Vol.2015-NL-222 No.14
2015/7/16

言い換え技術の文書レイアウト最適化への応用

城戸 祐亮[1,2,a)]　横野 光[2,b)]　Goran Topić[2,c)]　相澤 彰子[2,1,d)]

概要：計算機による言語表現の自動言い換え技術は，自然言語処理の分野で近年盛んに研究されており，文章読解支援や文書要約など多様な用途に応用されている．本稿では言い換え技術の新たな応用として，文書レイアウト最適化問題を取り上げ，その具体的な枠組みを検討する．文書レイアウト最適化問題とは，ワープロ・組版ソフトウェアによるテキストや画像などの文書要素の自動配置に関する問題であり，出版物の製作のみならずパソコン・スマートフォンでの表示など急速に応用範囲が広がっている．従来のレイアウト最適化では文章自体を変更することは想定されていなかったが，提案手法では，自動言い換え技術の適用により文章の長さを変更することで，より柔軟な文書レイアウトの調整を可能にする．また，英語 LaTeX 文書から widow などの問題箇所を自動検出し，ナイーブな言い換えによってそれらを回避するシステムを実装し，提案手法の有効性を示す．

1. はじめに

二つの異なる自然言語表現の意味が近似的に等価であるとき，一方の表現は他方の表現の言い換えであるという [12]．言い換え技術，とくにある表現に対してその言い換えを自動的に生成する言い換え生成技術は，自然言語処理のさまざまな目的に応用される．一つには，人間が文章を読む際にその読解を支援するための補助としての応用がある．特に子どもや外国人 [23]，失語症などの障害がある人 [3], [5] など，文章読解に困難がある人に対する補助を意図した研究が多くなされている．もう一つは，他の自然言語処理のタスクへの前処理としての応用で，入力に対して言い換えを適用することによって複雑な文構造をあらかじめ簡単にしておき，元のタスクの精度を向上することを目指すものである．この応用は広範なトピックにわたり，構文解析 [11]，関係抽出 [17]，意味役割ラベリング [22] などさまざまなタスクに対して研究がなされている．

本稿では言い換え技術の新たな応用として，文書レイアウト最適化問題を取り上げ，その具体的な枠組みを検討する．文書レイアウト最適化問題とは，ワープロ・組版ソフトウェアによるテキストや画像などの文書要素の自動配置に関する問題である．これまでの研究は間隔やサイズの調整・順番などの基本的な調整を対象とするもので，文章の内容を変更することによってレイアウトを調整する研究はなされてこなかった．一方提案手法は，自動言い換え技術の適用により文章の長さを変更することで，より柔軟な文書レイアウトの調整を可能にする．また言い換え技術の応用という観点から見ても，これまでの研究は意味内容を変更することが目的のものであったが，一方本研究は文書のレイアウトという視覚的な要素を考慮に入れて言い換えを行う点で，それらとは異なる．

文書レイアウトに関する問題はフォントサイズの選択や段落などのブロックの配置などさまざまなレイヤにわたる．たとえば改行，改ページ位置の決定などはしばしば最適化問題として論じられ，貪欲法や動的計画法などの最適化手法が適用される [13], [21]．ページ内での文章や図表の配置は，雑誌や書籍などの商業的な出版物の作成において重要な問題であり，自動的に最適なレイアウトを求める手法が研究されている [6], [10]．

さらに文書レイアウトは，近年ますます多様化が進むパソコン，スマートフォン，タブレットなどの各種デバイスでの文書表示においても重要な問題である．Jacobs ら [9] は，一つの文書を多様なディスプレイに適合させて表示することの重要性を述べ，*manifold content* と呼ぶ文書最適化の枠組みを提唱した．これは文書中に含まれる文書，画像などのコンテンツに対してバリエーションを用意することで，実際にディスプレイ上に表示する際にそれぞれのコンテンツのバリエーションから最適なものを選びその環境に適用する内容を動的に生成するものである．Jacobs ら

[1] 東京大学 大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[2] 国立情報学研究所
National Institute of Informatics
[a)] y.k@is.s.u-tokyo.ac.jp
[b)] yokono@nii.ac.jp
[c)] goran_topic@nii.ac.jp
[d)] aizawa@nii.ac.jp

