音声ユーザインタフェースにおける処理エラーによるユーザフラストレーションに関する調査

呉越　思瑤†　　酒井　哲也†

† 早稲田大学基幹理工学部情報理工学科　〒 169-8555 東京都新宿区大久保 3 － 4 － 1
E-mail: †shiyoh@ruri.waseda.jp, ††tetsuyasakai@acm.org

あらまし　近年、音声認識技術の向上により、Siri や Alexa のような音声ユーザインタフェース（VUI）が日常生活の中で使われるようになった。しかし、音声ユーザインタフェースにはまだ音声の誤認識やシステム設計の不備など様々な課題があり、ユーザとのインタラクションが破綻することが多い。本研究では、ユーザのフラストレーションの度合いは対話破綻タイプに応じて異なるという仮説を立て、これを検証するため、音声操作によるスケジュール登録の VUI システムを開発し、被験者が 10 人のタスクベースのユーザ実験を実施した。そこで、ユーザ実験で観察した対話破綻を 7 つのタイプに分類し、インタビューから得た各被験者が各タスクで感じたフラストレーションの評点を用いて、各対話破綻タイプに対しての平均フラストレーション値を算出し、仮説の検証を行った。その結果として、各対話破綻タイプが起こった時の平均フラストレーション値の間に統計的有意な差は得られなかった。また、対話破綻した際にシステムが返すフィードバックの情報の不十分さに対し、ユーザがフラストレーションを感じていることが分かった。そのため、VUI アプリケーションを設計するにあたって、対話破綻した際のフィードバックの内容を工夫することが必要になることが分かった。

キーワード　HCI, 音声ユーザインタフェース

1. 研 究 背 景

音声認識技術の向上により、Siri や Alexa のような音声ユーザインタフェース (VUI) が日常生活の中で使われるようになった。Amazon や Google などの大手企業は、次々と Amazon の Alexa 対応デバイスや Google Home などの VUI 製品を発売し、世に出している。既にスマートスピーカや音声アシスタントとの会話は日常生活の一部になりつつある。

しかし、音声インタフェースはまだたくさんの課題に直面している。ユーザと音声インタフェースがいかに自然にかつスムーズなやりとりを実現できるかは、音声認識の精度や自然言語処理の技術に大きく依存する。しかし、現状ユーザが直面する対話破綻の原因の大半は音声認識誤りであり、また、これ以外にも、システム側の処理エラーや開発時の会話設計の不備が挙げられる [3]。このように、対話破綻には様々な原因が考えられる中、現状は「ごめんなさい、よくわかりませんでした」のような画一的なレスポンスしか返していない。Porcheron ら [6] や Pelikan ら [1] は、このようなレスポンスの後にユーザが次のやりとりに進むことが難しいと指摘している。そのため、現状の音声インタフェースのレスポンスも徐々に工夫されつつある。例えば、画面付きの音声インタフェースでは、音声に加えて視覚的なレスポンスを返すことができ、ユーザはより多くの情報を得ることが可能である。ただし、このような改善はあくまでも音声認識の不足分を補うための最初の一歩となっており、他の原因による対話破綻についてはまだ考慮されていない。そのため、様々な対話破綻のタイプに応じたマルチモーダルなレスポンスを返し、ユーザとの有効な対話の継続を極力可能にすることが必要になってくる。

既存の研究の多くは、ユーザーが現在の VUI とどのようにやり取りするのか、そしてそれをどのように解決するのかを調べ、現状行われているやりとりの大枠を理解することに焦点を当てている。我々の研究は、これらの従来研究とは異なり、VUI との対話が破綻した際のシステムのレスポンスに着目している。ユーザと VUI の対話が破綻した際にも、システムのレスポンスがその現象に関する詳細をユーザに開示することにより、できる限り有効な対話に復帰できるようなシステムを構築することが本研究の最終目的である。このためには、まずユーザがどのような対話破綻に対しどのようなフラストレーションを感じるのか理解する必要があると考えた。そこで、本研究では、ユーザが感じるフラストレーションの度合いは破綻タイプに応じて異なるという仮説を立て、これを検証するためのプロトタイプを作成し、ユーザ実験を行った。そこで、観察したユーザと実験システムの対話破綻のタイプをシステム側の視点から定義し、それぞれの対話破綻タイプによってフラストレーションが異なるかについて検定を行なったが、統計的に有意な差は得られなかった。また、ユーザインタビューの結果から、対話破綻した際にシステムが返す情報の不十分さに対し、ユーザがフラストレーションを感じていることがわかった

