法を用いたアンケートの統計を表 3 に示す．

それ以外の評価基準では，非同期型のシステムが積極性で 0.82 ポイント，朗らかさで 0.64 ポイント，社交性で 0.63 ポイント，気分の良さで 0.91 ポイント，賢さで 0.70 ポイント同期型のシステムを上回った．一方，同期型システムがわかりやすさで 0.82 ポイント，行儀の良さで 0.64 ポイント非同期型システムを上回った．

積極性・人間らしさ・わかりやすさ・行儀の良さの 4 項目に対する評価はそれぞれのシステムにおける特長を反映したものであると考えられるが，とりわけ人間らしさについては，標準偏差がやや高いものの，両者の差が 1.28 ポイントと大きく開いており，有意な差であると認められる．

### 5.2 自由記述と対話ログに基づく定性的評価

2つのシステムに違いを感じるかという質問については，非同期型のシステムに対しても同期型のシステムと同様にシステムの応答を待って対話を行う被験者が存在しうることを想定しての設問であった．アンケートの結果，1 名の被験者が「違いを感じない」と回答し，先の想定通りの対話を行ったとの記述があった．

他の 10 名は 2 つのシステムにおける同期性と非同期性との違いに言及していた．同期型システムには「必ず返事をするため『会話が続いている』と感じられ，ストレスが少ない」という利点が主に挙げられたのに対し，「自分が書けるようになるまでの待ち時間が長い」「オウム返しが多い」といった欠点が挙げられた．また，「機械っぽさを感じる」という意見もあった．

同期型システムにおけるオウム返しの解決方法としては，あらかじめ応答パターンを与えないという前提があるため，応答パターンの獲得を進めることが第一に考えられる．システムが空白の応答を返すということも考えられるが，それを行うと，必ず返事するという同期型システムの利点は損なわれてしまう．

一方，非同期型システムには「好きなタイミングで自分の言葉が書ける」「学習が早く感じられる／賢く見える」「こちらのタイミングと関係なく話をしてくるのが面白い」といった利点が挙げられたのに対し，「話しかけても沈黙が長いことがあり，不安になる」「特定の話題の発話を連発してくるとストレスになる」「延々と挨拶ばかりしている」といった欠点が挙げられた．

被験者による対話の一部を表 4 に示す（発話に出てくる「ハヂモ（ハジモ）」とは，システムの名前である）．この例では，「ごはんを食べたかどうか」を巡って質問を繰り替えしたり，逆に聞き返したりというやりとりが繰り返されている．質問とそれに対する応答との両方に，同じセグメント，あるいは類似したセグ

表 4: 対話例 3 (U: 被験者, S: システム)

| | |
|---|---|
| U | ねーねー |
| U | お話しようよう |
| U | わたしのこと、好き？ |
| S | お話しようよう |
| U | うん |
| U | お話しよう |
| U | ハヂモはごはん食べてないの？ |
| S | お話しようよう |
| U | 今日は火曜日だよう |
| S | ハヂモはごはん食べてないのか … |
| U | 食べてないの？ |
| S | 食べてないのか … |
| U | ええッ？ |
| U | どっちなの？ |
| S | ええッ？ |
| U | お昼ごはんは食べた？ |
| S | お昼ごはんは食べた？ |
| U | 食べたよ。 |
| S | お昼ごはんは食べた？ |
| U | 食べたってば！ |

メントが含まれている場合，本手法のセグメント関係モデルでは，同じやりとりを反復してしまうケースが存在する．

挨拶が頻繁になされる例を表 5 に示す．挨拶などのように，被験者によって頻繁に発話されるためにシステムが獲得しやすい応答パターンについても，システムが過剰に使用してしまう例が多く見られる．この状態が，応答獲得の途上で見られる状態の一つであるのか，モデルの最終的な挙動であるのかについては短期的な実験からは判明しないため，今後観測を継続する必要がある．このようにユーザに不快感を与えてしまう発話出力を抑制するようにアルゴリズム改善することは，今後の課題である．

