特集 音声情報処理技術の最先端

話し言葉における言い直しの処理

船越 孝太郎
東京工業大学大学院情報理工学研究科
koh@cl.cs.titech.ac.jp

徳永 健伸
東京工業大学大学院情報理工学研究科
take@cl.cs.titech.ac.jp

言い直しは，話し言葉の特徴の1つであり，音声言語処理を困難にする．たとえば，言い直しによって，話速やピッチの変化，単語断片の混入が生じ，正確な音声認識が困難になる．また，文法から逸脱した表現や冗長な表現は，構文・意味解析を困難にする．本稿では，言い直しの言語現象を，その生成メカニズム（言い直し行為）と結果（言い直し表現）の2つの側面から解説する．そして，言い直し表現を実際に処理するための手法を言語処理の観点から紹介し，言い直し処理の現状と将来の展望について述べる．

■言い直しとは

何かを一度口にしてしまえば，もうそれを消しゴムで消すように取り消すことはできない．これは音声言語の宿命であり，紙に書かれる言葉との決定的な違いである．しかし人間は，発話に含まれる誤りなどを，発話を使って修復できる．たとえば，

右に えっと ちがった 左に押して　　(1)

という発話では，誤って発話した「右に」を「左に」で修復している．本稿で取り上げるこのような**言い直し**（self-correction, self-repair, speech-repair）は，上記のような修復行為の一種であり，話し言葉の持つ特徴の1つである．

発話を発話で修復する行為は，修復行為が行われるきっかけを作る開始者と実際に修復発話を発する修復者の違いによって，自己開始の自己修復，他者開始の自己修復，他者開始の他者修復，自己開始の他者修復に区別される[2]．言い直しは，このうちの自己開始の自己修復，その中でも特に修復される部分と修復する部分が同一発話単位の中に存在するものに相当する☆1．自己開始／他者開始といった概念は，会話分析などの分野で主に用いられ，音声言語処理の分野では，単に自己修復と言えば，言い直しのことを指すことが多い．

言い直しの例を，表-1に示す．言い直しの対象は，言い損じなどによって生じた不完全な語，すなわち単語断片（word fragment）や，表現の誤りの修復（error repair）だけに限らず，不足している情報の追加など発話の適切さの修復（appropriateness repair）にも及ぶ．また，中断によって途切れてしまった発話の流れを，語を繰り返すことによって修復したり，発話全体を改めて発話するといった，間違いの有無，情報の過不足などの観点からすれば必ずしも必要のない言い直しも存在する．これらは，聞き手の理解を助けるために行われるものと考えられ，冗長性という話し言葉のもう1つの特徴を生

☆1 発話の単位が何かを一般的に定めることは実は難しく，明確な定義は存在しない．音声言語処理の場面では，取り扱う対象に応じて，ポーズの長さ，話者の交替，などを基準に適宜定めることになるが，一般的にはおよそ書き言葉の「文」に相当するものと認識されている．必然的に，発話単位の認定基準によって，言い直しの定義も微妙に異なってくる．

分　類	表　現
繰り返し（同一語句）	どちらが遠い，遠いですか．
（言いかけた語）	建物の，なん，何階ですか．
言い替え（助詞）	この中で一番遠い店は，を教えて下さい．
（言いかけた語句）	では，丸井の場所をし，教えて下さい．
（語）	第一銀行，あ，第一金庫は，どういけばいいですか．
（文）	カフェはありますか，近くに．駅の近くにカフェはありますか．
挿入　（句）	この中で一番遠い，駅から一番遠い店はどこですか．

表 -1　言い直しの分類と例 [6)]

み出す．繰り返しには，話し続けることで対話の主導権を確保する役割もある．

「言い直し」という用語は，発話中に生じた問題を続く発話で修復する行為（言い直し行為）と，その結果としての発話（言い直し表現）との2つを意味し得るが，以後，特に断らない場合は，修復行為の結果生まれた言い直し表現を指す．ただし，区別が容易な場合には，特に断りなく行為を指すこともある．

