音声対話システムにおける不適格性の処理

船越　孝太郎　　徳永　健伸　　田中　穂積

東京工業大学 情報理工学研究科
E-mail: {koh, take, tanaka}@cl.cs.titech.ac.jp

音声対話システムが話し言葉に対応するためには，言い直し，助詞落ち，倒置などの不適格性とよばれる現象に対処する必要がある．これらの不適格性の中で特に問題となるのは，言い直しあるいは自己修復と呼ばれている現象である．しかし，自己修復に関する既存の手法は，自己修復表現を捉えるモデルと，その修正処理に問題点がある．本論文では，それらの問題点を改善した新しい手法を提案する．そして，提案手法を我々の作成した疑似対話コーパスに適用した結果を基に，提案手法の問題点について考察する．

Processing of Ill-formedness on Speech Dialogue Systems

Kotaro Funakoshi, Takenobu Tokunaga and Hozumi Tanaka

Department of Computer Science
Tokyo Institute of Technology
E-mail: {koh, take, tanaka}@cl.cs.titech.ac.jp

Speech dialog systems need to deal with various kinds of ill-formedness of speech inputs which appears in natural human-human dialogs. In particular self-correction (or repair) is one of the most problematic phenomena. There have been many proposals to deal with self-correction. However, they have limitations in both detecting the phenomena and recovering them. In this paper, we propose a new method overcoming these problems. We evaluate the proposed method by using our speech dialog corpus and discuss the limitation and future work.

1. はじめに

音声対話は，人間にとって機械との間のインターフェースとして最も望ましいものである．しかし，音声対話システムが日常にありふれた存在となるためには，人間の使用する曖昧で誤りの多い言葉，いわゆる話し言葉に対応できなければいけない．そのためには，繰り返し，言い淀み，言い直し，助詞落ち，倒置などの不適格性とよばれる現象に対処できる必要がある．

これらの不適格性の中で特に問題となるのは，言い直しあるいは自己修復と呼ばれている現象である．ユーザの発話中に自己修復表現が存在した場合，システムはその発話の中から不必要な語を取り除き，受理可能な発話を回復する必要がある．この自己修復に関する研究は，英語に関するものでは，4)1)2)9) などがあり，日本語に関するものでは，5)3)7) などがある．しかしながらこれらの論文で提案されている手法では，自己修復を捉えるモデルに不足があり，我々の作成した疑似対話コーパス[10] に見られるような表現をカバーできない．また，自己修復を検知した後の不要語の除去処理に関しても十分な手法を与えていない．

本論文では，日本語の自己修復に対処するための新しい手法を提案する．この手法では，従来の手法では捉えられなかった自己修復表現を捉える事ができるように自己修復表現のモデルを拡張する．そして，表層のマッチングと意味レベルでのマッチングを融合した，自己修復表現の解消法を提案する．

まず，2 節では不適格性とその中での自己修復の位置づけについて述べる．3 節では，本論文で用いるパーザと文法について述べる．4 節では，今回提案する自己修復表現の処理手法について述べる．そして 5 節では，提案手法をコーパス[10] に対して適用した結果を基に，提案手法の限界について考察する．

(1)　繰り返し（強調を意図したもの）
(2)　自己修復
　(a)　繰り返し（話者の思考の淀みによるもの）
　(b)　言い足し（新しい情報を付け足すもの）
　(c)　言い直し（新しい情報で置き換えるもの）
　　(i)　部分訂正（発話の一部分を訂正するもの）
　　(ii)　全訂正（発話を新しく始めるもの）
(3)　言い淀み

図 1　同一話者による冗長表現の分類

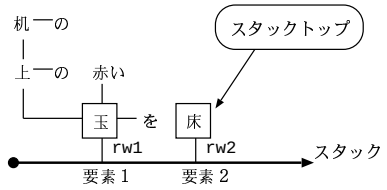

図 2　「机の上の赤い玉を床」という入力を解析した場合のスタックの一例

2. 不適格性

本論文で扱う不適格性は，助詞落ち，倒置，冗長表現の 3 つである．このうち，助詞落ちと倒置は，次節で述べる文法記述の方法によって，一般の構文解析と同じ枠組みで処理をする．

もう一つの不適格性である冗長表現は，同一話者によるものと複数話者によるものに分けられる[6]．本論文では，同一話者による冗長表現のみ考慮する．

同一話者による冗長表現を，本論文では**図 1** のように分類する．1 節で言及した自己修復はこの冗長表現の一種となる．自己修復はさらに，繰り返し，言い足し，言い直しに分ける．本論文では，強調を意図した繰り返しについては考慮しない．従って，全ての繰り返し表現は自己修復表現として扱われる．