2. 従 来 研 究

近年、VUI におけるユーザエクスペリエンスを調査する研究が多く行われている。Purington ら [7] は, Amazon.com から Amazon Echo に関する口コミをデータとして集め、分析し、デバイスに対する擬人化がインタラクションにおける社交性のレ

ベルや満足度に影響を与えると報告している。また、VUI のレスポンスについて、ユーザとの対話破綻が発生した場合、ユーザが対話を修復するのに必要な情報が不足していることが指摘されている [6] [1] [5]。Porcheron ら [6] は、システムがレスポンスとして何も返さなかった場合、ユーザがシステムが処理に失敗したと認識し、やりとりを中断する傾向があると報告している。また、Luger ら [5] は、現状の対話システムはユーザに返す情報が不足しており、ユーザにとってシステムがブラックボックスとなっていると報告している。このため、ユーザはシステムが実行可能なタスクを把握できず、そもそもシステムの備えている機能が限られていると認識してしまうと指摘している。

Pelikan ら [1] は、処理におけるエラーの原因を明らかにすることがユーザビリティの向上に繋がることを示している。これらの研究は全て、VUI におけるレスポンスの重要性について言及している。本研究では、VUI が処理に失敗した際のレスポンスに着目し、有効な対話を導くためのレスポンスを明らかにするため、まずユーザがどのような対話破綻に対しどのようなフラストレーションを感じるかについてユーザ実験を行った。

3. 実験の方針

3.1 実験の目的

VUI との対話破綻のタイプによってユーザが感じるフラストレーションの度合いが異なると仮説をたて、ユーザ実験で検証を行った。また、対話破綻のタイプについて、ユーザ側とシステム側からみた対話破綻のタイプは異なると考えた。システム側では異なる原因で対話破綻したが、ユーザに返すレスポンスが同じような場合、ユーザ側は同じ対話破綻に見える可能性がある。例えば、ユーザのある発話に対応する処理がシステムに予め用意されていなかったことに起因する対話破綻とシステムが発話の一部を認識し損ねたことに起因する対話破綻はシステム側から見ると異なる現象である。しかし、もしシステムがいずれの場合も「ごめんなさい、聞き取れなかったです」という同じレスポンスを返した場合、これらはユーザから見ると同じ現象となる。今回は、システム側から見た対話破綻のタイプに着目している。

3.2 実験システムの設計

スマートスピーカや音声アシスタント搭載端末の普及により、消費者は音声インタフェースを雑談目的のみならず、タスクを達成するために用いることが多くなりつつある。対話が成立しようがしまいが大きな支障はない非タスク指向の対話の場合と比べ、特定の目的を達成するためのタスク指向の対話においては、ユーザが対話破綻に対して寛容でなくなる可能性がある。そこで本研究では、音声インタフェースのタスク指向対話に着目し、実験システムを構成した。実際にどのような対話破綻が起こるかを観測し分析するため、本研究では以下の要件を満たしかつ対話破綻が起こりやすいアプリケーションを開発した。

- 一つのタスクを完成するのに複数のやりとりが必要
- ユーザがある程度自由に発話できる
- 実行可能なタスクを複数含む
- 操作の結果が確認できる

4. 実験システム

1. 章で述べたように、Myers ら [3] は、充分に複雑な操作を備えたタスク指向対話のアプリケーションとして、スケジュール管理のタスクを実験に用いた。本研究では、これにならい、かつ 3. 章で述べた要件を満たすスケジュール管理のアプリケーションを開発した。ユーザは音声入力によりスケジュールの登録や修正を行い、その操作結果として視覚的なフィードバックをパソコンのスクリーン上で確認できる。以下、試作した実験用アプリケーションを構成する音声インタフェースとフィードバックの部分について説明する。

