多重対話文脈を用いたロバストな実時間音声対話理解

中野 幹生 宮崎 昇 平沢 純一 堂坂 浩二 川端 豪
NTT 基礎研究所

1 はじめに

人間と自然な対話を行うシステムは, 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う必要がある[4), 8)]. 本研究の目標は, 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を構築することである.

実時間音声対話システムの言語理解モジュールは, 次のような要件を満たさなくてはならないと考えられる. まず, 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. そして, それまでの対話の文脈に照らし合わせ, その時点までの対話に関するシステムの理解の状態を変更しなくてはならない. 本稿ではこのような機構を実現する方法を考える.

今までの音声対話理解の研究で典型的なシステムは次のようなものであった. まず, ポーズなどを用いてユーザ発話の発話区間を検出し, 発話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, プランライブラリなどを用いて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[9)]. これをAIアプローチと呼ぶことにする. このアプローチには次のような問題がある. まず, 多くのシステムでは, 文単位の発声を仮定しているが, 自然な話し言葉では, 文の切れ目が明確にはわからない. このため, どの時点でシステムの状態を変更して良いのかわからない. また, ユーザがポーズをおくまで応答することができない. さらに, 現状の音声認識技術では, 常に正しい単語候補が得られるとは限らないので, 認識誤りに対処しなくてはならない. 認識誤りや誤解の解消などを扱ったプラン認識の研究はあるものの[7)], 複雑な処理を要し, 実時間対話理解に用いることは困難である.

AIアプローチとは対照的に, 音声理解からのアプローチでは, キーワードスポッティングなどを用いて, 自由なタイミングの発話からタスクに関係する重要な語のみを取出し, スロットフィリングなどを行う方法が用いられている[10)]. 小規模なタスクにおいては, この方法はロバスト性も実時間性も高く, 有効な方法といえる. しかし, キーワードスポッティングだけでは, 機能語の情報を用いることができないため, 発話理解に失敗する可能性が強くなる. 河原ら[5)]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. この方法では, 発話終了まで結果が得られないため, 即時の応答が困難である.

本稿では, 上記の問題を解決するため, 構文意味規則を用いながらも実時間で理解を進め, 必要な場合に応答ができる方法を提案する. 本方法は, 対話文脈と呼ぶデータを複数保持して理解を行うので, 多重対話文脈法と呼ぶことにする.

対話文脈は, 対話を逐次的に解析し, それに基づいて理解を行った結果を保持するデータ構造であり, 各時点で適切な応答を行うのに必要な情報を蓄えている. 構文意味規則の適用の仕方は複数有り得るので, 可能な対話文脈は一つではない. そこで, 複数の対話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, 優先度の最も高い対話文脈を用いる. 複数の対話文脈のうち, 優先度の低いものを捨て, 一定数のみを保持しておくことによって, 計算量の増大を防ぐことができる.

本方法では, 構文意味規則を用いることができるので, キーワードスポッティングよりも細かい理解を行うことができる. 対話文脈は, 逐次的な理解の結果を保持しているため, 常に最新の理解結果を参照することができ, 実時間応答が可能である. 各々の対話文脈をつくる過程では, 入力を, 対話のタスクに関係ある発話の連続ととらえるのではなく, 関係ある発話の間に, 無関係な部分, すなわち, システムが理解を行なわない部分があっても良いとする. これにより, 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 対話文脈

本節では, 対話文脈を説明する. 対話文脈は, 逐次的な解析を行い, それに基づいて理解を行った結果を表現するデータ構造である. 1つの対話文脈は, 不活性弧列, 理解状態, 優先度の3つのデータからなる.

不活性弧列は, ユーザの発話とシステムの発話との履

Robust Real-Time Dialog Understanding using Multiple Dialog Contexts
Mikio Nakano, Noboru Miyazaki, Jun-ichi Hirasawa, Kohji Dohsaka, and Takeshi Kawabata (NTT Basic Research Labs.)
E-mail: nakano@atom.brl.ntt.co.jp

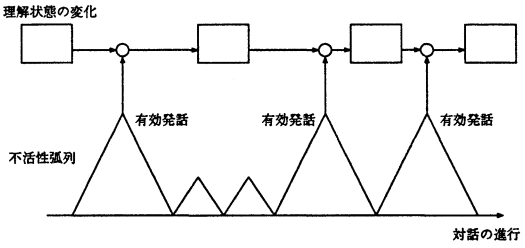

図 1: 対話文脈

歴に, 文法を適用して, well-formed substring を作ったものである. 対話の最初から現時点までが, well-formed substring の列, すなわち, チャート解析[6] の不活性弧の列として表されている. 不活性弧は部分木と言い換えることもできる. 不活性弧列は, チャートの最初の節点から最後の節点までを不活性エッジの列で結んだパスに相当する. 不活性弧列をつくる操作を発話解析と呼ぶ. 解析用の文法は, 構文意味規則に加え, ユーザ発話とシステム発話を結びつける談話的な規則も含む. これらの規則を解析規則と呼ぶ.