言い淀みは，話者が単語を最後まで言わなかったために，単語の断片が残る現象である．この言い淀みは，自己修復と共に現れる場合が多いため，多くの研究では自己修復の枠組みの中に取り込んでいる．しかし現在の音声認識技術を考えたときに，単語断片が正確に認識されることは仮定しにくい．そこで本論文では，言い淀みの処理は自己修復とは切り放し，未知語の誤認識を除去するための一般の不要語処理に委ねる．この不要語の処理についてはここでは議論しない．

3. パ　ー　ザ

不適格性を解消するための処理はすべて，構文解析と平行してパーザの上で行う．以下，本論文で用いるパーザとパーザの使用する辞書について説明する．

3.1　係り受け解析を用いた漸進的な構文解析

構文解析の手法として，文節ベースの係り受け解析を採用した．我々の解析手法では内容語が重視され，機能語は内容語に付属するものと捉える．構文解析はスタックを用いて漸進的に行なう．本論文では詳細は省くが，このパーザの使用は，将来，提案手法を組み込む予定の音声対話システムに，漸進的な処理を行わせることを目的としている．

パーザは，解析の途中に生成される複数の構文仮説を，別々のスタックに保持する．スタックの各要素には，依存関係で表現された構文木が納められる（**図 2**）．各スタックの要素である部分構文木のルートになっている語を「ルート語」と呼ぶことにする☆．

スタックに新しい単語がプッシュされると解析が行われる．プッシュされた語が機能語であった場合には，単純にスタックの上から 2 つ目の要素のルート語にその機能語を付ける．既に機能語が付いている場合には，「に・も」のように連接が可能な場合を除いて，機能語が言い直されたと考えて新しい機能語に置換する．次にプッシュされる予定の語が機能語でなければ，ここで係り受け解析が行われる．係り受け解析はスタックトップの要素のルート語（$rw2$）とトップのすぐ下の要素のルート語（$rw1$）の関係を見て行う．すなわち，$rw1$ が $rw2$ に係り得るかどうか（もしくはその逆）を調べる．$rw1$ が $rw2$ に係ることができないならば，スタックは変化しない．しかし，$rw1$ が $rw2$ に係ることができるならばこのスタックをコピーし，片方には $rw1$ が $rw2$ に係る仮説（H1），もう片方には $rw1$ が $rw2$ に係らない仮説（H2）を保持する．仮説 H1 を保持するスタックでは，一度上 2 つの要素をスタックから取り出したあとに，新しい係り受け関係を作った要素 1 つだけをスタックトップに戻す．仮説 H2 を保持するスタックは変化しない．また，H1 については同様の操作を再帰的に行う．従って，

　[（君が）|（玉を）>

という仮説スタック☆☆に「押せ」という単語がプッシュされると，

　[（君が）|（玉を）|（押せ）>
　[（君が）|（(玉を）押せ）>
　[（(君が）（玉を）押せ）>

という 3 つの仮説スタックが生成される．

3.2　文法表現と辞書

本手法では，文節構造以外の文法に相当するものは，全て単語辞書の中に単語毎に用意する．

☆ 図 2 において四角で囲まれた語．
☆☆ [がスタックの底，| が要素間の区切り，> がスタックのトップ，() が係り受け関係を表す．

```
押して VERB IMP+ PUSH+ DEGREE:#STD SPEED:#STD
   !<OBJECT> 1 *  NOUN は|を|も|-  INSTANCE+
   !<AGENT>  1 *  NOUN は|が|も|-  ANIMATE+ INSTANCE+
    <TO>     1 *  NOUN に|へ|-     LOCATION+
    <FROM>   1 *  NOUN から        LOCATION+
    <EXTENT> 1 *  ADV  -           DEGREE:*
    <SPEED>  1 *  ADV  -           SPEED:*
```

図 3 命令動詞「押して」の辞書エントリ

ある内容語 c1 がある内容語 c2 に係る時，c1 は c2 に対して特定の役割を担っていると考える．例えば，「押して」という命令動詞の辞書のエントリは**図 3** のように記述する．

図 3 の第一行目は,「押して」という語が，左から順に，

- 動詞である
- 命令（IMP+）である
- PUSH+という動作を表す素性を持つ
- 指定の無いときの動作の程度は#STD である
- 指定の無いときの動作の速さは#STD である

