音声言語情報処理 22－4
（1998．7．24）

多重文脈を用いた逐次的な発話理解

中野 幹生 宮崎 昇 平沢 純一 堂坂 浩二 川端 豪
NTT基礎研究所

実時間音声対話システムにおいて，ユーザの発話を逐次的に理解する方法を提案する．対話システムが必要な時にいつでも適切に応答するためには，その言語理解部は，音声認識部が単語候補を出力する度に，逐次的に理解を進めて行く必要がある．しかし，逐次理解では，発話の意味的な区切りがあらかじめ分らないので，どの時点でシステムの理解状態を変更して良いのかが不明であるという問題がある．本稿で提案する方法は，部分解析の手法を用いて，理解の単位となる句を発見しながら同時に逐次的に理解を進めていくが，理解の状態の曖昧性を扱うために複数の文脈を保持しておく．そして最も優先度の高い文脈に基づいて応答を行う．

Incremental Utterance Understanding Based on a Multi-Context Model

Mikio Nakano, Noboru Miyazaki, Jun-ichi Hirasawa, Kohji Dohsaka, and Takeshi Kawabata
NTT Basic Research Laboratories

This paper proposes a method for incremental understanding of user utterances in a real-time spoken dialog system. For a spoken dialog system to respond appropriately whenever a response is required, its language understanding module must incrementally understand word hypotheses that are incrementally sent from the speech recognition module. In incremental understanding, however, the system cannot determine when it should change its understanding state. The proposed method incrementally analyzes with the partial parsing technique and uses a multi-context model to deal with ambiguities of understanding states.

1 はじめに

我々の目標はユーザフレンドリなマンマシンインターフェースの実現であり，特に，音声対話システムにおけるユーザの快適性の向上について研究を行っている．そのためには，ユーザの話し方に制限を加えなくてもシステムが理解することや，システムの内部状態を適切にユーザに開示することが必要であると考えている．これらのことの実現のため，ポーズなどの位置を気にせず自由なタイミングで話されるユーザの発話を実時間で理解し，必要に応じて遅れなく応答を行える音声対話システムの実現を目指している[7),14)]．本稿では，このような実時間音声対話システムにおける言語理解モジュールについて述べる．

音声対話システムの言語理解モジュールは，音声認識の結果を受け取り，その内容を解析して，システムの理解状態の変更を行い，応答生成モジュールに渡すという処理を行うものとする．対話システムが上記の要件を満たすためには，言語理解モジュールは少なくとも次のような条件を満たさなくてはならないと考えられる．(1) 遅れなく理解するため，音声認識モジュールが逐次的に出力する単語候補を受け取り，逐次的に理解を進めていかなくてはならない．(2) ユーザがいつ話すのを中断しても応答できるように，また，必要な時には割り込んで話すことができるように，いつでもその時点までの対話に関するシステムの理解の状態を出力できなくてはならない．(3) ポーズや文法に関して制限のないユーザの発話を理解できなくてはならない．

従来の音声発話理解の研究では，あらかじめポーズやイントネーションを用いてユーザ発話の発話区間を検出し，これを理解の単位としていた．そして，この区間毎に音声認識を行い，その後構文解析によってその意味表現を求めて理解を行っていた．この方法では，発話区間が終了するまで理解が進まないの

で，発話区間中には応答ができない．また，発話の意味的な区切りと発話区間の区切りが一致しない場合に，構文解析に失敗してしまう．

しかしながら，後に示すように，逐次理解では，理解の単位があらかじめ決まっていないので，どの時点で理解の状態を変更して良いのかわからないという問題がある．構文解析を用いずキーワードの抽出のみで理解を行えば，この問題は回避できるが，正確な理解に失敗する可能性が高くなる．

本稿では，上記の問題を解決するため，あらかじめ理解の単位を決めずに，逐次的に理解していく方法を提案する．本方法は，その時点までの構文解析の中間結果や，それに基づく解釈を複数保持しながら処理を進める．これらの情報は一般に文脈と呼ばれるものに相当する．構文意味規則の適用の仕方は複数有り得るので，可能な文脈は一つではない．そこで，複数の文脈を保持することにより，この曖昧性に対処する．応答が必要になったときには，優先度の最も高い文脈に基づいて応答する．なお，本方法では，部分解析 (partial parsing)[10] の手法を用いて，キーフレーズの抽出を行いながら理解していくので，誤認識があっても，理解不能に陥ることなく対話を進めることができる．