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[1].

理解状態の変更は, 発話解析の結果を用いて行う. 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. 逐次的な発話解析の結果, 有効発話が見つかる度に, その内容に応じて理解状態を変化させる. これにより, 逐次的に理解を行うことができる. 図1に, 発話解析の結果により理解状態が変化する様子を示す. この図の場合, 1, 4, 5 番目の不活性弧は有効発話で, 理解状態を変化させる. 2, 3 番目の不活性弧は有効発話ではないので, 理解状態に影響を及ぼさない.

伝統的なモデルにそって言えば, 発話解析が構文意味解析に, 有効発話が文に, 理解結果の操作が談話処理に当たると言える. ただし, 有効発話と言語学上の文が同じである必要はなく, 対話のドメインやタスクに応じて定義すればよい. 人間同士の対話が文よりも小さい単位を用いて行われることに着目した研究が行われているが[3], 有効発話も文より小さい単位にした方が良い. また, 理解用文法の構文規則も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[2] などを用いることができる.

優先度は, どの対話文脈が尤もらしいかを表す数である. 最も優先度が高い対話文脈の理解結果に基づいて, 応答が行われる.

3 多重対話文脈法

1 つの対話に対して複数の対話文脈を作ることができる. それは, 解析規則の適用の仕方には, (1) 規則が適用できるときに, 実際に規則を適用するかどうか, (2) 複数の規則が適用できるときに, どの規則を適用するか, という任意性があるからである. 不活性弧列は, チャートの最初の節点から最後の節点までを不活性弧の連続で結んだパスなので, 可能なパスの数だけ, 対話文脈を作ることができる. これは, 書き言葉の文の解析で, 同じ範囲の解析結果として, 複数の構文木が得られるのと同じである.

各々の対話文脈で, 発話の切目や, 理解状態が変化するタイミングが異なる. 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.

この問題の解決のため, 本稿では, 複数の対話文脈を用いて理解を進めていく方法を提案する. これを, 多重対話文脈法 (multiple dialog context method) と呼ぶ.

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. まず, その単語の統語意味情報を求め, 語彙弧を作り, それまでに存在した対話文脈候補の各々の不活性弧列に加える. また発話生成モジュールから, システム発話の情報が送られてきたときも, 同様に不活性弧列に加える操作を行う. これらの対話文脈に対して, 解析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, あらたな対話文脈を作る. 有効発話が発見された場合, その内容を用いて, その対話文脈の理解状態を変更する. この規則の適用と理解状態の変更の処理を繰り返し, すべての可能な対話文脈を作成したのち, 優先度の高いもののみを残す.

本方法では, 対話の途中で優先度の低い対話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.

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

単語入力後:

0-1	0	会議室 (第一)

規則適用後:

1-1	[0-1]	0	会議室 (第一)
1-2	[0-1, A]	10	会議室設定 (第一)

図 3: 対話文脈の変化 (1)

4 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. アスタリスクのついているカテゴリは有効発話である. 「* 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. また対話文脈は最大 3 個まで保持できるとする.

図 3, 図 4, 図 5は, 「第一会議室」, 「2 時」, 「30 分」という単語列が入ってきたときの, 対話文脈の変化を示す. まず, 「第一会議室」が入力されると, 対話文脈が 1 つ作られ, その不活性弧列には, 「会議室 (第一)」がはいる. これは, カテゴリが「会議室」である句で, その意味が「第一会議室」であることを単純化して記述したものである. この対話文脈を 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). これに規則を適用することにより, 1-1, 1-2 の二つの対話文脈ができる. [0-1] は, この対話文脈が 0-1 から作られたことを示す. また, [0-1, A] は, 0-1 に規則 A を適用して作られたことを示す. その右の数字は優先度を示す. 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. 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「第一会議室」が入れられる.