ということを表している．第二行目以降は,「押して」に係ることのできる語の制約を役割毎に示している．例えば第二行目の，<OBJECT>（目的格）という役割は，左から順に，

- 必須格（!が示す）である
- 「押して」に対して 1 つしか存在しない
- 「押して」に係るときは前方依存 (F) ／後方依存 (B) のどちらでも良い（*はワイルドカード）
- 名詞しかこの役割は取れない
- その名詞についている機能語は「は」「を」「も」「-（無標)」のどれか
- INSTANCE+素性を持っている語でなければならない

という事を表している．

3.1 節で述べたパーザは，この辞書を用いて解析を行う．内容語 c1 が内容語 c2 に係ることができるかどうかは，c1 が c2 のエントリに示された役割の内のどれかを満たすことができるかどうかによって決まる．そして解析の段階で，全ての係り受けにその意味的な役割が割り当てられる．

このパーザにより，不適格性の内，助詞落ち，倒置は解決できる．すなわち，助詞落ちは上記の文法での-(無標) の場合として扱い，倒置は後方依存が可能かどうかを辞書に記述することで対処する．またこのように係り受け解析の段階で，語の役割を特定することで，後述する自己修復の意味的な処理が可能になる．

4. 自己修復表現の処理

4.1 過去の研究の問題点

我々の疑似対話コーパス[10] の中に現れる自己修復表現に過去の研究で提案されていた手法を適用したところ，対処できない例が見られた．

対処できない理由は大きく分けて 2 つある．1 つは，自己修復のモデルの不備，もう 1 つは自己修復を検知した後の修正処理の不備である．

まず，自己修復のモデルの不備を説明する．既存の研究では，どれも 9) の Repair Interval Model（RIM）に類するモデルを用いて自己修復を捉えている．RIM は，入力文の上で，修復を受けるものを含む区間を REPARANDUM（以下 RPD），修復するものを含む区間を REPAIR（以下 RP），RPD と RP の間に現れるフィラーや休止，手がかり句（cue phrase）☆を含む区間を DISFLUENCY（以下 DF）としたときに，

... RPD DF RP ...

の関係になるというモデルである．例えば，

「赤い玉を えっと 青い玉を 押して」

という発話の場合,「赤い玉を」が RPD,「えっと」が DF,「青い玉を」が RP に相当する．そしてこのモデルでは，手がかり句やフィラー以外の語が RPD にも RP にも含まれずに RPD と RP の間に現れることはないという前提をしている．従って，RIM は次のような自己修復を扱えない．

「赤い玉を 押して 君の前の玉を」

この例では動詞「押して」が，RPD である「赤い玉を」と RP である「君の前の玉を」の間に入っており，モデルの前提を破っている．

次に，自己修復を検知した後の修正処理についての不備について説明する．従来の手法では，検知した RPD の部分をまるごと発話から削除する方法が用いられている．実際，英語，日本語に関わらずこの処理が正しく機能する場合は多い．しかし，次のような発話では，この方法は重要な情報を落としてしまう．

「さっき押した赤い玉を 遠くに押したやつ
をもってきて」

この自己修復表現の処理は単なる削除だけでは不十分で，もっと複雑な処理が必要である．

4.2 自己修復の再分類

図 1 の内，自己修復に関する部分を，実際の処理に合わせて**図 4** のように再分類する．この分類の中で

☆ 自己修復を示す手がかり句は編集表現と呼ぶ.「ごめんなさい」,「じゃない」,「ちがう」等.

(1) 言い足し (繰り返しも含む)
  (a) 構造隣接 「赤い玉を 大きい玉を 押して」
  (b) 構造非隣接 「赤い玉を 押して 大きい玉を」
(2) 言い直し
  (a) 明示的
    (i) 構造隣接 「赤い玉を ごめん 青い玉を 押して」
    (ii) 構造非隣接 「赤い玉を 押して ごめん 青い玉を」
  (b) 非明示的
    (i) 構造隣接 「赤い玉を 青い玉を 押して」
    (ii) 構造非隣接 *「赤い玉を 押して 青い玉を」
(3) リスタート
  (a) 明示的 「赤い玉を ごめん 馬は 前に 行って」
  (b) 非明示的 「赤い玉を 馬は 前に 行って」