の手法は，レイアウト上の要請にしたがって領域内のコンテンツを入れ替えて表示する点で，本研究で提案する自動言い換えに基づくレイアウト最適化と共通点があるが，manifold content におけるコンテンツのバリエーションは，文書を編集する人間があらかじめデータとして埋め込んでおく必要があり，その生成は人間の手作業に委ねられている点が異なる．

以下，本稿では文書レイアウト最適化への自然言語処理技術の応用の一例として，言い換えによるレイアウト調整について説明する．これは，文書レイアウトにおける調整を，**問題箇所**の解決作業ととらえ，文書から問題箇所を自動検出，それを言い換えによって自動で回避することを目指すものである．この手法は一連の作業を 4 つの段階に分けて考え，それぞれについて具体的な手法を検討する．また，英語 LaTeX 文書から widow などの問題箇所を自動検出し，ナイーブな言い換えによってそれらを回避するシステムを実装し，提案手法の有効性を示す．

2. 手法

提案手法における処理の流れを図 1 に示す．提案手法では，文書を入力として受け取り，文章に言い換えを適用することで，レイアウト上の問題箇所の修正を試みる．今回の実装では特に英語で書かれた LaTeX 文書を対象とする．LaTeX 処理系の実行結果を事前に予測するのは困難であるため，修正においては実際に LaTeX を実行して，先頭から順番に 1 つずつ問題箇所に対応する．これを修正できる問題箇所がなくなるまで繰り返すことで文書全体を最適化する．

提案手法における問題箇所の修正は，(1) 問題箇所の検出，(2) 言い換え候補の生成，(3) 言い換え候補の選別，(4) 言い換え候補の選択と適用，の四つの処理手順にしたがう．以下では，それぞれの処理手順の詳細について説明する．

2.1 問題箇所の検出

まず入力された文書の LaTeX ソース，出力 PDF および LaTeX 処理系の出力ログを走査して，レイアウト上の問題箇所を列挙する．問題箇所がなければ，その時点での出力 PDF ファイルを最終版として出力し，処理を終了する．問題箇所が見つかった場合は，それぞれの問題箇所に対して 1) その問題箇所を修正するために文章中のどの範囲を書き換えればよいか（**対象領域**），2) どれだけの量の変更が必要か（**必要変更量**）の二つの情報を計算し，次の候補選択モジュールへ受け渡す．提案手法では，想定するレイアウト上の問題タイプごとに，問題箇所の検出および上記 2 種類の情報を計算するモジュールをあらかじめ実装しておかなければならない．今回の実験では，**ウィドウ**と**ハイフネーション**の 2 種類の問題タイプを対象とした．

ウィドウとは，文献により定義が分かれるが，ここでは改ページが段落の途中に入ってしまった結果段落の最終行のみが次のページに分かれてしまうもの（その最終行）を指す．ウィドウは一般に英文組版において完成した文書中に含まれてはならないとされているものであり [1]，多くの場合は組版ソフトウェアによって自動で回避される．しかしながらこういった調整は主に一ページに含まれる行数を変えることによって行われ，二ページ以上にわたって連続でウィドウが発生する可能性がある場合には回避できなかったり，またその調整によって不自然な空白がページに出来てしまったりする．一方ハイフネーションは，単語中に改行が入るためにハイフンが語中に挿入されることを指し，特に二行以上にわたって連続で出現することは好ましくない．ここでは連続であるか否かを顧みず，単にハイフネーションの個数が少ないほうが望ましいと近似的に仮定した．これら二つの事象は文書編集では頻繁に見られるものであり，ウィドウは段落単位を対象とする問題の，ハイフネーションは行単位を対象とする問題の，典型例としてここで扱う．

レイアウト上の問題箇所が検出された場合，それを回避するためには大きく分けて二つの戦略が考えられる．ひとつは，その箇所およびその箇所より前の部分のテキストを短くすることで問題の改ページ・改行を後ろにずらす方法，もうひとつはその箇所より前の部分のテキストを長くすることで問題の改ページ・改行を前にずらす方法である．今回は簡単のため前者のみを考慮し，各問題箇所に対して対象領域中の表現を書き換え，計算した必要変更量以上に短くするという戦略をとる．したがって，ウィドウに対しては対象領域はページの先頭から [*1] そのウィドウの出現する位置まですべて，必要変更量はウィドウの行の長さとなり，ハイフネーションに対しては対象領域はその段落の先頭からハイフネーションの行まで，必要変更量はハイフンを挿入された語の後半部分の長さとなる．

