自然言語処理 112－15
（1996．3．15）

Case-Based Text Planning における文脈情報の取り扱い

谷口 実　　上原 邦昭
神戸大学 工学部 情報知能工学科

従来のプランに基づく対話管理機構の問題点を解消するために，新たな対話管理機構の枠組として Case-Based Text Planning を提唱している．Case-Based Text Planning は，対話管理機構からユーザモデルや推論のための知識を排除しているため，発話状況に応じた柔軟な応答が困難であるという問題があった．我々は，Case-Based Text Planning における問題点をインプリメントに基づいて明らかにし，ユーザが発話の直前にとる行動に関する情報 (文脈情報) を取り扱い，上記の問題点を解消する手法を示す．また，文脈情報の利用によってもユーザの発話意図を特定できない場合には，ユーザの発話を観測しながらプランを逐次修正するリアクティブプランニングの枠組を導入している．

Dealing with Contextual Information in Case-Based Text Planning

Minoru Taniguchi　　Kuniaki Uehara
Department of Computer and Systems Engineering, Kobe University

In text planning, many rule-based system have been researched but the most of them have some problems such as increasing computational costs and complexity of knowledge representation. We have already proposed the new framework of case-based text planning to solve them. Case-based text planning retrieves many analogous cases to adapt for the current situation and thus it is difficult to select the appropriate one among them. In this paper, we present the approach to solve the problem by dealing with contextual information which shows user's previous behavior. We describe that our framework can retrieve the best case out of many cases and construct more cooperative dialogue. Furthermore, we apply reactive planning mechanism while performing the plan so as to modify the retrieved plan.

1 はじめに

事例ベース推論 (Case-Based Reasoning [1][2] : 以下 CBR) は，新しく発生した問題の解決を過去に獲得した問題解決の事例から求める推論方式である．CBR は事例を過去の問題解決の過程を縮約した一種のマクロとして定義し，類似性に基づく検索を採用しているので，推論や検索の中間過程を節約でき，正確な検索条件がなくても柔軟に事例を検索できる．これらの利点を踏まえ，我々は，自然言語対話 (Text Planning) という膨大な多様性をもった対象を取り扱うために，抽象化されたルール集合を利用する従来の手法 (Classical Text Planning) でなく，事例に基づく対話管理手法 (Case-Based Text Planning) という，新たな問題解決の方向性を提案する．本手法は，UNIX コマンド利用に関する対話を取り扱う対話管理システム Assist-R[3] において実現されている．

CBR 一般に生じる問題として，検索される事例によって問題の解決方法が異なるため，現在の問題に対して最適な解決法を生成できる事例候補を選択しなければならないという問題 (最適事例の選択問題)，類似事例が複数存在する場合は，より修正の容易な事例を選択しなければならないという問題 (修正事例の選択問題) などがある．また，対話に特有の問題として，ユーザの発話が曖昧であった場合，応答プランを獲得するための情報が不足し，適切なプランを生成できないという問題 (発話の曖昧性問題) がある．

本稿では，Case-Based Text Planning における問題点を明らかにし，解決のアプローチとして，対話における文脈情報を考慮した対話管理の枠組を提唱し，その有効性を検討する．

2 事例に基づく対話管理手法における事例の取り扱い

本手法で取り扱う対話事例は，ユーザとシステムの発話表現を，図 1 に示すような述語論理形式による内部表現で表し，システム内の事例ベースに蓄えられている．システムの発話はユーザの発話に起因しているために，ユーザの初期発話を事例のインデックスと定義し，システムの発話を応答プランと定義する．

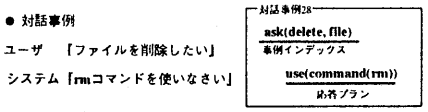

図 1: 事例表現

Case-Based Text Planning における応答プランの生成は，検索，修正の枠組によって実現している．まず，ユーザからの発話文の入力に対し，過去の対話を格納した事例ベースを類似度に基づいて検索し，最も類似度の高い事例を検索する．全く一致したプランが検索されれば，その事例が応答プランとなる．類似事例が検索された場合はシステム内部の知識に基づいて修正され，ユーザの発話に応答するプランとして生成される．

