対話コーパスからの対話制御情報の取得

奥 智岐　西本卓也　荒木雅弘　新美康永

京都工芸繊維大学工芸学部電子情報工学科
〒 606-0962 京都市左京区松ヶ崎御所海道町
Email tomoki@vox.dj.kit.ac.jp

あらまし　本稿では, タグ付けされた対話コーパスから話題の階層構造などの対話制御情報を取得する方法について報告する. これは, 我々が開発したタスクに依存しないフレーム駆動型対話制御システムにおいてタスクの変更を容易に行うために, 必要な対話制御情報をコーパスから取得することを目的としている. 我々のシステムでは, 対話の進行に伴う対話履歴を話題フレームの木構造として管理している. 一方, 談話セグメントタグの付与された対話コーパスから話題の遷移を抽出することで, 新しいタスクの話題フレームの作成が容易になる. また, ユーザ発話を談話解析する段階で必要な情報（動詞と談話行為の関係,caseframe の定義など）も談話行為タグ付コーパスから取得する. この手法を 3 つのタスクに対して適用し, 木構造情報の容易な取得と 80 %以上の精度で談話解析ができる談話解析情報の取得を実証した.

キーワード コーパス, フレーム駆動型, 木構造, 談話解析,caseframe

Obtaining Information for Dialog Control From Dialog Corpus

Tomoki OKU , Takuya NISHIMOTO , Masahiro ARAKI , Yasuhisa NIIMI

Dept. of Electronics and Information Science
Faculty of Engineering and Design, Kyoto Institute of Technology
Matsugasaki,Sakyo-ku,Kyoto 606-0962,JAPAN
Email tomoki@vox.dj.kit.ac.jp

Abstract This paper describes a mothod for obtaining information for dialog control from dialog corpus with tags. It makes it easy to change task in the frame-driven task-independent dialog control system we developed. In our system, the dialog history is represented by a tree of which nodes are topic frames. Since the movement of topics can be seen easily in dialog corpus with discource segment tags, topic frames for a new task can be obtained almost autometicaly.The information which requires to interpret user's utterances, that is, relationships between a verb and a dialog act, definitions of the caseframe, etc, can be obtained from corpus with discourse act tags. This method was applied to three tasks. It wad proven from this application that topic frames can be obtained easily and extracted information for the discourse analysis can interpret user's uttreance with an accuracy of over 80 %.

Key words corpus, frame-driven, tree construction, discourse analysis, caseframe

1 はじめに

我々はタスク変更時のコスト並びにに労力の削減を目的としてタスクに依存しない音声対話の制御方式を提案した. 対話制御方式としては, 拡張性に富み主導権交代型の対話の制御が比較的簡単に記述できるフレーム駆動型の対話制御

方式を用いた. これは最初テキストベースの対話システム [1] で用いられたが, 最近では音声対話システムでも用いられるようになった [2][3]. またよく知られているように, 目的指向型の対話では, タスクに依存した木構造に従って対話が展開される [4]. 我々はこの事実を利用して, フレーム駆動型の対話制御方式をより大規模なタスクに適用できるように拡張した [5].

このシステムを実際に新しいタスクに適用するには, 目標とするタスクの対話例から対話の流れや発話される単語, 文法の情報を取得する必要がある. 現在の研究を推進するために実際の音声対話データの書き起こしテキストにタグを付与したタグ付きコーパスの整備が進められている. これは人工知能学会の談話タグワーキンググループがまとめた発話単位タグマニュアルに従って発話単位ごとにタグを付与したもので, セグメント情報を明示する談話セグメント（発話境界）タグ [7] や発話内行為を表す談話行為（発話単位）タグ [8] が付与されている. 本研究ではこのタグに着目し, 談話セグメントタグや談話行為タグの付与された対話コーパスから対話の木構造と談話解析に必要な情報を取得することを試みた. 談話セグメントタグの遷移は対話の木構造と考えることができ, また談話行為タグは本システムの談話目標と類似したものであることから談話解析情報を取得できる.

本稿では3つのタスクにおいて談話セグメント, 談話行為タグの付与された対話コーパスから木構造や談話解析に必要な情報を取得し, 対話システムにおけるタスク変更を容易に行えることを検証した.

