2-405-57
2017/3/2

推敲過程を用いた文章作成支援システムの提案

松村 竜也[1,a)]　丸山 一貴[1]

概要：文章を作成する際に消してしまった文章を再利用したいと考えたが思い出せず，文章の作成がうまくいかない問題がある．また，これを防ぐために文章をメモ等の形で残しておこうとするあまり，文章の作成に集中できなくなってしまう事もある．本研究では，コンピュータで文章を作成する際に思いついた言い回しを入力しては削除するという推敲動作を行っている事に着目した．その推敲動作をデータとして保存し，そのデータを活用して文章作成を補助するシステムを提案する．このシステムを使用することによって，推敲を積極的に行いやすい環境を用意し，良い文章が書きやすくなる事を目的とする．目的のためにプロトタイプを作成し，実際に文章作成に使うことによって，効果はあるのかを検証した．

Recording Improvement Process of Text Writing Helps Our Writing

Tatsuya Matsumura[1,a)]　Kazutaka Maruyama[1]

Abstract: When we are writing sentences, such as an essay and a research paper, we find some representations for one sentence and choose the one of them. Then the choice would seem to be inappropriate and we would choose another representation to discard the one already chosen. Since the trial and error process, an elaboration, also occurs in writing by editor applications, we try to keep the deleted phrases into the copy-and-paste buffer or some comments of a typesetting system like LaTeX to reuse them. Consequently, the management of the phrases distracts us from the elaboration. Now, we focus on the process of writing and deleting some phrases, and keeping deleted phrases automatically helps us to keep our concentration on the elaboration. In this paper, we propose a system which records the improvement process and provides alternative representations at the correspondent positions. We have implemented a prototype to study the effectiveness of our proposal.

1. はじめに

私たちは，論文を始め様々な文章を日々作成している．文章を作成する時には，同じ意味の違う表現を考えついて，どちらの表現が良いか吟味する推敲が行われる．推敲をする際，文章を一度削除し，その後代わりの表現を当てはめ削除前の文章と比べてどちらが良いかを比べるという一連の動作が行われる．この動作を推敲動作と本研究では定義する．

この推敲動作を行っている最中に，一度文章を消す動作がある．消した際に，短い推敲なら問題ないが，これが大量になってきたり，長い表現を推敲する場合など，覚えておかなければならない表現が多くなった場合，忘れる可能性がある．また，その問題を意識するあまり，表現を覚えていることに集中してしまい，推敲に集中しきれなくなるという問題がある．そこで，推敲動作を読み取り，それを画面に表示してくれるシステムがあれば，この問題が解消されよりよい文章が書けるようになるのではないかと考えた．

この推敲動作は推敲をすると自然に現れる動作である．これを読み取ることによって，利用者に特別な操作を要求すること無く支援が可能である．操作を要求しないことにより，利用者は普段と変わらない操作量で今まで以上の文章を書くことができるようになる．また，特別な操作を必要としない事で，必要だと思わない削除まで読み取ることが出来，不要だと思って消してしまったが，後で必要になったという状況に対応できる．

また，思考の中で起きている推敲をデータの形にした際

[1] 明星大学 情報学部
School of Information Science, Meisei University
[a)] 13j5137@stu.meisei-u.ac.jp

にどのような事になるかということにも着目した．文章を推敲する際には文章中の一つの箇所について複数の候補が脳内に現れる．これを可視化すると，図 1 のような形になると思われる．

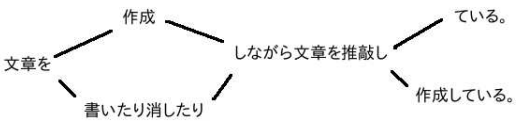

図 1　推敲の概念図
Fig. 1　Conceptual structure of elaboration.

これはグラフの形であり，この形を元にしたデータ形式でファイルを作成すれば，より文章推敲に適した形のデータになるのではないかと考えた．この，推敲をデータ化したものを以降ログファイルと呼ぶ．

本研究の目的は，推敲動作に着目し，これをグラフ状に保存し，利用することによって推敲を支援することは可能なのか検証する事である．

2. 関連研究

2.1 バージョニングシステム

編集を管理する手法としてバージョニングシステムがある．

Git[1] は，おもにソースコードの編集履歴を管理するためのツールである．ソースコードを書いているときにコミットという操作をすることによって，その時のファイルが保存され，コミット毎の差分やコミット時のファイルを，閲覧することができる．また，Git ではブランチモデルを使用しており，これは本研究が提唱するグラフ状のデータ形式と非常に近い．しかし，Git はブランチモデルを編集履歴の管理に使用しており，文章の管理に使用する本研究とは異なる．

