音声対話システムにおける日本語自己修復の処理

船越 孝太郎[†]　　徳永 健伸[†]　　田中 穂積[†]

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，提案手法を音声対話コーパスに適用した結果を基に，提案手法の有効性と問題点について考察する．

キーワード:　不適格性，自己修復，言い直し，話し言葉の解析，音声対話システム

Processing Japanese Self-correction in Speech Dialog Systems

Kotaro Funakoshi[†]　Tokunaga Takenobu [†] and Tanaka Hozumi[†]

Speech dialog systems need to deal with various kinds of ill-formed speech inputs that appear in natural human-human dialog. Self-correction (or repair) is a particularly problematic phenomenon. Although many methods of dealing with self-correction have been proposed, they have limitations in both detecting and correcting this phenomenon. In this paper, we propose a new method overcoming ill-formedness of speech inputs. We evaluate the proposed method using a speech dialog corpus and discuss its effectiveness and limitation.

KeyWords:　*Ill-formedness, Self-correction, Repair, Spoken language analysis, Speech Dialog System*

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 (山本, 小林, 中川 1992; 伝 1997).

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

[†] 東京工業大学大学院 情報理工学研究科 計算工学専攻
Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

るものでは，(Hindle 1983; Bear,Downing and Shriberg 1992; O'Shaughnessy 1992; Nakatani and Hirschberg 1993; Heeman and Allen 1997; Core and Schubert 1999) などがあり，日本語に関するものでは，(佐川, 大西, 杉江 1994; Kikui and Morimoto 1994; 伊藤, 佐川, 大西 1996; 伝 1997; 中野, 島津 1998; Heeman and Loken-Kim 1999) 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十分な点があり，ソフトウェアロボットとの疑似対話コーパス (東京工業大学田中・徳永研究室 2001) に見られるような表現をカバーできない．また，自己修復を検出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，日本語の不適格性，特に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を捉える事ができるように自己修復のモデルを拡張する．そして，表層及び意味レベルでのマッチングを用いた自己修復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，本論文で提案する自己修復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に対して適用した結果に基づいて考察する．

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，構文解析の枠組みの中で処理をする．

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 (川森, 島津 1996)．本論文では，同一話者による冗長表現のみを考慮する．

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同一話者による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として扱う．自己修復は，(Hindle 1983) と同様に，パーザに処理機構を埋め込むことで対処する．

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．言い淀みは，自己修復と共に現れる場合が多いため，既存の研究では自己修復の枠組みの中に取り込んでいる．これらの研究では，単語断片が単語断片として正確に認識されることを仮定しているが，現在の音声認識技術では難しい仮定である．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不要語処理に委ねる．これは，解析の途中で不要語の可能性がある語を読み飛ばすことで行う．ある語が不要語であるかどうかを局所的に見極めることは難しいので，それを不要語と見なした場合と，そうでない場合と，2 つの仮説をパーザは保持する．そしてパーザは，できるだけ多くの入力語を用いる仮説を優先することで，不必要に読み飛ばしを行うことを避ける．

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

3 パーザ

不適格性を解消するための処理は全て，構文解析と平行してパーザの上で行う．本論文では，音声認識の結果が直接パーザに与えられ，パーザは入力された単語列を不適格性を解消しながら解析し，意味構造に変換して談話処理部に出力することを想定する．発話は，必ずしも音声認識器の出力単位に対応する必要はない．ただし，ここでは，発話の終端はなんらかの手法によって判断できるものと仮定する．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を重視し，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[1]．

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属させる．既に機能語が付属している場合には,「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を行う．係り受け解析はスタックトップの要素のルート語 (rw_2) とトップのすぐ下の要素のルート語 (rw_1) の関係を見て行う．すなわち，rw_1 が rw_2 に係り得るかどうか (もしくはその逆) を調べる．rw_1 が rw_2 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，rw_1 が rw_2 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

[1] 図 2 において四角で囲まれた語.