2 従来の発話理解法の問題点

従来の音声対話システムの研究では，大部分が次のような構成を用いている[2),15)]．まず，ポーズなどを用いてユーザ発話の発話区間 (以後音声区間と呼ぶ) を検出し，音声区間終了後に直前の区間の音声認識を行う．次にその結果を言語解析して意味内容を求め，プランなどを用いて談話処理を行い，過去の文脈と結びつけて解釈する．その結果に従って応答発話のプランニングを行い，音声で応答する．この一連の処理を繰り返すことによって対話をすすめる．

本稿では，ユーザ意図の理解や対話の状況に関するシステムの内部状態を理解状態と呼ぶことにする．また，理解状態を変えることを理解を進めるという[1]．あるフレーズの意味に基づいて理解状態を変えるとき，そのフレーズが理解の単位となる．言い換えれば，理解の単位はひとつの発話行為を行うとも言える．これらの用語を用いて言うと，従来の発話理解法は，次のような特徴をもつ．

(I) 音声区間が理解の単位となると仮定している．

[1]理解状態がフレームで表されるとするなら，理解を進めることは，スロット値を変化させることに相当する．

えーと金曜日のー2時 (ポーズ) 30分からです
第1会議室 (ポーズ) ...

図 1: 音声区間が理解の単位と合わない例

(II) 音声区間終了後に理解を進める．

この他に，多くのシステムではひとつの音声区間が言語学上の文に相当すると仮定しているが[15)]，これについては，本稿では議論しない．本稿では，理解は節以下の細かい単位で行うことを仮定する[5),14)]．

上記の特徴は，次に述べるような点で，ユーザに負担を強いる．

(a) (I) より，各音声区間は，理解の単位となり得るように，構文意味的にまとまった句になっていなくてはならない．したがって，対話のタスクで重要な働きをするフレーズの途中にポーズが入ると，システムはうまく理解できない[2]．例えば，会議室予約対話で，図1のようなユーザ発話があったとする．「2時」と「30分から」がポーズで分断されているため構文的に結びつけることができない．また，「30分からです第1会議室」と2つの句をポーズなしで続けて発話しているので，システムは理解できない．

(b) (I) より，ある音声区間全体の構文解析に失敗すると，その音声区間の発話内容を理解状態に反映させることができない．したがって，1音声区間の音声認識結果中に一部でも誤認識があれば，構文解析に失敗してシステムが理解できなくなることがある．したがって，極めて高い認識率が必要となるが，現在の音声認識性能を考慮すると，高い精度で実時間認識を行わせるのは困難である．

(c) (II) より，音声区間が終了するまで，理解を進めることができない．したがって，ユーザが話している途中にシステムがあいづちをうったり，割り込んで応答したりできない．したがって，ユーザは自分が話している間，システムが理解しているかどうかが分らないため不安になる[7),9)]．また，音声区間が終了し，その音声認識結果が得られてから理解を進めるので，音声認識に少しでも遅れがあると，ユーザの音声区間

[2]日本語の対話の音声区間の5%ないし6%は構文的にひとまとまりにならないという報告がある[13),17)]．

終了後, 即座に応答することができない. システムの応答が遅れると, ユーザは不安になって, 再び話しはじめてしまうことがあり, 対話がスムーズに行かなくなる. 人間同士の対話では, 聞き手の応答は, 即座に行われ, 時には音声区間終了前に行われる場合もある[12] ことから, 即座の応答が重要であると考えられる.

音声認識の言語モデルの研究では, (I) の仮定を用いず, 各音声区間が構文意味的にまとまりのある句になっていなくても認識できる方法が検討されている[1), 17)]. しかしながら, これらの認識結果をどう用いて言語理解を行うかは述べられていない.

問題点 (b) の音声認識の誤りに対処するため, Allen ら[2)] は, confusion matrix を用いて認識誤りを修復する方法を用いている. 竹林ら[16)] の方法では各音声区間からキーワードをスポッティングし, その候補から LR 構文解析を用いて有効なキーワード列を抽出する. また, 河原ら[8)] はキーワードスポッティングの計算量の多さを解決するために, フレーズスポッティングによる発話理解方式を提案している. これらの方法によって, 音声認識の誤りにより構文解析が失敗して理解が進まない場合を減少させることができる. しかし, これらの方法では, (II) の方法を用いており, 音声区間終了まで結果が得られない.