2.2 編集単位でのログ保存

生田ら [2] は，論文作成中の削除された文字列を集めて書いている論文以外の研究に活用できないかを探った．本研究でも，利用者の何気ない削除のデータを保持しておき，それを活用するという基本的な思想は同じである．しかし，本研究では削除された文字列を，その文章を作成する事に活用する点で異なる．

中小路ら [3] では，テキスト編集支援環境を提案した．それは，複数候補を同時に画面上に表示することで，変更後の文章の見た目がどうなるのかと言う事が見て取れるというものだ．また，最終的な文章の出来上がりのための表現だけでなく，作りつつある物のための表現が必要である，ということを論じている．本研究も，作りつつある物のための表現を作成している．しかし，推敲動作に着目し，削除操作から推敲を検知する仕組みを採用している点で，本研究と異なる．

AZURITE[4] では，Selective Undo を通して利用者に支援環境を提供している．一般の編集操作である Undo の単位で文章の編集操作とその位置を保存し，それを利用して，指定した編集操作のみの Undo 等に対応している．編集単位での記録やその編集操作同士の比較ができる点で本研究と思想が似ている部分がある．しかし本研究では，推敲を支援することを念頭に置いており，そこからくる記録の提示方法などに差異が現れる．

Microsoft Word は，文書を作成するためのソフトウェアである．このソフトウェアの機能の中に校閲を行う機能が存在する．図 2 は校閲機能を使用して，「本日は晴天なり」という文章を「昨日は曇天なり」に校閲した結果である．これは文章を書いた後に執筆者以外の人がその文章を編集するような操作をすると自動的に編集操作を保存して，最終決定するまで編集操作されたことがわかりやすい表示になる機能である．この機能は，操作を読み取りどちらを選んだ方がよい文章になるか吟味するという点で本研究と同じ支援方法である．しかし，校閲の言葉の通り著者とは別の人が文章をチェックする目的で作られているため，今書き上げている文に対しての推敲と言う点では，一人で使う際にも一つ一つの変更を破棄か確定するかの操作をしなくてはならないなど，普段より多い操作量が必要になってしまう．本研究はできるだけ利用者の別段の操作を必要としないことを目指しているので，操作量と言う点で本研究とは違う．

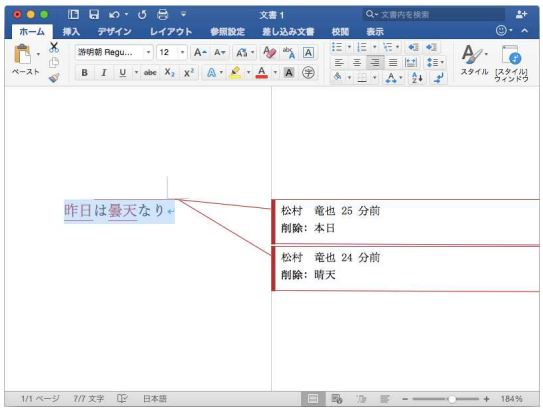

図 2　Microsoft Word の校閲機能
Fig. 2　Review function of Microsoft Word.

3. 提案手法

本研究では，文章の入力と削除を読み取り，データを構築し，それを表示する事によって利用者の支援を行う．まず本システムは図 3 のような単一ウィンドウに二つのパネルからなるテキストエディタのような形をとっている．上部パネルがテキストを入力できるパネルである．利用者は

このパネルを通常のテキストエディタを操作するような形で入力と削除を行う．入力済みの文章の削除を行うと本システム内で推敲をグラフにしたデータが作成され，それを読み取って下部のパネルに推敲過程が表示される．推敲過程の表示では，現在の文章を一番上に表示し，推敲動作を読み取って推敲したと思われる表現を現在の文章の対応した位置の下に縦に並べて表示する．削除された表現を表示する際の位置は，その左端が，本文に含まれていた際の場所に一致している．つまり，表示されている表現の始まりと本文との位置は一致しているが，表現の終わりは本文の位置と一致しているとは限らないということである．また，削除した順番が古いほど上に表示されるようになっている．

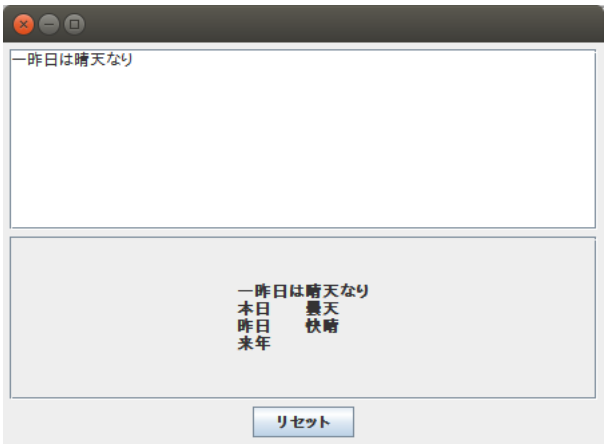

図 3　提案システムを実行した図
Fig. 3　System overview.