図 4 自己修復の処理上の分類

は，繰り返しを言い足しに含めている．ここで構造隣接という言葉を使っているのは，従来の 6) などの分類で使われている表面的な隣接性を示す言葉と区別するためである

言い足しは，“addition”，“appropriate repair” などと呼ばれているもので，RPD の中に間違った情報は含まれていないものである．言い足しの中で，構造隣接に含まれるものが今までのモデルでも扱える部類である．構造非隣接に含まれるものが，4.1 節で述べた，今までのモデルでは扱えないものである．この言い足しの RPD と RP の間には次のような制限がある．

> RPD と RP は同じ物，同じ様態，同じ動作を示していなければならない．[1]

言い直しは，“self-correction”，“repair”，“error repair” などと呼ばれるもので，RPD の中に間違った情報が含まれているものである．言い直しは，手がかり句が挿入されている**明示的**な言い直しとそうでない**非明示的**な言い直しとに区別する．これは，非明示的な構造非隣接に分類される類いの発話を人間が通常することはなく，仮に発話されても人間の聞き手ですら混乱し，理解できないと仮定するからである．おそらく，「赤い玉を 押して 青い玉を」などと指示された場合には，通常の人間であれば発話者に真意を問い質すだろう．

リスタートは，“restart”，“fresh start”，“full sentence repair” などと呼ばれているもので，これも明示的なものと非明示的なものとに分ける．漸進的な処理において，早い段階でリスタートを言い直しと区別することは難しい．RP に連体修飾句が含まれていたりすると，かなり先まで解析が進まないと判別できない場合もある．しかし，日本語においては「名詞句+は」やある種の手がかり句などの検出をすることで，ある程度のリスタートは正しく検出するできる可能性がある．明示的なリスタートの場合には既に訂正を行う意思表示を示す編集表現が与えられているので，この言い直しの手がかり句とその後に現れる情報を組み合わせることで，リスタートの処理を行える．一方，非明示的なリスタートは，ポーズの他にも，アクセントや身ぶりなど，パラリンガルな要素を考慮しないと，有意な検出は難しい．そこで，本論文では明示的なリスタートのみを考慮する．

4.3 自己修復の処理

本手法では 4) と同様に，自己修復個所の検知と修正を，構文解析と平行して行う．ただし，4) が決定的であるのに対し，本手法では自己修復の処理においても複数の仮説が生成される．複数の仮説が生成される場合仮説に尤度を与える必要があるが、その方法については本論文では省略する。

新しい単語が仮説にプッシュされて係り受け解析が行われると，係り受け解析が終わった仮説から順に自己修復表現に対する処理を受ける．自己修復の検知と修正の処理は，係り受け解析と同様にスタックの上 2 つの要素を処理することで行う．すなわち，スタックトップの要素の木が RP で，スタックの上から 2 番目の要素の木が RPD である（あるいは RPD を含んでいる☆）と考える．要素を 2 つ取り出した後の処理は，言い足し及び言い直しとリスタートの 2 つで別れる．図 4 では，言い足しと言い直しは別に分けたが，実際の処理は似ている部分が多いので 1 つにまとめて処理をする．

4.3.1 言い足し及び言い直しの処理

まず，構造非隣接型の自己修復をどのように捉えるべきかを考える．ここで，コーパスの観察などから，構造非隣接に分類されるタイプの発話は，

... RPD ... 動詞 DF RP ...

という形しか取らず，RPD は，主格や目的格として必ず動詞に係っていると仮定する．この結果，構造非隣接で RPD と RP になることができる組は，名詞句の組か副詞句の組しか無いことになる．

この構造非隣接型の冗長性を解消するための解釈の仕方は 2 通り考えられる．1 つは，RP が動詞に後ろから係ると考える方法（A）である．この時既に動詞に係っている RPD は，後から述べられた RP によって置き換えられると捉える．つまり，この解釈では自己修復が倒置と組み合わさったものと考える．

もう 1 つは，RP の後ろに来るべき動詞が省略されたものと解釈する方法（B）である．つまり，

... RPD ... 動詞 DF RP (動詞)...