2.2 言い換え候補生成

次に，検出されたそれぞれの問題箇所に対して対象領域内で適用可能な言い換えを生成し，そのリストを候補として出力する．これにはどのような既存の言い換え技術でも適用可能であり，今回の実装では二つの単純な手法を利用して生成した候補を合わせた．ひとつは同義語による単語レベルの置換，もうひとつはフレーズレベルの置換である．詳細は 3 章で詳しく説明する．

2.3 言い換え候補フィルタリング

以上の過程で生成された言い換え候補は，可能な書き換

[*1] 実際には文書の先頭からそのウィドウの出現する位置までが対象領域となるが，計算時間の短縮という狙いと，今回の言い換え生成技術の未熟さを考慮して不必要な書き換えをなるべく少なくする狙いから，範囲を狭めた．

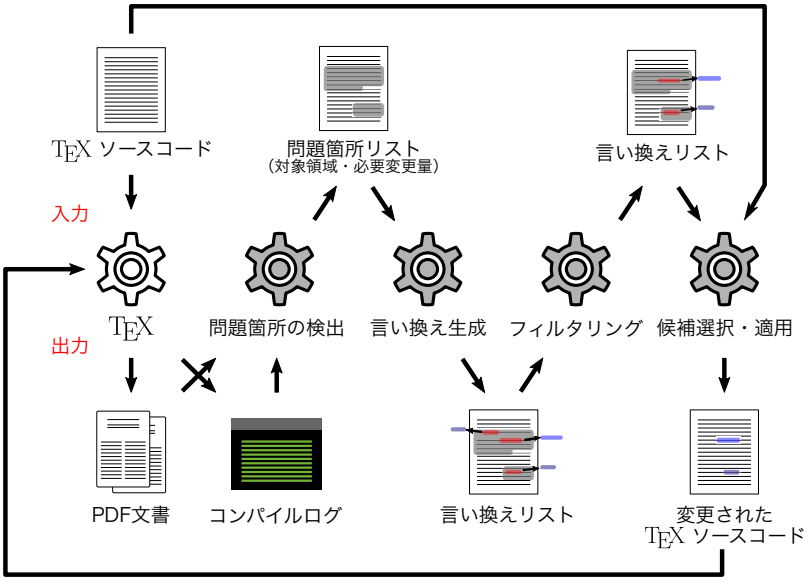


図 1 提案手法の処理の流れを示す概念図.

えの単なる列挙に過ぎず，実際に文書に対して適用するにはさらなる選別が必要である．そこで次に，不適切な候補の除去（フィルタリング）を行う．

まず文法的に適格ではない候補をなるべく取り除くため，3 グラム言語モデルに基づくフィルタリングを適用する．すなわち，書き換え前の文字列とその前後 2 語ずつについて，言語モデルに基づく確率 P_{Before} を計算し，書き換え後の文字列で同様に得られる確率 P_{After} と比較して閾値以上に確率が小さくなってしまうとき，具体的には

$$\frac{P_{\text{After}}}{P_{\text{Before}}} < \alpha \qquad (1)$$

となる場合には，候補が不適格であるとみなす．実験では，Google Ngram Corpus [14] を N グラム確率計算用のコーパスとして利用し，実験的に $\alpha = 0.1$ とした．

くわえて，書き換えを適用することによって表現の意味内容が変わってしまうことも問題となる．そこで，書き換える前の語と書き換えた後の語のそれぞれのベクトル $\mathbf{w}_{\text{Before}}, \mathbf{w}_{\text{After}}$ を計算し，コサイン尺度により計算した両者の意味的類似度が閾値を下回るとき，具体的には

$$\frac{\mathbf{w}_{\text{Before}} \cdot \mathbf{w}_{\text{After}}}{|\mathbf{w}_{\text{Before}}|\,|\mathbf{w}_{\text{After}}|} < \beta \qquad (2)$$

となるとき，書き換えによって大幅に意味が変更されているとみなして，この候補を削除する．ベクトルの計算には word2vec [16] を用い，同梱のモデル (`demo-word.sh` により生成される `vectors.bin`) とデフォルトのパラメタ設定を使った．β の値は実験的に 0.1 とした．

