係り受け関係を用いた即時発話理解

−音声対話メールシステムにおける手法−

伊藤 傑[†1] 松原 茂樹[†2†4] 河口 信夫[†3†4] 外山 勝彦[†3†4] 稲垣 康善[†3]

[†1]名古屋大学工学部 [†2]名古屋大学言語文化部 [†3]名古屋大学大学院工学研究科

[†4]名古屋大学統合音響情報研究拠点 (CIAIR)

{masaru,matu,kawaguti,toyama,inagaki}@inagaki.nuie.nagoya-u.ac.jp

1 はじめに

近年、親しみやすく、使いやすいユーザインタフェースの実現を目指して、音声対話システムに関する研究が盛んに行われている。ユーザにとって適切なタイミングでシステムが応答を返す適時応答性は、ユーザが快適にシステムを使用するための重要なポイントとなる。応答するタイミングの決定は、当然ながら、応答する内容を決定した後となるため、システムは、ユーザの発話をできる限り即座に理解する機能、すなわち、即時理解機能を備えることが望ましい。

本稿では、日本語音声発話に対する即時発話理解と適時応答生成のための手法を提案する。即時発話理解手法では、日本語の文節が入力されるたびに、その係り受けに関する情報をもとに、今後入力される文節を予測する。これにより、早い段階での発話理解を実現する。またその際、タスクドメインを限定することにより、係り先を限定しやすいという利点を利用する。一方、適時応答手法では、発話内容によりユーザにとって最適なタイミングでの応答を実現する。

我々はこれまでに、話し言葉の漸進的解釈に基づくマルチモーダルメールツール Sync/Mail[1] を作成している。Sync/Mail は、ユーザの発話を即座に解釈して、応答するシステムであるが、応答手段は画面上での検索結果の表示に限られており、応答は結果が出たらすぐに出力するという即時応答機能を備えたシステムとして設計されていた。しかし、音声による応答では、即時応答性よりもむしろ、適時応答性が必要となる。

そこで本稿では、応答を音声で行うメールシステムを設計し、発話の即時理解と応答の適時生成を、このシステムで実現する。

2 即時発話理解と適時応答生成

2.1 発話の即時理解

我々人間が会話をするとき、相手が発話途中の段階でも、その発話意図を理解できることがある。すなわち、発話による情報がたとえ不完全な状態であっても、会話の状況や文脈に関する情報からそれらを補うことができる。例えば、ファーストフード店でメニューを注文する場面で、店員が「こちらで」と言ったら、おそらく次に「お召し上がりですか？」と言うことを予測できる。これは、メニューを注文するという状況を考慮すると、食べる場所を知りたいという意図を店員が抱くことを予測できるためである。人間が文脈や状況に依存して、早期に意図理解できるように、システムもまた、その対話ドメインなどがあらかじめ特定されていれば、同様に理解できる可能性がある。

2.2 係り受け関係を用いた即時発話理解手法

前節で述べたように、相手の発話内容を予測できれば、即時発話理解が可能になる場合がある。日本語では、文節を構成する自立語の品詞や助詞などによって、その修飾先の文節のタイプを限定できる。すなわち、入力された文節が修飾し得る文節を特定することによって、後に入力される文節を予測可能となることが期待できる。このことは、係り受け関係を用いることにより実現できると考えられる。一般に、日本語の係り受け規則では、係る文節の自立語の品詞と助詞の種類によって、それを受ける文節の品詞が決まる。しかし、この規則だけでは係り得る文節は多数存在するため、係り先を限定することは困難である。そこで本稿では、タスクドメインを限定して、その上での文節の意味も考慮して係り受け関係を定めることにより、受け文節を限定する。