2 フレーム駆動型対話制御方式

2.1 音声対話システムの概要

本研究で想定している音声対話システムの構成を図1に示す. システムは音声認識部, 言語解析部, 対話制御部, 応答文生成部, 音声合成部の五つのモジュールから構成される.

音声認識部で認識されたユーザ発話文の単語列は言語解析部に送られ, 構文解析および意味解析が行われ, ユーザ発話情報として話題, 焦点, 談話目標が抽出される [6]. このユーザ発話情報は対話制御部に入力されて談話履歴として記憶されると共に, 対話システムの行動を決定する. システムの応答は, ユーザ発話情報と同形式のもので, 応答文生成部に送られ, 発話文が生成される.

2.2 話題フレーム

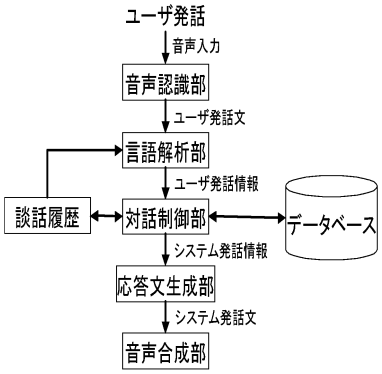
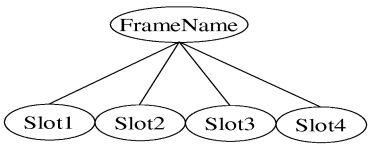


図1：音声対話システムの構成

先に提案した対話制御方式は, 人工知能の分野でよく用いられているフレームという知識の表現法を利用したものである.

Frame Name	State	[Frame ID]	[P_frame ID]	
Slot1	[Value]	Before process1	After process1	Priority1
Slot2	[Value]	Before process2	After process2	Priority2
Slot3	[Value]	Before process3	After process3	Priority3
Slot4	[Value]	Before process4	After process4	Priority4

(a)話題フレーム



(b)話題木

図2：話題フレームと話題木

フレームは図 2(a) のような構造になっている. 本稿では対話制御に用いるために話題フレームと呼んでいる. フレームにはそれぞれ名前がついており, 固有の ID 番号も付けられている. また, フレームにはいくつかのスロットがあり, 同じ階層で展開される話題を扱うことができる. 各スロットではそれぞれ値を持つことができ, ここに対話に関するデータが格納される. スロットにはこの値を求めるための処理とそれを求めた後の処理, また処理の優先順位が記述されている.

このフレームは図 2(b) のようにフレーム自体を上位話題, 各スロットを下位話題とした木構造に相当すると考えられる. 話題フレームのスロット値として, 他の話題フレームをとる事を許しているので, 話題フレーム全体は大きな木構造を構成している.

また, この話題フレームはお互いにリンクを

張ることができ, これによって階層構造として展開される話題を扱うことを可能としている. 従って, システムが持っている話題フレーム全体は１つの木とみなすことができる. これを静的話題木と呼ぶ. これに対し, 実際の対話中に話題が展開することで生成される話題木を動的話題木と呼ぶ. この動的話題木は静的話題木の部分木となっている. システムは対話中に動的話題木を生成することでユーザとの対話の内容, つまり談話履歴を管理している.

2.3 対話制御としてのフレームハンドラ

本システムでは対話を進めながら動的に話題木を生成する. 対話が進行してある程度の階層までいくとそこから下の階層はデータベースになっており, システム側はデータベースの検索を行い, 具体的な事例 (例えば, 観光案内タスクであればであれば, 庭園の特徴や社寺の建立者など) をユーザに提示する. フレームでの対話制御ではシステムが主導権をとった場合とユーザが主導権をとった場合ではその動作が異なる.

システム側はなるべく早くユーザに具体的事例を提示するように対話を進める. つまり動的話題木のなかで, 焦点とラベル付けされた話題から始めてまだ埋まっていない優先度の一番高いスロットを埋めようと努力する. 基本方針としては深さ優先探索 (depth-first search) の方針をとりながら対話を行う.