2.4 候補選択

最後に，フィルタリングの結果残った言い換え候補群からいくつかを選択し，実際に文書に適用する．すなわち，いまある一つの問題箇所が検出されており，それに対してそれを修正しうるような言い換え候補がいくつか提示されている．しかしながら，問題設定から，不必要な編集を行うことは望ましくなく，最小限の候補を採用することで問題箇所を修正したい．本手法では，ここで問題箇所を修正するために必要な変更量と，それぞれの言い換え候補がもたらす変更量が分かっていることから，この問題はナップザック問題に帰着できると考え最適解を求める．このとき，採用する候補の組み合わせの数は指数的に増大し，言い換え生成が提示する候補が増えれば増えるほど計算時間が増大してしまう．今回の実装では，変更量に関する貪欲法による解法を実装し，もしすべての候補を採用しても問題箇所が修正できない場合には，余計な編集によって内容を壊すことを避けるためその問題箇所について何もしないこととした．また，一つの問題箇所を修正した場合は，その変更による他の箇所への影響を再計算するため，再度 TEX により文書をコンパイルする．この処理は当然時間がかかる部分であり，一連の処理を文書の先頭から末尾に向かっての順序で処理することで可能な限りこの再コンパイルを減らすことを狙う．

3. 言い換え候補生成

続いて，今回の実装で用いた言い換え候補生成の手法について説明する．前述したように，言い換え候補の生成には任意の手法を用いることができるが，本研究では言い換えの目的がテキスト長の比較的軽微な調整に限定されること，レイアウト最適化の自由度を高めるためにはなるべく多くの言い換え候補を数え上げる必要があることから，ここでは辞書や単純なパターンに基づく単語およびフレーズレベルの言い換え手法を実装した．

3.1 単語レベルの言い換え生成

対象領域中にあるすべての単語について，辞書に登録された同義語を参照し，置換によって長さが短くなるようなものを全列挙する．ここでは同義語辞書として PPDB [7] を用いる．PPDB は複数のコーパスを統合して作られた言い換えのデータベースであり，登録された言い換え対は種類ごとに分類されている．今回はこのうち「語彙的言い換え (Lexical)」で，言い換えによって文字数が変化する 66,557 対を用いた．PPDB の言い換え対にはそれぞれ POS タグが付けられており，本手法でも POS タグまで含めて一致する場合にのみ言い換えの候補に含めた．

3.2 フレーズレベルの言い換え生成

第二に，フレーズレベルの言い換えとして，新たに辞書資源を構築した．これは，Wikipedia の編集履歴を分析した知見に基づいて作られたものである．いわば辞書的に構築された PPDB とは違い，実際に文書を校正する過程を参考にしたものであり，より人間の編集作業に近い言い換えを適用できると期待できる．

Wikipedia を言語資源として活用するというアイデアは広く知られたものであり，多くの研究があるが，それらは大きく二種類に分けられる．ひとつは Simple English 版 Wikipedia を用いるものである．Simple English 版 Wikipedia とは，一般的に話されている英語の文法をもとに語彙・文法構造を簡略化して作られたベーシック・イングリッシュ [19] で記述される Wikipedia であり，もともと英語版にあったものを移植・簡略化して作られた記事も多く含まれる．したがって，Simple English 版 Wikipedia と英語版 Wikipedia の対応する記事を比較し，機械翻訳の枠組で簡略化を学習する研究がされている [4], [8], [24], [26]．もう一つは，Wikipedia の編集履歴を用いるものである．Wikipedia はすべての記事・メタデータなどを容易に利用可能な形で配布しており，とくに，これらの記事のデータにはその記事が初めて作成された時点からのすべての版が含まれる．これを使って，各版を比較することで一回一回の「編集」を取り出すことができ，これから文章校正や文法誤り訂正などのための学習をする試みがなされている [15], [18], [25]．本研究においては，英語版 Wikipedia の編集履歴について検討する．

まず，英語版 Wikipedia の 2014 年 10 月 18 日時点でのダンプデータをすべてダウンロードし，2010 年以降になされた編集を全件抽出した．これらの編集には，単に情報を付け足すための加筆，誤字の修正，いたずら目的，あるいはそれを取り除くためといったさまざまな目的でなされたものが含まれている．本研究にとってはこういった目的の編集は利用価値が小さく，内容を大きく変更することなく読みやすさ・見た目を改善する目的でなされたものだけを選び出す必要がある．そのために以下の基準で編集を選別した．