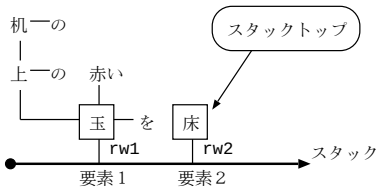

図 2: 「机の上の赤い玉を床」という入力を解析した場合のスタックの一例

```
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

rw_1 が rw_2 に係る仮説 (H1)，もう片方には rw_1 が rw_2 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[2]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．

このパーザは，2 節に述べたように言い淀みを処理するために読み飛ばしも平行して行うが，本論文では詳細は述べない．

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．

ある内容語 c_1 が別の内容語 c_2 に係る時，c_1 は c_2 に対して特定の役割を担っていると考える．「赤い玉」という名詞句であれば，「赤い」は「玉」に対してその色に関する情報を与える役割を持っていると見なす．「馬は前行って」という文であれば，「馬」は「行って」に対して，その動作主を特定する情報を与える役割を持ち，「前」は「行って」に対して，その進行方向を特定する役割を持つと見なす．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

[2] “[” 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．

に記述する[3].

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 c_1 が内容語 c_2 に係ることができるかどうかは，c_1 が c_2 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，倒置は前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，語の意味役割を特定することで，後述する自己修復の意味的な修正処理が可能になる．

4　自己修復の処理

4.1　過去の研究の問題点

我々の疑似対話コーパスの中に現れる自己修復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 2 つある．１つは，自己修復のモデルの問題，もう 1 つは自己修復を検出した後の修正処理の問題である．

まず，自己修復のモデルの問題について説明する．既存の研究では，どれも (Nakatani and Hirschberg 1993) 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

[3] 現在は語彙数が少ないので，活用語は活用形の見出し語として辞書に登録している．

復するものを含む区間を REPAIR(以下 RP) [4]，RPD と RP の間に現れるフィラーや休止，手がかり句 (cue phrase) [5]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．そして，RPD と DF の境界，不適格性が始まる点，を中断点 (interruption site/point) と呼ぶ．例えば，

「[赤い玉を]$_{RPD}$ [えっと]$_{DF}$ [青い玉を]$_{RP}$ 押して」

となる．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仮定する．従って，RIM は日本語に現れる次のような自己修復を扱えない．

「[赤い玉を]$_{RPD}$ 前に押して [えっと]$_{DF}$ [君の前の玉を]$_{RP}$」　(1)

この例では「前に押して」が，RPD と RP の間に入っており，モデルの前提を破っている．

パターンマッチングや統計的言語モデルによる自己修復の検出[6](Bear et al. 1992; Nakatani and Hirschberg 1993; Heeman and Allen 1997; Heeman and Loken-Kim 1999; Spilker et al. 2000) は，人間が 'self-monitoring' によって即座にエラーを訂正する (Levelt 1989) ため，自己修復が局所的であり，かつ RPD の始端から RP の終端までが 3,4 語程度の短いものが大多数であると仮定している．従って，パターンマッチングや統計的言語モデルを用いて (1) のような自己修復を検出することは，原理的には可能でも，精度を悪化させることが予想される．またこれらの手法では，DF の始端を RPD の終端として定義し，検出した RPD は単純に削除してしまうので，(1) のような例を正しく修正できない (この場合,「前に押して」まで削除される).

(Kikui and Morimoto 1994) は (黒橋, 長尾 1992) の並列構造推定手法を用いて RPD の始端を推定しており，他のパターンマッチング手法とやや性質を異にするが，仮に正しく RPD の始点を推定できたとしても，RPD の終端は DF の始端に固定されているので，(1) を正しく修正できない点は変わらない．また，中断点の位置が予め適切に与えられること，2 つの並列構造が完全に単語列の中に含まれていることなどが要求されるので，漸進的な処理には向かず，実用上も疑問が残る．(Spilker et al. 2000) も中断点が音響的に検出されることを前提にしているが，その報告では，中断点の音響的な検出の再現率は 50%に満たない.

(伝 1997) は，自己修復を RPD から RP への係り受け関係として扱うため，(1) に類する自己修復は原理的に扱えない．単一化を基にした句構造規則で自己修復を扱う (中野, 島津 1998) の場合，現在提案されている規則では (1) のような表現を扱うことはできない．新しい規則を追加すれば扱えるかもしれないが，その場合，条件の判定に必要な仕組みなどを新たにパーザに加える必要があり，規則の追加だけで一般的に解決することはできないだろう．(Core and Schubert 1999) も句構造規則で自己修復を扱うが，検出自体は (Heeman and Allen 1997) の手