音声区間の終了以前に構文解析を始めることにより, 構文解析結果が得られるまでの時間を短縮する研究が行われているが[6)], 音声区間終了まで構文解析結果が得られないので, 理解を進めることはできない.

3 構文解析による逐次理解の問題点

本稿では, 上記の (I), (II) の方法を用いない理解法を考える. (II) を用いないということは, 音声区間が終わる前に構文解析と理解を始めるということを意味する. さらに, (I) の仮定を用いないことから, 音声区間の終了が, 理解の単位の終了ではないので, 音声区間内であっても理解状態を変更する必要がある. 本稿では, 単語単位で逐次的に構文解析を進め, 必要なときに理解状態を変更する方法を考える. これを単語逐次理解, または単に逐次理解と呼ぶことにする.

(I), (II) の方法を用いず, タスクに関係する重要なキーワードを実時間で抽出しながら, スロットフィリングなどを行いつつ理解を進めていく方法が考えられる. しかし, この方法では, 構文規則を用いた理解を行うことができないので, 発話理解に失敗する可能性が強くなる.

しかしながら, (I), (II) の方法を用いず, かつ, 構文解析を用いようとすると, 次のような問題が生じる. 逐次理解を行うことと, (I) の仮定を用いないことから, どこが理解の単位の終りかはわからない. 構文解析は, フレーズの終わりがわかってはじめて結果を出すことができるので, どこが理解の単位の終わりかわからないと, いつまでも理解を進めることができない. 例えば, 会議室予約の対話で, 予約の開始時間に関して, ユーザが「2 時」と発話したとする. そこで理解の単位が終わると分っていれば, 2 時から予約したいと思っていると理解を進めればよい. しかし, そこは本当は理解の単位の終わりではなく, そのあとに「30 分」と続くかもしれない. すると実はユーザの意図は, 2 時 30 分から予約することだったということになる. さらにここが理解の単位の終わりでない可能性もあり, いつまでも理解を進めることができない. このように, 構文解析と逐次理解を同時に実現することには困難がある.

4 多重文脈を用いた逐次理解

本稿では, 前節で述べた問題の解決のため, キーワードスポッティングではなく, 構文解析を行いながらも, 逐次的に理解を進める方法を提案する. 本方法は, 構文解析の中間結果や理解の状態を複数同時に保持しながら理解を進めていく. これらの情報はひとまとめにして文脈と呼ぶことができる.

4.1 文脈による理解状態の表現

本方法では, 文脈をスタック, 理解状態, 優先度の 3 つのデータで表現する.

- スタックは, shift-reduce パーザで用いられるプッシュダウンスタックである. これは音声認識モジュールが逐次的に出力する単語候補を shift-reduce 法で解析した時にできるものである.
- 理解状態は, ユーザの意図を理解した結果や, 対話の履歴をまとめて保持しておくものであり, 逐次的に変更が可能なものである. 会議室予約の対話を例にとると, 予約したい日, 開始時間, 終了時間, 会議室, 人数などの情報や, ユーザに確認を求めたか, ユーザは確認したかといった情報が書き込まれる. 理解状態をどのようなデータ構造で表現するかは本稿では議論しないが, 複雑なタスクでなければ, フレームを用いてスロットフィリングを行なうことにより遂行できると考えられる[3)].

理解状態の変更は，スタックの状態に応じておこなう．解析用文法では，開始記号として，有効発話 (significant utterance) というカテゴリを用いる．有効発話は理解状態を変更する力を持つ．解析用文法は，どのような発話が有効発話であるかを規定するものである．これは，書き言葉の文法が文とは何かを規定しているのと同じである．スタックの先頭が有効発話であった場合に，その内容に応じて理解状態を変化させ，スタックを空にする．この時，スタックに残っていた句はすべて無視されることになるが，これは理解状態を変化させないような句を無視するということであり，ロバスト性に寄与する．

この方法は部分解析[10] を行いながら逐次的に理解をすると考えることができる．また，解析用文法の構文規則は，言語学的に正しい構造を受け付ける規則だけではなく，領域に依存した規則[4] を用いることができる．

