TR-IT-0171

音声対話システムの実装

巖寺 俊哲　竹澤 寿幸　田代 敏久*　加藤 直人　森元 逞

1996.4

概要

より優れた音声言語翻訳システムあるいは音声言語処理システムを構築するためには、音声認識・言語解析等の各機能の性能を高めるとともに、各機能を有機的に結合することが必要である。また、統合時に起こり得る問題を考慮する必要がある。音声対話システムは、このような音声言語処理で必要とする一連の機能を統合し、音声による人間対システムの対話を行なうためのシステムの試作である。本報告書では1995年秋の研究発表会に展示したシステムを中心にその実装方法を解説する。

*現在 マイクロソフト株式会社

目次

はじめに

より優れた音声言語翻訳システムあるいは音声言語処理システムを構築するためには、音声認識・言語解析等の各機能の性能を高めるとともに、各機能を有機的に結合することが必要である。また、統合時に起こり得る問題を考慮する必要がある。音声対話システムは、このような音声言語処理で必要とする一連の機能を統合し、音声による人間対システムの対話を行なうためのシステム [1] の試作である。本報告書では 1995 年秋の研究発表会に展示したシステムを中心にその実装方法を解説する。

謝辞　本報告書の作成にあたって、株式会社国際電気通信基礎技術研究所企画部開発室の妹尾正身、林輝昭、谷田泰郎、松島徹、大久保吉徳[1]各氏の多大な寄与がありました。記して感謝します。

[1]現在 NTTソフトウェア株式会社

第 1 章

概要

音声対話システムは、「奈良観光における宿泊の予約」という課題をユーザとシステムの音声による対話を通じて遂行する。課題・領域の詳細については付録を参照されたい。その主な機能は以下の通りであり、主な機能構成は図 1.1の通りである。ユーザ音声およびその解析されたデータをユーザ発話データと総称し、問題解決で生成された応答およびその解析されたデータをシステム発話データと総称している。

音声認識 ユーザの音声を入力し、音声認識を行なってポーズ単位の部分木を出力する。

言語解析 音声認識結果に構文解析と意味解析を施し、意味構造を出力する。

対話構造認識 対話履歴から対話構造と話題を取り出す。

照応解析 対話構造と話題を含む対話履歴を用いて、意味構造の多義解消と補完を行なう。

問題解決 補完された意味構造で表された問題を予約データベースを使って解決し、応答を生成する。

音声合成 応答を音声出力する。

ユーザの発話毎の処理の主な流れを以下に示す。

1. ユーザ音声を音声認識・言語解析する。
2. 言語解析結果の単語列から、対話構造認識で対話構造と話題を取り出す。
3. 対話構造と話題を用いて、照応解析で言語解析結果の意味構造の多義解消と補完を行う。
4. 補完された意味構造で表された問題を、問題解決で予約データベースを使って解決し、応答を生成する。

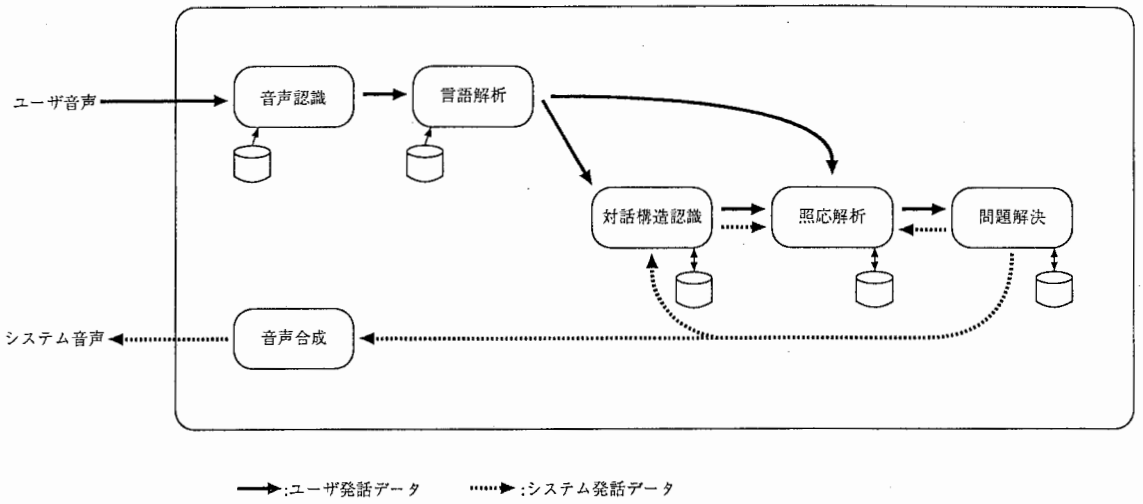


図 1.1: 主な機能構成

5. 応答の単語列から、対話構造認識で対話構造と話題を取り出す。

6. 応答の対話構造と話題を、照応解析で記録しておく。

7. 応答を、音声合成で音声出力する。

その他の機能として、メッセージを用いて全体の制御・上記機能間のデータの転送を行なうシステムマネージャと、画面に発話・対話構造・話題・意味構造・処理経過などを表示する画面表示がある。

第 2 章

全体設計

2.1 概要

音声対話システムは、サーバ・クライアントモデルを採用し、システムマネージャという唯一のクライアントと音声認識・言語解析などの「モジュール」と呼ばれる複数のサーバとから構成されている。全てのモジュール間のデータのやりとりはシステムマネージャを介して行なわれる、モジュールの独立性の高い疎結合型のシステムである。

ユーザの発話毎に処理を行なってユーザに応答し、対話を進めていくために、モジュールは以下の 4 つの手続きと内部状態「対話経過情報」を持つ。

初期化 対話開始前に対話経過情報を対話開始時の状態にする。入出力はない。

発話処理 システムマネージャにユーザから入力されたユーザ発話開始の指示により、音声認識が開始され、各モジュールは図 2.1のデータフローで定義されている入出力の流れに沿って、入力が揃ったモジュールからユーザ発話データ・システム発話データを処理していく。

入力「発話処理入力情報」と対話経過情報を参照して出力「発話処理出力情報」を出力し、対話経過情報を今回の処理を反映した状態に更新する。

システムマネージャにユーザから対話終了が入力されるまで、ユーザの発話毎に発話処理をする。

発話取消 システムが発話処理に失敗したりユーザが再発話したいとき、対話経過情報を 1 発話前の状態に戻す。入出力はない。

プログラム終了 プロセスを終了する。入出力はない。

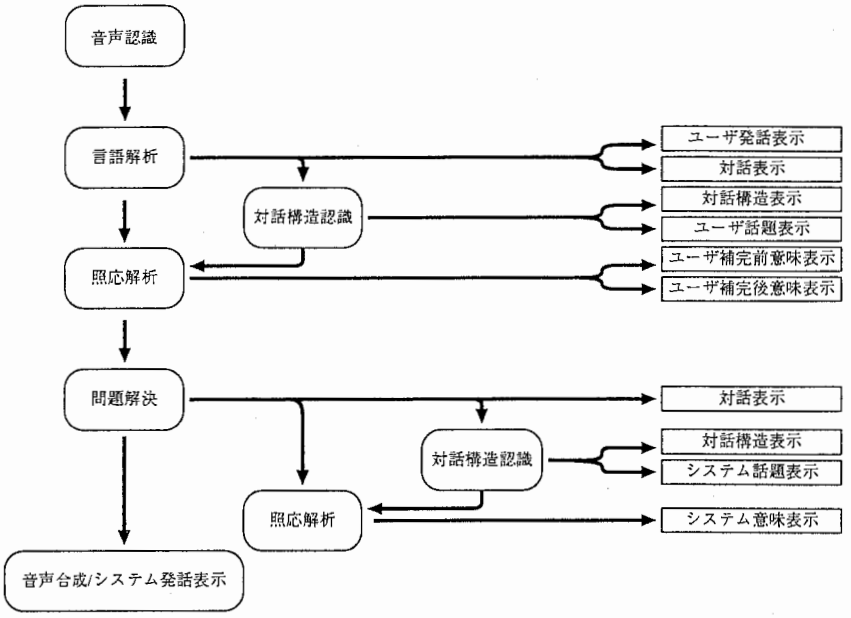


図 2.1: データフロー

2.2 モジュール一覧

第1章で挙げた主な機能は、音声合成を除き各々一つのモジュールとして実装されている。また、画面表示は表示情報毎にモジュールとして実装されている。なお、音声合成は独立したモジュールでなく、システム発話表示と合わせて1つのモジュールとなっている。以上をまとめて、音声対話システムには以下のモジュールが存在する。

モジュール	入力概要	出力概要
音声認識	ユーザ発話開始指示	ポーズ単位の部分木
言語解析	ポーズ単位の部分木	意味構造 単語列
対話構造認識	単語列	対話構造 話題
照応解析	補完前意味構造	補完後意味構造
問題解決	補完後意味構造	意味構造 単語列
音声合成/システム 発話表示	単語列	-
ユーザ発話表示	単語列	-
対話表示	単語列	-
対話構造表示	対話構造	-
ユーザ話題表示	話題	-
ユーザ補完前意味表示	補完前意味構造	-
ユーザ補完後意味表示	補完後意味構造	-
システム話題表示	話題	-
システム意味表示	意味構造	-
経過表示	処理モジュールの起動/終了	-

また、音声認識、言語解析、対話構造認識、照応解析、問題解決、音声合成/システム発話表示を「処理モジュール」、ユーザ発話表示、ユーザ話題表示、ユーザ補完前意味表示、ユーザ補完後意味表示、音声合成/システム発話表示、システム話題表示、システム意味表示、対話表示、対話構造表示を「表示モジュール」とまとめて呼ぶ。音声合成/システム発話表示はどちらにも入る。

2.3 実行環境

現在運用している環境を示す。使用しているワークステーションは以下の通り。

マシン名	製品名
atrh01	HP9000 Series 700
atrh34	HP9000 Series 700
as08	SPARCstation 5
as10	SPARCstation 10

モジュールを実行しているワークステーションは以下の通り。

モジュール	使用マシン名
システムマネージャ	atrh01
音声認識	atrh01
言語解析	atrh01
対話構造認識	atrh01
照応解析	atrh01
問題解決	as08
音声合成/システム発話表示	atrh34 + as10
表示モジュール	atrh34
経過表示	atrh34

ユーザとの入出力は、起動／音声入力／画面出力は atrh01 で行ない、音声合成のみ『しゃべりん坊』[1]が HP に対応してないため、(atrh01 と物理的に近い位置にある SPARCstation である)as10 で行なう。

画面操作イベントを常時受け付けている表示モジュールを高速で処理するため、処理モジュールと並列に、atrh34 で実行している。特に、音声合成/システム発話表示は atrh34 で動き、その中で音声合成のみ as10 上で『しゃべりん坊』を動かしている。

モジュールを起動するマシンは、モジュールが対応している範囲で立ち上げ時の指定で変更可能にしてある。

2.4 処理時間

音声認識にフレーム同期型 SSS-LR [2]を使用した時の、付録の対話例の処理時間を表 2.1に示す。

[1]音声対話システムで使用している音声合成装置。音声合成環境 (第 8.3.5 節) を参照されたい。

[2]音声認識の機能概要 (第 4.1.1 節) を参照されたい。

表 2.1: 発話処理時の各処理モジュールの処理時間 (秒)

	ユーザ発話の処理						システム発話の処理			
発話番号[2]	音声認識	言語解析	対話構造認識	照応解析	問題解決	問題解決まで	音声合成	対話構造認識	照応解析	応答完了まで
0&1	-	-	4.3	0.6	1.8	7.5	3.7	3.9	0.4	12.8
2&3	17.2	8.3	4.9	0.5	1.0	33.1	7.4	4.4	0.5	40.5
4&5	10.5	6.5	4.3	0.7	1.5	24.2	8.3	4.1	1.0	32.6
6&7	12.0	6.4	3.9	1.0	1.0	25.1	6.5	4.6	1.0	31.7
8&9	11.0	8.5	3.6	1.1	1.1	26.2	9.0	4.2	1.2	35.2
10&11	9.9	5.8	4.2	1.4	1.3	23.5	9.7	4.5	1.6	33.2
12&13	10.9	11.6	3.8	1.7	1.2	30.1	12.4	4.6	1.9	42.5
14&15	14.6	7.0	4.2	2.3	1.0	30.0	4.1	4.2	2.3	37.1
16&17	8.6	5.7	3.9	2.3	1.5	22.7	4.4	4.3	2.5	30.1
18&19	17.0	7.3	4.1	2.4	1.0	32.7	19.9	4.9	2.7	52.6
20&21	4.5	5.0	4.0	2.9	1.3	18.5	4.5	3.9	3.1	26.2