(1) 編集者によって「細部の編集 (minor edit)」であるとマークされていること．Wikipedia の記事を編集する際には，編集者は重大な加筆・訂正でないことを他の編集者に伝えるために「細部の編集」であることを宣言できる．意味の変更を伴わない単純な言い換えはこの「細部の編集」に該当することが普通であると考えられる．
(2) 変更量が少ないこと．一度にたくさん変更された場合，情報の追加・削除がされている可能性が高い．具体的には，編集前と編集後を比較したときに Levenshtein 距離が 30 未満であるとき変更量が少ないとした．
(3) 編集者のコメント欄に「shorten」という語が含まれていること．編集者のコメント欄は Wikipedia の編集の分類に多くの情報をもたらす [18], [25] ．コメント欄に「shorten」という語が含まれている場合編集の意図が表現の短縮であった可能性が高い．

こうした選別の結果，全部で 1,983 記事に対する 2,234 件の編集が得られた．これらの編集のうち無作為抽出した 503 件について目視で確認をし，以下の 6 種類に分類した．

(1) 150 件 (29.8%) はウィキ構文に関する編集を含むものなどで，言い換えには含まれないものであった．
(2) 136 件 (27.0%) は図表のキャプションや章のタイトルなどの本文以外の領域における編集であった．こういった部分では，冠詞が省略されたり完全な文ではなく名詞句になっていたりと，普通の文と異なる表現が使われることも多く，今回の目的には適さない．
(3) 18 件 (3.6%) は確かに言い換えであったが，周辺の文脈に応じて適用可能かどうかが分かれ，その判断により深い言語理解を要求されるものであった．今回の実験では用いないこととした．
(4) 69 件 (13.7%) は略語・略記の導入．
(5) 52 件 (10.3%) は冗長な表現を簡潔にする編集．
(6) 24 件 (4.8%) は「A of B」の形の句を「B's A」に置き換えるもの．

この分析に基づいて，今回の目的に不適切な (1) 〜 (3) に分類される候補を除いて書き換え規則のデータベースを構築した．このデータベースは 142 対の言い換えを含むものとなった．例を表 1 に示す．

3.3 予備実験

これらの二つの言い換え生成手法の効果を確かめるため，実際の科学論文を用いて予備実験を行った．実験に用いる TEX ファイルとして arXiv Bulk Data Access から得た英語論文のソースを用意した．これらのファイルから本文のみをプレーンテキストで抽出し，実装した言い換え生成モジュールを使ってどれだけの量の言い換えを提示できるかを調査した．論文は物性物理学や天文学など 4 分野

表 1 フレーズレベルの言い換えの例.

言い換え前		言い換え後
1992	→	'92
in the year 2007	→	in 2007
second edition	→	2nd edition
as well as	→	and
in order to	→	to
has a slower speed	→	is slower
make use of	→	utilize
is as follows	→	follows

から計 100 本で，平均で 70.1 段落 376,211 単語の長さであった．結果，単語ベースの言い換えでは論文 1 本あたり平均 1,380 箇所，フレーズベースの言い換えでは平均 15.7 箇所の言い換えを提示できることがわかった．語数あたりでは，単語ベースで平均 2.72 単語につき 1 単語，フレーズベースで平均 239 単語につき 1 箇所となる．

4. 実験

本手法の有効性を検証するため，すでに述べたように実験システムを構築し，実際に LaTeX 文書に対して適用する実験を行った．対象文書として，英語による科学論文を想定したテスト文書を用意した．実際の科学論文のソースとしてウェブ上で配布される自由に利用可能なもので，手元の環境で問題なくコンパイルできるものが存在しないため [*2]，論文を模したテンプレートを用意し，それに文章を流しこむことでテスト文書とした．また実世界の論文で多く使われるレイアウトである一段組み（シングルカラムレイアウト）と二段組み（ダブルカラムレイアウト）の二種類のサンプルを用意した．今回は対象とする問題箇所としてウィドウとハイフネーションの二つを扱うため，本システム適用前と後でそれらの問題箇所の個数を比較する．