本手法では、文節が入力された段階でそれを受ける文節の候補を調べ上げる。その時点で候補が唯一に定まれば、それを受け文節として決定し、さらに、その文節に対する受け文節の決定を試みる。唯一に定まらなければ、その時点では受け文節を決定せず、次に入力された文節の情報を用いて同様の処理を行う。先の例において、文節「こちらで」が入力された時点で、その状況のもとで考えられる受け文節として、「お召し上がりですか」のみを候補にあげることができるので、発話「こちらでお召し上がりですか」に対する係り受け構造を作成することができる。その結果、早い段階でユーザの意図を理解することが可能となり、即時発話理解を実現できる。

2.3 適時応答生成

即時発話理解が実現すると、即時応答生成も可能となる。しかし、システムが音声によって応答する場合は、必ずしも即座に応答をすることが望ましいとは限らない。発話内容によっては、ユーザが発話途中であってもそれを遮ってシステムが応答した方が良い場合もあれば、ユーザが発話し終わるまでシステムは応答しない方が良い場合もある。そこで、応答タイミングを決定するポイントとして以下の3点を考える。

- ユーザの気分を害することを避ける。
- ユーザの無駄な発話を避ける。
- ユーザの発話行為を支援する。

これらの各視点の重要度は、もちろんユーザによって異なるが、本稿では一般的に望ましいと思われる基準として、以下のような基準を用い、それに従って、発話タイミングを決定する。

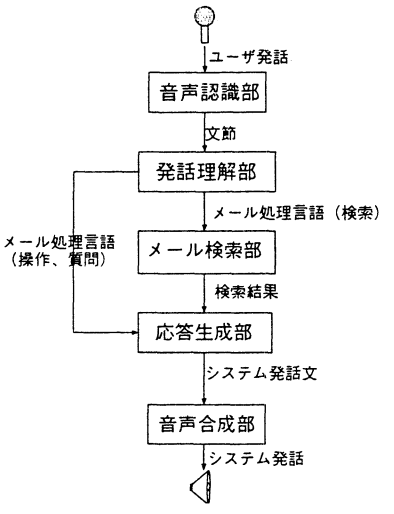

図 1: システム構成図

(1) 基本的には、ユーザの発話が終わった時点で応答する。システムは、ユーザの発話途中で応答する内容が決定しても発話せず、ユーザの発話に一定の無音部分があったら発話する。

(2) ユーザにすぐにでも伝えた方がよい情報があれば、ユーザ発話に割り込んで応答する。これによりユーザは、無駄な発話を減らすことができ、すぐに言い直しをすることが可能となる。

(3) ユーザの発話が一定時間途切れた場合は、ユーザにとって必要な情報を提供して、ユーザの次の発話を促す。

3 音声対話メールシステム

前節で述べた即時発話理解手法、並びに適時応答生成手法に基づき、音声対話メールシステムを実現した。本システムでは、メールの読み上げを依頼すること、メールの差出人やタイトルなどを尋ねること、数あるメールの中から該当するメールを検索すること、などを音声対話を通して行える。図1にシステム構成を示す。システムは、音声認識部、発話理解部、メール検索部、応答生成部、音声合成部の５つのモジュールから構成される。以下では、そのうち発話理解部、応答生成部について詳述する。

3.1 発話理解部

発話理解部では、音声入力された文節が認識されるたびに、その係り受け関係の情報をもとに、メール処理言語と呼ばれるタスク依存言語への変換を行う。それがメールの検索に関するものであればメール検索部に、メールに関する操作や質問に関するものであれば応答生成部にそれぞれ送る。以下では、話し言葉からメール処理言語への変換処理について述べる。