例として，図 3 は，以下のように入力や削除を行った例を表している．
(1)「 本日は曇天なり」と入力
(2)「 本日」を削除しその場所に「 昨日」を入力
(3)「 昨日」を削除しその場所に「 来年」を入力
(4)「 来年」を削除しその場所に「 一昨日」を入力
(5)「 曇天」を削除しその場所に「 快晴」を入力
(6)「 快晴」を削除しその場所に「 晴天」を入力

内部では，いくつかの文字を一つの単位としてそれをグラフ状につなげることによって，文章を管理している．そして，現在の実装の場合，1 文字ずつを一つの単位として生成する．

また，保存操作をすることで，現在作成している文章を保存すると同時に (作成文章ファイル).log というファイル名のテキストファイルをログファイルとして保存する．このログファイルは，上述の内部での保存単位のデータを json 形式で記述したものである．

また，リセットボタンを下部に備え付けており，これを実行することで，現在作成している文章と内部で管理している文章を削除し，初期状態に戻す事ができる．

4. 利用例

本論文中の第 1 章の一部を，実際に作成した提案システムで執筆を行った．その結果，様々な知見が得られた．

まず，システムを使うことによりどの表現で迷っているかがわかりやすくなり，推敲がしやすくなった．また，後で使うかもしれないという文章を消す時に，いままでは LaTeX のコメントアウト等を使用して文章を残す作業をする手間をかけていたが，消した文章は残るので，この手間をかけずに文章を作成することに専念できた．

どのような時に推敲が行われているかだが，以下のような内容の時に推敲動作が行われている事が分かった．
(1) 誤字修正
(2) 助詞の推敲
(3) 単語の推敲
(4) 文の推敲

(1) は推敲とは関係ない削除と再入力の操作であり，本来推敲過程の表示に含むべきものではない．それ以外の (2)(3)(4) はいずれも推敲であり，推敲過程の表示はこの三つを表示するべきである．(2) は図 4 のような助詞の推敲である．対象が助詞のため，1 字や 2 字のごく少ない文字数の推敲になる．(3) は図 5 のように一単語でどちらの単語を使うかという推敲である．(4) は図 6 のように，長い文字数での推敲である．これは，表現を変えるよりも大きく文章の意味を吟味するような推敲である．自然と文字数も単語の推敲より大きく，筆者が今回入力した文章の場合は 6 文字より多くなるような傾向が見られた．

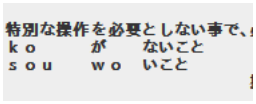

図 4　助詞の推敲の例
Fig. 4　Example of elaborating particles.

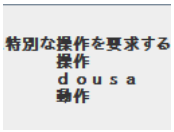

図 5　単語の推敲の例
Fig. 5　Example of elaborating words.

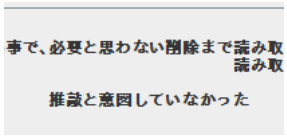

図 6　文の推敲の例
Fig. 6　Example of elaborating sentences.

現在の実装に起因する問題として，以下の二点が挙げられる．第一に，削除始めの位置のみを画面で確認できるようになっており，表示されている表現の後ろが現在書いている文章のどこに対応付けられているのかが分からない．推敲時に表現が長い場合，対応付けが把握し辛く，推敲に支障が出た．第二に，推敲する時下のパネルの表示が上のパネルの文章だと，どこなのかが分かり辛く，推敲に支障が出た．

5. 考察

5.1 改善点

第 4 章で得られた知見より，いくつかの改善点が見つかった．第一に，誤字の修正が表示する表現に入ってきてしまう点である．これにより推敲支援に支障が出てしまうので，これを表示する時点で取り除く必要がある．取り除く方法だが，まず利用者が手動で行なう方法がある．これは，表示している表現に対してクリック等の操作でその表現を非表示にする機能を追加することである．この方法は，利用者の意図が現れるため，推敲したい表現を非表示にしてしまうなどの間違いが無いのが利点である．しかし，欠点としてこの提案システムの特徴である，推敲以外の特別な操作を必要としない事が損なわれる．他の方法として修正された文字数を参照し，閾値以下の場合に非表示にするという手法が考えられる．これは，手動で取り除く方法と比べて利点と欠点が逆であり，推敲以外の特別な操作はいらないが推敲したい表現を非表示にしてしまう可能性がある．推敲の種類で言うと，助詞の推敲などはごく短い文字数になるのでこの方法で非表示になってしまう可能性が高い．どちらの改善が効果的かは複数の利用者と様々な文章による実験により確かめる必要があると考えている．