☆ 構造非隣接の場合

(1) スタックの上 2 つの要素を，上から要素 2，要素 1 として取出す．ただし，要素 1 が編集表現ならば，明示的な自己修復であるというフラグを立てて，要素 1 の下のスタックの要素を取り出して，それを要素 1 とする．要素 1 のルート語が $rw1$，要素 2 のルート語が $rw2$ である．

(2) $rw1$ と $rw2$ の品詞が同じ場合（構造隣接）

(a) $rw1$ と $rw2$ が置き換え条件を満たせば，$rw2$ は $rw1$ の自己修復である可能性があると返して終了．

(3) $rw1$ と $rw2$ の品詞が異なる場合（構造非隣接）

(a) $rw1$ が動詞でなければ，自己修復である可能性は無いと返して終了．

(b) $rw2$ が名詞でも副詞でもなければ，自己修復である可能性は無いと返して終了．

(c) $rw1$ に係っている内容語の中に $rw2$ と置き換え条件を満たすもの（d_i^{rw1} とする）があれば，$rw2$ は d_i^{rw1} の自己修復である可能性があると返して終了．ただしこの時，明示的でないならば $rw2$ と d_i^{rw1} は 4.2 節の [1] の制約を満たさなければならない．

図 5 言い足しと言い直しの検出手順

という解釈をする．この場合，構造非隣接は見かけ上の存在であり，本質的には構造隣接と同じになる．つまり，省略された動詞を補完することで，

... RPD DF RP ...

という形に還元でき，これは従来の RIM で捉えられるパターンとなる．この解釈の場合，動詞を補完した後の処理は構造隣接の場合の処理とほぼ同じになるために，統一的な解釈ができるという利点がある．

しかしながら，(B) の場合，動詞の補完をするまでの処理を特別に用意しなければならず，これは（A）の処理に必要な手続きのほとんどを含む上に更に多くの処理が必要になる．さらに，動詞を補完してしまえば動詞が省略されなかった場合と全く同じように処理してよいのかは疑問である☆1．このような理由に加え，著者の内省では（A）の解釈の方が自然に思えたため，本論文では（A）の解釈を取った．

4.3.1.1 検出処理

ある仮説スタックが与えられると，検出は**図 5** の手順で行われる．図 5 の手順の中で使用されている置き換え条件とは，$rw1$ あるいは d_i^{rw1}(図 5 参照) と $rw2$ がそれぞれに付属している機能語も含めて満たさなければならない条件である．この条件が満たされるとき，$rw1$（あるいは d_i^{rw1}）は $rw2$ で置き換えられる．この置き換え条件は 7) の分類 A の (I) ☆2 とほぼ同じ

- P1 = P2
- N1 = N2 and P2 ≠ nil
- N1 ∼ N2 and P1 = nil

図 6 名詞の置き換え条件．P1，P2 はそれぞれ名詞 N1，N2 に付いている機能語である．= は全く同じ語であること，∼ は N1 と N2 が同じ意味クラスに入ることを意味する．P1 = nil は，N1 には機能語が付いていないことを示す．

である☆3．置き換え条件の一部（名詞に対する条件）を**図 6** に示す☆4．名詞 N1 と名詞 N2 の組に関して，図 6 のどれかを満たせば良い．

4.3.1.2 修正処理

既存の研究での修正処理は，単純に RPD を削除することで行われていた．しかしこれでは，RPD の中には存在するが RP では省略されてしまった情報まで削除してしまい好ましくない．そこで本手法では，$rw1$ をルートとする部分木（$t1$）が $rw2$ をルートとする部分木（$t2$）で置き換えられるときに，$t1$ には存在するが $t2$ には存在しない情報を，$t1$ から $t2$ に移し替える処理を行う．これは，具体的には $rw1$ に係っている語の内，$t2$ では省略された語を $rw2$ に付け替えることで行う．ただし，この時 $rw2$ に付け替えることで矛盾が起きるような語は付け替えないで捨てる．

例えば，

$t1$:（(赤い)（(ニワトリの）前の）玉を）

$t2$:（(青い）玉を）

という例の場合は，(ニワトリの）前の）が $t2$ で省略されているので $t2$ の「玉を」に付け替える．その結果 $t2$ は，

$t2$:（(青い)（(ニワトリの）前の）玉を）

となる．

この処理は部分木のルートだけではなく，その中間ノードに対しても再帰的に適用する必要がある．例えば，

$t1$:（(馬は）（前の）玉を）押して）

$t2$:（((青い）玉を）押して）

のという自己修復で，$t1$ と $t2$ のルートの「押して」に対してのみこの処理を適用した場合 $t2$ は，

$t2$:（(馬は）（(青い）玉を）押して）

となり「前の」が落ちてしまう．これを防ぐためには，$t1$ の「玉を」と $t2$ の「玉を」を対応づけて，「玉を」に関しても同様に付け替え処理を行う必要がある．