データフローの制約から、ユーザ発話の処理は逐次、表 2.1中の左から順に行なわれる。一方、システムの発話に対する対話構造認識・照応解析は音声合成と並行して処理が行なわれている。

各モジュールの処理時間は、音声合成を除き、システムマネージャがデータを送信してからシステムマネージャが出力を受信するまでの時間である。音声合成のみ、システムマネージャがデータを送信してから実際の音声が出力し終るまでの時間である。また、「問題解決まで」は、ユーザ発話開始指示からシステムマネージャが問題解決の出力を受信するまでの時間である。「応答完了まで」は、ユーザ発話開始指示から、音声合成が実際の音声を出力し終え、かつ、システム発話に対する照応解析が終るまでの時間である。この二つの処理時間にはシステムマネージャの処理にかかった時間も含めており、モジュールの処理時間の合計ではない。

表 2.1には記述されてないが、発話処理失敗時に長時間 (30 〜 60 秒) かかることがある。

表 2.2に、表 2.1と同じ条件でのユーザ体感待ち時間を示す。ユーザ体感待ち時間は、ユーザの発話終了から音声合成が物理的な音声をスピーカーから出力し始めるまでの時間である。

初期化時の処理時間を表 2.3に示す。初期化はデータフローの制約がないので、各モジュールが並行して処理を行なっている。よって、全体の初期化に要する時間は、対話構造認識の初期化の時間となる。

[2]付録の対話例において、発話順に各発話に付与された番号である。ユーザの発話とそれに応答するシステムの発話の番号をまとめて示す。

表 2.2: 発話処理時のユーザ体感待ち時間 (秒)

発話番号	待ち時間
0&1	(8.6)
2&3	24.1
4&5	20.3
6&7	23.7
8&9	21.3
10&11	21.6
12&13	23.6
14&15	32.2
16&17	24.4
18&19	29.5
20&21	18.5

表 2.3: 初期化時の各処理モジュールの処理時間 (秒)

音声認識	言語解析	対話構造認識	照応解析	問題解決	音声合成	初期化完了まで
-	-	39.3	0.1	0.4	-	39.3

2.5 今後の課題

2.5.1 高機能化

モジュールメッセージ交換機能のテンプレート化 モジュール側でのシステムマネージャとの通信機能・各手続きの振り分け・起動判定をマクロまたは (クラス) ライブラリで提供できるようにしたい。

起動タイミング判断とか出力結果の振り分けはプログラム可能にして、デフォルトはテンプレートでパラメータのみの設定で可能としたい。

次発話の予測による音声認識精度向上 文献 [1] でも述べられているが、対話構造認識・照応解析から次発話の予測を行ない、この予測を利用して音声認識の精度を向上させることが検討されている。

次発話予測を行なうモジュールが、次発話タイプを出力するのであれば、現在の音声対話システムの枠組で、組み込み可能である。

モジュールの処理単位の柔軟化 発話処理単位が、現在は全てのモジュールで 1 発話となっており、1 発話は (複数解を考えないと) 意味構造 1 個に対応している。また、発話は必ずユーザとシステムが交替に行なうとされている。

ユーザの 1 発話が長くなり、複数の意味構造に対応するようになった場合、もしくは、ユーザが連続して発話する (システムが発話を終えないうちに次の発話を始め

てしまう) 場合を考えてみる。この時、仮に、言語解析が 1 発話に対し複数回出力 (各出力は意味構造 1 個) を行なうようになれば、システムマネージャが言語解析の複数回出力を別のデータとして流すようにするだけで、言語解析以降の処理モジュールは改造を行なわずに使用可能にできる。

多プログラミング言語対応　音声対話システムは、基本的に C で作成されており、一部 (言語解析の一部・対話構造認識・照応解析) で Lisp が使用されている。ソフトウェアとしての柔軟性やソフトウェア開発がオブジェクト指向へ移行しつつある情勢を考えると、Java,C++,python 等で開発されたモジュールとも容易に結合できるようにシステムマネージャや上記テンプレートをオブジェクト指向の方向で整備していくことが考えられる。

2.5.2 高速化

表 2.2に示されるユーザ体感待ち時間を短くするため、全体的に発話処理の高速化の必要がある。それ以外にも以下が高速化の課題として挙げられる。

発話処理失敗時の中止機能　発話処理で長時間処理したあげく失敗することがある。一定時間後に強制中止して、発話取消をする機能が必要ではないか。

対話構造の初期化　対話構造の初期化が遅い。原因は、対話構造認識の初期化自体が遅いのと、システムマネージャが (ユーザ発話用とシステム発話用で) 2 回初期化を送信していることにある。

現状、システムマネージャが対話構造の初期化を行なっているのはシステム起動時と対話終了時なので、対話開始選択後発話がすぐに始められないといった事態にはなってないが改良が必要である。

第 3 章

システムマネージャ

3.1 機能

システムマネージャは、メッセージを用いて全体の制御・モジュール間のデータの転送を行なう。

システムマネージャとモジュールがプロセス間通信する情報 (文字列) を「メッセージ」と呼ぶ。メッセージは、モジュールに起動を依頼している手続き名 (モジュールの持っている手続きである初期化、発話処理、発話取消、プログラム終了がある。その他にユーザ発話開始指示、開始通知、終了通知がある。) とその手続きの入力データからなる。メッセージによって、発話時の処理は以下のように行なわれる。

1. システムマネージャからモジュールに発話処理メッセージが送信される。
2. モジュールが、システムマネージャから発話処理メッセージを受信し、受信したメッセージで発話処理に必要な入力データが揃ったならば発話処理を行なう。[1] 図 2.1のデータフローで出力が記述されているモジュールでは処理結果のメッセージをシステムマネージャに送信する。
3. モジュールから処理結果のメッセージを受けとったシステムマネージャは、データフローの記述に従ってそのメッセージを他のモジュールに送信する。

こうして、ユーザが発話する度に、音声認識から始まって、各モジュールの発話処理結果がメッセージとしてデータフローに従って順次出力され、最後にシステムの応答が音声合成される。

システムマネージャとモジュールは、実際には UNIX ネットワーク上の INET ドメインのソケットによるプロセス間通信を用いてメッセージを交換する。システムマネージャを受動側、モジュールを能動側としている。

3.2 動作

発話処理以外も含んだ、システムマネージャ全体の状態遷移を図 3.1に示す。

[1] 入力データの種類は、種類毎にソケットを用意してどのソケットからの入力かで判断している。

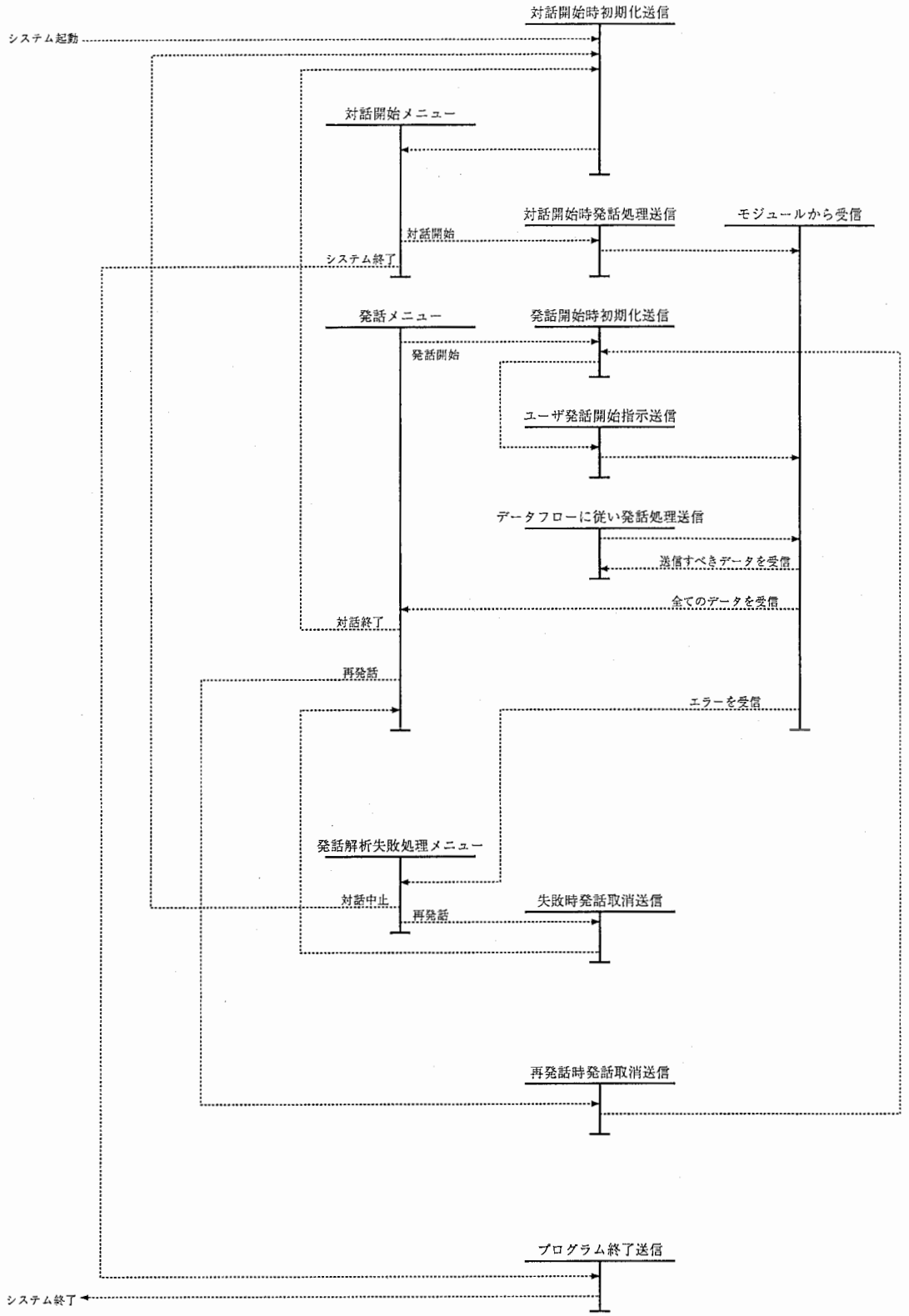


図 3.1: システムマネージャの状態遷移

図3.1では、状態をT字型の図形で表し、横線の上に状態名を記す。状態から状態への遷移は矢印付の点線で示し、その遷移が起こる条件を点線の脇に記す。例えば、「対話開始メニュー」で、対話開始メニューを画面に表示しユーザの選択を待っている状態を表している。そして、ユーザが「対話開始」を選択したら「対話開始時発話処理送信」に遷移し、ユーザが「システム終了」を選択したら「プログラム終了送信」に遷移することを状態間の矢印で表している。

以下で、図3.1の中の各状態を説明する。

対話開始メニュー 「対話開始」・「システム終了」という項目からなるメニューを表示し、ユーザに1項目選択させる。

発話メニュー 「発話」・「シミュレーション」・「再発話」・「対話終了」という項目からなるメニューを表示し、ユーザに1項目選択させる。「発話」ならマイクからユーザ音声を入力する発話処理の発話開始、「シミュレーション」ならシミュレーションとしてユーザがメニューから選んだ文字列を入力とする発話処理の発話開始、「再発話」なら再発話、「対話終了」なら対話終了を選択したものとする。