これに対してユーザ発話の解析では, ユーザは話題階層の上位, 下位といったことには関係なく, 現話題の近傍のことについてしゃべるものと仮定する. つまり, 現話題 (ラベル付けされた枝) からの等距離探索を基本方針として, ユーザ発話に含まれている話題に適応するスロットを検索する. このようにユーザが決まったことしかしゃべれないのではなく, 現話題の近傍のことをしゃべれることでシステム側とユーザ側の主導権交代型の対話システムが可能となる.

これまで述べたように, 本方式ではタスクに依存した情報はすべてフレームに記述されている. つまり, この方式であれば, タスクを別のものに変更するときには, 基本的に知識データベースとフレームを新たに構築するだけでよい. しかしフレームを操作するフレームハンドラはタスク独立である必要があり, 対話制御をするためのフレームハンドラとして動的話題木の生成や焦点・話題の移動 (動的話題木の探索), システム発話の生成, ユーザ発話からのスロット値抽出, そしてデータベース検索などの機能を持っている. また対話の流れが変わるような条件分岐の場合の動作は, 例えば After Process の所にデーモンを埋め込むことにより実現している.

3 コーパスからの情報取得

[2：挨拶 :]
00:05:896-00:08:024 0000 L:こちら道路情報センターでございます
00:08:168-00:08:328 0001 R:はい
[2：現在位置とレストラン :]
00:09:032-00:29:096 0002 R:F えとですね*ちょっと今今ちょっとまだ京都駅の近くにいるんですけど*ちょっとこのあと烏丸御池のへんでちょっ*と用事があって行くんですけど**ちょっとその近辺でちょっとレストラン教えてもらいたいんですけど
00:29:064-00:30:328 0008 L:はいわかりました
[2：定食屋 :]
00:31:448-00:35:704 0009 R:F えとですね定食屋みたいなのはありますかね
00:36:456-00:36:566 0010 L:はい
[1：定食屋 :]
00:36:566-00:45:000 0011 L:F えそういたしましたら簡単な定食をお出しします*F え和風のお店かもし中華料理のお店もございますが
00:45:400-00:46:952 0013 R:F あ和風のほうがいいんですけど

(a) 談話セグメントタグつきコーパス

[< 対話開始 ><>] → ①慣用的な発話
00:05:896-00:08:024 0000 L:こちら道路情報センターでございます
[< 対話開始 ><>]
00:08:168-00:08:328 0001 R:はい
[< 依頼 >< 関係無 >] → ②働き掛けタグ
00:09:032-00:29:096 0002 R:F えとですね*ちょっと今今ちょっとまだ京都駅の近くにいるんですけど*ちょっとこのあと烏丸御池のへんでちょっ*と用事があって行くんですけど**ちょっとその近辺でちょっとレストラン教えてもらいたいんですけど
[< 了解 ><>]
00:29:064-00:30:328 0008 L:はいわかりました
[< 未知情報要求 >< 関係無 >]
00:31:448-00:35:704 0009 R:F えとですね定食屋みたいなのはありますかね
[< その他の応答 >< 0009 >]
00:36:456-00:36:566 0010 L:はい
[< 未知情報要求 >< 関係無 >]
00:36:566-00:45:000 **0011** L:F えそういたしましたら簡単な定食をお出しします*F え和風のお店かもし中華料理のお店もございますが
[< 未知情報応答 >< **0011** >] → ③応答タグ (0011 に応答)
00:45:400-00:46:952 0013 R:F あ和風のほうがいいんですけど

(b) 談話行為タグつきコーパス

図３：タグ付コーパス

本研究では図３に示すようなタグ付コーパスから対話制御に必要な情報を取得することを試みる。

図 3(a) は各発話（発話境界）に対して「やりとり」,「話題」などのセグメント情報を明示している談話セグメントタグの付与されたコーパスであり、その発話から始まる話題名や話題の変化の大きさなどが記述されている。談話セグメントタグは,

$$[topic_break_index : topic_name : topic_options]$$

という形式で記述されている.*topic_break_index* は「話題境界らしさ」を表す指標で, 基本的な「やりとり」を単位として, 前後の内容の関連性の強さを主観的に判断し, 数値として記述される. この数値は値が大きいほど話題が大きく変わっていることを示し,2 か 1 のいずれかが与えられる.*topic_name* は話題名で, その直後の発話からどのような話題が展開されているかが記述される. これら 2 つの値は必須であるが,*topic_options* は話題の遷移に関する情報の記述で必須ではない.