[4] (Heeman and Allen 1997) では alternation, (Spilker,Klarner and Gorz 2000) では reparans.

[5] 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.

[6] 本論文でいう検出とは，自己修復の範囲の同定までを指す．(Nakatani and Hirschberg 1993; Heeman and Allen 1997; Spilker et al. 2000) などでは，検出 (detection) とは中断点の検出のみを指し，範囲の同定は含まない.

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

法に頼っており，上のような例は扱えない．(佐川他 1994; 伊藤他 1996) も，提案手法のままでは (1) のような表現は扱えない．

次に，自己修復を検出した後の修正処理の問題について説明する．先にも触れたように，従来の手法では，検出した RPD の部分全体を発話から削除する．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]$_{RPD}$ [遠くに押したやつを]$_{RP}$ もってきて」

この自己修復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．(伊藤他 1996) では，動詞句の自己修復の場合に，RPD には含まれるが RP に含まれていない格情報を保存することの必要性について言及しているが，具体的な方法は述べていない．(Core and Schubert 1999) は，RPD を単純に削除してしまうと，RP 内の代名詞が RPD 内の名詞を参照する場合に問題が起きることを指摘しているが，RPD も意味解析モジュールに渡す必要があるというだけで，意味解析モジュールでの具体的な処理法については言及していない．

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 (川森, 島津 1996) 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[7]．

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

[7] (川森, 島津 1996) の分類では,「赤い玉 青い玉」という言い直し表現の二つの「玉」は表面的に離れているために非隣接であると分類される．本論文の自己修復処理手法では，この表面的な隣接／非隣接は問題にはならない．

違った情報は含まれていないものである．図 1 の言い足しと繰り返しに対応する．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．　(2)

言い直しは，“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．図 1 の部分訂正に対応する．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．図 1 の全訂正に対応する．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 (Hindle 1983) と同様に，自己修復個所の検出と修正を，構文解析と平行して行う．ただし，(Hindle 1983) 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。

本手法では，自己修復が検出されるとすぐに修正処理が行われるので，出力を見てもどのような自己修復が存在したのかは判らない．パーザの出力は，修正され冗長性を除去された構文木，もしくは辞書に与えられた役割を基に変換された格フレームの形で出力されるからである．冗長表現はそれ自体が曖昧で，言い直し，リスタート，言い淀みのどれと解釈して処理を行っても得られる結果が同じである場合が多い．結果が同じであるにも関わらず，冗長性の解釈の違いで異なる仮説を保持することは，不必要な曖昧性と計算量の増加を招くだけである．従って本手法では，冗長性の解釈のされかたには関心を持たず，冗長表現の検出と修正処理を同時に行って極力曖昧性を無くす．解析途中で同じ結果をもつ仮説が複数生成された場合は，1 つだ

け残して残りのものは破棄する．

新しい単語が仮説スタック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の検出・修正処理を受ける．自己修復の検出と修正の処理は，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[8]) 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)...

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[9]．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

[8] 構造非隣接の場合，ルート語に係っている部分木のいずれかが RPD である．

[9] 「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 (もちろん語勢やイントネーションの補助もあってである).

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 rw_1，要素 2 のルート語が rw_2 である．
2. rw_1 と rw_2 の品詞が同じ場合 (構造隣接)
   (a) rw_1 と rw_2 が置き換え条件を満たせば，rw_2 は rw_1 の自己修復である可能性があると返して終了．
3. rw_1 と rw_2 の品詞が異なる場合 (構造非隣接)