第二に，表示されている表現の対応付けが分かり辛い問題だが，表示されている表現の対応付けを線で表現するという方法がある．イメージとしては，図 7 のようになる．図 7 を見て分かる通り，本文との対応付けは確認できるようになったが，表示が増えて直感的に理解しづらくなってしまった．実際どの程度効果があるか比較実験をする必要があるだろう．

第三に，下のパネルの文章上の位置が分かり辛い問題だが，これは現在のカーソルの位置を下のパネルで強調表示するなどの改善策が挙げられる．

5.2 さらなる利用例

今回見られた推敲の種類は第 4 章でも挙げた通り，最大でも文単位での変更である．この様な結果になった理由は，今回の利用例では，200 文字程度で区切ってログを出力し，リセットを実行していたためであると考えられる．しかし，今回の利用例では見られなかったが，さらに大きな単位での推敲が考えられる．日常で文章を作成していて

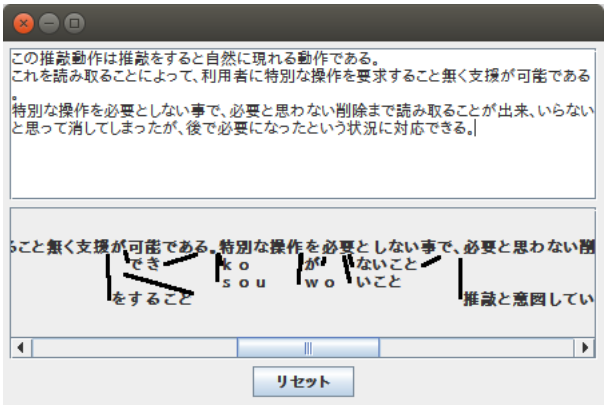

図 7 表現の対応付けの改善案

Fig. 7 Lines to clarify the correspondence between some expressions.

行った推敲例として，文を他の場所に移動させるという例がある．これは例えば，例をいくつか挙げる文章で，例の順番を変えたような時や，文をそのまま他の章に移動したときなどに見られる．これは現在の実装の場合単なる削除や挿入と認識され，削除は推敲履歴としてその場所に表示され，挿入は推敲履歴とは関係なく本文に表示されるだけであり，移動したという認識はされない．しかし，表現のブロックを移動したという認識の方が正しいだろう．システムの表示と利用者の認識を一致させるために，改善を施す必要がある．改善策としては，表現のブロックをドラッグ&ドロップで移動できるようにする，コピー&ペーストの場合は移動処理と考えて処理する，などが考えられる．

また，ログファイルの活用も考えられる．本システムの実装は，利用者の推敲動作から生成したグラフ状の文章データをログファイルとして書き出すことができる．これは，本システムの推敲表示機能などとは切り離されており，単独でデータとして使用できるようになっている．例えば，プログラミングをする際に使用することによって，実装を迷った場所や，どの順番で記述していったのか，などを読み取ることができる．これにより，例えば学校で生徒が実装を迷った部分を本システムで抽出し，教師がどこを指導するかの指針に活かすなどの利用例が思いつく．

6. まとめ

本研究では，文章を推敲する際に現れる動作を読み取り表示することによって，推敲を支援する手法を提案した．提案システムを実際に使用し，推敲支援に活用できる可能性が認められたが，推敲過程の表示に課題がある事を確認した．今後は，今回得られた知見を元に改良を施し，複数人の被験者実験を行い，推敲支援に適した表示方法を探していきたい．

参考文献

[1] Git, Git (online), available from ⟨https://git-scm.com/⟩ (accessed 2016-12-21).
[2] 生田泰章，才記駿平，西本一志：文章作成過程における棄却テキスト 断片の活用に関する一検討，インタラクション 2016 論文集，pp. 302–305 (2016).
[3] 中小路久美代，小田朋宏，山本恭裕：編集中のテキスト 情報を利用した書くことを支援するインタラクション機構のデザイン指針と試作，人工知能学会全国大会論文集，Vol. 26, pp. 1–3 (2012).
[4] Yoon, Y. S. and Myers, B. A.: Supporting selective undo in a code editor, *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, IEEE Press, pp. 223–233 (2015).