発話解析失敗処理メニュー 「再発話」・「対話終了」という項目からなるメニューを表示し、ユーザに1項目選択させる。

対話開始時初期化送信 音声認識・言語解析以外の全てのモジュールに、対話経過情報の初期化と画面クリアのため、初期化メッセージを送信する。(音声認識・言語解析は、対話経過情報を持たず、画面クリアも行なわないので送られない。)

発話開始時初期化送信 1発話内での情報の画面表示をクリアするため、1発話内での情報のみを表示している以下のモジュールに初期化メッセージを送信する。

ユーザ発話表示 ユーザ話題表示 ユーザ補完前意味表示 ユーザ補完後意味表示
音声合成 / システム発話表示 システム話題表示 システム意味表示 経過表示

対話開始時発話処理送信 以下を、言語解析からの出力[2]のダミーとする発話処理メッセージを対話構造認識と照応解析へ送信する。

```
(
((<pt-of-sp> システム呼出し))

(
((MAIN HELLO))
)

)
```

[2]言語解析の発話処理出力情報 (第5.1.4節) を参照されたい。

ユーザ発話開始指示送信 音声認識にユーザ発話開始指示メッセージを送信する。マイクからユーザ音声を入力するか、シミュレーションとしてメニューから選んだ文字列の入力で済ませるかも指定する。

データフローに従い発話処理送信 図2.1のデータフローに記述されている、あるモジュールの入力データを受信したら、そのデータをそのまま発話処理メッセージの入力データとして該当モジュールに送信する。

失敗時発話取消送信 発話取消を、図2.1のデータフローに従い発話処理が行なわれた以下のモジュールに、発話処理が行なわれた回数、発話取消メッセージを送信する。

モジュール	発話取消を送信する最大回数	モジュール側の動作
対話構造認識	2回	
照応解析	2回	発話処理入力情報が両方とも入力されていたら発話取消を行なう。
問題解決	1回	
対話表示	2回	
対話構造表示	2回	

再発話時発話取消送信 発話取消を以下のモジュールに以下の回数、発話取消メッセージを送信する。

モジュール	発話取消を送信する回数
対話構造認識	2回
照応解析	2回
問題解決	1回
対話表示	2回
対話構造表示	2回

プログラム終了送信 全てのモジュールにプログラム終了メッセージを送信する。

モジュールから受信 図2.1のデータフローに記述されている、あるモジュールの入力データとなっているデータを受信したら、「送信すべきデータを受信」としてこの状態を終了する。

図2.1のデータフローに記述されている、データを全て受信したら、「全てのデータを受信」としてこの状態を終了する。

エラーを受信したら、「エラーを受信」としてこの状態を終了する。

以上が、図3.1の中の各状態の説明だが、図3.1に記述してないシステムマネージャの処理として、発話処理時に、処理モジュールに開始指示または発話処理を送信すると、経過表示モジュールにその処理モジュールの開始通知を送信し、処理モジュールからデータを受信すると、経過表示モジュールにその処理モジュールの終了通知を送信している。

3.3 メッセージの仕様

以下の「"」で囲まれた文字列をメッセージとして使用する。

システムマネージャからモジュールへ

手続き名	メッセージ
初期化	"INIT"
発話処理	"SS<改行><入力データ><改行>SE"
発話取消	"UNDO"
プログラム終了	"QUIT"
ユーザ発話開始指示	"START <mode>"
開始通知(経過表示に対してのみ)	"SS <module_id>"
終了通知(経過表示に対してのみ)	"SE <module_id>"

モジュールからシステムマネージャへ

手続き名	メッセージ
発話処理正常終了と出力データ送信	"SS<改行><出力データ><改行>SE"
発話処理異常終了	"ERR"

なお、<改行>、<入力データ>、<出力データ>、<mode>、<module_id> はメタ記号で、その値と意味は以下の通り。

<改行>　　改行文字

<入力データ>　　UNIX path name
<出力データ>　　UNIX path name
full path で指定された file の内容は ASCII コードまたは EUC コードで、
各モジュールの発話処理出力情報の節の通り。
発話処理出力情報のデータ形式は、BNF によって記述する。
'*' で 0 回以上の繰り返しを表し、'+' で 1 回以上の繰り返しを表す。
'[' と']' で囲んで 0 回または 1 回の生起を表す。

<mode>　　SPEECH または SIMULATION
音声認識を実際に行なうかシミュレーションで済ますかを指定する。

<module_id>　　1 から 6 の数字
以下の対応で、処理モジュールを表す。
1 : 音声認識
2 : 言語解析
3 : 対話構造認識
4 : 照応解析
5 : 問題解決
6 : 音声合成 / システム発話表示

3.4 今後の課題

発話処理出力情報ファイルの複数化　各モジュールの発話処理出力情報は、現在 1 つのテキストファイルにすべて書き込まれている。これを複数のファイルに別内容を書き込み可能とする。

メリットとしては、(1) 発話処理出力情報を利用するモジュールが、自分が必要な情報を取り出す手続きが簡単になる、(2) 発話処理出力情報の形式変更時に調整するモジュールが少なくなる、といった点が挙げられる。

発話処理の起動の柔軟化　現在のシステムマネージャの実装では、データフローとして定義されている各メソッドの実行は1発話に1回しか実現されず、発話処理メソッドを同じ発話に対して再実行できない。

例えば、言語解析の結果を承けて条件を変化させて音声認識を行ない、新しい音声認識結果を基に再度言語解析を行なう、という任意回実行可能なループは定義できない。

しかし、現在のシステムマネージャの枠組を拡張することによって対応可能である。そうすると、発話取消メソッドと組み合わせてバックトラックも出来るようになる。

第 4 章

音声認識機能

音声対話システムにおける音声認識について述べる。音声認識は全体で一つのモジュールをなし、システムマネージャと交信して機能を実現している。

4.1 音声認識

4.1.1 機能概要

音声認識では、音声入力、音声波形表示、音声認識を行なう。

音声入力 接話型マイクから入力された音声を DATLINK を用いて A/D 変換をしてサンプリング周波数が 12KHz の 16bit データを作成する。

音声波形表示 入力された 1 発話分の音声データの波形を表示する。なお、水色で囲まれた範囲がポーズ単位に切り出された波形を表していて、横軸の単位は時間 (msec) である。

音声認識 音声認識は SSS-LR 連続音声認識プログラムを使用している。

SSS-LR 連続音声認識プログラムは、HMM による音素照合部、LR パーザによる統語解析部とから構成されている。

音声認識用文法を拡張 LR パーザで駆動して音素予測をすることにより音素列仮説を作成する。つぎに、予測した音素を音素環境依存 HMnet(SSS) を用いてビタービサーチにより入力音声データの探索をして音響尤度を計算する。音素列仮説木を横型探索する N ベスト探索を行なうことにより連続音声認識を行なう。

現在 SSS-LR 連続音声認識プログラムには、音素同期型 SSS-LR [3] とフレーム同期型 SSS-LR [4, 5] の 2 種類がある。いずれも、文法で受理できる音素列だけではなく、単語区切りまで統語解析が終了した音素列も部分木として出力できるようになっている [8]。音素同期型にも、同じ音素系列に対する複数の文法スタックは一つにパッキングする改良がされている。

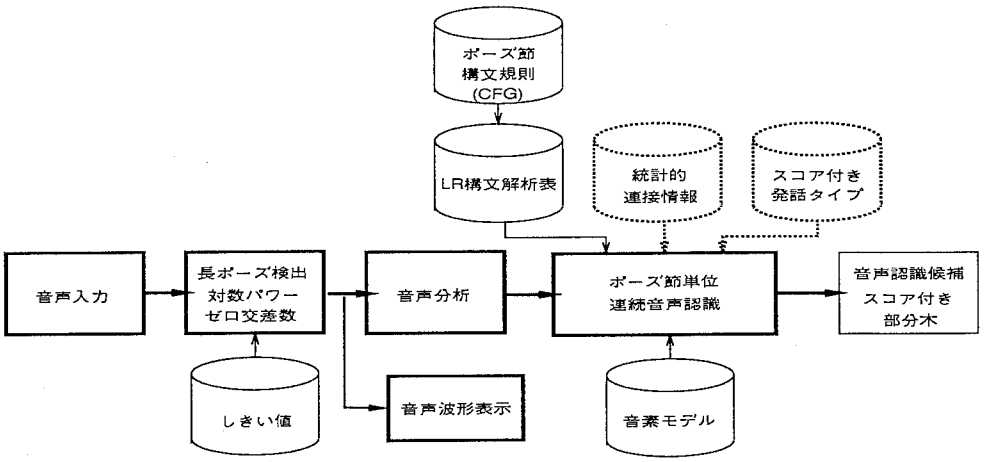


図 4.1: 音声認識部の機能構成図

音声対話システムでは、音素同期型 SSS-LR およびフレーム同期型 SSS-LR を利用することが出来る。

モジュールの手続きとしては、発話処理とプログラム終了を持つ。発話内の情報のみを参照して、対話経過情報を持たないので、初期化及び発話取消はない。上述の処理は全て発話処理で行なわれる。

4.1.2 機能構成

図 4.1に機能構成を示す。ここで、破線は音声対話システムでは実装していないが将来実装する予定の機能である。

4.1.3 実行環境

ワークステーション HP9000/700 シリーズあるいは Sun SPARC station シリーズで動作する。現状では 音声入力装置として DASBOX あるいは DATLINK が必要である。

4.1.4 ユーザ発話開始指示メッセージ

以下のメッセージをシステムマネージャから受信する度に、音声認識を行なう。

- ユーザ発話開始指示

ユーザ発話開始指示の mode が SPEECH なら、音声入力装置を介してユーザが発話した音声を認識する。ユーザ発話開始指示の mode が SIMULATION なら、ユーザがメニューから選択した文字列を音声認識したものとする。

4.1.5 発話処理出力情報

音声認識を行なう度に、以下で定義されている音声認識結果をシステムマネージャに送信する。

```
音声認識結果 ::= 文の始まり 改行 {ポーズ単位の音声認識候補 改行}+ 文の終り

文の始まり ::= SS

ポーズ単位の音声認識候補 ::= ポーズ開始 改行 {[i:j] 部分木 改行}+ ポーズ終了
ポーズ開始 ::= PS
ポーズ終了 ::= PE
i ::= 自然数
j ::= 自然数
部分木 ::= (非終端記号 部分木 *) | 終端記号
非終端記号 ::= ASCII 文字列
終端記号 ::= EUC 文字列

文の終り ::= SE
```

4.1.6 対話経過情報

なし。

4.1.7 発話処理実行例

「それぞれおいくらなんですか」というユーザ音声に対する認識成功時、以下を発話処理出力情報としてシステムマネージャに送信する。

```
SS
PS
[1:1]
(
(<advp>
   (<adv>  それぞれ))
)
:
:   < 省略 >
:
PE
PS
[1:1]
(
(<_start>
   (<cl1>
      (<vp-sfp>
         (<vaux-sfp>
```

```
                (<vaux-noda-syusi>
                   (<vp-no>
                      (<vp-rentai>
                         (<verb-cop-rentai>
                            (<np>
                               (<wh-pro>  おいくら))
                            (<aux-cop-da-rentai>  な)))
                      (<p-jun>  ん))
                   (<aux-cop-desu-syusi>
                      (<auxstem-desu>  で)
                      (<vinfl-spe-su>  す)))
                (<aux-sfp>  か)))))
)
:
:   < 省略 >
:
PE
SE
```

認識失敗時は、以下を発話処理出力情報としてシステムマネージャに送信する。

```
SS
PS
[0:1]
(
  (DUMMY DUMMY)
)
PE
SE
```

4.1.8 SSS-LR の統語解析部

同じ音素系列の仮説に対して構文木の曖昧さにより複数の文法スタックが生じるが音声認識では、これを一つにパッキングして処理をする必要がある。

フレーム同期型 SSS-LR では既に解説されている [4] ので、ここでは音素同期型 SSS-LR の方法を説明する。

いずれの方法も複数の音素系列をトライ構造に展開していくときの、1つのノードに対する LR アルゴリズムとなっている。図 4.2のようにセルから LR スタックを分離したデータ構造にして、同じ音素系列のセルになる LR スタックは、すべてマージするような LR アルゴリズムを考える。