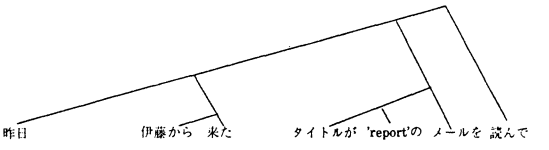

図 2: 一般的な係り受け構造

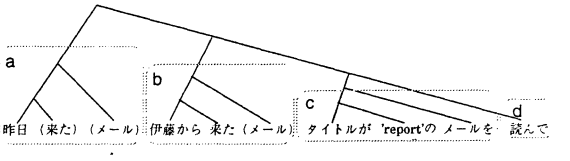

図 3: 文節を補足した係り受け構造

3.1.1 メール処理言語

メールの読み上げを依頼する発話「昨日、伊藤から来たタイトルがレポートのメールを読んで」を日本語の一般的な係り受け構造で表すと図2のようになる。しかし、この発話は実際には、「昨日来たメールであり、伊藤から来たメールであり、タイトルがレポートであるメールを読んで」という意味であるとみなせる。そこで、この文に文節を補って係り受け構造を作成すると、図3のようになる。この構造から、a,b,c 各部分構造が文節「読んで」に係っており、それぞれが意味的に一つのまとまりになっていることがわかる。このことから、不足語を補って係り先を予測することにより、意味のまとまりが明確になると考えられる。そこで、メールの検索、メールに関する質問、操作など、メール処理が実行可能な意味のまとまりを定めるメール処理言語を設計した[1]。図3の a,b,c,d それぞれが、メール処理言語の処理可能な最小単位に相当する。これが定まった時が、ユーザ発話に対するタスクの実行タイミングである。

メール処理言語の文法の一部を図4に示す。非終端記号 <SELECT>,<OPERATE>,<ANSWER> が処理可能な最小単位であり、<ATTRIBUTE> には date, from, subj などメールの属性を表す終端記号が、<VALUE> には人名、タイトル、日付など属性の値が入る。ユーザ発話１文に対して、生成されたメール処理言語は図5に示すような構造になる。これは、<SELECT> が<OPERATE> または<ANSWER> に係る構造になっており、これに基づいてタスクを実行することができる。例えば、ユーザ発話

「伊藤から来た、タイトルが'report' のメールを読んで」

をメール処理言語に変換すると

```
find from 'ITO'      (伊藤から来た)
find subj 'report'   (タイトルが'report' のメールを)
read                 (読んで)
```

となる。これらは上から「伊藤からのメールを検索」、「検索されたメールの中からタイトルが'report' のメールを検索」、「検索されたメールを読み上げる」という意味になる。

表 1: 文節辞書の例

入力文節	受け範疇	係り先範疇	意味
差出人が	ATTRIBUTE	VALUE(person)	(from , *)
タイトルが	ATTRIBUTE	VALUE(title)	(subj , *)
'person' の	VALUE(person)	MAIL	(* , 'person')
'title' の	VALUE(title)	MAIL	(* , 'title')
'person' が	VALUE	ATTRIBUTE(from)	(* , 'person')
'person' からの	ATT-VAL	MAIL	(from , 'person')
'person' から	ATT-VAL	COME	(from , 'person')
昨日	ATT-VAL	COME	(date , yesterday)
差出人の	ATTRIBUTE(from)	MAIL	(from , *)
差出人は	ATTRIBUTE	QUESTION(who)	(from , *)
誰ですか	QUESTION(who)	.	(* , *)
誰から	ATTRIBUTE	COME-Q	(from , *)
いつ	ATTRIBUTE	COME-Q	(date , *)
来た	COME	MAIL	(* , *)
来た	COME-Q	QUESTION(mail)	(* , *)
メール	MAIL	.	(* , *)
メールですか	QUESTION(mail)	.	(* , *)
読んで	READ	.	(* , *)

```
<SENTENCE> ::= <SELECT> <SENTENCE>(選択して処理)
             | <OPERATE>                    (操作)
             | <ANSWER>                     (質問)
<SELECT>   ::= find <ATTRIBUTE> <VALUE>   (検索)
<ANSWER>   ::= inform <ATTRIBUTE>   (質問に応答)
<OPERATE>  ::= read                        (読む)
```

図 4: メール処理言語の文法 (一部)