表 2 に実験結果を示す．各行がひとつのテスト文書に対応しており，レイアウト（カラム数）とページ数，本手法を適用する前後のハイフネーション・ウィドウの個数が示されている．また問題箇所の個数については，元の文書にはなかったが編集過程で新たに増えてしまった問題箇所を考慮した増減数がかっこ内に示されている．これらの数字から分かる通り，いずれのテスト文書においてもハイフネーションの個数を減らすことが出来ており，ウィドウについても一部は減らすことに成功している．図 2 に実際のテスト文書の例を示す．またこの実験で実際に使われた言い換えの例を表 3 に示す．

5. 考察

実験の結果から，提案手法はウィドウとハイフネーションの個数を減らすという目的に対して有効であると言える．しかしながらすべての問題箇所を解決できたわけではない．すなわち，一部の問題箇所に対しては，我々の用意した言い換え候補生成モジュールが提案する書き換えをすべて適用してもそれを解決できない場合があった．したがって，異なる言い換え技術を組み合わせるなどして生成される言い換え候補自体の個数を増やすことで対処できる問題箇所の数を増やすことが出来ると期待される．また，とくにウィドウについては本手法を適用しても直せない場合が多かったが，今回ウィドウを解消するために文短縮のみを考慮に入れたため，ひとつのウィドウを解消するのにおよそ一行分の長さの短縮を施す必要があり，前述の候補の少なさと相まって効果を小さくしてしまっていることが考えられる．これに対しては，短縮のみならず文の長さを長くする方向での修正も考慮に入れるという改良があり得るが，候補選択における最適化問題の評価はより複雑になり，注意深い設計が必要となる．

また，ある言い換えを適用して文章を短縮しても，期待されたような行数の変動が見られない場合があった．これは TeX が余白を調整するなどして柔軟に配置を制御したため，本手法による書き換えが相殺されてしまったためと考えられる (cf. [2])．つまり，TeX がどのようにレイアウトを決定するかの内部的な実装を考慮に入れて，同じアルゴリズムで言い換え適用後のレイアウトを予測する [20] ことで，無駄な書き換えを減らすという改良が考えられる．

くわえて，意味内容の保持という観点は今回の実験だけでは十分に検証されていない．表 3 に示した例でも，名詞である *interaction* を動詞である *interact* に置換するなど

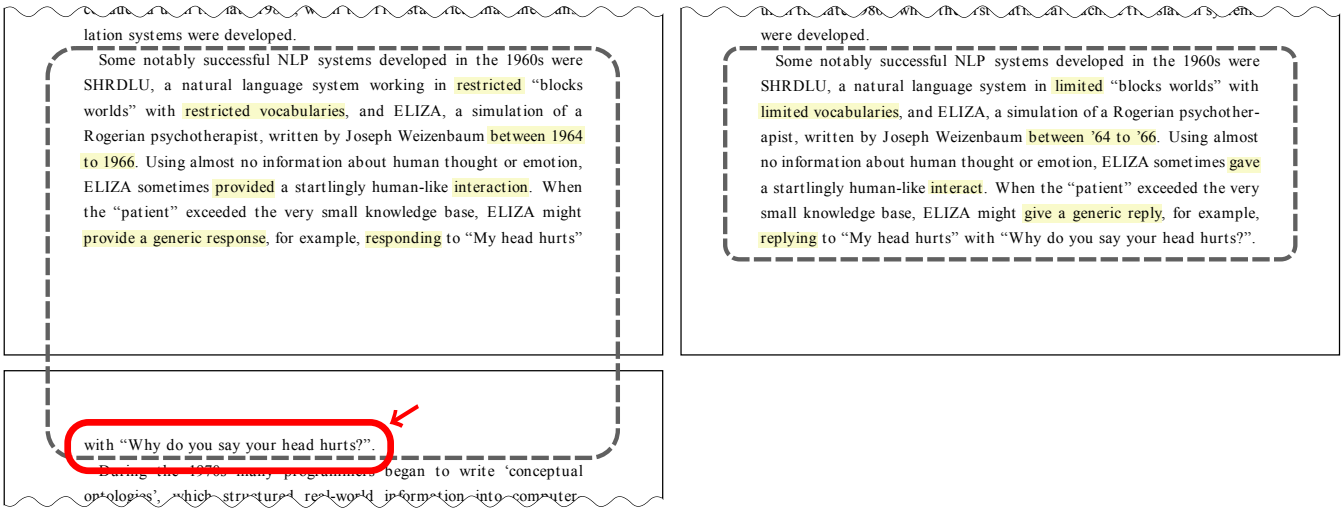
表 2 テスト文書の概要と実験結果．テスト文書の生成に用いた文章は，Project Gutenberg (1, 5, 7) および英語版ウィキペディア（他）から用意した.