- REDUCE 処理部

  1. すべてのセルに対して
  2. すべての LR スタック列に対して
     (a) LR スタックの top state を求める

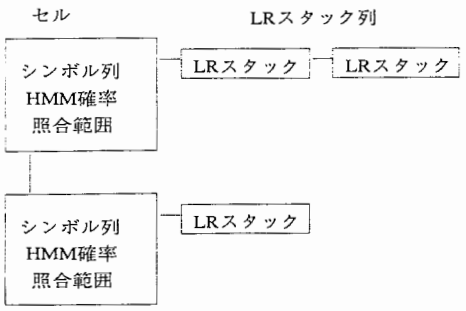


図 4.2: データ構造

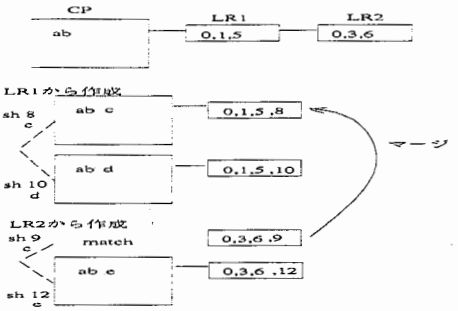


図 4.3: SHIFT 処理

(b) 予測されるアクションが REDUCE であれば

i. LR スタックのコピーを作成

ii. REDUCE 処理を行なう

iii. LR スタック列にリンクする

- REDUCE 処理直後で音声データ終了まで達しているセルは部分解リストへ残す
- ACCEPT と SHIFT 処理部

1. すべてのセルに対して
2. すべての LR スタック列に対して
   (a) LR スタックの top state を求める
   (b) 予測されるアクションが ACCEPT で音声データ終了まで達しているセルであれば、受理解リストへ残す
   (c) 予測されるアクションが SHIFT であれば
      i. newLR スタックを作成する
      ii. new state を push する
      iii. 予測されたシンボルがすでに音素照合していれば
         A. newLR をその LR スタック列にマージする
      iv. 音素照合していなければ
         A. new セルを作成する
         B. 音素照合する
         C. newLR を new セルにリンクする

以上のアルゴリズムは HMM-LR(拡張 LR) 用であるので、SSS-LR(音素環境依存) 用にはさらに triphone 予測 (次音素の次の音素も予測) に拡張する必要がある。その方法は次のようにする。

1. triphone の先行音素はセルのシンボル履歴を見れば分かるが、中心音素と後続音素は LR スタックの top state により異なるので、LR スタック毎に後続音素を保持するようにする。
2. REDUCE のときには、REDUCE 操作を行なった newLR スタックの後続音素を REDUCE の予測音素とする。

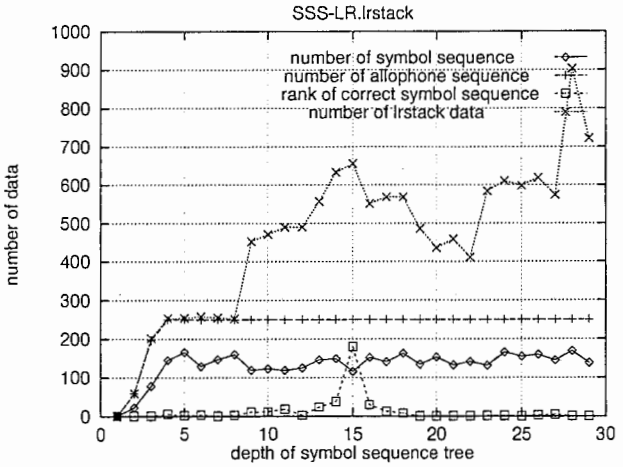


図 4.4: データ構造を改善した音素同期型 SSS-LR における異なる仮説数の推移

3. SHIFT のときには、分岐する条件が異なる中心音素と後続音素分だけある。したがってすべての組合せのなかで中心音素と後続音素が同じものはマージすることができる。

図 4.4に、データ構造を改善した音素同期型 SSS-LR で、1 文についてビーム幅 250 で認識途中の重複を省いた音素列仮説数、異音列仮説数、正解候補の順位および LR スタック総数を音素ツリーの深さごとにプロットしてみた。
従来は LR スタックの数が枝刈りのビーム幅であったが、構文木の曖昧性をすべてパッキングすることにより triphone を考慮した異なる音素系列の数がビーム幅になっていることがわかる。このデータではビーム幅 (250) の約 57%(143) が平均の異なる音素系列仮説数となっている。

4.1.9 探索手法

入力音声データのポーズ情報に着目し、ポーズで区切られた区間 (ポーズ単位とよぶ) を音声認識の単位とする横型探索手法を用いている。ポーズの正確な判別の課題はあるが、ポーズ単位を統語解析するための音声認識文法を用いて音素系列仮説木を予測し単語区切りに到達した音素系列仮説を音響スコアの高い順に出力する。音素系列仮説木を横型探索する手法は音素系列仮説木の深さが深くなると処理時間がかかる欠点があるが、文よりも短いポーズ単位を認識の単位とすることにより探索空間を小さくして処理時間の増大を防ぐことができる。

4.1.10 部分木出力用文法による SSS-LR の性能評価

語彙数の異なる 3 種類の文法を使って音素同期型 (PS)SSS-LR とフレーム同期型 (FS)SSS-LR の性能評価を ATR 音声言語データベースから選んだ対話音声により行なった。実験条件を以下にしめす。

音響モデル　話者適応モデルでフレーム長が 5msec(MAP-VFS, 状態数201, 混合数5) とフレーム長が 10msec(VFS, 状態数201 混合数5)
音声データ　TAS12009(ホテルの部屋のキャンセル) の申込者の発話である 24 ポーズ単位 (13 文)
音声データ長　1380msec/ポーズ単位
実行マシン　HP 9000/755/125

音声認識文法の諸元

名称	単語 (形態素) 数	規則数	perplexity	
			音素	単語 (形態素)
S-2-1	317	1395	2.80	18.57
M-2-1	561	1567	3.21	39.06
L-2-1	1010	1809	3.87	71.23

この文法は部分木を単位とする音声認識用日本語文法 [6] で、動詞と後置詞句の共起制約を規則化した制約をもっている。

上位 100 位までのポーズ単位累積認識率の結果を図 4.5に示す。認識率は正解のポーズ単位音素系列と認識候補の音素系列が全く同じものから求めている。また、上位 20 位までの単語認識率の結果を図 4.6に示す。これは言語モデルとして音声認識用日本語文法だけを使ったときの SSS-LR の性能である。

また音声対話システム用に収録した朗読発声の音声データ 18 ポーズ単位 (11 文) における認識結果を次に示す。

音響モデル　話者適応モデルでフレーム長 10msec (状態数401, 混合数1)
音声データ長　1361msec/ポーズ単位
音声認識文法　247 単語, 409 規則, perplexity 2.47/音素 26.8/単語
実行マシン　HP 9000/755/125

	ビーム幅	ポーズ単位認識率 (TOP20) (%)	CPU 処理時間 (msec/ポーズ単位)
音素同期型 SSS-LR	B100	66.7	4096
フレーム同期型 SSS-LR	B500	72.2	2002

- 音素同期型のビームの単位は音素系列数であるが、フレーム同期型は音響モデルの状態 (GRID) 数である。
- 音声対話システムではこのビーム幅を採用している。

この実験結果より次のようなことが分かる。

- 音声対話デモシステム用の文法では認識率は約 70% 程度で認識処理速度 (CPU Time) は音素同期型で音声データ長 (1361msec) の 3 倍、フレーム同期型で 1.5 倍程度である。

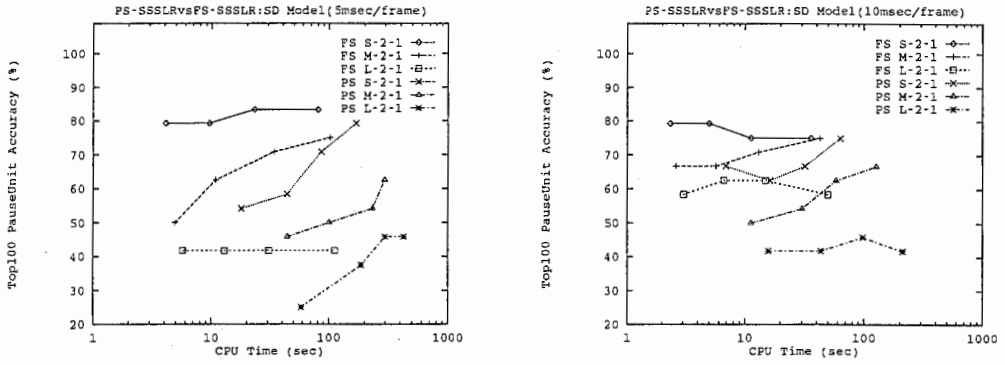


図 4.5: 上位 100 位までのポーズ単位累積認識率 (5msec モデルと 10msec モデル)

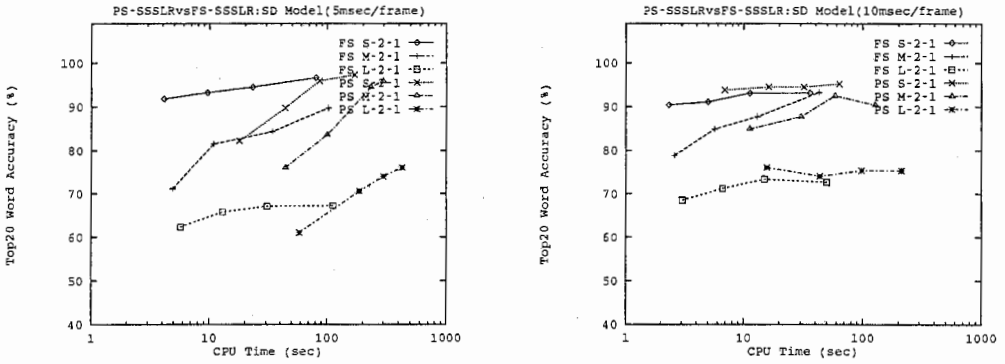


図 4.6: 上位 20 位までの単語認識率 (5msec モデルと 10msec モデル)

- 朗読発話では両者の認識率の差はそれほど無いが、自由発話に近くなるとビームが小さい方ではフレーム同期の方が性能が良い。例えば、「-,t,o,-」と発話している音声データ (605msec) ではフレーム同期では正解が出力されるが、音素同期では「-,t,o,o,o,o,o,-」「-,t,o,o,o,o,o,o,-」が出力される。

- 出力候補数は音素同期の方が圧倒的に多く、フレーム同期型のほうが寡黙であるが、正解を多く含んでいるのはフレーム同期型である。

- 文法的に受理するものだけを出力するものより音素認識率は良くなっている [7] が、文末が無意味な数詞や活用語尾で終る非文が多数出力される。

- 文法 S-2-1 で 3 ポーズ単位を認識したときの主記憶使用量は音素同期 (B250) で 29.5Mbytes、フレーム同期 (B1000) で 8.5Mbytes である。バッチ処理で 1 会話を処理する時は、これの 3 倍から 5 倍は必要である。

4.1.11 今後の課題と予定

- 文法による制約だけでは L-2-1 のような文法は認識が困難である。局所的な制約条件としてはたらく統計的言語モデルを併用して制約を強くした認識手法 [7] をデモシステムでも採り入れていく予定である。

- 認識処理時間に関しては図 4.6に示しているが、これは CPU 処理時間であるので実時間では更に 2 倍近く必要となる。効率の良い探索手法はグラフ構造に仮説を保持することであろうが、認識でパーザを使って単語単位のマージを行なうには次のような方法が考えられる [9, 10]。