■音声言語処理と言い直し

音声言語処理の観点から見たとき，言い直しは次の2点において大きな問題となる．

- 自動音声認識を困難にする．
- 構文解析・意味解析を困難にする．

自動音声認識を困難にする理由

話者が言い直しをするとき，発話速度や声の高さ・大きさといった音響的な特徴が変化する．この変化が，自動音声認識器が持つ音響モデル（音の並びのもっともらしさのモデル）の許容範囲を超えると，正しく音声を認識することができなくなる．

また，言い直しを行うために発話を中断したことで単語断片が生じれば，それも音声認識を阻害する．現行の自動音声認識は，あらかじめ登録された単語しか認識できない．言い直しに伴う，「ええっと」，「じゃなくて」のような表現（編集表現）に関しては，あらかじめ登録しておくことである程度対処できるが，単語断片に対してそのような方法で対処することは困難である．

単語断片に関しても，各単語の最初の2音節程度までが単語断片になりやすいといった傾向を利用して，対象領域で使用される単語のリストから発生する期待値の高い単語断片を抽出することは可能である．しかし，そのような単語断片のリストをそのまま辞書に加えても，言い直しでない部分での誤認識を増加させ，結果として認識能力を下げてしまう．

さらに，現行の音声認識は，言語モデル（単語の並びのもっともらしさのモデル）としてコーパスから抽出した統計値や，人が与えた文法を用いる．これらの統計値や文法には，言い直し表現の混入によって生じる非文法性（同じ助詞の繰り返しなど）が反映されていないので，正しい推定が行えない．

構文解析・意味解析を困難にする理由

仮に言い直しの音声認識に成功したとする．あるいは，人手で書き起こされたテキストを処理することを考える．それでもなお，問題は残る．

前述のとおり，通常の文法は言い直しを含むようには設計されていない．そのため，言い直しが含まれる発話をそのまま構文解析することができない．構文解析ができなければ意味解析もできないため，その先の処理に進むことができない．構文解析を行わず，キーワードや語の意味情報だけから発話の意味を抽出するキーワードスポッティングと呼ばれる手法もあるが，それらの手法で対応できるのは，航空券予約システムのようにほとんど固有名詞や数値表現しか現れない限られた対象領域である．

また，言い直しが発生した場合，言語表現中に不必要な情報が存在することになるので，どの情報が不要で，どの情報が必要なのかを適切に判断しなければならない．

■言い直しのモデル

言い直しを計算機で処理するためには，まずそれをモデル化する必要がある．初めに，言い直し行為のモデル（生成モデル）を説明し，次に言い直し表現のモデルを説明する．

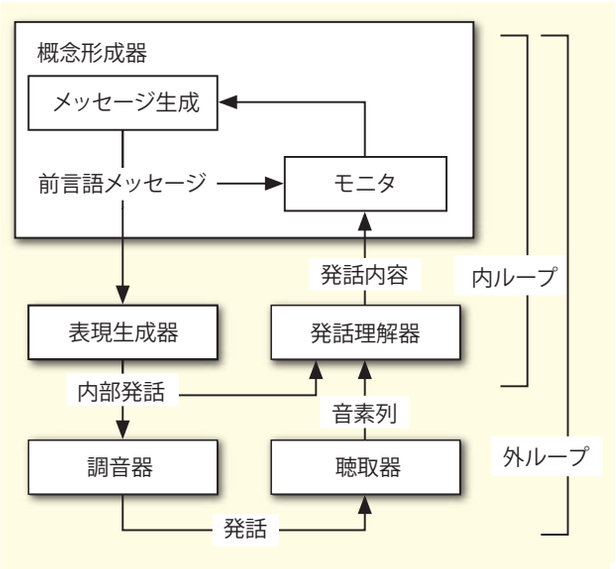

図-1　自己監視の perceptual loop 理論

言い直し行為のモデル

話し手は，自分自身が何をどのように発話したかを常に監視している．これを自己監視（self-monitoring）という．自己監視によって，話し手は自己修復（言い直し）を行うことができる．自己監視のモデルはいくつも存在するが，ここではLeveltのperceptual loop理論を紹介する[3]．

perceptual loop理論では，監視モジュールが生成モジュールの出力を監視することで問題を発見する☆2．ただし，監視機構は，他者の発話を理解する場合と同じように自己の発話を監視するので，監視のために用意しなければならないモジュールを最小限にとどめることができる．perceptual loop理論の構成を図-1に示す．

perceptual loop理論には，問題を発声の前に検出する内ループと，発声後に耳から聞いて検出する外ループとがある．この2つのループの存在によって，環境雑音の大小のために，ある種の誤りの修復される割合が変化することや，修復されるまでの時間が変化することなどが説明できる．