4.1 音声インタフェース

スケジュール管理のアプリケーションは、Amazon Alexa Skills Kit(注1)を使い、Amazon Echo (注2)上で開発した。3. 章で述べた方針に基づき、本システムでは実行可能な機能として、イベント登録、削除と修正という複数の機能を備えている。

本研究ではシステム側からみた対話破綻に着目しているため、VUI アプリケーションのシステム側でどのように音声を処理しているのか、またどの部分にエラーが起こりやすいかについて説明する。

（1） 音声から文字への変換

まず、VUI アプリケーションの最初の処理として、ユーザの発話を音声認識によりテキストに変換し、形態素解析を行う。現状の音声認識精度では、ユーザの発話した内容と異なるテキストに変換される場合がある。

（2） 発話と該当する処理をつなぎ合わせる

タスク指向のアプリケーションでは、ユーザの特定の発話に対して特定の処理を行う必要がある。Alexa Skills Kit には、ユーザの発話をシステムが行う処理に結び付けるインテントという概念が存在する。システム開発の時点で、どの発話をどのインテントに対応させるか予め設定しておく。例えば、「今日の天気は？」という発話を受け取ったら、これは「天気を調べる」インテントを持つものと解釈し、実際に「天気を調べる」処理を行うよう設定する。この発話とインテントを対応づける際に、多様な表現に対応できるようにしておかないと、実際に対話を行う際にユーザの発話を正しいインテントにマッピングできなくなる。例えば、「天気教えて」という発話が設計段階で記述されていない場合、この発話は「天気を調べる」インテントに分類されないことになる。

（3） 発話から情報を取得する処理

Alexa Skills Kit では、ユーザの発話から情報を取得し、Slot Filling [2]、すなわち変数 (スロット) に値を入れることが可能である。また、スロットの型指定も可能である。例えば、「AMAZON.DATE」というスロットの型を指定し、かつサンプル発話が「今日の予定は○○から」と設定された場合、ユーザの発話「今日の予定は 13 時から」から、スロット値として「13 時」が取得される。しかし、システム開発時の会話の設計が不十分

(注1)：https://developer.amazon.com/ja/alexa-skills-kit

(注2)：Amazon.com が開発したスマートスピーカー

である場合や、提供されている API を開発者が正しく理解せず用いた場合に、Slot Filling に失敗する場合がある。

4.2　操作結果のフィードバック

スケジュール登録の VUI アプリケーションの場合、音声による操作が正しく行われているのかについて確認できる画面が必要である。そのため、本実験システムでは、パソコンのスクリーン画面を使い、スケジュール登録画面 (図 1) を表示することにした。この画面は、音声インタフェースと連携し、音声の操作結果を表示している。

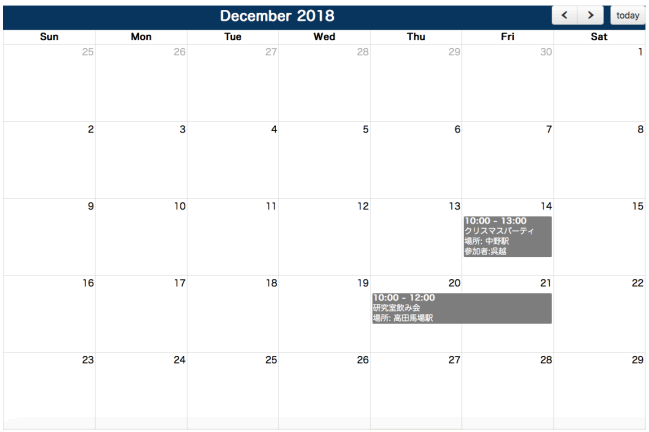

図 1　実験システムのスケジュール登録画面

5.　ユーザ実験

3.1 節で述べた仮説の検証を行うため、ユーザ実験を実施した。この章ではユーザ実験の詳細について説明する。

5.1　被　験　者

今回は 10 名の被験者にユーザ実験を実施した。被験者は全員男性で、20 代の早稲田大学の学生である。被験者のうちスマートスピーカ所有者が 7 人おり、このうち 3 人は開発経験がある。