事例の検索では，現在のユーザの発話を表す内部表現と事例のインデックスの類似度を算出し最大の類似度を持つ事例を選択する．類似度検索は，システム内部に持つ概念階層木上の概念距離を計算して求めるようにしている．

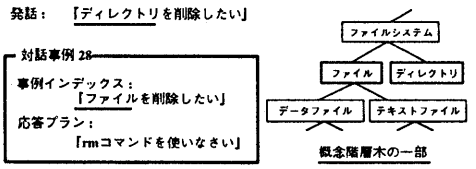

図 2: 新たな事例の生成

図 2 はユーザが「ディレクトリを削除したい」と発話した場合で，発話の内部表現と事例ベース内の個々の事例インデックスとの類似度を計り，最も類似した事例として「ファイルの削除」に関する事例 28 が獲得されている．

検索された事例を現在の状況に一致させるために，UNIX コマンドに関する知識の関係をネットワークで表現したコマンドネットワークを用いて事例の修正を行なっている．コマンドネットワークは，発話の概念表現と UNIX コマンドに関する領域知識を関連づけたもので，概念表現を妥当なコマンドに修正するために用意されている．

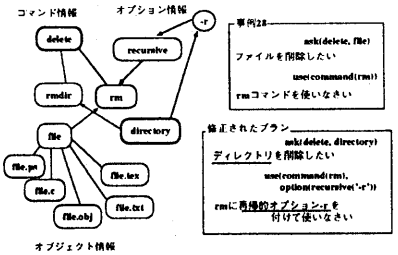

図 3: コマンドネットワークによる事例の修正

図 3 は，類似検索された「ファイルの削除」に関する事例の修正の様子を表している．「ファイル」と「ディレクトリ」表現の差分から検索事例のインデックスを修正し，コマンドネットワーク上で修正されたインデックスをたどる．コマンドネットワークは，「rm コマンド」を使うためには「再帰的オプション -r 」の付加が必要な知識であることを示している．したがって，「ディレクトリの削除」に対する新たな応答プランが生成される．

3 事例に基づく対話管理に伴う問題点

本研究では，事例ベースとして 108 種類の UNIX コマンドに関する 406 の対話例を収集し，Assist-R の開発を行なった．しかし，現状では，典型的な対話を除き，適切なシステムの応答が生成できない場合が多く見られた．そこで我々は，ユーザの発話に対する適切な応答が失敗する状況を解析し，以下に示すような 3 つの問題点を明らかにした．

最適事例の選択問題

対話の選択は発話表現に基づいて行なわれるために，検索された事例のうちユーザの発話が全く同じであるものについては，いずれを選択すれば良いか決定することができない．例えば，あるコマンドの質問に対して詳細な応答をするのか簡単な応答をするかは，ユーザの持つ知識レベルによって応答を選択する方が好ましい．また，ユーザが目標に向かって何らかの行動をしている場合，ユーザの発話が同じでも，状況に応じた事例の選択が必要である．

修正事例の選択問題

ユーザの発話が一致しない場合には，類似に基づく検索によって候補を決定する必要があるが，類似度が同じ事例が複数存在する場合には，一意に決定することができない．効率の良い事例の利用のためには，事例候補の中で，より修正の簡単な事例を選択する必要がある．

発話の曖昧性問題

ユーザの発話表現そのものが曖昧で意図が特定できない場合，事例の候補を選択できない．例えば，ユーザが「ファイル」と発話した場合，「データファイル」や「テキストファイル」に関する事例が候補として求められても，いずれを選択するか決定できない．