- 優先度は，理解の状態の尤もらしさを表す数である．これは以下で述べる多重文脈の利用の際に用いる．

4.2 多重文脈の利用

1つの談話，すなわち，ユーザ発話の連続に対して複数の文脈を作ることができる．それは，shift-reduce パーザにおいて，(1) reduce できるときに，shift するか reduce するか，(shift-reduce conflict) (2) reduce の際に複数の規則を用いることができるとき，どの規則を用いるか (reduce-reduce conflict) という2つの曖昧性があるからである．各々の文脈で，発見される有効発話，理解状態が異なる．したがって，各々の時点で規則の適用の仕方を一意に決定してしまうと，その決定が誤っていた場合には，間違った理解をしてしまう．これは，前節で述べた，構文解析を用いた逐次理解の問題点に相当する．

この問題の解決のため，本稿では，複数の文脈を用いて理解を進めていく方法を提案する．本方法では，音声認識モジュールから単語が入力される度に，次のような操作を行う．

1. その単語の統語意味情報を求め，その時点で存在するすべての文脈のスタックに shift する．
2. これらの文脈に対して，規則の適用を試みる．規則が適用可能な場合，その結果を用いて，あらたな文脈を作る．有効発話が発見された場合，その内容を用いて，その文脈の理解状態を変更する．

```
A  会議室設定  → 会議室
B  時間設定    → 時間
C  時間        → 時間 分
```

図 2: 規則の例

3. この規則の適用と理解状態の変更の処理を繰り返し，すべての可能な対文脈を作成したのち，優先度の高いもののみを残す．このとき，スタックと理解状態の両方の内容が同じ文脈が複数あるときには，もっとも優先度の高いものだけを残す．

本方法では，対話の途中で優先度の低い文脈を捨ててしまうので，すべての可能な場合を調べつくす方法に比べれば信頼性は落ちるが，対話では問い返しや確認によって理解の誤りを解消することができるため，ある程度の曖昧性が扱えれば問題はないと考えられる．

優先度を決定する一般的な方法は現時点では得られていない．しかしながら，実験システム構築の経験で，現状の音声認識率が100% には遠いことと，後の対話で誤解を解消できることから，比較的単純なヒューリスティックスを用いても問題はないことが判明している．

5 例

一例として，簡単な会議室予約の対話を考える．このタスクでは，ユーザがシステムに予約したい会議室や開始時間を言うことにより予約を行なう．図2にこの例で用いる解析用規則を示す．説明を簡単にするため，単純な規則のみを用いる．ゴシック体は有効発話であることを意味する．「会議室設定」はカテゴリが有効発話で，そのタイプが会議室設定であることの略記である．規則に付属する細かな条件は省略する．また文脈は最大3個まで保持できるとする．

図3に，「第1会議室」，「2時」，「30分」という単語列が入ってきたときの文脈の変化を示したものである．四角で囲んであるものが理解状態で，その上にスタックを示した．スタックは上に行くほど深いとする．

まず，「第1会議室」が入力されると，文脈が1つ作られ，そのスタックには，「会議室 (第1)」が shift される．これは，カテゴリが「会議室」である句で，その意味が「第1会議室」であることを単純化して