  - 文法を規則と辞書に分離した 2 段 LR を採用する。
  - 辞書は前終端記号予測付き LR 表を利用する。
  - 同一時刻に規則 LR により予測された前終端記号が同じものはすべてマージをして辞書探索をする。
  - 規則 LR により作成される前終端記号の木構造のノードに統計的言語モデルの計算値を与えておくことにより、現在照合中の単語の言語尤度を計算する。

- デモシステムでの対話開始、途中解除、終了などの合図、あるいは対話中の次発話や話題の予測が可能であれば、その状況に応じて認識に使用する言語モデルを変更することが出来るようにすることも考えていく予定である。

第 5 章

言語解析機能

音声対話システムにおける言語解析について述べる。言語解析は全体で一つのモジュールをなし、システムマネージャと交信して機能を実現している。

5.1 言語解析

ここでは、統語解析、意味解析が実現する機能を総称して言語解析とよぶ。この言語解析全体の機能をまず記述し、その後、統語解析、意味解析それぞれの機能を記述する。

5.1.1 機能構成

言語解析は一つのモジュールとしてシステムマネージャと結合される。

5.1.2 機能概要

発話処理は、音声認識結果を入力として統語解析を行ない、その出力を入力として意味解析を行なう。統語解析、意味解析については、各々第 5.1.8 節、第 5.1.9 節を参照されたい。

モジュールの手続きとしては、発話処理とプログラム終了を持つ。対話経過情報がないので、初期化及び発話取消はない。

5.1.3 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに対する言語解析を行なう。

- 音声認識結果

5.1.4 発話処理出力情報

受信した発話データに対する言語解析を行なう度に、以下で定義されている言語解析結果をシステムマネージャに送信する。

```
言語解析結果 ::= (単語列 補完前意味構造列)

単語列 ::= (単語 +)
単語 ::= (品詞 表層文字列)
表層文字列 ::= EUC 文字列
品詞 ::= ASCII 文字列
```

補完前意味構造列は、多義の時、複数の補完前意味構造からなる。

```
補完前意味構造列 ::= (補完前意味構造 +)
補完前意味構造 ::=   ((MAIN HELLO))
                   | ((MAIN REQUEST)
                      (OBJ MAIN_OBJ 素性値) [(ANAPHORA ANAPHORA 素性値)])
                   | ((MAIN INFORM) (OBJ MAIN_OBJ 素性値))
                   | ((MAIN YES))

ANAPHORA 素性値 ::= ((MAIN 文字列))

MAIN_OBJ 素性値 ::=  ((MAIN RESERVE) (OBJ VERB_OBJ 素性値))
                    | ((MAIN TELL) (OBJ VERB_OBJ 素性値))

VERB_OBJ 素性値 ::= ((MAIN {TOUR| 変数})
                     [(COURSE TOUR_COURSE 素性値)]
                     [(PARTICIPANT PARTICIPANT 素性値)]
                     [(DATE_FROM DATE 素性値)]
                     [(DATE_TO DATE 素性値)]
                     [(HOW_LONG DAYS 素性値)] )
変数 ::= ?X1 | ?X2 | ...

TOUR_COURSE 素性値 ::= ((MAIN {文字列 | 変数})
                        [(ROOM_TYPE ROOM_TYPE 素性値)]
                        [(DINNER DINNER 素性値)])
ROOM_TYPE 素性値 ::= 変数 |((MAIN 変数) (HOW_MUCH HOW_MUCH 素性値))
DINNER 素性値 ::=  ((MAIN {文字列 | 変数}) [(COURSE DINNER_COURSE 素性値)])
DINNER_COURSE 素性値 ::= 変数 | ((MAIN 変数) (HOW_MUCH HOW_MUCH 素性値))
HOW_MUCH 素性値 ::= ((MAIN 変数) [(DEGREE 最小)])
PARTICIPANT 素性値 ::= ((MAIN PERSON)
                        [(NAME NAME 素性値)]
                        [(PHONE_NUMBER PHONE_NUMBER 素性値)])
NAME 素性値 ::=  文字列 [1]
PHONE_NUMBER 素性値 ::= 文字列 [1]
```

[1] '((MAIN 文字列))' と定義すれば、他の素性値の定義と統一できる。

DATE 素性値 ::= 文字列 [1]
DAYS 素性値 ::= 文字列 [1]

なお、補完前意味構造中の変数は全て異なる。以上の定義に従って記述された補完前意味構造が表す意味は、例えば、

```
((MAIN REQUEST)(OBJ  ((MAIN RESERVE) (OBJ ((MAIN TOUR)
 (COURSE ((MAIN 奈良観光)))))))
```

において、'((MAIN REQUEST)(OBJ' は OBJ 以下を要求する要求表現であることを表わし、'((MAIN RESERVE) (OBJ ((MAIN TOUR)' は「対象が宿泊である予約」を表わし、'((MAIN TOUR) (COURSE ((MAIN 奈良観光))))' は「コースが奈良観光である宿泊」を表わす。

5.1.5 対話経過情報

なし。

5.1.6 参照情報

統語解析、意味解析を参照されたい。

5.1.7 発話処理実行例

音声認識の発話処理実行例 (第 4.1.7 節) の発話処理出力情報が発話処理入力情報の時、以下を発話処理出力情報としてシステムマネージャに送信する。

```
(
((<ADV> それぞれ) (<WH-PRO> おいくら) (<AUX-COP-DA-RENTAI> な)
 (<P-JUN> ん) (<AUXSTEM-DESU> で) (<VINFL-SPE-SU> す) (<AUX-SFP> か))

(((MAIN REQUEST)
  (OBJ  ((MAIN TELL)
         (OBJ ((MAIN ?X1)
               (COURSE ((MAIN ?X2)
                        (ROOM_TYPE ((MAIN ?X3)
                                    (HOW_MUCH ((MAIN ?X4)))))))))))
  (ANAPHORA ((MAIN それぞれ))))
 ((MAIN REQUEST)
  (OBJ ((MAIN TELL)
        (OBJ ((MAIN ?X2)
              (COURSE ((MAIN ?X3)
                       (DINNER ((MAIN ?X4)
                                (COURSE ((MAIN ?X5)
                                         (HOW_MUCH ((MAIN ?X6)))))))))))
  (ANAPHORA ((MAIN それぞれ)))))
)
```

5.1.8 統語解析

認識誤りや非文法的な文を含む音声認識結果を効率的にかつ精度良く統語解析するために、多種多様な言語知識を有効に利用でき、外部モジュール[2]と容易にリンクできる構文解析ツールキット [2] を用いて実現した。

機能構成

理想的には、構文解析ツールキットに用意されている依存構造解析モジュール[2]、格構造解析モジュール、格構造併合モジュールから構成されるべきであるが、モジュールで利用される知識の信頼性、安定性等の理由[3]から、音声対話システムにおいては、依存構造解析モジュールのみの構成となっている。

機能概要

複数の部分木からなる音声認識結果を、タスク・ドメインに依存しない統語的な知識 (構文解析規則に係り受けのタイプを付与したもの) を用いて解析し、意味解析に渡すための依存構造を作成する。

入力情報

音声認識結果

出力情報

出力 ::= (依存構造 +)
依存構造 ::= 素性構造
素性構造 ::= ((素性名 素性値)+)
素性名 ::= 見出し | カテゴリ | 引数
素性値 ::= (素性構造 +) | 文字列

参照情報

構文解析規則に、係り受け関係のタイプを付与したものを書き換え規則として参照している。

参照情報 ::= (係り受けタイプ付き構文解析規則 +)
係り受けタイプ付き構文解析規則 ::= (構文解析規則 係り受けタイプ)
構文解析規則 ::= (左辺 右辺 +)
係り受けタイプ ::= 0 | 1 | 2 | 3 | 4
左辺 ::= 文字列
右辺 ::= 文字列

---

[2]ここでいうモジュールは、音声対話システムの用語としてのモジュールではなくソフトウェア一般の用語である。

[3]音声対話システム作成時には、一般的でかつ信頼性の高い格解析及び格構造併合のための知識が用意されていなかった。

統語解析の参照情報イメージ

```
(((<start>  <_start>) 0)
 ((<_start>  <cl>) 0)
 ((<_start>  <cl1>) 0)
 ((<_start>  <cl2>) 0)
 ((<cl>  <interj-pre>) 0)
 ((<cl>  <interj-post>) 0)
 ((<cl1>  <np> <interj-post>) 1)
 ((<cl2>  <conj> <cl1>) 1)
 ((<cl2>  <adv-cl> <cl1>) 1)
 ((<cl1>  <vp>) 0)
 ((<cl1>  <vp-sfp>) 0)
 ((<vp>  <verb-cop-syusi>) 0)
 ((<vp>  <vaux-te-meirei>) 0)
 ((<vp>  <vaux-masu-syusi>) 0)
 ((<vp>  <vaux-ta1-syusi>) 0)
          :
          :
```

付与した係り受けタイプの説明は以下の通り。

0 : 右辺の要素の数が１つの場合は、単に右辺の情報を伝搬する。

1 : 右辺の要素の数が２つで、右側の語が統語上の head となる場合、
左側の語を右側の語の子（娘）要素とし、右側の語の情報を伝搬する。

2 : 右辺の要素の数が２つで、左側の語が統語上の head となる場合、
右側の語を左側の語の子（娘）要素とし、左側の語の情報を伝搬する。
日本語では、めったにないが、メカニズム上用意している。

3 : 右辺の要素の数が２つで、２つの語の間に親子関係は存在しないが、
意味的な中心は左側の語である場合、２つの語を１つのノードに併合し、
左側の語の情報を伝搬する。
日本語の句構造規則では、動詞語幹と語尾の関係がこれに相当する。

4 : 右辺の要素の数が２つで、２つの語の間に親子関係は存在しないが、
意味的な中心は右側の語である場合、２つの語を１つのノードに併合し、
右側の語の情報を伝搬する。
日本語の句構造規則では、接頭語と名詞等の関係がこれに相当する。

実行例

音声認識の発話処理実行例 (第 4.1.7 節) の発話処理出力情報が入力情報の時、以下を出力する。

```
（(見出し か)
```

```
(カテゴリ <aux-sfp>)
(引数
  ((見出し です)
   (カテゴリ <auxstem-desu>)
   (引数
     ((見出し ん)
      (カテゴリ <p-jun>)
      (引数
        ((見出し な)
         (カテゴリ <aux-cop-da-rentai>)
         (引数
           ((見出し おいくら)
            (カテゴリ <wh-pro>)))
         (引数
           ((見出し それぞれ)
            (カテゴリ <adv>)))))))))))
```

5.1.9 意味解析

統語解析結果から、問題解決に必要な意味情報を効率的にかつ精度良く抽出するために構文解析ツールキット [2] を用いて実現した。

機能構成

本モジュールは、構文解析ツールキットに用意されている木構造書き換えモジュールのみから構成されている。語の係受け関係の逆転、語の挿入、語の削除等、素性構造に対する任意の変形操作のみを行ない、素性構造の分割や併合は行なわない。

機能概要

統語解析の出力である依存構造を、タスク・ドメインには依存するが談話情報には依存しない意味的な知識を利用して再順序付けするとともに、問題解決で使用する意味構造のスケルトン (省略要素及び照応対象が未決定の意味構造) である「補完前意味構造」を作成する。

なお、意味解釈の段階で曖昧性が生じ、選好されることなく複数の命題が出力されることもあるため、出力は意味表現のリストとなる。

入力情報

入力は統語解析の出力 (第 5.1.8 節) である。

出力情報

出力は言語解析の発話処理出力情報 (第 5.1.4 節) である。

参照情報

意味解釈をするための木構造書き換え規則を参照する。

実行例

統語解析の実行例 (第 5.1.8 節) の出力が入力情報の時、言語解析の発話処理実行例 (第 5.1.7 節) の発話処理出力情報を出力する。

第 6 章

対話処理機能

音声対話システムにおける対話処理機能について述べる。ここでは、対話構造認識・照応解析が実現する機能を総称して対話処理機能と呼ぶ。対話構造認識・照応解析は各々一つのモジュールをなし、システムマネージャと交信して機能を実現している。