   (a) rw_1 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) rw_2 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) rw_1 に係っている内容語の中に rw_2 と置き換え条件を満たすもの ($d_i^{rw_1}$ とする) があれば，rw_2 は $d_i^{rw_1}$ 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 rw_2 と $d_i^{rw_1}$ は 4.2 節の (2) 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

- $P_1 = P_2$
- $N_1 = N_2$ and $P_2 \neq$ nil
- $N_1 \sim N_2$ and $P_1 =$ nil

P_1，P_2 はそれぞれ名詞 N_1，N_2 に付いている機能語である．$=$ は全く同じ語であること，$\sim$ は N_1 と N_2 が同じ意味クラスに入ることを意味する．$P_1 =$ nil は，N_1 には機能語が付いていないことを示す．

図 6: 名詞の置き換え条件

文では (A) の解釈を取った．

検出処理　ある仮説スタックが与えられると，検出は図 5 の手順で行う．図 5 の手順の中で使用されている置き換え条件とは，rw_1 あるいは $d_i^{rw_1}$(図 5 参照) と rw_2 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，rw_1(あるいは $d_i^{rw_1}$) は rw_2 で置き換えられる．この置き換え条件は (中野, 島津 1998) の分類 A の (I) [10] とほぼ同じである[11]．置き換え条件の一部 (名詞に対する条件) を図 6 に示す[12]．名詞 N_1 と名詞 N_2 の組に関して，図 6 のどれかを満たせば良い．

[10] 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い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．

[11] (中野, 島津 1998) 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．

[12] これは，(中野, 島津 1998) 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．

修正処理　検出した自己修復には，続いて修正処理を行う．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，rw_1 をルートとする部分木 (t_1 すなわち RPD) が rw_2 をルートとする部分木 (t_2 すなわち RP) で置き換えられるときに，t_1 には存在するが t_2 には存在しない情報を，t_1 から t_2 に移し替える処理を行う．これは，具体的には rw_1 に係っている語の内，t_2 では省略された語を rw_2 に付け替えることで行う．ただし，この時 rw_2 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，

「[ニワトリの前の赤い玉を]$_{t_1}$ [青い玉を]$_{t_2}$ ...」
t_1:　(((ニワトリの)　前の)　(赤い)　玉を)
　　<GEN>　<LOC>　<COL>
t_2:　((青い)　玉を)
　　<COL>

という例の場合 (<GEN>, <LOC>, <COL> などは，そのすぐ上の語にパーザが与えた役割を示す)，t_2 で省略されている “((ニワトリの) 前の)” を，t_2 の「玉を」に付け替える．その結果 t_2' は，

t_2':　(((ニワトリの)　前の)　(青い)　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，

「[馬は前の玉を押して]$_{t_1}$ [青い玉を押して]$_{t_2}$」
t_1:　((馬は)　((前の)　玉を)　押して)
　　<AGNT>　<LOC>　<OBJ>
t_2:　((青い)　玉を)　押して)
　　<COL>　<OBJ>

という自己修復で，t_1 と t_2 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 t_2' は，

t_2':　((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，t_1 の「玉を」と t_2 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．

語 rw_1 をルートとする部分木 t_1 を，語 rw_2 をルートとする部分木 t_2 で置き換えるとする．この時，rw_1 に係っている m 個の語を $d_1^{rw_1}, d_2^{rw_1}, ..., d_m^{rw_1}$ とする．rw_2 に係っている n 個の語を $d_1^{rw_2}, d_2^{rw_2}, ..., d_n^{rw_2}$ とする．この t_1 と t_2 に対して，付け替え処理を行う関数 $dmerge(rw_1, rw_2)$ 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．

仮定 1　$d_1^{rw_1}, ..., d_m^{rw_1}$ の中の任意の語 $d_i^{rw_1}$ と同じ語は $d_1^{rw_2}, ..., d_n^{rw_2}$ の中にただ 1 つ $d_j^{rw_2}$ しかなく，その逆もまた成り立つ時に限り，$d_i^{rw_1}$ と $d_j^{rw_2}$ のそれぞれの rw_1 と rw_2 に対する役割が例え異なろうとも，$d_i^{rw_1}$ は $d_j^{rw_2}$ によって置き換えられたと認識できる．