話し手は，問題を検出すると，原則としてすぐに発話を中断し修復する．ただし，発声中の語が誤っているのでない場合は，その語の発声が完了するまで中断が引き延ばされることもある．修復が行われるまでの時間にある程度の幅があり，また修復される場合とされない場合とがあるのは，話し手がどのレベルの問題にどれだけの注意を向けているかが，対話の文脈によって変化するからである．たとえば，公式な場とそうでない場とでは，敬語使用の誤りに対する修復の割合は異なる．

☆2 これはエディタモデルに分類されるモデルの特徴である．コネクショニストモデルに分類されるモデルの場合は，生成と監視はモジュールの区別でなく，情報伝搬の方向の違いで説明される．

言い直し表現のモデル

次に，言い直し表現のモデルを説明する．言い直しの言語処理に関する研究は多く存在するが，基本的にはどれもNakataniとHirschberg[4]がRepair Interval Model（RIM）と呼んでいるモデルを採用している．本稿でも，以降，RIMを中心に説明する．

RIMでは，言い直しを，修復対象区間（reparandum interval），非流暢区間（disfluency interval），修復区間（repair interval）の3つの区間に分け，それらが必ず連続すると仮定する．それぞれを，*RPD*，*DF*，*RP*と略して表現すると，言い直しは，... *RPD DF RP*... と表すことができる．例文（1）の場合は，次のようになる．

［右に］*RPD*［えっと　ちがった］*DF*［左に］*RP*押して

*RPD*の終端は，中断点（interruption site，以後*IS*）と呼ばれ，ここを起点に言い直しの検出と処理を行う．中断点および非流暢区間，修復区間には，話速，声量など音声の特徴に変化が見られることが多い．

例文（2）のように，発話上には*IS*と*DF*だけが現れる潜在的修復（covert repair）と呼ばれる修復は，問題が内ループで検出され，問題にかかわる語をまだ発声していなかった場合に起こる．

第一金庫は　えっと　どこですか？　　（2）

■言い直しの言語処理

ここでは，言い直しの言語処理について概観する．
言い直しの処理には，次の2つの手法が存在する．

（A）構文解析の前処理として行う
（B）構文解析と同時並行に処理する

手法（A）の場合には，音声認識器，あるいは形態素解析器からの出力に対して言い直しの検出を行い，不要な部分を取り除いた結果を構文解析器へ渡す．手法（B）の場合には，構文解析と同時に言い直しの検出と修正を行う．これには，言い直し表現を文法規則として記述し，従来通り構文解析を行う方法と，構文解析器に言い直しを取り扱う特別な仕組みを持たせる方法とがある．

（A），（B）いずれの方法にせよ，次の3つの処理を行

うことになる．

(1)言い直しの存在を検出する
(2) *RPD*，*DF*，*RP*の範囲を特定する
(3)言い直しの修正を行う

(1)，(2) の処理は，手法によって，順番に行われる場合と，同時に行われる場合とがある．

言い直しの検出

言い直しの検出は，前述の中断点 (*IS*) を発見することに相当する．*IS*の周辺には，音声的／言語的な特徴が現れるので，それをもとに検出を行う．

以下に音声情報と言語情報のそれぞれを用いた検出方法を説明する．ただし，実際には両方の情報を同時に利用しないと，正確な検出は難しい．

音声情報を用いた検出

先にも述べたように，*IS*の周辺では音声に特徴的な変化が現れる．この変化を捕まえることで，言い直しの存在を検出できるはずである．しかしながら，人間が聞けばそこに存在することが分かる特徴も，それを機械に認識させるのは難しいのが現状である．