図 3(b) は発話内行為を表す談話行為タグの付与されたコーパスで、依頼, 示唆, 肯定・受諾など, 本稿で提案する対話システムの談話目標に類似した記述がされている. 談話行為タグには, 大きく分けて①慣用的な発話に対するタグ、②働き掛けタグ、③応答タグがある。慣用的な発話とは、対話の開始、終了、了解等である。働き掛けタグは新たなやり取りのための始まりを表し、以前の会話との関係の有無も記述される。応答タグとは働き掛けに対する応答であり、どの発話に対するかが発話番号によって記されている。

3.1 対話制御に必要な情報

先に提案した対話制御システムを新しいタスクに適用するには, 対話制御に必要な情報として主に以下の 2 つのものが必要となる.

前節で説明したように, 本システムでタスクに依存する情報は全てフレームとデータベースに記述されており, タスク変更時には静的話題木を用意する必要がある. 静的話題木は話題の展開を木構造で表現したものであり, 変更しようとするタスクの対話例から話題の展開を推測して木構造情報を取得する.

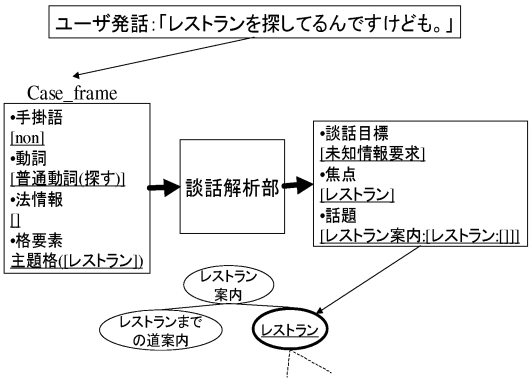


図 4 ：談話解析部の入出力関係

また, 図 1 で説明したように, 対話制御部へは構文解析, 談話解析を経たユーザ発話情報が入力として与えられる. 本システムの談話解析部の入出力関係を図 4 に示す。まず構文解析部ではユーザ発話用の文法や意味マーカなどから発話文の手掛語, 動詞, 法情報, 格要素が抽出される. この 4 つの特徴をリストにしたものを caseframe（発話フレーム）と呼んでいる. 談話解析部ではこの caseframe から談話目標と焦点, 話題を抽出しており, 談話解析をするには動詞と意味マーカの関係を表した caseframe 定義や動詞と談話目標の関係を用いている. もちろん caseframe の定義はタスクに依存した情報であり, タスク変更時にもっともコストのかかる部分の一つと考えられる

3.2 木構造情報の取得

木構造情報, つまり静的話題木の骨組みは対話の流れである. そこで, この対話の流れを談話セグメントタグの遷移から取得することを考える.

談話セグメントタグは前述したように,

$[topic_break_index : topic_name : topic_options]$

という形式で記述されている. このうち $topic_options$ は必須ではないので情報取得には使用しない. 本稿ではそれ以外の 2 つの指標から話題の木構造を推測する. まず $topic_name$ は話題木の各要素に相当し, そのときの $topic_break_index$ が話題の上下, つまり話題の階層関係を表していると考えられる. 例えば,$topic_break_index$ が 1 ならば, 現在の話題（$topic_name$）と一つ前の話題は同じ階層であると考え,1 つの話題フレーム内のスロットとして記述する. また $topic_break_index$ が 2 ならば階層が上下関係にあると考え, 新しい話題フレームを生成する. ただし,$topic_name$ には挨拶や各種応答も含まれているのでこれらは考えない. このような談話セグメントの遷移をいくつかの対話で見ていくことで, ある程度の木構造情報が得られると考えられる.

3.3 談話解析情報の取得

本システムで談話解析を行うには caseframe 定義や動詞と談話目標の関係の情報が必要となる.caseframe 定義とは動詞・法情報と, それに関係する意味マーカ（格要素）の関係を定義したものであり, ユーザ発話から話題・焦点を解析するための情報である. 動詞と談話目標の関係情報とは, ユーザ発話に登場する動詞とそれが表している談話目標の関係のことで, 談話目標解析のための情報である.