6.1 目的

対話処理機能の目的は、他の機能に対して先行する発話から得られる情報を提供することである。

6.2 目標

ここでは、対話処理機能全体の目標について述べる。

- 領域、課題について独立であること。少なくとも可搬性があること。
- 機能を実現するのに必要とする知識、情報は、自動的に、あるいは、低コストで作成可能であること。
- 頑健であり、かつ、効率的であること。
- 翻訳処理への適用を考慮して、他の機能とは独立に機能すること。

6.3 アプローチ

上記の目標を実現するためのアプローチについて述べる。

- 表層表現と対話中での出現位置に基づく発話の分類を使用する。(発話タイプの使用)
- 対話コーパスから得られる統計情報を利用する。

- 問題解決後のみに得られる情報は利用しない。

6.4 特徴

- 3 階層 (Exchange,Move,Act) から成る対話構造モデルの使用。
- 対話構造と話題を利用した参照解決。
- 統計情報と対話構造の利用。
- 対話の制御は行なわず、モニタするのみ。

6.5 対話構造認識

6.5.1 目標

ここでは、対話構造認識の目標について述べる。

- 質問−応答のような局所的な発話間の関係を認識する。
- 発話毎に話題の候補を出力する。

6.5.2 アプローチ

上記の目標を実現するためのアプローチについて述べる。

- 表層表現、特に機能表現のパタンを働きかけ型、応答型のように分類する。(発話タイプ)
- 話題マーカ (表層表現) による話題の分類。(話題タイプ)

6.5.3 特徴

- 3 階層 (Exchange,Move,Act) から成る対話構造モデルの使用。この構造により問い返し (Clarification) を扱える。
- 対話構造に応じて発話タイプ、話題タイプを動的に解釈する。

6.5.4 機能概要

対話の進行をモニタすることにより対話の構造 (発話間の関係) と各発話のタイプを各発話時点毎に決定する。また、発話時点毎に話題候補を認識する。これらを、各発話時点での対話の状態 (発話状況) として対話経過情報に記録する。文献 [11] では、対話構造認識を行なう独立したプログラム「対話構造認識プログラム」について記述してある。これを核として、音声対話システム用にインタフェース部分を追加して本モジュールは作成してある。

6.5.5 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに対する対話構造認識を行なう。

- 単語列[1] (音声認識結果から)

なお、ユーザ発話かシステム発話を判別し、その情報を話者 ID(音声対話システムでは、ユーザなら 1、システムなら 0) として発話表現に付加して対話構造認識プログラム [11] の入力とする。

6.5.6 発話処理出力情報

受信した発話データに対する対話構造認識を行なう度に、以下で定義されている対話構造認識結果をシステムマネージャに送信する。対話構造認識プログラム [11] の出力をそのまま送信している。

```
対話構造認識結果 ::= (ACT 列 照応解析用データ)

ACT 列 ::= (話者 ID ACT+)
話者 ID ::= { 1 | 0 }  ; 1: ユーザ, 0: システム
ACT ::= (ACT 文字列 (話題 *) (ex_id mv_id act_id) (act_label*))
ACT 文字列 ::= EUC 文字列
話題 ::= EUC 文字列
ex_id ::= 整数
mv_id ::= 整数
act_id ::= 整数
act_label ::= ASCII 文字列

照応解析用データ ::= (短期話題列 長期話題列)
短期話題列 ::= (短期話題 *)
長期話題列 ::= (長期話題 *)
短期話題 ::= EUC 文字列
長期話題 ::= EUC 文字列
```

6.5.7 対話経過情報

対話開始時に以下の情報を初期化する。発話処理時に以下の情報を参照／更新する。発話取消時に以下の情報のうち最新の発話に対応する部分を削除する。

- 対話構造の状態
- 話題候補の履歴

[1]現在は分かち書きのみ利用しているが順次より詳細情報を利用可能とする。

6.5.8 参照情報

対話開始時に以下の情報を読み込む。発話処理時に以下の情報を参照する。

- 発話分割表現テーブル
- 発話タイプ－表層表現対応テーブル
- 話題マーカテーブル

6.5.9 発話処理実行例

対話開始後、付録の課題対話例の発話番号 0 ～ 3 の発話に続いて、ユーザ発話として言語解析の発話処理実行例 (第 5.1.7 節) の発話処理出力情報が発話処理入力情報の時、話者 ID を 1 として対話構造認識プログラムを起動し、以下を発話処理出力情報としてシステムマネージャに送信する。

```
((1 (それぞれおいくらなんですか
     (古都コース 万葉コース 希望 お部屋コース 奈良観光 宿泊 予約
      それぞれおいくら)
     (3 5 5)
     (<inform> <confirmation-question> <acknowledge> <yn-question>)))
 ((古都コース 万葉コース 希望 お部屋コース 奈良観光 宿泊 予約 それぞれおいくら))
)
```

6.6 照応解析

6.6.1 目標

- 言語解析から出力される意味構造中で参照表現とみなされるもののうち、問題解決に必要とするものについて参照先表現 (先行詞) の候補を決定する。

6.6.2 アプローチ

上記目標を実現するためのアプローチについて述べる。

- 基本的には、意味構造の空欄のスロットに対して、そのスロットについての制約を満たす値を文脈から満たすことにより行なう。

6.6.3 特徴

- 対話構造を利用した先行詞探索先の順序つけ
- 対話構造と話題を利用した先行詞 (候補) の決定

6.6.4 機能概要

問題解決を行なうのに必要とする情報が発話から欠落していた場合、その情報を補完する。補完は、先行する発話から抽出される情報を利用して行なわれる。

6.6.5 発話処理入力情報

発話処理入力情報として以下の発話データを両方ともシステムマネージャから受信する度に、受信した発話データに対する対話構造認識を行なう。

- 意味構造列 (言語解析結果または問題解決結果から)
- 照応解析用データ (対話構造認識結果から)

問題解決結果の意味構造は補完後意味構造であり、入力がそのまま出力の補完後意味構造となる。

6.6.6 発話処理出力情報

受信した発話データに対する照応解析を行なう度に、以下で定義されている照応解析結果をシステムマネージャに送信する。

照応解析結果 ::= (補完後意味構造 表示用補完前意味構造列 表示用補完後意味構造)

補完後意味構造は問題解決の出力でもある。

```
補完後意味構造 ::=   ((MAIN HELLO))
                   | ((MAIN REQUEST) (OBJ MAIN_OBJ 素性値))
                   | ((MAIN INFORM) (OBJ MAIN_OBJ 素性値))
                   | ((MAIN CONFIRM) (OBJ MAIN_OBJ 素性値))
                   | ((MAIN THANKS))
                   | ((MAIN YES))

MAIN_OBJ 素性値 ::=  ((MAIN RESERVE) (OBJ VERB_OBJ 素性値)))
                    | ((MAIN TELL) (OBJ VERB_OBJ 素性値)))
                    | VERB_OBJ 素性値 [2]

VERB_OBJ 素性値 ::= ((MAIN TOUR)
                     [(COURSE TOUR_COURSE 素性値)]
                     [(DINNER DINNER 素性値)] [2]
                     [(DATE_FROM DATE 素性値)]
                     [(DATE_TO DATE 素性値)]
                     [(HOW_LONG DAYS 素性値)]
                     [(PARTICIPANT PARTICIPANT 素性値)]
                     [(PHONE_NUMBER PHONE_NUMBER 素性値)] [2]
                    )
```

[2]問題解決が出力する補完後意味構造でのみ見られる。

```
TOUR_COURSE 素性値 ::= ((MAIN 文字列)
                        [(ROOM_TYPE ROOM_TYPE 素性値)]
                        [(DINNER DINNER 素性値)])

ROOM_TYPE 素性値 ::=  ROOM_TYPE 単数素性値
                    | ((MAIN &)(SET (ROOM_TYPE 単数素性値 +)))
                    | ((MAIN かつ)(SET (ROOM_TYPE 単数素性値 +))) [2]
                    | ((MAIN いずれか一つ)
                       (SET (ROOM_TYPE 単数素性値 +)))
ROOM_TYPE 単数素性値 ::= ((MAIN {文字列 | 変数})
                          [(HOW_MUCH HOW_MUCH 素性値)])
HOW_MUCH 素性値 ::= ((MAIN {変数 | 最小 [3] | 文字列})) | 文字列 [2]


DINNER 素性値 ::=  DINNER 単数素性値
                 | ((MAIN いずれか一つ)(SET (DINNER 単数素性値 +)))
DINNER 単数素性値 ::=  ((MAIN 文字列)[(COURSE DINNER_COURSE 素性値)])

DINNER_COURSE 素性値 ::=  DINNER_COURSE 単数素性値
                        |((MAIN &)(SET (DINNER_COURSE 単数素性値 +)))
                        |((MAIN かつ)
                          (SET (DINNER_COURSE 単数素性値 +))) [2]
                        |((MAIN いずれか一つ)
                          (SET (DINNER_COURSE 単数素性値 +)))
DINNER_COURSE 単数素性値 ::=  ((MAIN {文字列 | 変数})
                              [(HOW_MUCH HOW_MUCH 素性値)])


DATE 素性値 ::=  DATE 単数素性値
               | ((MAIN いずれか一つ)(SET (DATE 単数素性値 +)))
DATE 単数素性値 ::=  ((MAIN 文字列)) | 文字列 [4]

DAYS 素性値 ::= ((MAIN 文字列)) | 文字列 [4]


PARTICIPANT 素性値 ::= ((MAIN person)
                        [(NAME NAME 素性値)]
                        [(PHONE_NUMBER PHONE_NUMBER 素性値)])
NAME 素性値 ::=  ((MAIN {文字列 | 変数})) | 文字列 [4]
PHONE_NUMBER 素性値 ::= ((MAIN {文字列 | 変数})) | 文字列 [4]

表示用補完前意味構造列 ::= (表示用補完前意味構造 +)
表示用補完前意味構造 ::= ((素性名 補完前素性値)*)
表示用補完後意味構造 ::= ((素性名 補完前素性値 補完後素性値)*)
補完前素性値 ::= 文字列 | 変数
補完後素性値 ::= 文字列 | 変数
変数 ::= ?X1 | ?X2 | ...
```

[3]問題解決の入力とするため特例として導入した。

[4]言語解析が出力する補完前意味構造を入力とした照応解析の出力でのみ見られる。

6.6.7 対話経過情報

対話開始時に以下の情報を初期化する。発話処理時に以下の情報を参照／更新する。発話取消時に以下の情報の最新の発話に対応する部分を削除する。

- 対話構造の状態
- 話題候補の履歴

6.6.8 発話処理実行例

対話開始後、付録の課題対話例の発話番号 0 ～ 3 の発話に続いて、ユーザ発話として言語解析の発話処理実行例 (第 5.1.7 節) の発話処理出力情報と対話構造認識の発話処理実行例 (第 6.5.9 節) の発話処理出力情報が発話処理入力情報の時、以下を発話処理出力情報としてシステムマネージャに送信する。

```
(((MAIN REQUEST)
  (OBJ ((MAIN TELL)
        (OBJ ((MAIN TOUR)
              (COURSE ((MAIN 奈良観光)
                       (ROOM_TYPE ((MAIN &)
                                   (SET (((MAIN 万葉コース)
                                          (HOW_MUCH ((MAIN ?X4))))
                                         ((MAIN 古都コース)
                                          (HOW_MUCH ((MAIN ?X5))))))))))))))
 (((OBJ ?X1) (COURSE ?X2)(ROOM_TYPE ?X3)(HOW_MUCH ?X4))
  ((OBJ ?X2)(COURSE ?X3)(DINNER ?X4)(COURSE ?X5)(HOW_MUCH ?X6)))
 ((OBJ ?X1 宿泊パック)
  (COURSE ?X2 奈良観光)
  (HOW_MUCH ?X4 ?X4)
  (ROOM_TYPE ?X3 万葉コース)
  (ROOM_TYPE ?X3 古都コース)))
```

6.6.9 今後の課題

言語解析は 1 発話に対応した意味構造を出力している。一方、対話構造認識は 1 発話を分割して複数の ACT にし、各 ACT に対する対話構造や話題が出力している。