*IS*の直後，すなわち*DF*の先頭部には，200msec前後のポーズが存在することが多い．現在の技術で言い直しの検出に利用できそうな音声情報は，このポーズくらいしかない．しかし，この程度の長さのポーズはさまざまな場所に現れるし，もっと長いポーズを伴う言い直しの数も少なくない．

NakataniとHirschbergは，人間が音声を聞いて各種の特徴をタグ付けしたコーパスを用いて決定木の学習を行い，93.9%の精度と，83.4%の再現率で言い直しを検出することに成功したと報告している[4]．今後音声解析技術が発達すれば，高い精度で言い直しを検出できることが期待できる．

言語情報を用いた検出

*DF*の区間には，「じゃなかった」，「ごめん」といった言い直しを特徴付ける表現 (手がかり句) が現れることが多い．手がかり句は，誤りの修復の場合の方が，適切性の修復の場合よりも現れやすい．また，「えーっと」，「あー」といった有声休止 (フィラー) もよく現れる．*DF*の位置に現れる手がかり句や有声休止を，編集表現 (editing expression) と呼ぶ．しかしながら，言い直しが編集表現を必ず伴うわけではないし，これらの表現が言い直しの場合だけに現れるわけでもない．したがって，これらの表現は言い直しを検出するための手がかりにはなるが，決定的な要素とはならない．

基本的に*RP*は，*RPD*の繰り返し，あるいは*RPD*をいくらか編集した表現であるので，*RPD*と*RP*の間には類似性がある．したがって，そのような類似した区間が並んでいる，あるいはいくつかの編集表現や有声休止を挟んで並んでいる場所をパターンマッチングなどを用いて探すことで，言い直しを検出できる．また，*RPD*と*RP*の間には，一定の規則性が存在し，コーパス上の言い直しを観察すると，それらを抽出できる．たとえば，「右に」を「左に」と言い直すことはあっても，「下から」という表現で言い直すことはほとんどない．したがってこのようなパターンをもう一段階抽象化して利用することで，誤検出を減らすことができる．しかしそれでも，類似性だけでは，誤検出を防ぎきれない．

言い直しの範囲の特定

言い直しの範囲の特定は，上で説明した*RPD*と*RP*間の類似性を用いて行う．*RPD*と*RP*の類似性を利用することで，言い直しの検出とその範囲の特定を同時に行える．

手法 (B) では，類似性に加えて，構文解析と並行して行うことで，文法的な制約を利用できる．例として，

その赤い椅子を右に，青い椅子の右に置いて　(3)

という発話を考える．類似性だけに注目すると，「赤い椅子を右に」という表現を「青い椅子の右に」という表現で修復していると判断してしまいかねない．しかし，文法的な制約を利用できれば，「赤い椅子を」は「置いて」に係る句であり「右に」という部分とは独立していることが分かる．一方「青い椅子の」は「右に」に依存する句であり「右に」とひとまとまりにして扱う必要があることが分かる．したがって，「青い椅子の右に」が修復しているのは「右に」だけであると分かる (**図-2** (a) 参照)．反面，手法 (B) は，

机を右に，赤い机を右に置いて　(4)

といった発話に対応することが難しい．文法的な対応関係，すなわち「机を」と「赤い机を」，「右に」と「右に」の2つの対応関係が交差していて，木構造で表現できる範囲を本質的に超えているからである (**図-2** (b) 参照)．この問題に対処するために，手法 (B) では，交差をカバーする特殊な文法を導入したり，例外的な処理を導入する必要がある．あるいは，(4) のような表現を，発話の先頭から*IS*までの全区間が取り消されるフレッシュスタート (fresh start) として扱うことで，問題を回避す