	カラム	ページ	ハイフネーション			ウィドウ		
#1	二段	7	31	$\xrightarrow[(+6,-11)]{}$	26	3	$\xrightarrow[(+0,-0)]{}$	3
#2	二段	11	201	$\xrightarrow[(+56,-106)]{}$	151	0	$\xrightarrow[(+0,-0)]{}$	0
#3	二段	11	194	$\xrightarrow[(+50,-71)]{}$	174	2	$\xrightarrow[(+0,-1)]{}$	1
#4	二段	11	144	$\xrightarrow[(+25,-49)]{}$	120	3	$\xrightarrow[(+1,-3)]{}$	1
#5	二段	14	113	$\xrightarrow[(+26,-51)]{}$	88	4	$\xrightarrow[(+4,-4)]{}$	4
#6	一段	12	49	$\xrightarrow[(+20,-33)]{}$	36	3	$\xrightarrow[(+0,-3)]{}$	0
#7	一段	11	39	$\xrightarrow[(+4,-17)]{}$	26	5	$\xrightarrow[(+0,-4)]{}$	1

表 3 図 2 のテスト文書を処理する際に使われた言い換えの例.

言い換え前		言い換え後
restricted vocabularies	→	limited vocabularies
between 1964 to 1966	→	between '64 to '66
provided an interaction	→	gave an interact
responding to	→	replying to
accurate	→	exact
simply by supplying	→	just by supplying
significant increases	→	major increases
learning approaches	→	learning methods

[*2] たとえば予備実験に用いた TeX 文書もスタイルファイルなどの不足からそのままでコンパイルできない.



(a) (b)

図 2 実際のテスト文書に本手法を適用する様子．(a) 段落の末尾の一行が次のページに分かれてしまっている（ウィドウ）．(b) 本手法を適用しいくつかの言い換えによって文章を短くした結果，段落がページ内に収まるようになった．

文法的な妥当性を侵すような書き換えが見られる．現段階でこのシステムによって実世界の文書を編集するためにはまだ不十分であり，実際には人間に判断を任せるいわば文書編集支援のためのシステムとしての利用の方がより妥当である．

また，西暦年の *1992* を *'92* に置換するような言い換えも，ニュース記事を対象とするような場合では適切であるが，科学論文においては不適切となる．したがって，実際に適用する対象の文書に応じて言い換え生成の技術を適切に選択する必要があると言える．

6. おわりに

本稿では言い換え技術の新たな応用先として文書レイアウト最適化を提案した．また具体的な手法として，言い換えによる文書レイアウト最適化を 4 段階に分けてそれぞれについて分析を加えた．さらに実験的ではあるが提案手法を実際の文書に適用するシステムを実装し，評価実験の結果を示した．