対話問題の最大の特徴として，ユーザの発話時の状況や知識レベルによって問題解決に利用できる事例の最適性は様々に変化する事があげられる．上記の問題の原因は，ユーザの発話時の情報が発話そのものの表現に現れることは少なく，発話に至る文脈的な情報を考慮しなくては解決することができない点にある．この問題は，柔軟なヒューマンインターフェースの構築を目的とする研究のなかでも Augmented Reality の問題 [4] として論じられており，特に対話においては，発話情報と発話の文脈情報をいかに抽出して統合し，実際のユーザの意図に近似させるかが問題となっている．

4 文脈情報の取り扱い

4.1 コマンド履歴による文脈情報

本手法に伴う問題点を解消するために，我々は，発話時の文脈情報を考慮するアプローチを採用している．これは，本研究で取り扱う対話を UNIX のコマンドに関する利用支援に限定しているので，ユーザが発話以前に実行したコマンドの履歴を文脈情報として利用することができるためである．これは，ユーザがシステムに対して質問を行なう場合，全くの初心者の場合を除き，ユーザが意図を満たすために，既に何らかの行動とっている事が期待できるからである．コマンドの履歴が表す文脈情報としては，次の 5 つの情報が考えられる．

1. 知識レベル：ユーザが発話内容について持っている知識に関する情報

2. 発話補足：ユーザの曖昧な発話内容を補足するための情報
3. 発話状況：ユーザが操作している状況に関する情報
4. 手続きのマクロ化：ユーザがある意図にしたがって行なった一連の手続きの情報
5. ユーザの話題: ユーザが応答を望んでいる話題を示すための情報

4.2 文脈情報を用いた候補事例の選択

文脈情報を示すインデックスとして，各事例には過去の対話時に伴ったユーザのコマンド履歴を付加している．これをサブインデックスと呼ぶ．メインインデックスである発話表現によって獲得された候補事例の中から，サブインデックスと現在のユーザのコマンド履歴を比較することによって，現在の文脈にあった最も適切な事例が選択される．

事例の選択には，コマンドネットワークと呼ぶ領域知識を利用している．コマンドネットワークは，ユーザの操作要求(ACTION)，操作対象(OBJECT)，操作状況(OPTION)を，UNIX コマンドと関連づけたネットワークである．コマンドネットワーク上の各コマンドは，操作対象や状況に応じて，表1に示すようないくつかの関連コマンドからなるカテゴリを形成している．

表 1: コマンドカテゴリの例

カテゴリ分類	UNIX コマンド
プロセス関連	nohup, sleep, jobs, kill gcore, ps, pagesize, nice, bg, fg
ファイル操作関連	find, wc, sum, chgrp, chmod umask, chown, cp, ln, link unlink, mv, install, rm, split
ファイル変換関連	jistoeuc, jistosj, euctojis euctosj, sjtojis, sjtoeuc nkf, jtops, dos2unix, unix2dos
ディレクトリ関連	cd, ls, mkdir, pwd, rmdir popd, pushd
プリンタ関連	lp, cancel, lpstat, lpq, lpr, lprm, lptest, script

サブインデックスとユーザのコマンド履歴とのマッチングは，以下のルールにしたがっている．

- 発話に直接関わるような情報は，UNIX コマンドの制約から比較的直前のものに限られる．したがって，発話の直前の3つのコマンド 履歴を用いてサブインデックスを構成する．中でも，発話の1つ直前のコマンド履歴を最も重要なサブインデックスとする．
- コマンド自体のマッチングには，コマンドの利用対象や仕様によって，分けられたコマンドカテゴリを用いる．コマンド履歴上に現れるコマンドとサブインデックスのコマンドが同一のカテゴリに含まれればコマンドの記述がマッチしていると見なす．
- コマンド履歴の，オプション，オブジェクトのそれぞれ独立に事例のサブインデックスとマッチングをおこなう．

4.3 各文脈情報に基づく候補事例の選択例

発話表現のみの検索では不可能であった候補事例の選択が，文脈情報を用いることにより可能となる．以下では，発話表現によって類似度選択された最類似事例の候補に対して，文脈情報を用いて適切な事例の選択が可能となる例を示す．