当モジュールでは、今のところ 1 発話分の ACT を 1 つにまとめて対話構造や話題を意味構造に対応させている。

将来、発話の意味構造・ ACT への分割の仕方が複雑になったならば、 ACT と意味構造の対応のとり方を検討する必要がある。

第 7 章

問題解決機能

音声対話システムにおける問題解決について述べる。問題解決は全体で一つのモジュールをなし、システムマネージャと交信して機能を実現している。

7.1 問題解決

7.1.1 目標

- ユーザ発話に対応する意味構造が入力された場合に適切な応答を出力する。

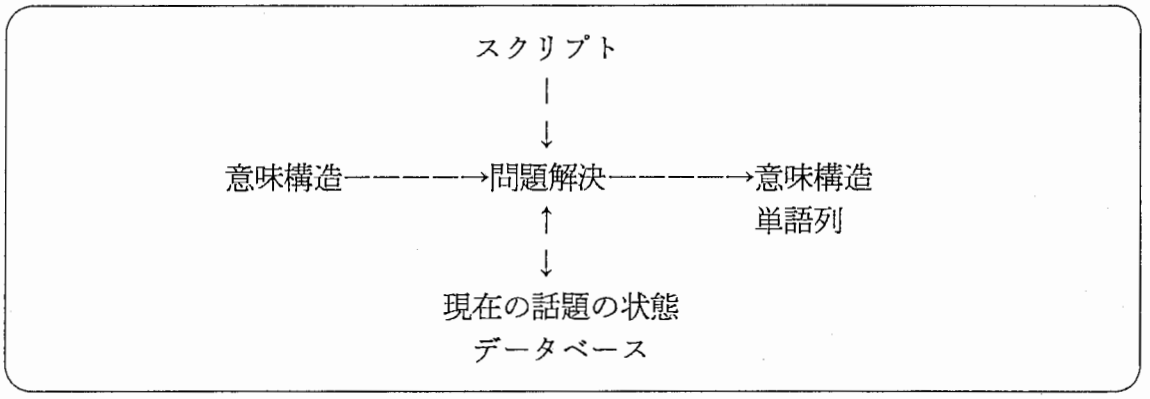
7.1.2 アプローチ

- 話題の状態遷移とその状態で必要な処理を記述したスクリプトによる処理。

7.1.3 特徴

- 特になし。

7.1.4 機能構成



7.1.5 機能概要

意味構造を入力として、現在の話題の状態とスクリプト(話題の状態遷移を記述したもの)とデータベース(ホテルの予約状況を管理したもの)に従い処理をし、必要なら現在の話題の状態やデータベースを遷移/更新し、入力に対する質問/回答/確認となる意味構造とその言語表現にあたる単語列を出力する。ホテル予約のスクリプトの場合には次のように記述してある。

話題の状態	処理	出力する意味構造	出力する意味構造の表層表現
ホテルの予約			
↓			
部屋コースの決定	部屋コースの質問	(((MAIN REQUEST)(OBJ ...)))	…部屋コースを教えて…
↓			
食事の有無の決定	食事の有無の質問	(((MAIN REQUEST)(OBJ ...)))	…食事の有無を教えて…
↓			
:	:	:	:

例えば、現在の話題の状態が「部屋コースの決定」で、ユーザ発話の意味構造が「部屋コースの質問」の答になっているなら、部屋コースに関する決定をデータベースに追加して、現在の話題の状態を「食事の有無の決定」に遷移させる。そして、「食事の有無の質問」のための出力(=「…食事の有無を教えてください。」)を行なう。

7.1.6 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに対する問題解決を行なう。

- 補完後意味構造(照応解析結果から)

7.1.7 発話処理出力情報

受信した発話データに対する問題解決を行なう度に、以下で定義されている問題解決結果をシステムマネージャに送信する。

問題解決結果 ::= (単語列 (補完後意味構造))

補完後意味構造の形式は照応解析の発話処理出力情報(第6.6.6節)を参照されたい。

単語列の中の品詞は全て<pt-of-sp>であり、単語列は補完後意味構造を言語化したものである。

言語解析と出力の形式を合わせるため、補完後意味構造は常に一つにも拘らずリストにしてある。

7.1.8 対話経過情報

対話開始時に以下の情報を初期化する。発話処理時に以下の情報を参照／更新する。発話取消時に以下の情報のうち最新の発話に対応する部分を削除する。

- 現在の話題の状態
- データベース

「現在の話題の状態」とは、スクリプトにあらかじめ記述された「話題の状態」のうちの一つで、次の発話処理開始時の「話題の状態」となる。

データベースは、空室の日付・部屋のタイプ・料金等のホテルの状況を管理したものである。

7.1.9 参照情報

- スクリプト

7.1.10 発話処理実行例

対話開始後、付録の課題対話例の発話番号0～3の発話に続いて、照応解析の発話処理実行例 (第6.6.8節) の発話処理出力情報が発話処理入力情報の時、以下を発話処理出力情報としてシステムマネージャに送信する。

```
(
((<pt-of-sp> 万葉コース)(<pt-of-sp> は)(<pt-of-sp> ２５０００円)
 (<pt-of-sp> です)(<pt-of-sp> 。)(<pt-of-sp> 古都コース)(<pt-of-sp> は)
 (<pt-of-sp> １５０００円)(<pt-of-sp> です)(<pt-of-sp> 。))

(((MAIN INFORM)
  (OBJ ((MAIN TELL)
        (OBJ ((MAIN TOUR)
              (COURSE ((MAIN 奈良観光)
                       (ROOM_TYPE ((MAIN かつ)
                                   (SET ((MAIN 古都コース)
                                         (HOW_MUCH １５０００円))
                                        ((MAIN 万葉コース)
                                         (HOW_MUCH ２５０００円)))))))))))))

)
```

第 8 章

ユーザインタフェース機能

音声対話システムにおけるユーザインタフェースとユーザインタフェース機能について述べる。ここでは、音声入力・入力音声波形表示を除いたユーザインタフェースの機能をユーザインタフェース機能と呼ぶ。ユーザインタフェース機能はモジュール群をなし、各モジュールがシステムマネージャと交信して機能を実現している。

8.1 機能概要

音声対話システムにおけるユーザインタフェースは、音声入出力とマウスによる選択と画面表示からなる。画面は、図 8.1に示すようにウィンドウにより構成される。図 8.1の左の列のウィンドウは、上から順に

- ユーザ発話の音声波形
- ユーザ発話の表層文字列
- システム発話の表層文字列
- 対話開始以降の全発話の ACT 文字列、対話構造

を表示する。右の列のウィンドウは、上から順に

- 対話開始以降の全発話の表層文字列 (図 8.1では表層文字列は表示されていない。「対話履歴」というタイトルのみ表示されている。)
- ユーザ発話の話題
- システム発話の話題
- 省略補完前のユーザ発話の意味構造
- 省略補完後のユーザ発話の意味構造
- システム発話の意味構造

- 処理モジュール稼働状況

を表示する。表示のうち、ユーザ発話データは青、システム発話データは赤で表示される。

ユーザインターフェースのうち、ユーザ音声入力とユーザ発話の音声波形の表示については音声認識で行なわれている。マウスによる対話 / 発話の選択についてはシステムマネージャで行なわれている。各々第 4.1.1 節、第 3.2 節を参照されたい。残る音声出力と音声波形以外の画面表示とをユーザインタフェース機能で行なっている。

ユーザインタフェース機能は、ユーザ発話の音声波形の表示以外の上述のウィンドウと 1 対 1 に対応する以下のモジュールからなる

モジュール	ウィンドウ
ユーザ発話表示	ユーザ発話の表層文字列
音声合成/システム発話表示	システム発話の表層文字列
対話構造表示	対話開始以降の全発話の ACT 文字列、対話構造
対話表示	対話開始以降の全発話の表層文字列
ユーザ話題表示	ユーザ発話の話題
システム話題表示	システム発話の話題
ユーザ補完前意味表示	省略補完前のユーザ発話の意味構造
ユーザ補完後意味表示	省略補完後のユーザ発話の意味構造
システム意味表示	システム発話の意味構造
経過表示	処理モジュール稼働状況

各情報を表示するウィンドウは独立して動作しており、常にシステムマネージャからのメッセージとユーザからのマウスによる画面操作イベント (スクロールなど) のイベント待ち状態にある。

これらのモジュールでは、各手続きの機能は

初期化 画面クリア。対話構造表示・対話表示では対話経過情報初期化も行なう。

発話処理 画面表示。

発話取消 対話構造表示・対話表示では対話経過情報の最新発話対応分削除と画面再表示。それ以外ではなし。

となっている。

8.2 ユーザ発話表示

8.2.1 機能概要

言語解析が出力する単語列を受けとり、その表層文字列をユーザ発話として表示する。

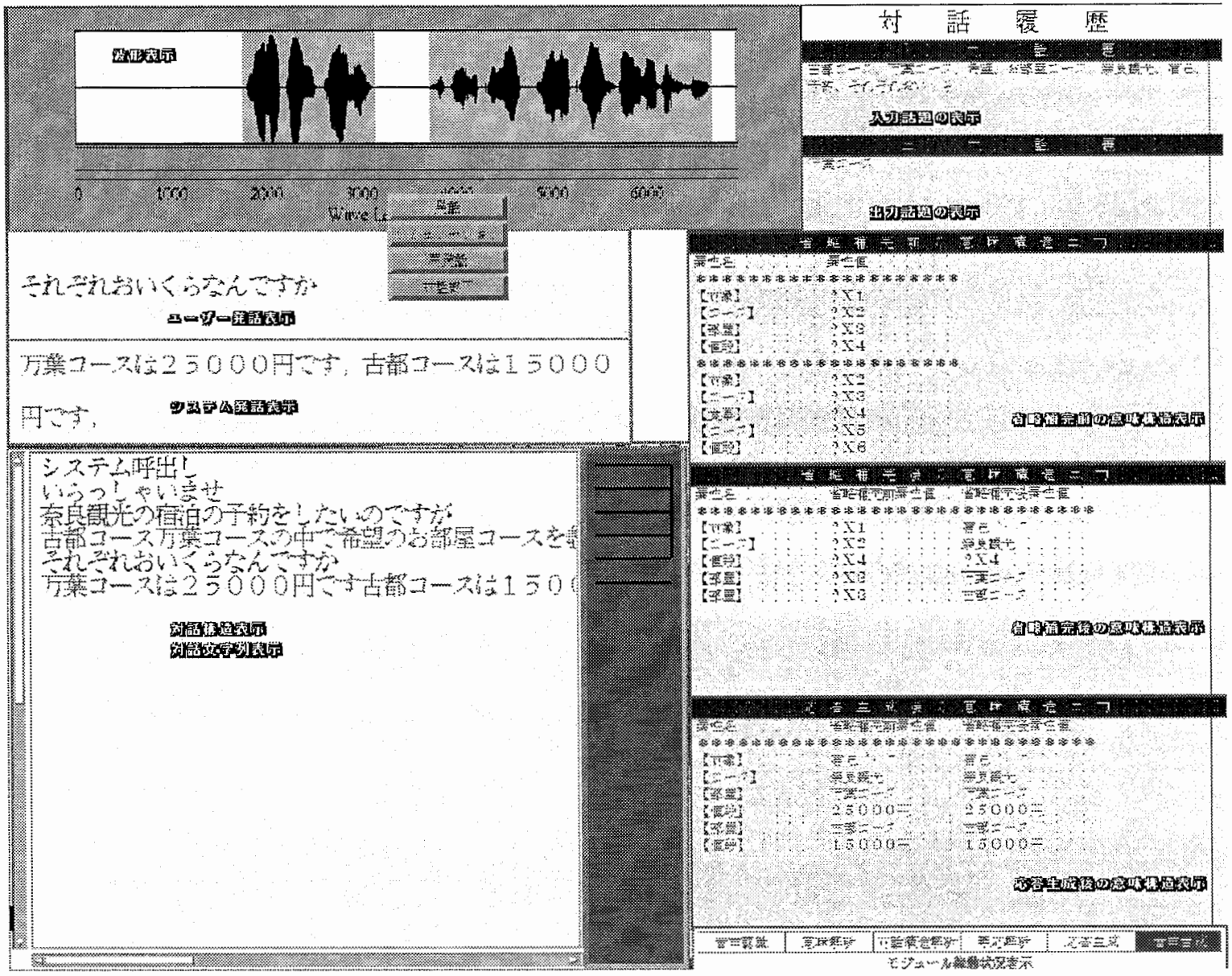

対話履歴
波形表示
入力話題の表示
出力話題の表示
それぞれおいくらなんですか
ユーザー発話表示
万葉コースは２５０００円です，古都コースは１５０００
円です，
システム発話表示
システム呼出し
いらっしゃいませ
奈良観光の宿泊の予約をしたいのですが
古都コース万葉コースの中で希望のお部屋コースを
それぞれおいくらなんですか
万葉コースは２５０００円です古都コースは１５０
対話構造表示
対話文字列表示
省略補完前の意味構造表示
省略補完後の意味構造表示
応答生成後の意味構造表示