☆1 「赤い玉を 押して 青い玉を」は理解しがたいが，「赤い玉を 押して 青い玉 を押して」は多少の戸惑いはあるものの言い直しであるのだろうと解釈できる．

☆2 これは，RPD と RP が同じ構文カテゴリの句である場合に，自己修復表現であるならば満たさなければいけな条件を示したものである．名詞句，助詞句，助詞，動詞句，連体詞，副詞の自己修復の場合に分けて述べられている．例えば名詞句の場合，

（**I-1**） 同じ名詞句（例: [角] [角] ですか）

（**I-2**） 同じ意味カテゴリの名詞句（例: [こ こ] あ [受け付け] におりますが）

と分類されている．

☆3 7) の分類 A の（I）では形容詞の場合について触れられていない．ただし，形容詞の場合は副詞と同じでよい．

☆4 これは，7) の分類 A の（I）の名詞句と助詞句に関する条件をまとめたものである．

語 $rw1$ をルートとする部分木 $t1$ を，語 $rw2$ をルートとする部分木 $t2$ で置き換えるとする．この時，$rw1$ に係っている m 個の語を $d_1^{rw1}, d_2^{rw1}, ..., d_m^{rw1}$ とする．$rw2$ に係っている n 個の語を $d_1^{rw2}, d_2^{rw2}, ..., d_n^{rw2}$ とする．この $t1$ と $t2$ に対して，付け替えを行う関数 $dmerge(rw1, rw2)$ のアルゴリズムの概要を**図 7** に示す．このアルゴリズムは人間の自己修復の認識に関する下の 3 つの仮定を満たすようになっている．

- $d_1^{rw1}, ..., d_m^{rw1}$ の中の任意の語 d_i^{rw1} と同じ語は $d_1^{rw2}, ..., d_n^{rw2}$ の中にただ 1 つ d_j^{rw2} しかなく，その逆もまた成り立つ時に限り，d_i^{rw1} と d_j^{rw2} のそれぞれの $rw1$ と $rw2$ に対する役割が例え異なろうとも，d_i^{rw1} は d_j^{rw2} によって置き換えられたと認識できる．
- $d_1^{rw1}, ..., d_m^{rw1}$ の中の任意の語 d_i^{rw1} と同じ役割を持つ語は $d_1^{rw2}, ..., d_n^{rw2}$ の中にただ 1 つ d_j^{rw2} しかなく，その逆もまた成り立つ時に限り，d_i^{rw1} と d_j^{rw2} が例え異なる語であろうとも，d_i^{rw1} は d_j^{rw2} によって置き換えられたと認識できる．
- 上の 2 つの認識に食い違いがない場合にのみ，d_i^{rw1} は d_j^{rw2} によって置き換えられたと認識できる．

つまり，図 7 のアルゴリズムは，d_i^{rw1} が語そのものか役割によって d_j^{rw2} と 1 対 1 に対応付けが可能な場合で，なおかつ語と役割に関する対応付けに食い違いがない場合のみ，d_i^{rw1} と d_j^{rw2} を対応づけ，$dmerge(d_i^{rw1}, d_j^{rw2})$ を再帰的に実行する．この条件を満たさない d_i^{rw1} と d_j^{rw2} に関しては処理は何も行われずに無視される．これは，RPD（$t1$）と RP（$t2$）の間での対応関係が曖昧な場合には人間も特定の解釈をすることができないと仮定するからである．このアルゴリズムで，語と語の対応を取るのに役割を使うのは，基本的には助詞が同じかどうかを見ることと同じである．しかしながら，助詞は省略される場合があるので，助詞を見るだけでは解決できない場合がある．例えば，

$t1$:（((赤い）玉を）（君が） 押して）

$t2$:（((大きい）やつ） 押して）

という例の場合，単語も助詞も異なるため，意味すなわち役割を考えないと対応を取ることができない．また，もともと助詞がない場合 (名詞に係る形容詞など) も，すでに役割のラベルが与えられているので，新たに意味素性などの情報を用いて対応関係を計算する必要がなく，処理が効率的になる．

このアルゴリズムは，我々のコーパス内の事例と我々が解釈可能であると考え出した例とを満足する．しかしながら，我々の持つコーパスは規模が小さく，このアルゴリズムが一般的であるという主張はできない．このアルゴリズムの妥当性の検証は今後の研究課題である．

$dmerge(rw1, rw2)$: $rw1$ をルートとする部分木から $rw2$ をルートとする部分木への付け替え処理を行う

(1) $d_1^{rw1}, ..., d_m^{rw1}$ と $d_1^{rw2}, ..., d_n^{rw2}$ の間で同じ語であることを基準に対応を取る（対応 1）．