4.3.1 知識レベルを考慮した候補選択

候補事例中に含まれる文脈情報が，ユーザの知識レベルを表している場合の選択例を示す．この事例選択は，ユーザの知識レベルは発話表現そのものに現れる事が少ないために，発話表現のみに基づく手法では獲得できなかったものである．

発話例: `# cp file.txt newfile.txt`
「ファイル名を変更したい」

検索事例 92: #
「ファイル名を変更したい」
「`mv` コマンドを使いなさい，書式は
"`mv` 変更元ファイル名 変更ファイル名 "です」

検索事例 103: `# cp file.txt ..`
「ファイル名を変更したい」
「`mv` コマンドでファイル名の変更ができます，
書式は `cp` と同じです」

検索事例 117: `# mv file.txt ..`
「ファイル名を変更したい」
「ファイル名の変更にも `mv` コマンドが使えます」

92. コマンドについての知識がない
103. ファイル操作のコマンドについて何らかの知識を持っている
117. ファイルの移動コマンドとして `mv` を知っている

この例では，文脈情報からユーザがファイル操作に関して何らかの思考錯誤ができる程度の知識レベルを持っていることがわかる．したがって，事例 103 を適切な候補として選択することが可能である．

4.3.2 発話状況を考慮した候補選択

候補事例中に含まれる文脈情報が，ユーザの発話状況を表している場合の選択例を示す．この事例選択も，発話表現中にはユーザの発話時の状況が現れず，事前の発話に至る文脈を考慮しなければ選択できなかったものである．

```
発話例:  # emacs file.txt
          「日本語を入力したい」

検索事例 231:  # tgif
          「日本語を入力したい」
          「kinput2 をバックグラウンドで実行しなさい」

検索事例 201: # emacs
          「日本語を入力したい」
          「Contrl-\ で日本語入力モードになります」

検索事例 126: # kterm
          「日本語を入力したい」
          「uum を実行しなさい」

231. tgif 上で日本語を入力したい
201. emacs 上での日本語の入力の仕方を知りたい
126. ターミナル上で日本語の入力がしたい
```

この例の場合は，発話状況として emacs を実行しているために，emacs 上での日本語入力を意図していると考えられる．したがって，コマンド履歴として emacs という発話状況を持っている検索事例 201 が候補として選択される．

4.3.3 発話の曖昧性の解消

ユーザの発話自体が曖昧で，発話表現のみでは適切な事例候補を選択できなかった場合，文脈情報に含まれる曖昧な発話の補足情報を用いれば，適切な事例を選択できるようになる．

```
発話例:  # jlatex file.tex
          「ファイルを PS 形式に変換したい」

検索事例 112：# vi file.txt
          「テキストファイルを PS 形式に変換したい」
          「 a2ps コマンドを使いなさい」

検索事例 94：# jlatex file.tex
          「 dvi ファイルを PS 形式に変換したい」
          「 dvi2ps コマンドを使いなさい」

112. ファイルはテキストファイルである
 94. ファイルは TEX 形式のファイルである
```

ユーザの発話に現れる「ファイル」の表現は曖昧で，ファイルの種別を特定できないが，ファイルの種別が応答プランに大きく関わる場合には，曖昧な表現を解消する必要がある．上記の例では，文脈情報を考慮すれば，「DVI 形式のファイル」であると特定した事例 94 を候補事例として選択することができる．

4.3.4 手続きのマクロ化を考慮した事例選択

UNIX コマンドの利用において，ユーザは要求を充足するために，複数のコマンドを組み合わせて利用することが多い．一連の手続きに伴う応答は，発話の意図を特定しにくい点，応答プランが複雑である点など，事例の選択・生成上，問題が多く取り扱えないものであった．