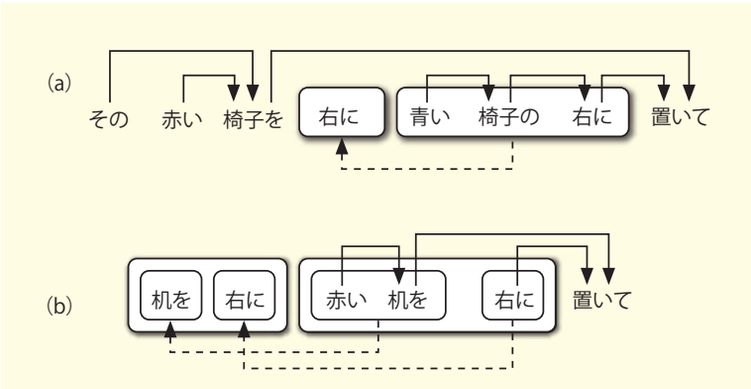

図 -2　構文構造と言い直しの対応関係

ることもある．

一方，手法 (A) では，(4) のような表現には強いが，(3) のような表現には弱い．文法的な制約を利用しにくいという短所を克服するために，言い直しのパターンを確率的に学習することなどで対応する手法も提案されている[5]．

言い直しの修正

言い直しの修正は，範囲を特定した RPD と DF を単純に削除するのが一般的である．しかし，この方法では重要な情報を失ってしまうことがあるので，RPD から必要な情報を取り出す必要があることが指摘されている[1]．これについては，後でもう少し詳しく説明する．

■言い直しの処理手法

実際に言い直しを処理する手法を簡単に紹介する．手法 (A) の例として Spilker らの手法[5]，手法 (B) の例として船越らの手法[1] を取り上げる．

確率的機械翻訳を応用した言い直しの処理

Spilker ら[5] は，確率的機械翻訳の手法を応用して言い直しの範囲を特定する手法を提案している．

確率的機械翻訳は，ある単語の並びが別の言語の単語の並びに変換される確率を，同じ内容を異なる言語で記述したパラレルコーパスから得た統計情報を用いて計算する手法で，同一語族内の 2 言語など，比較的構造の似た言語間の翻訳に用いられる技術である．Spilker らは，修復区間 RP から修復対象区間 RPD への変換を，原言語の単語列 S から目的言語の単語列 T への翻訳という従来の確率的機械翻訳の問題へ対応付けた．IS の直前の語を w_i とすると，RPD と RP はそれぞれ w_{i-s} ... w_i と w_{i+t} ...w_{i+u} になる (s,t,u $>$ 0, t $<$ u)．このとき，RP から RPD への変換確率を最大にする s,t,u を見つけることが，言い直しの区間を特定することに相当する．単語レベルでの RPD と RP の対応を学習しようとすると学習データが不足してしまうため，実際には品詞レベルでの対応と語の意味クラスのレベルでの対応の 2 つを組み合わせ，合計 3 つのレベルでの変換確率を計算し，統合する．

彼らの手法は，まず別に用意された検出モジュールを用いて IS を検出する．報告によれば，その検出能力は 549 個の言い直しに対し，音響情報のみを利用した場合で，再現率 49%，適合率 70%，単語断片が必ず検出できると仮定した場合で，再現率 71%，適合率 85% であった．IS の検出後，上記の確率的機械翻訳の手法を用いて，言い直しの区間を特定する．範囲特定単独の能力は評価されていないが，上の 2 種類の言い直し検出それぞれの場合における範囲特定の能力は，音響情報のみを利用した場合で，再現率 47%，適合率 70%，単語断片が必ず検出できると仮定した場合で，再現率 62%，適合率 83% であった．

係り受け解析器の上での言い直しの処理

船越らの手法では，言い直しを扱うための仕組みをスタックを用いた構文解析器に埋め込んでいる．構文解析は文節単位の漸進的な係り受け解析である．この解析器の上で，言い直しを構文解析と並行して処理する．

彼らの手法は，倒置も許した係り受け解析の過程で言い直しを処理するので，例文 (5) のような倒置と言い直しが複合した表現も扱うことができる．このような表現は RIM では扱えない．

[その本]$_{RPD}$ とって [ ]$_{DF}$ [その黒いやつ]$_{RP}$　(5)

また，彼らの手法は，RPD と RP を融合し，必要な情報を保存する．これが可能なのは，言い直しの処理

を構文解析と並行して行うため，*RPD*と *RP*の中の構文構造を把握できるからである．*RPD*と *RP*の融合は，*RPD*と *RP*の間の構文的な対応関係を求め，*RPD*にあって *RP*にない情報を *RP*へ移し替えることで行う．