図 8.1: 画面構成

8.2.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに基づく表示を行なう。

- 単語列 (言語解析結果から)

8.2.3 発話処理出力情報

なし。

8.2.4 対話経過情報

なし。

8.3 音声合成/システム発話表示

8.3.1 機能概要

問題解決が出力する単語列を受けとり、その表層文字列に対して音声合成を行ない、また、システム発話として表示する。

8.3.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに基づく音声合成と表示を行なう。

- 単語列 (問題解決結果から)

8.3.3 発話処理出力情報

なし。

8.3.4 対話経過情報

なし。

8.3.5 音声合成環境

音声合成のために、当モジュールからプログラム JTalk を起動している。

JTalk は、NTT-IT 製日本語音声合成ボード『しゃべりん坊 HG』を使用した簡易日本語発声プログラムであり、SPARCstation 上で動作する。JTalk では、しゃべりん坊の拡張 BOX 版ライブラリを使用しているため、拡張 BOX に挿入された『しゃべりん坊 HG』が RS-232C ポート経由でワークステーションに接続されている事を前提とする。

プログラム名と起動オプション

JTalk　(Japanese TALK)

-d device-name　: しゃべりん坊拡張 BOX が接続されている RS-232C ポートのデバイス名を指定する。
省略した場合の default 値は "/dev/ttyb"。

-s default-sex　: 発声文字列中に音声発声者の性別の指定がない場合の値を以下の通り指定する。
m : 男性音声
f : 女性音声
省略した場合の default 値は'f' となる。
なお発声文字列中に音声発声者の性別指定がある場合は、その値が優先される。

8.4 対話構造表示

8.4.1 機能概要

ユーザ及びシステムの発話に対して対話構造認識が出力する ACT 列を受けとり、「ACT 対応情報」に、Exchange に基づく ACT の対応関係を生成しておき、ACT 毎の ACT 文字列と ACT の対応関係を表示する。

8.4.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、ACT 対応情報と受信した発話データに基づく表示を行なう。

- ACT 列 (対話構造認識結果から)

8.4.3 発話処理出力情報

なし。

8.4.4 対話経過情報

対話開始時に以下の情報を初期化する。発話処理時に以下の情報を参照／更新する。発話取消時に以下の情報のうち最新の発話に対応する部分を削除する。

- ACT 対応情報

ACT 対応情報には、ACT 毎に、話者 ID,ACT 文字列,ex_id,act_id, 同じ Exchange に属する ACT の act_id が記録してある。

8.5 対話表示

8.5.1 機能概要

ユーザ発話に対して言語解析が出力する単語列およびシステム発話に対して問題解決が出力する単語列を受けとり、「対話情報」にユーザ発話 (言語解析出力時) かシステム発話 (問題解決出力時) かの分類とともに記録しておき、対話開始以降の表層文字列を表示する。

通常はタイトルのみ表示されており、タイトルをクリックするとウィンドウが開き上述の表示をする。

8.5.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、対話情報と受信した発話データに基づく表示を行なう。

- 単語列 (言語解析結果または問題解決結果から)

8.5.3 発話処理出力情報

なし。

8.5.4 対話経過情報

対話開始時に以下の情報を初期化する。発話処理時に以下の情報を参照／更新する。発話取消時に以下の情報のうち最新の発話に対応する部分を削除する。

- 対話情報

対話情報には、発話毎のユーザ発話かシステム発話かの分類と表層文字列が記録してある。

8.6 ユーザ話題表示

8.6.1 機能概要

ユーザ発話に対して対話構造認識が出力する ACT 列を受けとり、その話題を表示する。話題がない時は「話題なし」を表示する。

8.6.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに基づく表示を行なう。

- ACT 列 (対話構造認識結果から)

8.6.3 発話処理出力情報

なし。

8.6.4 対話経過情報

なし。

8.7 システム話題表示

8.7.1 機能概要

システム発話に対して対話構造認識が出力するACT列を受けとり、その話題を表示する。

8.7.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに基づく表示を行なう。

- ACT列(対話構造認識結果から)

8.7.3 発話処理出力情報

なし。

8.7.4 対話経過情報

なし。

8.8 ユーザ補完前意味表示

8.8.1 機能概要

ユーザ発話に対して照応解析が出力する表示用補完前意味構造列を受けとり、表示する。

8.8.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに基づく表示を行なう。

- 表示用補完前意味構造列(照応解析結果から)

8.8.3 発話処理出力情報

なし。

8.8.4 対話経過情報

なし。

8.9 ユーザ補完後意味表示

8.9.1 機能概要

ユーザ発話に対して照応解析が出力する表示用補完後意味構造を受けとり、表示する。

8.9.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに基づく表示を行なう。

- 表示用補完後意味構造 (照応解析結果から)

8.9.3 発話処理出力情報

なし。

8.9.4 対話経過情報

なし。

8.10 システム意味表示

8.10.1 機能概要

a システム発話に対して照応解析が出力する表示用補完後意味構造を受けとり、表示する。

8.10.2 発話処理入力情報

発話処理入力情報として以下の発話データをシステムマネージャから受信する度に、受信した発話データに基づく表示を行なう。

- 表示用補完後意味構造 (照応解析結果から)

8.10.3 発話処理出力情報

なし。

8.10.4 対話経過情報

なし。

8.11 経過表示

8.11.1 機能概要

発話処理時の処理モジュールの起動／終了毎にシステムマネージャが送信する開始通知／終了通知[1]を受信し、どの処理モジュールが現在稼働しているのかを表示する。また、各処理モジュールの最新出力を表示する。

8.11.2 初期化メッセージ

初期化をシステムマネージャから受信すると、全ての処理モジュールの表示を待機中にする。

8.11.3 開始通知メッセージ

開始通知をシステムマネージャから受信すると、開始通知の<module_id>に対応するモジュールの表示を稼働中にする。

8.11.4 終了通知メッセージ

終了通知をシステムマネージャから受信すると、終了通知の<module_id>に対応するモジュールの表示を待機中にする。

8.11.5 処理モジュール出力表示

画面の処理モジュールの稼働状況の表示がクリックされると、ウィンドウを開いてその処理モジュールの最新の発話処理出力情報を表示する。ただし、音声合成は除く。

[1] メッセージの仕様 (第 3.3 節) を参照されたい。

参考文献

[1] 巖寺 俊哲, 竹澤 寿幸, 田代 敏久, 加藤 直人, 石崎 雅人, 森元 逞 : “ポーズ単位に基づく音声言語統合処理と発話状況管理の統合 – 音声対話システムの試作 –,” 1996年電子情報通信学会総合大会講演論文集 SD-4-3, pp331 – 332 (1996).

[2] 田代 敏久 : “音声言語処理のための構文解析ツールキット ユーザーズマニュアル,” *ATR Technical Report*, TR-IT-0142 (1995).

[3] 永井明人, 嵯峨山茂樹, 北研二: “HMM-LR 連続音声認識における音素環境依存型パーザの実現アルゴリズム”, 信学技報, SP91-23, pp. 41-48, (1991-6).

[4] 門前聖康, H.Singer, 松永昭一: “フレーム同期型 SSS-LR による連続音声認識”, *ATR Technical Report*, TR-IT-0080 (1994-4).

[5] Shimizu, Monzen, Singer, Matsunaga: “Time-synchronous continuous speech recognizer driven by a context-free grammar”, *Proc. of ICASSP95*, pp. 584-587, (1995).

[6] 竹沢寿幸, 田代敏久, 衛藤純司: “部分木を単位とする音声認識用日本語文法”, *ATR Technical Report*, TR-IT-0110 (1995-04).

[7] 竹沢寿幸, 田代敏久, 森元逞: “自然発話の言語現象と音声認識用日本語文法”, 情報処理学会研究報告, 95-SLP-6-5, pp. 27-34 (1995-05).

[8] 竹沢寿幸, 森元逞: “部分木を単位とする構文解析と前終端記号のバイグラムを利用した連続音声認識”, 信学技報, SP95-89, (1995).

[9] 北研二, 森元逞, 嵯峨山茂樹: “到達可能照合機構を用いた 2 段階 LR パーザと音声認識への応用”, 電子情報通信学会 NLC92-12 (1992-07).

[10] 伊藤克亘: “連続音声認識システムに関する研究”, (1993-03).

[11] 巖寺 俊哲, 石崎 雅人, 森元 逞 : “対話のインタラクション構造と話題の認識,” 情報処理学会 自然言語研究会 94-NL-104, pp119 – 126 (1994).

付録 A

音声対話システムで対象とする課題・領域

A.1 対象とする課題

音声対話システムで対象とする課題は、ホテルの予約関連とする。すなわち、ホテルの予約、予約の変更、キャンセル、ホテルの紹介・問い合わせである。(現在は予約のみが可能である。) システムは、これらの課題を人間と対話することにより遂行、達成する。

これらの課題は、システムのアクションと次のように対応する。

課題	システムのアクション
ホテルの予約	データベースへのデータの追加
予約の変更	データベースのデータの更新
キャンセル	データベースのデータの削除
問い合わせ	データベースのデータの表示

A.2 対象とする領域

音声対話システムで対象とする領域とは、システムの扱える対象世界のことである。この領域は、奈良周辺のホテルとする。

この領域について上記の課題を遂行するためには、以下のような知識を必要とする。

- ホテルに関する知識
- 奈良市内観光に関する知識

A.3 課題対話例

表 A.1: ★★ ホテルの予約 対話例 ★★

発話番号	話者	発話
0	ユーザ：	(システム呼出)
1	システム：	いらっしゃいませ。
2	ユーザ：	奈良観光の宿泊の予約をしたいのですが
3	システム：	古都コース、万葉コースの中で希望のお部屋コースを教えてください。
4	ユーザ：	それぞれおいくらなんですか
5	システム：	万葉コースは２５０００円です。古都コースは１５０００円です。
6	ユーザ：	安い方をお願いします
7	システム：	有、無の中で希望の食事の有無を教えてください。
8	ユーザ：	食事はつけて下さい
9	システム：	松コース、竹コース、梅コースの中で希望のお食事コースを教えてください。
10	ユーザ：	それぞれおいくらなんですか
11	システム：	梅コースは１０００円です。竹コースは３０００円です。松コースは５０００円です。
12	ユーザ：	安い方をお願いします
13	システム：	三月十五日、三月十六日、三月十七日、三月十八日の中で希望の旅行開始日を教えてください。
14	ユーザ：	三月十五日から十七日までの二泊をお願いします
15	システム：	お名前を教えてください。
16	ユーザ：	清水太郎です
17	システム：	お電話番号を教えてください。
18	ユーザ：	電話番号は、零六三二八六七五四です
19	システム：	確認いたします。奈良観光の古都コースで、食事有、梅コース。ご宿泊は三月十五日から三月十七日までの二泊。清水太郎様、零六三二八六七五四でよろしいでしょうか。
20	ユーザ：	はい
21	システム：	ご利用ありがとうございました。