このような情報は対話コーパスの発話単位ごとに現れる.caseframe 定義は, 発話単位ごとの動詞, 法情報, 意味マーカを関連付けすることで表現できる. 法情報は動詞に関連する助動詞に相当し, 意味マーカは動詞の前にある名詞句であると考られる. 対話コーパスを形態素解析 (chasen,chawan による) し, その結果から caseframe 定義を作成した.

また動詞と談話目標の関係は, 発話単位ごとに登場する動詞とそれに付与された談話行為タグの関係に等しいと考えられる. 本システムで扱っている談話目標と談話行為タグは名称が

多少異なるが大体同じようなことを表現している.

4 評価実験

前節で解説した方法に従い, 実際の対話例から新しいタスクに必要な情報を抽出した. 木構造情報の取得は特にツールの開発などは行っておらず, 談話セグメントタグの推移だけから手動で静的話題木の生成を考えた. 談話解析情報の取得に関しては, 対話コーパスを形態素解析し発話ごとにある程度の形式にまで自動で変換するツールを開発した. 但し, そのままでは単語にばらつきがあるために現段階では手修正を加えている.

4.1 各タスクの概要

今回扱ったタスクを表 1 にまとめる.

表 1 ：タスクの概要

タスク名	概要
レストラン案内	車での移動を想定し, 京都市内の各地にあるレストランの情報と現在地からの道案内
道案内	電車を使っての移動手段と所要時間, 料金などをナビゲート
トラブルシューティング	車が故障した場面を想定し, その原因追及と対処法の提示

4.2 結果と考察

4.2.1 木構造情報の取得

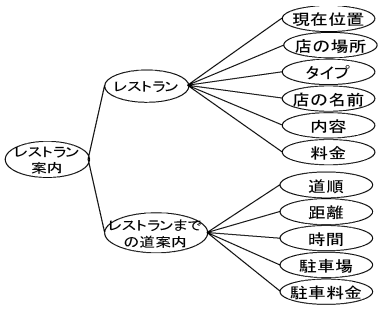
図 3(a) のような談話セグメントタグ付コーパスからタグだけを取ってきたものが図 5(a) であり, 対話 1 が図 3(a) に相当する. いくつかの対話例からこのようなタグの系列を取り出してきて木構造を推測した結果, 図 5(b) のような木構造を容易に作成することができた. 同様に道案内タスク, トラブルシューティングタスクでも容易に木構造を作成できた.

4.2.2 談話解析情報の取得

図 3(b) からは caseframe 定義や動詞と談話目標の関係などの談話解析情報を取得する. 本節の最初に述べたように, まず chasen,chawan を用いて形態素解析し品詞分解する. 法情報には動詞の後の助動詞を, 意味マーカには動詞の前の名詞句を関連付けて caseframe 定義を生成した. 図 6(a) がその例である. このままでは意味マーカやその他の単語に統一性がなく解析に用いにくいため, 図 6(b) のようにある程度の単語に修正した.

対話 1	対話 2	対話 3
[2：挨拶 :]	[2：挨拶 :]	[2：挨拶 :]
[2：現在位置とレストラン :]	[2：現在位置とレストラン :]	[2：現在位置とレストラン :]
[2：定食屋 :]	[1：確認：聞き返し]	[2：食事場所 :]
[1：定食屋 :]	2：道案内 :]	[2：料理の種類 :]
[1：定食屋 :]	[1：確認：聞き返し]	[1：確認：聞き返し]
[2：名前 :]	[1：道案内：復帰]	[2：イタリア料理 :]
[2：料金 :]	[1：道案内 :]	[2：店の名前 :]
[2：道案内 :]	[2：所要時間 :]	[1：店の名前 :]
[1：道案内 :]	[2：料金 :]	[1：店の名前 :]
[2：確認：聞き返し]	[2：料金 :]	[2：道案内 :]
[2：道案内：復帰]	[2：駐車場 :]	[1：確認：聞き返し]
[1：道案内 :]	[1：駐車場 :]	[1：道案内：復帰]
[1：道案内 :]	[1：確認：聞き返し]	[2：料理内容 :]
[1：道案内 :]	[1：駐車場：復帰]	[2：料理内容 :]
[1：道案内 :]	[2：駐車料金 :]	[2：駐車場 :]
[2：挨拶 :]	[1：確認：聞き返し]	[2：挨拶 :]
	[2：挨拶 :]	

(a) 談話セグメントタグの遷移



(b) タグの遷移から考えた木構造

図 5 ：木構造情報の取得

動詞と談話目標の関係は, 動詞と談話行為タグを発話ごとに関連付けすることで生成した. 図 7(a) がそれで, 図 7(b) はそれを修正したものである.