5.2　実 施 手 順

実験では、各被験者に実験システムを利用し、我々が与えたタスクを実行してもらった。参加した被験者全員に同じタスクを与え、全部で 4 個のタスクとなる。また、スマートスピーカの未経験者に対し、操作の不慣れが実験結果に影響を与えないようにするため、実験を行う前にスマートスピーカの基本的な使い方（起動方法や中断方法など）を指導した。

5.2.1　タ　ス　ク

タスクでは、イベントの作成、イベントの変更、およびイベントの削除といった一連の作業を順番に依頼している。また、イベントの作成のみ、2 つのタスクが含まれている。

（1）　音声 VUI の指示に従い、イベント登録する

（2）　音声 VUI に従わずにイベントの情報を一度に登録する

ここで、評価に偏りが起こらないように、上記 2 種類のタスクのみ被験者ごとに実行順番をランダムに変えている。実験で被験者に与えたタスクの詳細を図 2 に示す。

背景：
あなたは 12 月に二つのイベントがあります。
それぞれのイベントは以下のようになります
1）12 月 28 日 18:30~20:00
　イベント：バイト先飲み会
　場所：　銀座駅
2）12 月 31 日　9:00~11:00
　イベント：パーティ
　場所：高田馬場

タスク 1：
1）のイベントをアレクサを使って登録してください。
タスク 2：
2）のイベントをアレクサを使って登録してください
タスク 3：
1）のイベントについて、あなたは、登録した場所を間違えたことに気づきました、正しくは銀座駅ではなく、高田馬場駅でした。この間違えを修正してください。
タスク 4：
12 月 29 日のイベントが取り消しになったため、イベントを削除してください。

図 2　タスク詳細

5.2.2　インタビュー

実験では、各被験者が各タスクを完了すると、評点を記入してもらい、短いインタビューを行った。また、ユーザのフラストレーションの度合いを測るため、5 段階のリッカート尺度を用いた。各被験者は提示された「あなたはフラストレーションを感じたか」に対し、1（「全く感じなかった」）から 5（「非常に感じた」）という評点をつける。

5.3　実験データの記録

実験中の被験者とシステムの対話およびタスク終了後のインタビューは録音した。その他に、実験システムでは、Amazon Echo の音声認識で音声から文字に変換された発話を記録し、認識された発話がどの処理にマッピングされたか、およびその発話からどの情報を取得したかもキャプチャしている。

6.　実験結果と考察

本章では、実験結果の分析と考察を以下の順で行う。まず、6.1 節で、ユーザ実験で観察された対話破綻をシステム側の観点から 7 つのタイプに分類した結果について説明する。次に、6.2 節で、各ユーザが各タスクで感じたフラストレーションの評点と対話破綻タイプとの関係を議論する。6.3 節では、タスク終了後のインタビューの結果について述べる。

6.1　対話破綻のタイプ

まず、ユーザ実験で観測された全?件の対話破綻を第一著者が人手により分析し、システム側の視点から排他的であるよう、以下の 7 つの対話破綻タイプに分類した。

B1:　登録されていない発話

4.1 節で述べたように、ユーザの発話が予め登録されていた発話と一致した場合、　設定されたインテントに基づき該当する処理を行う。逆に登録されていない発話に関して例外処理に飛び、「もう一度教えてください」という文言を返すこととなっている。B1 はそのようなエラーを指している。