仮定 2　$d_1^{rw_1}, ..., d_m^{rw_1}$ の中の任意の語 $d_i^{rw_1}$ と同じ役割を持つ語は $d_1^{rw_2}, ..., d_n^{rw_2}$ の中にただ 1 つ $d_j^{rw_2}$ しかなく，その逆もまた成り立つ時に限り，$d_i^{rw_1}$ と $d_j^{rw_2}$ が例え異なる語であろうとも，$d_i^{rw_1}$ は $d_j^{rw_2}$ によって置き換えられたと認識できる．

仮定 3　上の 2 つの認識に食い違いがない場合にのみ，$d_i^{rw_1}$ は $d_j^{rw_2}$ 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，$d_i^{rw_1}$ が語そのものか役割によって $d_j^{rw_2}$ 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，$d_i^{rw_1}$ と $d_j^{rw_2}$ を対応づけ，$dmerge(d_i^{rw_1}, d_j^{rw_2})$ を再帰的に実行する．この条件を満たさない $d_i^{rw_1}$ と $d_j^{rw_2}$ に関しては処理は何も行われずに無視される．これは，RPD(t_1) と RP(t_2) 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．

仮定 1 によって，

「[あれを右から押して]$_{t_1}$ ごめん [右に押して]$_{t_2}$」
t_1:　((あれを)　(右から)　押して)
　　<OBJ>　<FROM>
t_2:　　((右に)　押して)
　　<TO>

という発話から，t_2'' ではなく t_2' を得ることができる．

t_2':　((あれを)　　(右に)　押して)
t_2'':　((あれを)　(右から)　(右に)　押して)

同様に仮定 2 によって,「あれを馬の前に押して ごめん 後ろに押して」という発話から「あれを馬の後ろに押して」という解釈を得ることができる．

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，

「[君赤い玉押して]$_{t_1}$ [大きいやつ押して]$_{t_2}$」
t_1:　((君)　((赤い)　玉)　押して)
　　<AGNT>　<COL>　<OBJ>
t_2:　　((大きい)　やつ)　押して)
　　<SIZE>　<OBJ>

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「玉」と「やつ」の対応を取ることができない．もともと助詞がない場合 (名詞に係る形容詞など) でも，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．

$dmerge(rw_1, rw_2)$: rw_1 をルートとする部分木から rw_2 をルートとする部分木への付け替え処理を行う

1. $d_1^{rw_1}, ..., d_m^{rw_1}$ と $d_1^{rw_2}, ..., d_n^{rw_2}$ の間で同じ語であることを基準に対応を取る (対応 1).
2. $d_1^{rw_1}, ..., d_m^{rw_1}$ と $d_1^{rw_2}, ..., d_n^{rw_2}$ 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 (d_x^{rw1}, d_y^{rw2}) を取り出し，$dmerge$ を再帰的に適用する．d_x^{rw1} が d_y^{rw2} よりも下位の意味クラスに属するならば，d_y^{rw2} を d_x^{rw1} で置き換える．
4. $d_1^{rw_1}, ..., d_m^{rw_1}$ 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を取り出し，それらをルートとする部分木だけを，rw_2 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

また，対応付けた 2 つの内容語のうち，より詳細な情報を持つ下位クラスの語を残すことによって，代名詞による繰り返しなどの場合も情報の損失を防げる (図 7 の 3.の後半)．上の例の場合，下の t_2'' ではなく，t_2' を得られる．

t_2':　((君)　((赤い) (大きい)　玉)　押して)
t_2'':　((君)　((赤い) (大きい)　やつ)　押して)

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．

リスタートの検出には，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 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　提案手法の評価と考察

提案手法を評価するために，提案手法を「ソフトウェアロボットとの疑似対話コーパス (東京工業大学田中・徳永研究室 2001)」(以後，疑似対話コーパス) に人手で適用した．その結果に基づいて，定性的な考察を行う．

5.1　疑似対話コーパスへの提案手法の適用

「ソフトウェアロボットとの疑似対話コーパス」は，全 15 対話，532 発話を含む．疑似対話コーパスは,「仮想世界内にある 2 体のロボットを操作して，同じく仮想世界内にある 4 つの球をあらかじめ指定された位置に配置する」という課題で収集された．収集に参加したのは，1 対話につき 5 人である．5 人の参加者の内訳は,

- ロボットに音声言語で指示を与える者 (指示者)
- 指示者の指示にしたがってロボットを操作する者 (ロボット操作者)
- 仮想世界の様子を映すカメラを操作する者 (カメラ操作者)
- 仮想世界を管理するシステムを担う者 (システム管理者)
- タスクの終了を判断する者 (終了判定者)