これらの情報を用いてどの程度談話解析できるのかを実験した. 図 4 のような入出力関係で行い, 入力は発話文を構文解析したユーザ発話情報, つまり caseframe(発話文の手掛語, 動詞, 法情, 報格要素) であり, 出力は談話目標, 焦点, 話題の 3 つの要素である. 出力が正解例と同じであるかを判断した. その結果が表 2 である.3 つのタスクともに 80 %以上の精度を得られ, 本方式によってある程度コーパスから情報を自動的に取得できることが確認された.

表 2 ：談話解析の実験結果

タスク (対話数)	正解数/評価数	正解率
レストラン案内 (4)	29/35	82.86%
道案内 (10)	149/186	80.11%
トラブルシューティング (4)	16/18	88.89%

```
cframe_def([普通動詞 (いる)], [], [対象主格 ([今今]), 属格 ([京都駅]), 主題格 ([近く])]).
cframe_def([普通動詞 (あっ)], [], [対象主格 ([ん]), 対象主格 ([ちょっと]), 属格 ([あと島丸御池]), 主題格 ([へん]), 主題格 ([用事])]).
cframe_def([普通動詞 (行く)], [], []).
cframe_def([普通動詞 (教え)], [], [対象主格 ([ん]), 主題格 ([ちょっと近辺]), 対象主格 ([レストラン])]).
cframe_def([普通動詞 (もらい)], [希望], []).
cframe_def([普通動詞 (わかり)], [了寧], []).
cframe_def([普通動詞 (あり)], [丁寧], [対象主格 ([定食屋みたい]), 属格 ([の])]).
cframe_def([普通動詞 (いたし)], [丁寧], []).
cframe_def([普通動詞 (出し)], [], [対象主格 ([簡単]), 主題格 ([定食])]).
cframe_def([普通動詞 (し)], [丁寧], []).
```

(a) コーパスから作成された caseframe 定義

```
cframe_def([普通動詞 (いる)], [], [属格 ([駅名：_]), 主題格 ([周辺])]).
cframe_def([普通動詞 (ある)], [], [属格 ([場所：_]), 主題格 ([周辺])]).
cframe_def([普通動詞 (教える)], [], [主題格 ([周辺]), 対象主格 ([レストラン])]).
cframe_def([普通動詞 (ある)], [丁寧], [対象主格 ([タイプ：_])]).
cframe_def([普通動詞 (出す)], [], [主題格 ([タイプ：_])]).
cframe_def([普通動詞 (かもし)], [], [属格 ([料理種別：_])]).
cframe_def([普通動詞 (出す)], [丁寧], [属格 ([料理種別：_]), 主題格 ([タイプ：_])]).
cframe_def([普通動詞 (ござる)], [丁寧], [主題格 ([値段：_])]).
cframe_def([普通動詞 (いう)], [丁寧], [主題格 ([地名：_])]).
cframe_def([普通動詞 (離れ)], [], [属格 ([タイプ：_]), 属格 ([地名：_])]).
```

(b)(a) に手修正を加えたもの

図 6 ：caseframe 定義の作成

```
verb_act([普通動詞 (いる)], [依頼]).
verb_act([普通動詞 (あっ)], [依頼]).
verb_act([普通動詞 (行く)], [依頼]).
verb_act([普通動詞 (教え)], [依頼]).
verb_act([普通動詞 (もらい)], [依頼]).
verb_act([普通動詞 (わかり)], [了解]).
verb_act([普通動詞 (あり)], [未知情報要求]).
verb_act([普通動詞 (いたし)], [未知情報要求]).
verb_act([普通動詞 (出し)], [未知情報要求]).
verb_act([普通動詞 (し)], [未知情報要求]).
```