B2:　必要とするスロット値が取得できなかった

ユーザの発話が該当するインテントに認識され、正しい処

理に結び付けられたが、その処理で必要となる情報をその発話からうまく取得できなかった時のエラーを指す。例えば、サンプルの発話が「{date} の {timestart} から {timeend} に {location} で {title} を登録して」で、ユーザの発話が「12 月 31 日の朝 8 時から夜 8 時までパーティを登録して」の場合、上記のサンプル発話と一致し、該当する処理に結び付けれるが、location に対応するスロット値がユーザの発話から抜けているため、エラーが起こることになる。

B3: B3 アレクサが音声を受け付けていない時に発話

アレクサが音声を受け付けていない状態でユーザが発話し、認識されなかった場合を指す。

B4: インテントの誤認識

ユーザの発話の一部が事前に登録しているサンプルの発話とマッチされ、ユーザの意図しない処理に移ることを指す。

B5: システムに向けた発話でないものをシステムが認識

ユーザの独り言や第三者に対する発言がシステム側に認識された場合を指す。

B6: 発話の一部しか捉えていない

ユーザの発話の一部しかシステムが認識してない時を指す。例えば、ユーザの「高田馬場、(沈黙 2 秒) 駅」という発話に対し、システムは「高田馬場」のみ捉えた。また、日付の設定の際、ユーザの「12 月 29 日」という発話に対し、システムは「29 日」しか認識できなかった。そのため、該当するインテントに正しく認識されず、エラーが起きる。このようなエラーは、ユーザの喋り方や喋る速度などにより起ることが多い。また、上述の B2 のタイプとは違い、B6 の場合はインテントに正しく認識される前の段階で起きるエラーとなっている。

B7: 音声認識の誤り

音声認識が主な原因となり、ユーザの発話を正しい文字に変換できなかったことを指す。今回の実験では、ユーザの「高田馬場駅」に対し、システムは「高田馬場　駅」や「肩　の　馬場駅」に変換していた。このように文字間にスペースを入れたり、誤った文字に変換することなどのミスにより、発話と処理のマッチングに大きな影響を与えている。

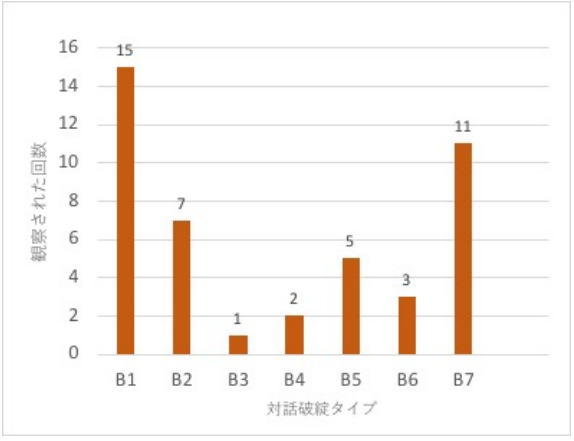

図 3　実験で観察された各対話破綻タイプの回数

6.2　フラストレーションと対話破綻のタイプの関係

3.1 節で述べたフラストレーションと対話破綻のタイプの仮説を検証するにあたって、実験で得たデータを用いて、各対話破綻のタイプに対してユーザが感じた平均フラストレーションを求めた。これらの平均間の差の統計的検定を行うため、Tukey HSD 検定 [4] を実施した。

まず、各対話破綻のタイプに対するフラストレーション値を算出するにあたって、実験データから、ある対話破綻タイプ Bz が観察されたユーザ u_x とタスク t_y の組み合わせの集合を $UT(Bz)$、ユーザ u_x がタスク t_y で感じたフラストレーション値を $F(u_x, t_y)$ とすると、対話破綻 Bz に対する平均フラストレーション値は以下のように求められる。

$$MF(Bz) = \frac{1}{|UT(Bz)|} \sum_{(u_x,t_y)\in UT(Bz)} F(u_x, t_y) \qquad (1)$$

式 1 により、7 つの対話破綻タイプに対する平均フラストレーションを求めた。図 4 はこれを視覚したものである。また、各対話破綻タイプが観察された回数を図 3 に示す。