発話の意図を特定しにくい候補事例の選択問題に対しては，文脈情報がユーザの一連の手続きを表している場合，手続きをマクロ化した情報として取り扱えば解決することができる．これは，ユーザがある目的にために何らかの手続きを行なっており，手続き上のどの段階にあるかを示す情報となる．したがって，手続き状態に応じてユーザの手順の不足や誤解をある程度明確にした事例の選択が可能となる．

```
発話例:  # rlogin liszt
          # xterm &
   「リモートのプログラムを実行したい」

検索事例 81: #
   「リモートのプログラムを実行したい」
   「rsh コマンドで実行できます」

検索事例 242: # rlogin kojiro
          # kterm
   「リモートのプログラムを実行したい」
   「setenv で DISPLAY 変数をローカルのホストに設定しなさい」

検索事例 260:  # rlogin kojiro
          # setenv DISPLAY jones:0.0
          # kterm &
   「リモートのプログラムを実行したい」
   「ローカルホストで xhost + を実行しなさい」

 81. 特に情報は存在しない
```

242. DISPLAY のセットを行なっていない
260. xhost + を実行していない

上記の例では，一連の手続きにしたがって行動している場合は，適切な応答が可能となる例を示している．しかしながら，手続き状態を示した文脈情報を持った事例は，何らかの手順を前提としているため，かなり限定された範囲でしか利用できないという欠点がある．

4.3.5 話題に応じた修正事例の選択

発話表現に基づいて検索された事例の候補は，コマンドネットワークによって修正され，現在の状況に応じた応答プランが生成される．この際，検索された候補の中から修正が容易な事例を選択できれば，修正コストを軽減することが可能となる．

言い替えると，「同じ話題に関する事例を修正する方が容易である」という前提のもとに，文脈情報としてユーザが発話する話題を特定できれば，修正に有効な事例を選択することができる．

発話例: # ls -l
「ディレクトリを削除したい」

検索事例 11: # ls
「ディレクトリを移動したい」
「 mv コマンドを使いなさい」

検索事例 24: # ps aux
「プロセスを削除したい」
「 kill コマンドに -KILL オプションを付けて実行しなさい」

11. ディレクトリに関する操作
24. プロセスに関する操作

この例では，発話表現の類似度検索によって事例 11, 24 が最類似候補として選択される．ユーザの発話表現は似ているが，事例 11 の場合，発話の内容はディレクトリに関する応答を示しており，事例 24 の場合はプロセスに関する応答を示している．現在のコマンド履歴から，ユーザはディレクトリに関する操作を行なっていることが分かるので，ユーザの関心はディレクトリに関する話題にあると見なすことができる．類似事例の再利用の観点から考えれば，事例 24 の例を修正して応答文を生成するよりも，現在の状況にあった事例 11 を用いて修正する方が自然である．したがって，事例 11 を修正し，「rmdir コマンドを使いなさい」という応答文が生成できる．

4.3.6 実際の選択例

今まで述べてきたように，様々な文脈情報を持った事例を事例ベースに蓄えれば，文脈情報の種類を考慮することなく，ユーザの発話時のコマンド履歴のマッチによって最適な候補を選択することができる．

発話例: # more file.txt
「漢字コードを jis にしたい」

検索事例 37: # cat file.txt
「ファイルの漢字コードを euc にしたい」
「 nkf コマンドに -e オプションを付けて
実行しなさい」
検索事例 70: # nkf file.txt > file.euc
「ファイルの漢字コードを euc にしたい」
「 EUC コードにするには -e オプションが必要です」
検索事例 202: # mule file.txt
「ファイルの漢字コードを euc にしたい」
「 Ctrl-x Ctrl-k f とキーを押した後に
euc-japan RET と入力しなさい」
検索事例 45: # kterm &
kill %1
「ターミナルの漢字コードを euc にしたい」
「 kterm に -km euc オプションをつけて起動しなさい」