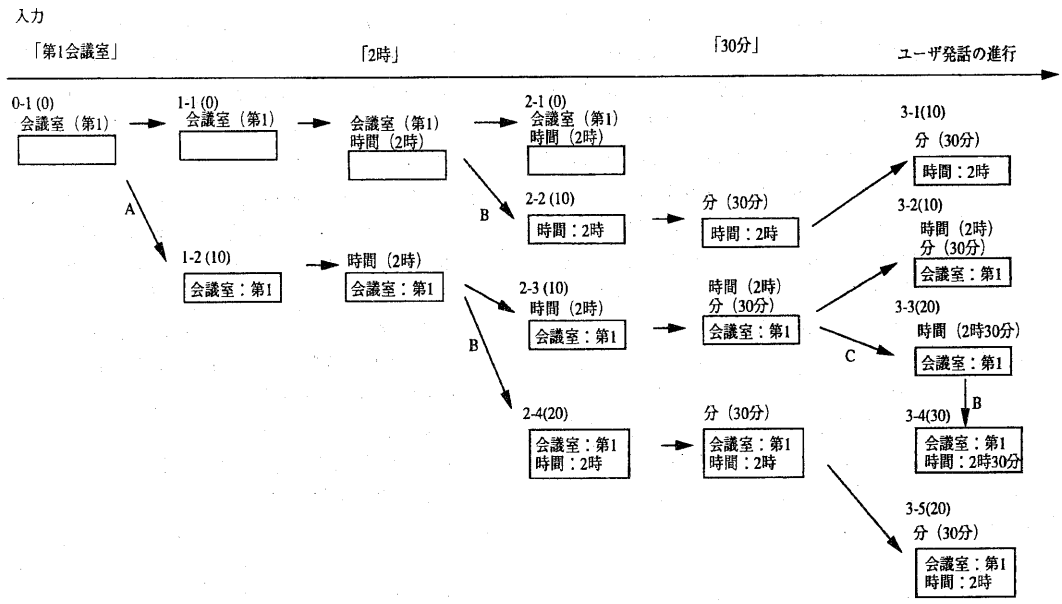

図 3: 文脈の変化

記述したものである．この文脈を 0-1 とする．'-' の左側の数字は何番目の単語まで処理したかを表す．また，括弧の中の数字は優先度を表わす．この例では，単純に，規則が用いられる度に優先度が 10 増えるとする．

これに規則を適用することにより，1-1, 1-2 の二つの文脈ができる．矢印に A と書いてあるのは，規則 A を適用して作られたことを示す．1-2 では，有効発話「会議室設定」が発見されたため，理解状態を書き換える．理解状態が単純なフレームで表されているとすると，会議室名スロットに「第 1 会議室」が入れられる．

次に「2 時」が入ってきた場合を考える．文脈 1-1, 1-2 のスタックに，「時間 (2 時)」が加えられる．これらに規則 B を適用し，1-1 から 2-2 が，1-2 から 2-4 ができる．これらの文脈の理解状態は，予約する時間が 2 時になる．一度に保持できる文脈の最大値が 3 であるので，優先度が低い 2-1 は以後用いられない．

「30 分」が入ってきた後も同様の処理を繰り返す．2-3 の「時間 (2 時)」と「分 (30 分)」の連続に規則 C を適用して 3-3 ができ，さらにそれに規則 B を適用して 3-4 ができる．3-3 の理解状態は，予約する会議室が第 1 会議室で，時間が 2 時 30 分である．この時点で，もっとも優先度の高いものは 3-4 である．「2 時」が入って来た時点では，時間が 2 時の状態がもっとも優先度が高かったが，30 分が入って来たあとでは，2 時 30 分の状態がもっとも優先度が高くなり，これに基づいて応答が行われる．

このように，逐次的に理解状態が変化し，その時点でもっとも優先度の高い理解状態を得ることができ，実時間で応答することが可能になる．

また，音声認識の誤りにより，「第 1 会議室，30 分，2 時」という単語列が入力された場合を考えると，「30 分」を前後の単語と結び付ける規則がないので，「30 分」は無視され，「第 1 会議室, 2 時」という単語列が入ってきた場合と同じ理解状態が得られる．このように，規則に合わない入力を無視することができ，ロバストな理解を行うことができる．これは，入力が完全でなくても，対話を進行させるため，システムの持っている知識に照らし合わせて理解を試みることを意味する．こうして得られた理解の結果は，実際のユーザの意図とは異なる可能性もあるが，確認対話の際に，誤解を解消することが可能である．

6 実現

上記の方法は，TV 番組の録画予約を受け付ける音声対話実験システム[11] で用いている．音声認識部は ISTAR Protocol (Incremental Structure Trans-

mitter And Receiver Protocol)[7] によって逐次的に単語候補を出力する．言語処理部では，語彙処理プロセス，言語理解プロセス，応答生成プロセス等のプロセスが並行して動作する．語彙処理プロセスは，音声認識部から入力された単語候補の構文意味情報を調べる．そして，文節毎の構文意味構造を構築し，これを言語理解プロセスに送る．単語毎ではなく文節毎に送るのは，処理量の削減のためである．言語理解プロセスは，本稿で提案した方法を用いて理解を行う．応答生成プロセスは，優先度の最も高い文脈の理解状態に基づき応答発話を生成する．現在のバージョンでは，ユーザがポーズを置いたときに応答する．このとき，それまでに理解した状態に基づいて即座に応答を行うので，音声認識の遅れによる応答の遅れは生じない．また，これ以外にも，ユーザの発話中に，理解状態に基づいてあいづちをうつ．