表 2: 変換規則

係り受け構造の範疇	メール処理言語
MAIL	find attribute(S) value(S)
QUESTION(x)	inform attribute(S)
READ	read

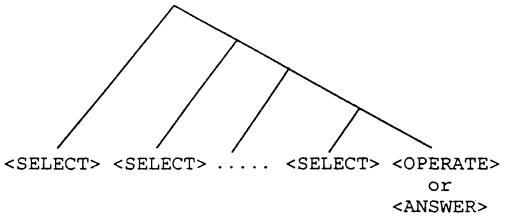

図 5: ユーザ発話 1 文の構造

3.1.2 係り受け関係を用いたメール処理言語への変換

ユーザ発話をメール処理言語文へ変換するために、係り受け構造を作成し、それをメール処理言語文へ変換する。

係り受け構造を作成の際には、表 1のような文節辞書を用いる。文節辞書は受け範疇、係り先範疇、意味からなる。範疇は、係り受け関係における文節の役割を表す。範疇には、文節が受け文節となるときの範疇 (受け範疇) と、その文節の係り先の文節の範疇 (係り先範疇) がある。係り先範疇は、文節の自立語の品詞と助詞による係り受け関係に加えて、文節の自立語の意味と、メールタスクというドメイン上での意味も考慮して定めた。なお、係り先範疇が "." である文節は係り先を持たない文節であることを示す。文節間に係り受け関係があるとき、係る文節の係り先範疇と受ける文節の受け範疇は同じである。このような係り受け関係によってできる係り受け構造の例を図 6に示す。ここで、文節の列 $p_1,\cdots,p_n$ に対する係り受け構造を s_n とすると、最後の文節 p_n の受け範疇を構造 s_n の範疇とする。また、表 1において、文節の意味とは、文節が持つメール属性とその値からなる組である。ただし、"*" は情報が未決定であることを表す。以下では文節 p_i の意味を $\mathrm{sem}(p_i)$ と書く。係り受け構造 s_n の意味 S も s_n が持つメール属性とその値からなる組であり、s_n を構成している各文節の意味から作成する。

ユーザ発話から係り受け構造とその意味を生成する手順を以下に示す。ただし、意味 (x,y),(u,v) の単一化を $(x,y) \sqcup (u,w)$ と書き、p_i の係り先範疇を $\mathrm{cat}(p_i)$ とする。

(step1) i=1、 S=(* ,*) とする。(step2) へ。

(step2) 発話入力された文節を p_i とする。(step3) へ。

(step3) $S = S \sqcup \mathrm{sem}(p_i)$ とし、 $C = \mathrm{cat}(p_i)$ とする。C = "." ならば (step5) へ。 $C = \mathrm{VALUE}(x)$ または $C = \mathrm{ATTRIBUTE}(x)$ ならば i=i+1 として (step2) へ。それ以外なら i=i+1 として (step4) へ。

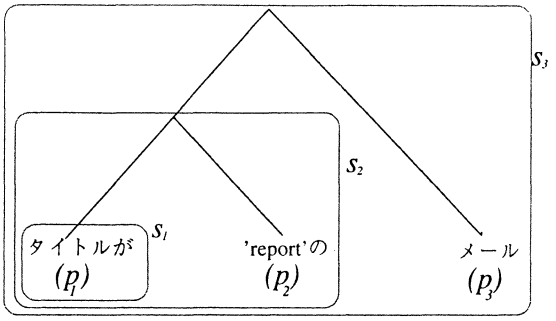

図 6: 係り受け構造

(step4) Cと同じ受け範疇を持つ文節を次入力予測文節 p_i とする。(step3) へ。

(step5) $p_1,\cdots,p_i$ に対する係り受け構造 s_i を生成する。

次に、作られた係り受け構造の範疇と表2の変換規則からメール処理言語文を生成する。係り受け構造の範疇が MAIL, QUESTION(x) のときは、係り受け構造の意味を S = (attribute(S),value(S)) とすると、 attribute(S), value(S) を用いてメール処理言語文を作成する。