37. 「漢字コード」とは「ファイルの漢字コード」の変換を意図している (発話補足情 報)
70. nkf の使い方を知っている (知識レベル)
202. mule 上での漢字コードの変換を意図している (発話状況)
45. kterm 起動時の設定を意図している (手続きのマクロ化)

発話の類似検索によって選ばれた候補事例の中から，コマンド履歴のマッチにより事例 70 が選択され，修正を受けて「nkf コマンドに -j オプションをつけて実行しなさい」という発話が生成される．

4.4 文脈情報の限界

4.4.1 文脈情報と発話の関係が希薄な事例の選択

次例のように，ユーザの発話によってはコマンド履歴に文脈情報が現れにくい場合がある．このような場合，事例が生成された時点のコマンド履歴を文脈情報として保持しておいても発話との関係は希薄であるので，事例を選択するための有効な指標とはならない．

発話例: 「ディスクの使用量を知りたい」

検索事例 315:
「ネットワーク上のディスク使用量を知りたい」
「 df コマンドを使いなさい」

検索事例 80：
「カレントディレクトリのディスク使用量を知りたい」
「 du コマンドを使いなさい」

一般の対話を考えても，このような応答の特定は難しく，経験的に使用頻度の高い事例によって答えることも考えられるが，いずれの事例を選択すべきかは依然として曖昧のままである．

4.4.2 文脈情報以外に依存する事例選択

UNIX コマンドの利用には，その性質上，システムの制約や現実世界の状況による制約を考慮しなくてはならない場合もある．そういった場合はシステム内部の情報やハードウェア上の情報などのユーザの意図以外の情報も必要となり，文脈情報だけでは状況を特定できない．

発話例:「ファイルを印刷したい」

検索事例 5 : #
「lpr コマンドを使いなさい」
検索事例 12 : #lpr file.txt
「印刷は PS 形式のファイルでなくてはなりません」

検索事例 39 : #a2ps file.txt > file.ps
#lpr file.ps
「プリンタに電源が入っていないようです」
検索事例 66 : #a2ps file.txt > file.ps
#lpr file.ps
「プリンタデーモンが起動していないようです」

5．特に情報が得られない
12．ファイルを印刷する手続きについて失敗がある
39,66．ファイルを印刷する手続きを行なっている

「ファイルの印刷」についての状況は様々に考えられるが，この例の場合，文脈情報から一意に特定できる事例は 5, 12 のみである．39, 66 の事例については，計算機内部や外的要因に関する情報が得られてはじめて状況を特定できる．つまり，状況を特定する情報が本研究で用いる文脈情報には現れず，これだけでは事例を選択するための有効な指標とはならない．

5 リアクティブプランニング

文脈情報の限界が示すように，事例に基づく対話管理に伴う問題が完全に解消されるわけではない．文脈情報の限界は，不適切なプランを生成させることになる．生成プランに従ったシステムの応答が不適切でユーザが理解できなかった場合は，新たな説明を加えたり，現在のプランを破棄し，他のプランの利用を試みるといった対処が必要である．これらの問題は，現在実行中のプランが否定されることから現在のプランの逸脱問題として取り扱うことができる．

本研究では，生成プランの逸脱問題に対して，プランを逐次的に実行し，ユーザとの対話を行ないながらプランを修正する，リアクティブプランニングの枠組を提案している．本枠組は，システムの応答に対するユーザの発話を観測する枠組と，観測された状況に応じてプランを動的に修正し，対話を続ける枠組からなる．

5.1 フィードバックの観測

システムがプランの逸脱を認識するためには，システムの発話に対するユーザの応答を観測する必要がある．ユーザから獲得される応答をフィードバックと呼ぶ．また，フィードバックから明確に認識が可能な情報を表 2 に示す．なお，フィードバックの獲得は，最初のシステムからの応答がなされた以後，ユーザの発話を自然言語処理によって内部表現に変換する際に行なわれる．