次に「2 時」が入ってきた場合を考える. 対話文脈 1-1, 1-2 の不活性弧列に, 「時間 (2 時)」が加えられ, 図 4のようになる. これらに規則 B を適用し, 1-1 から 2-2 が, 1-2 から 2-3 ができる. 一度に保持できる対話文脈の最大値が 3 であるとすると, 1-1 に規則を適用しなかっ

単語入力後:

1-1	0	会議室 (第一), 時間 (2 時)
1-2	10	会議室設定 (第一), 時間 (2 時)

規則適用後:

2-1	[1-2]	10	会議室設定 (第一), 時間 (2 時)
2-2	[1-1, B]	10	会議室 (第一), 時間設定 (2 時)
2-3	[1-2, B]	20	会議室設定 (第一), 時間設定 (2 時)

図 4: 対話文脈の変化 (2)

単語入力後:

2-1	10	会議室設定 (第一), 時間 (2 時), 分 (30 分)
2-2	10	会議室 (第一), 時間設定 (2 時), 分 (30 分)
2-3	20	会議室設定 (第一), 時間設定 (2 時), 分 (30 分)

規則適用後:

3-1	[2-3]	20	会議室設定 (第一), 時間設定 (2 時), 分 (30 分)
3-2	[2-1, C]	20	会議室設定 (第一), 時間 (2 時 30 分)
3-3	[3-2, B]	30	会議室設定 (第一), 時間設定 (2 時 30 分)

図 5: 対話文脈の変化 (3)

たものは, 優先度が低いので捨てられる. 2-2, 2-3 での理解状態は, 予約する時間が 2 時になる.

「30 分」が入ってきた後も同様の処理を繰り返す. 2-1 の「時間 (2 時)」と「分 (30 分)」の連続に規則 C を適用して 3-2 ができ, さらにそれに規則 B を適用して 3-3 ができる. 3-3 の理解状態は, 予約する会議室が第一会議室で, 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 3-3 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, 逐次的に理解の状態が変化し, 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, 「第一会議室, 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

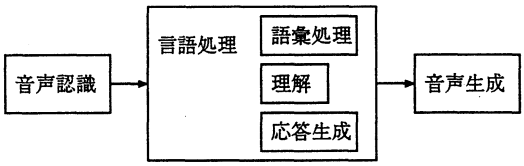

図 6: 実験システムの構成

ユーザ	すいませーん
システム	はい
ユーザ	予約したいんですけども
システム	はい
ユーザ	えっとー第一会議室
システム	はい
ユーザ	金曜日のーえー午後2時から
システム	第一会議室, 金曜日, 午後2時からということでよろしいでしょうか
ユーザ	はい
システム	かしこまりましたー

図 7: 対話例

分」は無視され,「第一会議室, 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るので, ロバストな理解を行うことができる. この理解の結果は, 実際のユーザの意図とは異なる可能性もあるが, 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により, 対話を進行させることができる. 確認対話の際に, 誤解を解消することが可能である.

5 実現

上記の方法を用いて, 実験システムを作成した (図 6). 本システムは, 音声認識モジュール, 言語処理モジュール, 音声生成モジュールからなる. 音声認識モジュールは入力音声を認識し実時間で単語候補を出力し, 言語処理モジュールにおくる. 認識用の文法は, 各音声区間が文節の連続からなると規定している. 言語処理モジュールでは, 語彙処理プロセス, 理解プロセス, 応答生成プロセスの3つのプロセスが並行して動作する. 語彙処理プロセスは, 入力された単語候補の構文意味情報をもつ構造を作る. 本実験システムでは, 単語ごとではなく文節ごとに作る. 理解プロセスは, 本稿で提案した多重対話文脈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対話文脈の理解状態に基づき応答を生成する. 図 7に実験システムとの簡単な対話の例を示す.

6 おわりに

本稿では, ロバストな実時間音声対話理解を行うための多重対話文脈法を提案した. 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, 実時間で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 川森雅仁氏, 田本真詞氏, 杉山 聡氏, 北陸先端科学技術大学院大学 島津 明 教授, 実験システムの作成にご協力頂いた NTT アドバンステクノロジ (株) 久保田 哲也氏に感謝いたします.

参 考 文 献

1) 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.

2) 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.

3) 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.

4) 平沢, 川端. わかってうなずくコンピュータの試作. 情報処理学会研究報告 SLP19-20, pp. 131–138, 1997.

5) 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.

6) M. Kay. Algorithm schemata and data structures in syntactic processing. Technical Report CSL-80-12, Xerox PARC, 1980.

7) S. W. McRoy and G. Hirst. The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4):435–478, 1995.

8) 島津. コンピュータと人間の会話：現状と課題. 情報処理学会誌, 39(3), 1998.

9) R. W. Smith and D. R. Hipp. *Spoken Natural Language Dilaog Systems*. Oxford University Proress, 1994.

10) Y. Takebayashi, H. Tsuboi, H. Kanazawa, Y. Sadamoto, H. Hashimoto, and H. Sinichi. A real-time speech dialogue system using spontaneous speech understanding. *IEICE Transactions on Information and Systems*, E76-D(11):112–119, 1993.