(2) $d_1^{rw1}, ..., d_m^{rw1}$ と $d_1^{rw2}, ..., d_n^{rw2}$ の間で同じ役割を持つことを基準に対応を取る（対応 2）．

(3) 対応 1 と対応 2 それぞれのなかで 1 対 1 の対応であり，なおかつ対応 1 と対応 2 の間で食い違いがない対応をもつ語のペアを取り出し，$dmerge$ を再帰的に適用する．

(4) $d_1^{rw1}, ..., d_m^{rw1}$ の内，対応 1・対応 2 のどちらからも対応づけを与えられなかったものだけ，$rw2$ にそのまま付け替える．

図 7 付け替え処理のアルゴリズムの概略

4.3.2 リスタートの処理

先に述べたように，リスタートに関しては明示的な場合，つまり編集表現が発話された場合しか扱わない．リスタートの場合重要なのは適切な検出のみで，処理に関してはリスタートの発生点より以前の入力を無視すれば良い．

リスタートの検知には，2 つの特徴を捉えることで対処する．1 つは，日本語の特性を利用し，編集表現の後に「名詞+は」が出現するかどうかを調べる．

もう 1 つは，4.3.1.1 で述べた，置き換え条件を逆に利用する．もし，編集表現の後に出現している部分木のルートが編集表現の直前に出来ていた部分木のルートと置き換え条件を満たさないならば，それは明示的な言い直しではなく，リスタートである可能性が高いと考えられる．この可能性は，編集表現の後に出現している部分木のルートの，編集表現からの形態素列上の距離が離れれば離れるほど高くなる．

5. 提案手法とシステムの検証

本論文で仮定した自己修復処理の手法を評価するために，我々の疑似対話コーパス[10] に人手で適用した．ここで，その結果判明した問題点について説明する．そして，今回提案した手法のその他の問題点とその解決方法について議論する．

5.1 コーパスによる検証

「ソフトウェアロボットとの疑似対話コーパス」[10] は，全 15 対話，532 発話を含む．今回対象とした不適格性の内，助詞落ち以外のものは 99 個所あった．この内，単純な倒置 15 個を除いた 85 個所が冗長表現で，14 個所の言い淀みを除く 71 個所が，単語断片ではないはっきりとした単語で構成される自己修復であった．

この 71 個所の内，正しく解決できないと思われる

ものが 16 個所存在した．ここで，正しく解決できる，とは妥当と思われる結果が第 1 候補に挙がる場合を意味する．第 2 候補以降に挙がるものもあるが，それは正しく解決できなかったものと見なした．

コーパスの規模と，人手での作業であることから，この結果の正解率などについては議論しない．その代わりに，対処できなかった冗長表現をタイプ別に分類し，それぞれに必要な処理について述べる．

● 7) の分類 A の（II）☆ に属するタイプ

これには 3 つ該当する例があった．その内の 1 つとして，

「黒の，ガンダムが黒の後ろに行って」

が挙げられる．係り受け解析を基本とする場合，このタイプのものが解決できないできないことは 7) で指摘されている．このタイプを係り受け解析の手法の上で解決するためには，構造非隣接の扱いを拡張するか，パターンマッチングによる対応付けが必要になる．あるいは，読み飛ばしや非明示的なリスタートの処理をポーズの情報などを用いて高い確信度で行うことができれば，うまく解決出来る可能性がある．

●単純な置き換えでは情報を損なうタイプ

これに含まれるものは 1 つであった．本手法では，RPD の単純な削除は情報の損失を招くとして，付け替えによる情報の保存を考えた自己修復の処理手法を提案した．しかし，本手法で提案した情報の保存は RP で省略された係り受け関係の移し替えのみを考慮していて，言い直された単語自体は単純に置き換えている．このままでは，次のような例で情報の損失を起こす．

「青を，そのブロックを押して」

この例では,「青」は青いものを示す代名詞として使われている．この「青」を単純に「ブロック」で置き換えてしまうと，折角話者が提供した「青色」という情報を失い，システムは曖昧性を正しく解決できない恐れがある．これを防ぐためには，単純に表層のシンボルの操作として自己修復を扱うのではなく，本論文で提案した手法よりもより深い意味の操作として自己修復を扱う必要がある．この例であれば，単語間においても単純な置き換えを行うのではなく，意味素性の引き渡しを行わなければならない．