表 2: ユーザのフィードバック例

フィードバック	発話表現
コマンド情報	ファイルを移動したい
意図の明確化	知りたいのはファイルの内容です
応答への理解	わかりました，なるほど
応答への同意	そうですね，その通りです
応答への承認	はい，実行します，暗黙の実行
応答への確認	本当ですか？
応答の促進	それで，続けて下さい
応答への停止	もういいです．黙りなさい
応答への否定	いいえ，実行できません
応答への不理解	わかりません，何ですか

5.2 対話修正の枠組

ユーザの発話がプランを逸脱していた場合どのような修正を行なうかは，Tweak Rule によって定義されている．Tweak Rule とは，観測されるフィードバックの種類から対応するプラン修正ルールを起動する if-then ルールである．

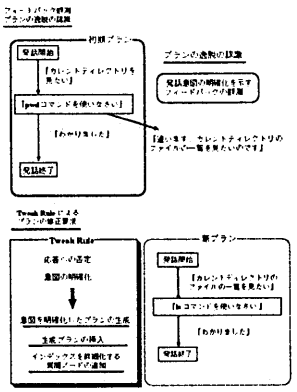

図 4: 逐次修正の枠組

図 4 はユーザのフィードバックが Tweak Rule によって修正され，新たなプランを生成する例を示している.「カレントディレクトリを見たい」というユーザの発話に対し，システムは事例の検索・修正によって「pwd コマンド」に関する応答プランを生成する．この初期プランは，システムの発話がユーザの意図にしたがった応答であれば「わかりました」というフィードバックが期待されることを示している．しかしながら，この例ではユーザの発話は期待されるフィードバックを返さない．したがって，プランにはたどるべきフィードバックを持つアークが存在しないために，初期プランの逸脱が認識される．

観測されるフィードバックにマッチする Tweak Rule が活性化され，曖昧さを解消する情報に基づいた再プランニングと，初期プランとのリンクが生成される．また，次回からの応答のために事例インデックスの詳細化情報を含んだ質問ノードが付加される．

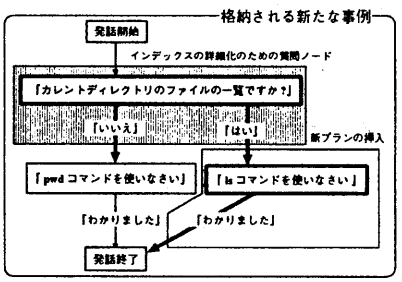

図 5: 新たな対話事例

したがって,「ls コマンド」のプランが挿入された新たな事例 (図 5) が生成される．ユーザの「違います」といった発話に対する応答としてシステムは「ls コマンド」に関する発話を行なう．ここでユーザが「はい」と答えれば対話は終了し，新たな事例として事例ベースに蓄えられる．こうして新たな事例が格納されるので同様の発話があった場合，この事例が検索されれば,「ディレクトリを見たい」という発話に対する応答が可能となる．

6 おわりに

本稿では，Assist-R のインプリメントを通して，CBR における一般的な問題の他に，対話特有の問題を明らかにし，問題の解決の新たなアプローチとして，発話の文脈情報を用いた枠組を提案した．また，候補選択に有効な文脈情報の様々な利点とともにその限界を明らかにし，リアクティブプランニングの枠組によって事例の再利用性を向上させ，対話における事例ベースの有効性を示した．

参考文献

[1] Hammond, K.: *Case-Based Planning*, academic press (1986).

[2] Riesbeck, C. and Schank, R.: *Inside case-based reasoning*, The Institute for the Learning Sciences Northwestern University Evanston, Illinois (1989).

[3] Uehara, K. and Ogino, H.: A Case-Based Approach to Text Planning, *Proc. of the IASTED International Conference on Artificial Intelligence, Expert Systems and Neural Networks*, pp. 123–127 (1994).

[4] Nagao, K. and Rekimoto, J.: Ubiquitous Talker: Spoken Language Interaction with Real World Objects, *Proc. of IJCAI-95*, pp. 1284–1290 (1995).