本実験システムにより，定性的にではあるが，本方法の有効性を確かめることができた．

7 おわりに

本稿では，多重文脈モデルと部分解析により，音声発話の単語逐次理解を行う方法を提案した．本方法では，入力の解析結果および理解状態を複数保持することにより，構文意味の曖昧性のみならず，発話の意味的な区切りの曖昧性も扱うことができ，逐次的に理解を進めることが可能である．また理解規則と合わない発話を無視することができ，ロバストに理解を行うことができる．

謝辞

日頃ご指導いただく石井健一郎情報科学研究部長，討論して頂いた対話理解研究グループの皆様，北陸先端科学技術大学院大学 島津 明 教授に感謝いたします．

参考文献

1) 秋葉, 伊藤. 休止を単位とした発話の文法. 情報処理学会研究報告 SLP-10-17, pp. 93–98, 1996.

2) J. F. Allen, B. W. Miller, E. K. Ringger, and T. Sikorski. A robust system for natural spoken dialogue. In *Proc. 34th ACL*, pp. 62–70, 1996.

3) D. Bobrow, R. Kaplan, M. Kay, D. Norman, H. Thompson, and T. Winograd. GUS, a frame driven dialog system. *Artif. Intel.*, 8:155–173, 1977.

4) J. S. Brown and R. R. Burton. Multiple representations of knowledge for tutorial reasoning. In D. B. Bobrow and A. Collins, editors, *Representation and Understanding*. Academic Press, 1975.

5) 堂坂, 島津. タスク指向型対話における漸次的発話生成モデル. 情報処理学会論文誌, 37(12):2190–2200, 1996.

6) G. Görz, M. Kesseler, J. Spilker, and H. Weber. Research on architectures for integrated speech/language systems in verbmobil. In *Proc. 16th COLING*, pp. 484–489, 1996.

7) 平沢, 中野, 川端. うなずき・相槌による音声対話システムの理解状態開示. 言語処理学会第 4 回年次大会発表論文集, pp. 182–185, 1998.

8) 河原, 北岡, 堂下. A* 探索に基づいたフレーズスポッティングによる頑健な音声理解. 電子情報通信学会論文誌, J79-D-II(7):1187–1194, 1996.

9) M. Kawamori, A. Shimazu, and K. Kogure. Roles in interjectory utterances in spoken discourse. In *Proc. ICSLP-94*, pp. 955–958, 1994.

10) D. D. McDonald. An efficient chart-based algorithm for partial-parsing of unrestricted texts. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pp. 193–200, 1992.

11) 中野, 堂坂, 宮崎, 平沢, 田本, 川森, 杉山, 川端. TV 番組の録画予約を受け付ける実時間音声対話システム. 情報処理学会研究報告 SLP-22, 1998.

12) 小坂. あいづちを中心とした会話音声の呼応関係の分析. 電子情報通信学会技術研究報告, SP87-107, 1987.

13) M. Seligman, J. Hosaka, and H. Singer. "Pause units" and analysis of spontaneous Japanese dialogues: Preliminary studies. In E. Maier, M. Mast, and S. LuperFoy, editors, *Dialogue Processing in Spoken Language Systems*, Lecture Notes in Artificial Intelligence, pp. 100–112. Springer-Verlag, 1997.

14) 島津, 中野, 堂坂, 川端. 細かな発話単位の処理に基づく対話システム. 人工知能学会全国大会論文集, 1998.

15) R. W. Smith and D. R. Hipp. *Spoken Natural Language Dialog Systems*. Oxford University Press, 1994.

16) 竹林. 音声自由対話システム TOSBURG II— ユーザ中心のマルチモーダルインタフェースの実現に向けて—. 電子情報通信学会論文誌, J77-D-II:1417–1428, 1994.

17) 竹沢, 森元. 部分木に基づく構文規則と前終端記号バイグラムを併用する対話音声認識手法. 電子情報通信学会論文誌, J79-D-II(12):2078–2085, 1996.