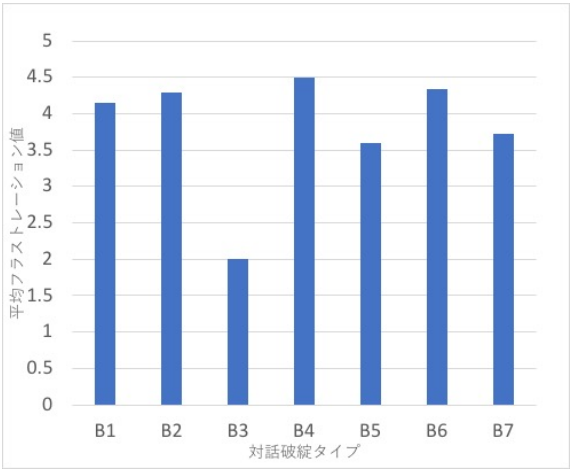

図 4　対話破綻のタイプと平均フラストレーションの関係

上記の結果を元に、各対話破綻タイプ間における平均値の比較を、R を用いて有意水準 $\alpha = 0.05$ で Tukey HSD 検定を行った。図 5、図 6 はその結果を図示したものである。これらの結果により、各対話破綻タイプ間の平均値に統計的に有意な差が得られなかったことがわかる。その理由として、対話破綻タイプの定義の仕方の問題と、サンプルサイズが不充分であったことの二点が考えられる。

まず、対話破綻タイプの定義の仕方の問題について述べる。今回定義した対話破綻タイプは、システム側・開発者側の視点から定義したが、これらは必ずしもユーザから見た現象と一致しない。実際には様々なタイプの対話破綻が起こっている場合でも、現状のシステムはユーザに対して同じような文言、例えば「ごめんなさい、もう一度教えてください」といったフィードバックを返している。このため、ユーザには異なる対話破綻タイプの発生が直接見えておらず、ユーザのフラストレーションも対話破綻タイプに呼応するものになっていない可能性がある。次に、サンプルサイズであるが、今回の被験者数はわずか

	diff	lwr	upr	p adj
B1-B3	1.53333333	-2.1211296	5.187796	0.8441334
B5-B3	1.60000000	-2.2761433	5.476143	0.8537933
B7-B3	1.72727273	-1.9684849	5.423030	0.7677930
B2-B3	2.28571429	-1.4970142	6.068443	0.5037807
B6-B3	2.33333333	-1.7524804	6.419147	0.5694877
B4-B3	2.50000000	-1.8336599	6.833660	0.5579331
B5-B1	0.06666667	-1.7605648	1.893898	0.9999998
B7-B1	0.19393939	-1.2106646	1.598543	0.9994487
B2-B1	0.75238095	-0.8672866	2.372049	0.7726994
B6-B1	0.80000000	-1.4378924	3.037892	0.9197884
B4-B1	0.96666667	-1.6969580	3.630291	0.9143252
B7-B5	0.12727273	-1.7812083	2.035754	0.9999922
B2-B5	0.68571429	-1.3861714	2.757600	0.9432598
B6-B5	0.73333333	-1.8507622	3.317429	0.9727443
B4-B5	0.90000000	-2.0604533	3.860453	0.9619131
B2-B7	0.55844156	-1.1523618	2.269245	0.9467594
B6-B7	0.60606061	-1.6986492	2.910770	0.9813831
B4-B7	0.77272727	-1.9472767	3.492731	0.9726004
B6-B2	0.04761905	-2.3941217	2.489360	1.0000000
B4-B2	0.21428571	-2.6227607	3.051332	0.9999837
B4-B6	0.16666667	-3.0634527	3.396786	0.9999983