これらの処理によって，

うどん いや そばを 食べた 彼と えっと 温かいやつを (6)

という，従来の手法ではうまく扱えない発話から，「彼と温かいそばを食べた」という内容を正しく得ることができる．

図-3に例文 (6) の処理過程を示す．［ 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．

言い直しの検出は，スタックの上2つ（編集表現は除外）の要素 *e1*，*e2*を検査することで行う．*e1*と *e2*があらかじめ定められた条件を満たすとき，言い直しが発生している可能性があると判定する．言い直しの存在はあくまで可能性なので，そこに言い直しを認める仮説と認めない仮説とを作り，別々のスタックに保存する．言い直しの範囲は，言い直しが検出されたときの *e1*と *e2*である．図-3の3.の場合は，*RPD*が「うどん」で，*RP*が「そばを」になる．

この手法を用いることで，助詞落ち，言い直し，倒置などを含む139の発話のうち，106発話（76%）を正しく構文解析できるようになったと報告している．

1. ［(うどん)＞
　　↓
2. ［(うどん)|(いや)＞
　　↓
3. ［(うどん)|(いや)|(そばを)＞
　　↓言い直しを処理：うどん＋そばを→そばを
4. ［(そばを)＞
　　↓
5. ［(そばを)|(食べた)＞
　　↓係り受けを処理：そばを→食べた
6. ［((そばを) 食べた)＞
　　↓
7. ［((そばを) 食べた)|(彼と)＞
　　↓係り受けを処理：彼と→食べた
8. ［((そばを) (彼と) 食べた)|(えっと)|(温かい)＞
　　↓
9. ［((そばを) (彼と) 食べた)|(えっと)|(温かい)|(やつを)＞
　　↓係り受けを処理：温かい→やつを
10. ［((そばを) (彼と) 食べた)|(えっと)|((温かい) やつを)＞
　　↓言い直しを処理：そばを＋温かいやつを→温かいそばを
11. ［(((温かい) そばを) (彼と) 食べた)＞

図-3　解析過程の例

■今後の展望

言い直しは，表層言語的な情報だけでは正しく検出できない．たとえば，“... twenty two twenty one forty ...”という言い直しを含む表現は，音声情報を利用しなければどの位置が正しい *IS*か分からず，可能な解釈（21:40, 22:40）のどれを選べば良いのか分からない[4]．言い直しの処理における，音声情報の利用は必要条件である．

言語処理レベルにおける言い直しの処理は，さまざまな手法が提案され，問題点も克服されてきている．音声認識も，文書読み上げなどの均整な音声に対しては，高い精度で認識できるようになっている．今後は，単語断片の認識など，従来の音声処理技術の中ではまだ未発達な状態にある技術の向上に期待したい．Spilkerらの報告からも分かるように，単語断片が認識できるだけでも，言い直し処理の性能の大幅な向上が期待できる．

また，言語処理レベルの手法においても，拡張性と問題領域からの独立性に優れる確率・統計を用いた手法に，文法的な制約を巧みに取り入れた手法が登場すれば，より高精度な言い直し処理を期待できる．

参考文献
1）船越孝太郎，徳永健伸，田中穂積：音声対話システムにおける日本語自己修復の処理，自然言語処理，Vol.10, No.4, pp.33-53 (2003)．
2）石崎雅人，伝　康晴：談話と対話，東京大学出版会 (2001)．
3）Levelt, W. J. M.: Speaking, The MIT Press (1989)．
4）Nakatani, C. and Hirschberg, J.: A Speech-first Model for Repair Identification and Correction, Proceedings of 31th Annual Meeting of ACL, pp.200-207 (1993)．
5）Spilker, J. and Klarner, M. and Gorz, G.: Processing Self-Corrections in a Speech-to-Speech System, Verbmobil: Fundations of Speech-to-Speech Translation (ed. Wolfgang Wahlster), Springer, pp.131-140 (2000)．
6）田中穂積（編）：自然言語処理－基礎と応用－, 電子情報通信学会 (1999)．
（平成16年7月13日受付）