である．参加者は対話毎に役割を交替した．指示者はモニタに映されるカメラの映像を通して仮想世界内の様子を知り，球を操作する 2 体のロボットに加えて，カメラの位置も言語によって操作する．また，指示者は自分や球の位置などをシステム管理者に質問することができる．1 対話の収集は，ロボットや球がランダムに配置された状態から始まり，指示者が指定された配置を完了したと終了判定者が判断した所で終る．コーパスの収集は,「発話者が指示を与え，それに対してロボットが動作する」といサイクルの繰り返しによって採集された．また，収集経過は市販のビデオカメラで撮影された．

このコーパスを調べた結果，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 84 個所が冗長表現で，14 個所の言い淀みを除く 70 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．70 個の自己修復を図 4 の分類に従って分類した結果は以下の通りである．

言い足し:	構造隣接 37 個 (52.9%)，構造非隣接 16 個 (22.9%)
言い直し:	明示的構造隣接 1 個 (1.4%)，明示的構造非隣接 0 個 (0%)
	非明示的構造隣接 10 個 (14.3%)，非明示的構造非隣接 0 個 (0%)
リスタート:	明示的 0 個 (0%)，非明示的 1 個 (2.9%)
分類不能:	5 個 (5.7%)

RP が文頭から始まっていて，言い足し／言い直しともリスタートとも解釈できる場合には，言い足し／言い直しとした．分類不能の 5 個は後で説明する挿入表現に該当する．

提案手法をこれらの自己修復に適用した場合，提案手法で解決できるものが 53 個，解決できないものが 17 個であった．疑似対話コーパスの調査，及び手法の適用に際しては,「ねえ」,「ですねえ」,「さあ」,「下さい」等の，間投詞的あるいは定型的な接尾表現は無視した．

コーパスの規模が小さいことと，人手での作業であることから，本論文ではこれ以上の定量的な評価は行わない．かわりにこのコーパスを用いて定性的な考察を行う．

例 (I)
修正前　[もう]$_{RPD}$，[もう]$_{RP}$ 90 度回り込む
修正後　もう 90 度回り込む

例 (II)
修正前　そうじゃなくて [反対側で]$_{RPD}$，[反対側に]$_{RP}$ できるだけ回り込んで
修正後　そうじゃなくて反対側にできるだけ回り込んで

例 (III)
修正前　[ザクは]$_{RPD}$，[ごめんなさい]$_{DF}$，[カメラは]$_{RP}$ 斜め 45 度くらいから映して
修正後　カメラは斜め 45 度くらいから映して

例 (IV)
修正前　[青い玉の近くまで押して]$_{RPD}$，[まっすぐ押して]$_{RP}$
修正後　青い玉の近くまでまっすぐ押して

例 (V)
修正前　どっちか手の空いてる方が，[その赤いやつを]$_{RPD}$ カメラのすぐ手前あたりまで持ってくる，[カメラのすぐ前の赤いやつを]$_{RP}$
修正後　どっちか手の空いている方が，カメラのすぐ前のその赤いやつをカメラのすぐ手前あたりまで持ってくる

例 (VI)
修正前　ザクは，[その青いやつを]$_{RPD1}$，[さっきのやつを]$_{RP1}$，[もうちょっとカメラから見て]$_{RPD2}$，[見て]$_{RP2}$ 右に押す
修正後　ザクは，その青いさっきのやつを，もうちょっとカメラから見て右に押す

例 (VII)
修正前　[[[右の]$_{RPD1}$，[右の]$_{RP1}$ 青を]$_{RPD2}$，[カメラから見て右の青を]$_{RP2}$，白から見て赤の反対側に置いて]$_{RPD3}$，[押して]$_{RP3}$
修正後　カメラから見て右の青を，白から見て赤の反対側に押して

図 8: 処理可能な発話の例

5.2　考察

図 8 に，今回使用した疑似対話コーパス中のデータで，提案手法が有効に動作する例を示す．今回使用した疑似対話コーパスに現れた自己修復を含む発話には，図 8 の例 (I) のような単純なものから，例 (VI)，(VII) のような複数の自己修復を含むものまであった．

例 (V) では構造非隣接の言い足しが起きているが，提案手法により問題なく対処できている．また，例 (IV)，(V)，(VI)，(VII) では修正処理によって RPD から RP へ適切に情報が残されている．例 (V)，(VI) では「その」という連体詞が，RPD にしか現れていない．「その」という連体詞は具体的な情報を与えるものではないが，「その」という語が発話されたということを意味解析以降の処理に伝えることで，対象の特定などにおいてある種の絞り込みを行なうことができる可能性がある．従って，このような語を削除せずに残しておけることにも意義がある．また，本手法は漸進的に処理を行なっていくので，例 (VII) のように自己修復が入れ子になっている場合でも問題なく処理できる．

次に提案手法では対処できなかった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．

● **挿入表現**　このタイプの自己修復は (中野, 島津 1998) では分類 A の (II) [13] に分類され，(田中 1999) では挿入と呼ばれている．これには 5 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 (ここで,「黒」は「黒いブロック」をさしている)．(伝 1997) のように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ことは (中野, 島津 1998) で指摘されている．しかし，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．また，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を別に導入することによって解決することも可能である．(中野, 島津 1998) は，(Heeman and Allen 1997) の様な手法は漸進的な処理に用いられないと述べているが，その理由は不明である．(中野, 島津 1998) は音声認識器の出力を解析することを前提にしており，そうであるならばパターンマッチング手法を併用することも (その結果を必ず信用するかどうか，あるいは修正処理までを行わせるかどうかは別として) 可能なはずである．(中野, 島津 1998) の手法自体も，挿入表現を扱う規則の条件から (脚注 13 参照)，読み飛ばし以上の事はできず，例えば，