図 5　Tukey HSD 検定の結果（p 値）

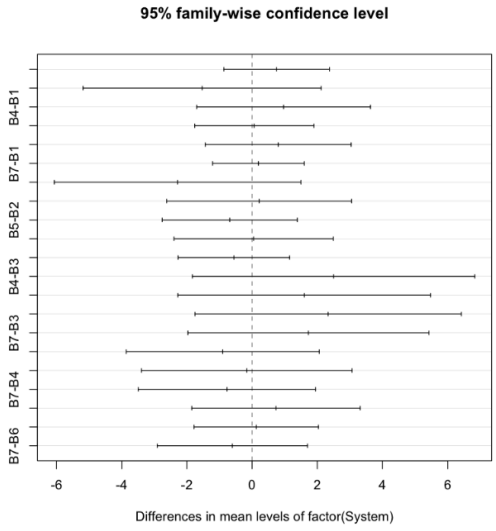

図 6　Tukey HSD 検定の結果（95%同時信頼区間）

10 であったため、各対話破綻タイプの観測数が充分とれず、統計的有意差が得られなかった可能性がある。また、図 3 より示している実験で観察された各対話破綻タイプの回数から、少なくとも本実験では「B1:登録されていない発話」の頻度が多く、その次に「B7: 正しく認識されていない」の回数が多かった。従って、VUI アプリケーションを設計する際、上記のような対話破綻タイプに対するフィードバックを重点的に実装していくことで、ある程度対話への復帰に繋がることが考えられる。

6.3　インタビューの結果と考察

実験で得た録音データについて、コード化を行った上で分析を行った。コード化とは録音の音声データをテキストに落とし、それらに抽象的なラベルを貼ることを通して、何らかのパターンを見出す手法となる (注3)。ユーザ実験で各ユーザに各タスクの中で感じたフラストレーションについてインタビューを行った。また、6.2 節で述べたように、システム視点からの対話破綻のタイプとユーザ視点からのが異なってくるため、ユーザインタビューでは、6.1 節で定義したシステム側の視点に基づく対話破綻タイプを意識したような回答は得られなかった。

以下に、VUI とのやりとりの中でどの部分にユーザがフラストレーションを感じたのか、またなぜ感じたかについてまとめる。

（1） システムのレスポンスの情報が足りない

実験システムで対話破綻が発生した際、「ごめんなさい、よくわかりませんでした」あるいは「もう一度教えてください」といった文言をレスポンスとして返している。

そのため、レスポンスに対してフラストレーションを感じたという意見が得られた。

- 「何回もトライしたのにも関わらず同じようなことしか言われないので、何がうまくいかなかったのか分からない、改善しようがない」
- 「アレクサがどういう言い方を求めてるのかが分からない、正しい言い方が思いつかなかった」
- 「自分は失敗する度に言い方を変えているのに、システムの聞き取れなかった理由はいつも同じような返答、ちゃんと聞いてくれてるのかが分からない」

（2） 予想していたことと違った

- 「イベント登録の時に途中で間違えたからイベント修正をしたいのにできなかった、やり直しができない」
- 「イベントの内容を一気に登録するの難しいと思うから、一番簡単シンプルなパターンで登録したのに、できなかった」
- 「イベント登録の時に時間指定で、6 時って言えば、システムから午後か午前かについて聞かれると思った」
- 「イベントの登録であれば、日付と詳細が最低限の要素だと思ったのに、それだけで登録できなかった」

一方、同じ対話破綻について、あまりフラストレーションを感じなかったユーザの意見を以下にまとめた。

- 妥協点が見つかったから

イベントの場所を「銀座駅」から「高田馬場駅」に修正するタスクに対し、実験システムは「高田馬場駅」をうまく認識できず、ユーザはなんども試したが、最終的に「高田馬場」が実験システムに認識され、登録を完了した。そこで、「最低限の「高田馬場」が認識されるのを知っていたので、妥協点があるからあまり感じなかった。もし妥協点が見つからなかったら、フラストレーションが溜まっていた」という意見が得られた。また、他にも「最初からうまくいかないより、途中までうまくいって最後ミスるならまだ許せる」や「結果として最後は登録