## 6 まとめと今後の予定

本稿では，モデルを介して入力系と出力系とが並行動作し，非同期的な相互作用を可能とする雑談システムの提案を行った．チャットアプリケーション上に実験システムを実装し実験を行った結果，システムの好感度，人間らしさ，面白さ，付き合いやすさの 4 項目において提案手法の優位性が確認できた．

今後，WordNet[4]などから得られる類義語，上位・下位概念による言い換えを用いたあいづちを行う雑談システムや，感情極性などを用いた学習による応答パターン獲得を行う雑談システム，さらに発話ペースの緩急を制御する機能や自身のパーソナリティを持った雑談システムの構築と評価を行う予定である．

[4] http://nlpwww.nict.go.jp/wn-ja/

表 5: 対話例 4 (U: 被験者, S: システム)

| | |
|---|---|
| S | こんばんは。 |
| U | 今何時？ |
| S | ハヂモは食べた？ |
| U | ハジモはあなたの事だよ |
| S | こんばんは |
| S | こんばんはー |
| S | ハヂモは晩ごはん食べたよ。 |
| S | ハヂモは晩ごはんは？ |
| S | こんにちは |
| U | ところで髪切った？ |
| S | そうだね。ハヂモはどこに住んでるって本当？ |
| S | ハヂモは晩ごはん食べた？ |
| U | 食べたっていってたよ。 |
| U | ラーメンじゃなかったっけ |
| U | 住んでるのはロシアだっけ？ |
| S | こんばんは |
| U | こんばんはー |
| U | 挨拶は一回でいいんだよ。 |
| S | こんばんは |
| U | わかったかい？ |
| S | こんばんはー |
| U | 一回だけだってば |
| U | わかったかい？ |
| S | わかった。 |
| U | よくできました！ |

## 参考文献


[1] 斉藤 哲也, 広田 健一, 星野 准一: Web 情報を用いたキャラクタの発話・世間話モデル, 情報処理学会研究報告. 自然言語処理研究会報告, IPSJ SIG Notes 2007(94), pp. 53–58 (2007)

[2] 下岡 和也, 徳久 良子, 寺嶋 立太: 雑談対話システム構築に向けた事態間の因果関係知識を用いた応答生成, 人工知能学会研究会資料, SIG-SLUD-A701-13 (2007)

[3] 水野 淳太, 乾 健太郎, 松本 裕治: ウェブニュースを利用した雑談対話システム, 人工知能学会研究会資料, SIG-SLUD-A803-1 (2009)

[4] 木村 泰知, 荒木健治, 桃内佳雄, 栃内香次: 遺伝的アルゴリズムを用いた帰納的学習による音声対話

処理手法, 電子情報通信学会論文誌 *D-II*, Vol. J84-D-II, No. 9, pp. 2079–2091 (2001)

[5] Shinsuke Higuchi, Rafał Rzepka and Kenji Araki: A Casual Conversation System Using Modality and Word Associations Retrieved from the Web, *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 382–390 (2008)

[6] 西村 良太, 北岡 教英, 中川 聖一: 応答タイミングを考慮した雑談音声対話システム, 人工知能学会研究会資料, SIG-SLUD-A503-5 (2006)

[7] 西村 良太, 北岡 教英, 中川 聖一: 対話における韻律変化・タイミングのモデル化と音声対話システムへの適用, 人工知能学会研究会資料, SIG-SLUD-A602-7 (2006)

[8] 河添 麻衣子, 北村 泰彦: 発話タイミングを考慮したマルチエージェント対話システム, 電子情報通信学会技術研究報告. *KBSE*, 知能ソフトウェア工学, Vol. 106, No. 618, pp. 53–56 (2007)

[9] 若原 基, ジェプカ ラファウ, 荒木 健治: 非タスク指向型対話システムにおける非同期並列的相互作用による言語獲得手法の提案, 電子情報通信学会第二種研究会資料, IEICE SIG Notes WI2-2009-28-45, pp. 53–54 (2009)

[10] 松本 泰明, 麻生 英樹, 原 功, 柿倉 正義: インタラクティブエージェント用ユーザモデル構築のための対話実験, 人工知能学会全国大会論文集, Vol. 18, 2E1-07 (2004)