(a) コーパスから作成された動詞と談話目標の関係

```
verb_act([普通動詞 (教える)], [依頼]).
verb_act([普通動詞 (もらう)], [依頼]).
verb_act([普通動詞 (かもし)], [未知情報要求]).
verb_act([普通動詞 (出す)], [未知情報要求, 未知情報応答]).
verb_act([普通動詞 (まぎ)], [未知情報応答, 確認]).
verb_act([普通動詞 (それ)], [未知情報要求]).
verb_act([普通動詞 (はじ)], [未知情報応答]).
verb_act([普通動詞 (離れ)], [真偽情報要求]).
verb_act([普通動詞 (しまう)], [真偽情報要求]).
verb_act([普通動詞 (突き当たり)], [未知情報応答]).
```

(b)(a) に手修正を加えたもの

図 7 ：動詞と談話目標の関係の作成

5 おわりに

本稿では, 以前提案したタスクに依存しないフレーム駆動型の対話制御方式において, 現在使用しているタスクから他のタスクへ移行する際にかかる労力, コストを減らすという視点から, タグ付対話コーパスからタスクに依存する情報を取得しその労力を軽減する方法について述べた. 対話コーパスに付与される談話セグメントタグや談話行為タグから, 対話の木構造や談話解析に必要な caseframe 定義などの情報を取得しタスク変更のコスト削減を考えた. 本稿ではこの情報取得の方法を 3 つのタスクに適応し, 容易に木構造情報を取得できることを実証し,80 %以上の精度で談話解析ができる談話解析情報を取得することができた. 本稿で行ったような情報取得の方法は本システムにとどまらず, 様々な音声対話システムにおけるタスク変更でも活かされると考えられる. 特に木構造情報の取得はどのような対話システムにおいても必要なことであり, 今後の有効利用も考えられる.

今後の課題としては, まだまだ自動的に情報取得ができていないということがもっとも大きな課題として考えられる. 例えば, 談話セグメントタグの情報として現在は *topic_options* の指標を用いていないが, これを用いることで挨拶や応答の発話を自動的に排除できるということも考えられる. また *topic_break_index* についても一つ前の発話だけに着目せず以前のいくつかの発話に着目すれば階層構造を上下のどちらに推移しているのかが分かるのではないかと考えている.caseframe 定義の情報取得に関しても, 現在は形態素解析の段階で単語の形式をそのままで行っているが,chasen で形態素解析する際に単語を基本形に変換することで単語（特に動詞）の種類が削減できると考えられる.

参考文献

[1] Bobrow, J.G., Kaplan, R., Kay, M., Morman, D., Thompson, H., and Winograd, T., "GUS: a frame-driven dialog system," Artificial Intelligence, vol.8, pp.155-173 (1977).

[2] Young, S.J. and Proctor, C.E., "The Design and Implementation of Dialogue Control in Voice Operated Database Inquiry Systems,"Computer Speech and Language, vol.13, no.4, pp.329-353 (1989).

[3] Godden, D., Meng, H., Polifromi, J., Seneff, S., and Busayaongchai, S., "A Frame-Based Dialogue Manager for Spoken Language Applications," Proc. of ISCLP'96, pp.701-704 (1996).

[4] Grosz, J.B. "Discourse Knowledge," in Walker, D.E. eds, Understanding Spoken Language, 229-337, North-Holland, New York, (1978).

[5] Niimi, Y., Takinaga, N., and Nishimoto, T.,"Dialogue Management in a Spoken Dialogue System, SDSKIT-3," Int. Workshop on Speech and Computer (SPECOM'98), pp.91-96 (1998).

[6] Niimi, Y., Takinaga, N., and Nishimoto, T.,"Extraction of the Dialog Act and the Topic from Utterances in a Spoken Dialog System," to appear in Proc. of ICSLP'98 (1998).

[7] 山下 洋一, 小磯 花絵, 村上 雄大, 石崎 雅人, "話題（談話セグメント）に関するタグ付けマニュアル（第 2.2 版）",(1997).

[8] 談話タグワーキンググループ, "発話単位タグ第２版マニュアル",(1997).