「私は [あの分厚い本を]$_X$，[図書館からご注文の本を運んできました]$_Z$」

というような発話は扱えない．

● **単純な置き換えによる情報の損失**　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表現**　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．

提案手法では，情報の保存のために行われる RPD と RP の対応づけが単語のレベルで行われるため，下の例のような場合正しく修正処理を行うことができない．

[13] (中野, 島津 1998) では「X(RPD に相当) の単語列が，Z(RP に相当) の単語列の部分列になっている場合」と定義され (() 内は著者の注),「[二十分]$_X$ [愛甲石田まで二十分もかからない]$_Z$ から」という例が挙げられている．

「それをガンダムの**前に**押して，前の**辺りに**」

この例の場合,「前に」は「辺りに」と対応づけられるために,「ガンダムの」は「辺りに」に付け替えられてしまい，“((それを) ((ガンダムの) (前の) 辺りに) 押して)” という結果が生成されてしまう．この例を正しく解釈するためには,「前に」が「前の辺りに」という複合表現と対応していることを理解できる必要がある．

また，本手法を含めて，表層のレベルで自己修復を扱う既存の手法はどれも，品詞が異なるために次のような簡単な表現も扱う事ができない．

「**赤い**，ごめん，**緑の**玉を押して」

当然,

「**白いのが入るぐらいに**映して，**白いのを**映して」

のような，表層上はかなり異なるが意味的にはほぼ同じと考えられる表現も扱う事ができない．

● **主辞の省略**　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．あるいは，交差する係り受けも許すような仕組みが必要である．

● **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表現**　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，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．

6　おわりに

本論文では，自己修復のモデルを拡張し，従来よりも多くの表現を検出する方法を示した．そして，RPD と RP の間で単語間の対応関係を取ることにより，従来の手法では単純に削除されてしまった情報を，自己修復の冗長性を修正した後にも適切に残すことができる手法を提案した．また，本論文で用いたパーザは，単語の役割を認識することで，日本語に見られる不適

格性の 1 つである助詞落ちを回復することができる．そして，自己修復の修正処理は，パーザによって構文木に付与される単語の役割を利用することで，適切かつ効率的に行なうことができる．提案した検出処理は，日本語，及び導入したパーザに依存するため，その他の言語への直接の応用は難しいかも知れない．しかし，修正処理は，日本語やパーザに関係なく応用が可能である．

本論文では，小規模のコーパスに人手で適用した結果をもとに定性的な考察を行った．今後の課題の 1 つとして，実際の音声対話システム上での利用を視野にいれたより定量的な評価を行う必要がある．

提案手法では，自己修復の検出に構文的な手がかりのみを用いた．構文的な手がかりを用いれば，ほとんどの自己修復の検出は可能である．しかし，非明示的なリスタートはこの限りではない．また，2 節で述べたように，本手法で用いたパーザは，多くの語を含んでいる仮説を優先する．従って，自己修復の可能性を検出することはできても，そこに自己修復の存在を認めない仮説の方が優先されてしまうことがある．例えば,「馬の右の，馬の左の玉を押して」という発話の場合,「馬の右の」を「馬の左の」で言い直しているとする仮説よりも,「馬の右の」が 2 番目の「馬」に係るとする仮説が優先される．この問題を解決するためには，(Bear et al. 1992; O'Shaughnessy 1992; Nakatani and Hirschberg 1993; Heeman and Allen 1997; Spilker et al. 2000) 等のように音響/音韻情報を導入して，自己修復として処理する仮説の方が尤度が高くなるような処理が必要になる．

検出後の修正処理のために提案したアルゴリズム (図 7) 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，このアルゴリズムが前提とする 3 つの仮定 (4.3.1 節) が，広く一般に成り立つものであるかどうかについては，他の分野のコーパスなどを用いて検証する必要がある．

また，本論文では，発話の終端は判るものと仮定して話を進めたが，実際にはこれは大きな問題であり，本論文で用いたパーザが漸進的な構文解析を行うのも，1 つにはこの問題を踏まえてのことである．発話が連続する状態では，自己修復として扱うべきか別々の発話として扱うべきかを決定できる枠組みが必要である．そのような枠組みとの連携も今後の課題である．

自己修復を検出する手がかりとして，編集表現と呼ばれるキーワードが用いたが，これらの表現は必ずしも自己修復だけに用いられるのではなく，通常の否定表現としても用いられる．自己修復とそれらの表現の区別をつけられる仕組みも必要である．

参考文献

Bear, J.,Downing, J.,and Shriberg, E. (1992). "Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog." In *Proceedings of 30th Annual Meeting of ACL*, pp. 56–63.

Core, M. G. and Schubert, L. K. (1999). "A syntactic framework for speech repairs and other