●より高度な意味処理が必要なタイプ

これには 9 つの例が含まれる．上に述べたタイプも，本手法よりも高度な意味処理を要求するものであるが，上のタイプはまだ比較的簡単な問題である．それよりも，ここに分類されるものは更に複雑な意味処理を要求するものである．

「それをガンダムの前に押して，前の辺りに」

「赤い玉押して，白いのの結構手前まで，1 マス位手前まで」

「白いのが入るぐらいに映して，白いのを映して」

この 3 つの例は，下に行くほど難しいと思われる例である．どの例も，単純な字面や意味カテゴリでの比較では，同じようなことを言い直しているのだとは判らない．

●主辞の省略補完を行わないと解決できないタイプ

このタイプのものは 2 つあった．2 つとも下の例のように「〜から見て」という句が動詞の後から付け足されている例である．

「右に押して，カメラから見て（右に）」

「カメラから見て」は「押して」に係るわけではないので，本論文で提案した構造非隣接に対する処理手法では解決できない．これを解決するためには「右に」あるいは「右に押して」までを，何らかの推論によって補完して考える必要がある．

●見かけは普通の言い直しだが，単純な言い直しとしては解決できないタイプ

このタイプのものは 1 つ見つかった．

「カメラもうちょっと右から映してくれる，右に回り込んでくれる」

この例の場合，一見動詞句の非明示的な言い直しのように見えるが，単純に「映して」を「回り込んで」で置き換えてしまうことはできない．ここでは，話者は「右から映す」ための手段として「右に回り込む」ことを依頼しているのであって，回り込みながら「(何かを）撮影する」ことが重要なのである．このような発話を正しく理解するためには，より統一的な，談話解析までも構文解析と並列化した仕組みが必要である．そして，外界の状況やユーザの意図に応じた処理を行わなければならない．そのようなモデルは 8) などで提案されており，本論文もそのような方向に発展させたいと考えている．

6. おわりに

本論文では，表層のマッチングに加え意味的なマッチングも用いた自己修復表現の処理手法を提案し，これまでに提案されていた手法よりも多くの自己修復表現を扱えることを示した．今後は今回提案した手法を

☆ 7) ではこのタイプは「X[RPD] の単語列が，Z[RP] の単語列の部分列になっている場合」と定義されている（[ ] 内は著者の注）.「二十分愛甲石田まで二十分もかからないから」という発話が例として挙げられている．

システムに実装し，提案手法の有効性を定量的に評価したいと考えている．しかし，音声対話システムの性能を精度や正解率などで測ることは難しい．そこで 8) のように，本研究で開発した手法をシステムに組み込んだ場合と，組み込まない場合とでの，ユーザのタスク達成までの時間や必要とされた発話の回数などを測定することで，性能の評価をする予定である．

参 考 文 献

1) Bear, J., Dowding, J. and Shriberg, E.: Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog, Proc. of 30th Annual Meeting of ACL, pp.56-63 (1992)
2) Core, M.G. and Schubert, L.K: A syntactic framework for speech repairs and other disruptions, Proc. of 37 th Annual Meeting of ACL, pp.413-420 (1999)
3) 伝 康治: 統一モデルに基づく話し言葉の解析, 自然言語処理, Vol.4, No.1, pp.23-40 (1997)
4) Hindle, D.: Deterministic parsing of syntactic non-fluencies, Proc. of 21st Annual Meeting of ACL, pp.123-128 (1983)
5) 伊藤 雅弘, 佐川 雄二, 大西 昇: オンライン言語処理モデルにおける自己修復文の解析手法, 情報処理学会研究報告, 96-SLP-10, pp.87-92 (1996)
6) 川森 雅仁, 島津 明: 話し言葉における冗長表現の解釈, 情報処理学会研究報告, 96-SLP-14, pp.31-38 (1996)
7) 中野 幹生, 島津 明: 言い直しを含む発話の解析, 情報処理学会論文誌, Vol.39, No.6, pp.1935-1943 (1998)
8) Nakano, M. et al.: Understanding unsegmented user utterances in real-time spoken dialogue systems, Proc. of 37th Annual Meeting of ACL, pp.200-207 (1999)
9) Nakatani, C and Hirshberg, J.: A speech-first model for repair identification and correction, Proc. of 31st Annual Meeting of ACL, pp.46-53 (1993)
10) ソフトウェアロボットとの疑似対話コーパス (http://tanaka-www.cs.titech.ac.jp/pub/qdc/)