(注3)：http://www.cshe.nagoya-u.ac.jp/asg/SummarizeData

できたから」などの意見もあった。

- 失敗しても代わりの手段があるから

「イベント登録に失敗しても、他の端末ですぐに操作できるし」といったような意見もあった。

また、ユーザの属性により、フラストレーションの感じ方が異なっていることもわかった。被験者のうち開発経験のあるユーザは、開発経験のないユーザと比べシステムの対話破綻について、比較的に寛容な態度をとる傾向があることがわかった。開発の知識があるため、彼らはシステムができる最低限のことを予測し、スムーズにいくよう文言を考えながらタスクを行っていた。例えば、イベントの時間を指定する際、「18 時半から 20 時の言い方をしたのは、システム側にとって理解しやすいだろうなと思ったから。普段ならそういう言い方はしない」と回答した被験者がいた。だが、もしも期待していた最低限のことができなかった場合、開発経験のあるユーザにもフラストレーションを感じるといった以下の回答も得られた。「色々試して、複雑なパターンがダメだったから、一番簡単なパターンを試したのに、それでもダメだった時はフラストレーションを感じた」

7. ま と め

7.1 結　　論

今回は、ユーザと実験システムのやりとりを観察し、そこでユーザが感じだフラストレーションについて調査する実験を行った。そこで観察したシステム側からみた対話破綻のタイプを定義し、対話破綻タイプによってフラストレーションが異なるかについて調べたが、各対話破綻タイプ間のフラストレーション平均値の間に統計的に有意な差が得られなかった。また、6.3 節で述べたインタビューの結果から、対話破綻した際にシステムが返したフィードバックからユーザは異なる対話破綻のタイプの発生を認識できなかったことが分かった。その原因として、現状のフィードバックはシステムの状況が把握できるほど十分な情報を含んでいないことが考えられる。さらにユーザは現時点のシステムがどういう状況なのか、どのように自分のリクエストを改善すればうまくいくのかが分からず、フラストレーションを感じると答えていた。また、図 3 に示したように、対話破綻タイプ「B1:登録されていない発話」や「B7: 正しく認識されていない」が比較的多く観察されたそのため、VUI アプリケーションを設計する際に、上記のような対話破綻タイプが発生した際に対応するフィードバックを工夫することが必要となり、そのフィードバックで有効な対話に復帰できるような内容を提供することも重要であることが今回の実験で分かった。

7.2 今後の課題

今回はシステムで発生する対話破綻がユーザにどのようなフラストレーションを与えているかについて調査した。しかし、本研究には改善の余地が残されている。まず、被験者を増やした実験を行うことにより、各対話破綻タイプの観測数を増やし、より信頼性の高い対話破綻タイプの分布を求めることが挙げられる。また、対話破綻タイプをシステム側ではなくユーザ視点から定義することで、新たな知見が得られる可能性がある。

文　　献

[1] Hannah R.M.Pelikan, Mathias Broth. Why that nao?: How humans adapt to a conventional humanoid robot in taking turns-at-talk. In *CHI '16 Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 4921–4932, 2016.

[2] Marilyn A. Walker,Diane J. Litman,Candace A. Kamm and Alicia Abella. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of NAACL 2000*, pp. 210–217, 2000.

[3] Chelsea Myers, Anushay Furqan, Jessica Nebolsky. Patterns for how users overcome obstacles in voice user interfaces. In *CHI '18 Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018.

[4] Tetsuya Sakai. *Laboratory Experiments in Information Retrieval: Sample Sizes, Effect Sizes, and Statistical Power*. Springer, 2018.

[5] Ewa Luger, Abigail Sellen. "like having a really bad pa": The gulf between user expectation and experience of conversational agents. In *CHI '16 Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 5286–5297, 2016.

[6] Martin Porcheron, Joel E. Fischer, Stuart Reeves, Sarah Sharples. Voice interfaces in everyday life. In *CHI '18 Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018.

[7] Amanda Purington, Jessie G. Taft, Shruti Sannon, Natalya N. Bazarova, Samuel Hardman Taylor. Alexa is my new bff": Social roles, user satisfaction, and personification of the amazon echo. In *CHI EA '17 Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 2853–2859, 2017.