disruptions." In *Proceedings of 37th Annual Meeting of ACL*, pp. 413–420.

伝康治 (1997). "統一モデルに基づく話し言葉の解析." 自然言語処理, **4** (1), 23–40.

Heeman, P. A. and Allen, J. F. (1997). "Intonational Boundaries, Speech Repairs and Discourse Markers: Modeling Spoken Dialog." In *Proceedings of 35th Annual Meeting of ACL*, pp. 254–261.

Heeman, P. A. and Loken-Kim, K. H. (1999). "Detecting and Correcting Speech Repairs in Japanese." In *ICPhS Satellite Meeting on Disfluency in Spontaneous Speech, Berkeley.*

Hindle, D. (1983). "Deterministic parsing of syntactic non-fluencies." In *Proceedigns of 21st Annual Meeting of ACL*, pp. 123–128.

伊藤雅弘, 佐川雄二, 大西昇 (1996). "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**, 87–92.

川森雅仁, 島津明 (1996). "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**, 31–38.

Kikui, G. and Morimoto, T. (1994). "Similarity-based Identification of Repairs in Japanese Spoken Language." In *Proceedings of ICSLP-96*, pp. 915–918.

黒橋禎夫, 長尾眞 (1992). "長い日本語文における並列構造の推定." 情報処理学会論文誌, **33** (8), 1022–1031.

Levelt, W. J. M. (1989). *Speaking.* The MIT Press.

中野幹生, 島津明 (1998). "言い直しを含む発話の解析." 情報処理学会論文誌, **39** (6), 1935–1943.

Nakatani, C. and Hirschberg, J. (1993). "A speech-first model for repair identification and correction." In *Proceedings of 31th Annual Meeting of ACL*, pp. 200–207.

O'Shaughnessy, D. (1992). "Analysis of False Starts in Spontaneous Speech." In *Proceedings of ICSLP-92*, pp. 931–934.

佐川雄二, 大西昇, 杉江昇 (1994). "自己修復を含む日本語不適格文の分析とその計算機による理解手法に関する考察." 情報処理学会論文誌, **35** (1), 46–52.

Spilker, J.,Klarner, M.,and Gorz, G. (2000). "Processing Self-Corrections in a Speech-to-Speech System." In Wahlster, W. (Ed.), *Verbmobil: Fundations of Speech-to-Speech Translation*, pp. 131–140. Springer.

田中穂積 (監修) (1999). 自然言語処理 –基礎と応用–. 電子情報通信学会.

山本幹生, 小林聡, 中川聖一 (1992). "音声対話文における助詞落ち・倒置の分析と解析手法." 情報処理学会論文誌, **33** (11), 1322–1320.

東京工業大学田中・徳永研究室 (2001). "ソフトウェアロボットとの疑似対話コーパス.". **(http://www.cl.cs.titech.ac.jp/pub/qdc/)**.

略歴

船越 孝太郎 (学生会員): 2000 年東京工業大学工学部情報工学科卒業．2002 年同大学院情報理工学研究科計算工学専攻修士課程終了．同年同大学院情

報理工学研究科計算工学専攻博士課程進学，在学中．音声対話に関する研究に従事．情報処理学会，人工知能学会，Association for Computational Linguistics，各会員．

徳永 健伸 (正会員)： 1961 年生．1983 年東京工業大学工学部情報工学科卒業．1985 年同大学院理工学研究科修士課程修了．同年 (株) 三菱総合研究所入社．1986 年東京工業大学大学院博士課程入学．現在, 同大学大学院情報理工学研究科助教授．博士 (工学)．自然言語処理, 計算言語学，情報検索などの研究に従事．情報処理学会, 認知科学会, 人工知能学会, 計量国語学会, Association for Computational Linguistics, ACM SIGIR 各会員．

田中 穂積 (正会員)： 1964 年東京工業大学工学部情報工学科卒業．1966 年同大学院理工学研究科修士課程終了．同年電気試験所 (現電子技術総合研究所) 入所．1980 年東京工業大学助教授．1983 年東京工業大学教授．1983 年東京工業大学教授．現在，同大学大学院情報理工学研究科計算工学専攻教授．博士 (工学)．人工知能，自然言語処理に関する研究に従事．情報処理学会，電子情報通信学会，認知科学会，人工知能学会，計量国語学会，Association for Computational Linguistics，各会員．

(1995 年 5 月 6 日 受付)
(1995 年 7 月 8 日 再受付)
(1995 年 9 月 10 日 採録)