ユーザ発話「タイトルが'report' のメール」に対する変換例を説明する。まず、文節「タイトルが」を p_1 とすると、S = (subj, $*$) として係り先文節を調べる。 p_1 の係り先範疇は VALUE(title) であるので、次の入力文節「'report' の」を p_2、 S = (subj,'report') として、係り先文節を調べる。 p_2 の係り先範疇は MAIL であるので、文節「メール」に係ることが予測でき、これを p_3 とする。 p_3 の係り先はないので、この時点で図6の係り受け構造を作る。この構造の範疇は p_3 の受け範疇である MAIL となる。ここで、表2より、メール処理言語文

find attribute(S) value(S)

を生成できる。 S = (subj,'report') であるので、 attribute(S) = subj、 value(S) = 'report' となり、メール処理言語文は

find subj 'report'

となる。

3.2 応答生成部

応答生成部では、メールに関する操作と質問を行うメール処理言語文と、メール検索部での検索結果を用いて、応答内容を決定する。その内容に応じて発話のタイミングを決定し、システム発話文を音声合成部に送る。本システムでは、処理可能な最小単位のメール処理言語文が生成されるとすぐに実行を開始する。したがって、メール処理言語文が生成されるとすぐに応答内容を決定し、応答内容によって適時応答する。実際に以下のように発話のタイミングを決定し、2.3節で述べた基準を達成する。

(1) ユーザの質問に答えたり、要求に応じたりする場合は基本的にユーザ発話が終った直後に応答する。ユーザ発話の無音部分により、発話の終わりを判断する。

[例] U「誰から来た？」
S「Aさんからです。」
(「来た」という発話の後の無音部分を認識してすぐ応答している。)

(2) メールの検索などで、該当するメールがない場合は、ユーザの発話途中でもすぐに応答する。

[例] U「差出人がAさんでタイトルが'news' のメ..」
S「該当メールはありません」
U「じゃあ、タイトルが'report' のメール読んで」
(ユーザの発話途中に対象となるメールがない場合、ユーザ発話に割り込んで応答することで、ユーザの無駄な発話をなくしている。)

(3) ユーザの発話が途切れた場合は、次の発話の手助けになるように、その時点での情報を応答する。

[例] U「昨日のメールで、えーっと」
U「....」
S「3通あります。」
U「誰から来てる？」
(一定時間の無音部分を認識したら、メール数を答える。)

4 おわりに

本稿では、音声対話システムにおける即時発話理解と適時応答の手法について提案した。本手法では、文節ごとに入力されるユーザ発話の係り先を予測することにより、早い段階でユーザ発話を理解する。また、タスクドメインを限定することにより、係り先が限定されることを利用する。これにより、早い段階で応答内容を決定することが可能となり、応答内容によりユーザにとって最適なタイミングでの応答を実現している。また本稿では、この手法に基づいて作成した音声対話メールシステムについて述べた。

なお、今後の課題として以下のものが挙げられる。

- あいまいな発話への対応

  本稿で提案した手法は、自立語の後の助詞によって係り先が変わるので、文法的に間違った助詞の使用や助詞の省略などには対応していない。したがって、他の言語解析手法の組み合わせるなどにより対処する必要がある。

- より良い応答タイミング

  最適な応答タイミングには個人差があり、一般にどのタイミングで応答するのが望ましいかは明らかではない。今後、いろいろな応答タイミングのシステムで実験することにより、より良い応答タイミングについて調査する必要がある。

参考文献

[1] 松永, 松原, 河口, 外山, 稲垣: 「Sync/Mail: 話し言葉の漸進的解釈に基づく即時応答インターフェイス」, 情報処理学会研究報告,SLP24-5,pp.33-40(1998).