謝辞 本研究は JSPS 科学研究費補助金 26540121 の助成を受けたものです．

参考文献

[1] *The Chicago manual of style*, The University of Chicago Press Chicago, 16th edition (2010).
[2] Bazargan, K. and Radhakrishnan, C. V.: Removing Vertical Stretch—Mimicking Traditional Typesetting with TEX, *TUGboat*, Vol. 28, No. 1 (2007).
[3] Carroll, J., Minnen, G., Canning, Y., Devlin, S. and Tait, J.: Practical Simplification of English Newspaper Text to Assist Aphasic Readers, *Proceedings of the AAAI Workshop on Integrating Artificial Intelligence and Assistive Technology* (1998).
[4] Coster, W. and Kauchak, D.: Learning to Simplify Sentences Using Wikipedia, *Workshop on Monolingual Text-To-Text Generation, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics* (2011).
[5] Feng, L.: Text Simplification: A Survey, *The City University of New York, Technical Report* (2008).
[6] Gange, G., Marriott, K. and Stuckey, P.: Optimal Guillotine Layout, *Proceedings of the 2012 ACM Symposium on Document Engineering* (2012).
[7] Ganitkevitch, J., Van Durme, B. and Callison-Burch, C.: PPDB: The Paraphrase Database, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2013).
[8] Horn, C., Manduca, C. and Kauchak, D.: Learning a Lexical Simplifier Using Wikipedia, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2014).
[9] Jacobs, C., Li, W. and Salesin, D.: Adaptive Document Layout via Manifold Content, *Proceedings of the 2nd International Workshop on Web Document Analysis* (2003).
[10] Johari, R., Marks, J., Partovi, A. and Shieber, S.: Automatic Yellow-Pages pagination and layout, *Journal of Heuristics*, Vol. 2, No. 4 (1997).
[11] Jonnalagadda, S., Tari, L., Hakenberg, J., Baral, C. and Gonzalez, G.: Towards Effective Sentence Simplification for Automatic Processing of Biomedical Text, *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers* (2009).
[12] 乾健太郎，藤田　篤：言い換え技術に関する研究動向，自然言語処理，Vol. 11, No. 5, pp. 151–198 (2004).
[13] Knuth, D. E. and Plass, M. F.: Breaking paragraphs into lines, *Software: Practice and Experience*, Vol. 11 (1981).

[14] Lin, Y., Michel, J.-B., Aiden, E. L., Orwant, J., Brockman, W. and Petrov, S.: Syntactic Annotations for the Google Books Ngram Corpus, *Proceedings of the ACL 2012 system demonstrations*, Association for Computational Linguistics, pp. 169–174 (2012).

[15] Max, A. and Wisniewski, G.: Mining Naturally-occurring Corrections and Paraphrases from Wikipedia's Revision History, *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)* (2010).

[16] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, *Computing Research Repository*, Vol. abs/1301.3781 (online), available from ⟨http://arxiv.org/abs/1301.3781⟩ (2013).

[17] Miwa, M., Saetre, R., Miyao, Y. and Tsujii, J.: Entity-focused Sentence Simplification for Relation Extraction, *Proceedings of the 23rd International Conference on Computational Linguistics* (2010).

[18] Nelken, R. and Yamangil, E.: Mining Wikipedia's Article Revision History for Training Computational Linguistics Algorithms, *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence* (2008).

[19] Ogden, C. K.: *Basic English: A General Introduction with Rules and Grammar*, No. 29, K. Paul, Trench, Trubner (1944).

[20] Piccoli, R. and Oliveira, J. B.: Balancing Font Sizes for Flexibility in Automated Document Layout, *Proceedings of the 2013 ACM Symposium on Document Engineering* (2013).

[21] Plass, M. F.: Optimal Pagination Techniques for Automatic Typesetting Systems, PhD Thesis (1981).

[22] Vickrey, D. and Koller, D.: Sentence Simplification for Semantic Role Labeling, *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics* (2008).

[23] Watanabe, W. M., Junior, A. C., Uzêda, V. R., Fortes, R. P. d. M., Pardo, T. A. S. and Aluísio, S. M.: Facilita: Reading Assistance for Low-Literacy Readers, *Proceedings of the 27th ACM International Conference on Design of Communication* (2009).

[24] Wubben, S., van den Bosch, A. and Krahmer, E.: Sentence Simplification by Monolingual Machine Translation, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1* (2012).

[25] Yatskar, M., Pang, B., Danescu-Niculescu-Mizil, C. and Lee, L.: For the Sake of Simplicity: Unsupervised Extraction of Lexical Simplifications from Wikipedia, *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2010).

[26] Zhu, Z., Bernhard, D. and Gurevych, I.: A Monolingual Tree-based Translation Model for Sentence Simplification, *Proceedings of the 23rd International Conference on Computational Linguistics* (2010).

正誤表

5 ページ 左カラム	誤	376,211 単語
	正	**3,762 単語**
6 ページ 図 2 キャプション	誤	実際のテスト文書に本手法を適用する様子． (a) 段落の末尾の一行が次のページに分かれてしまっている（ウィドウ）． (b) 本手法を適用しいくつかの言い換えによって文章を短くした結果，段落がページ内に収まるようになった．
	正	実際のテスト文書に本手法を適用する様子．**文書の内容は英語版Wikipedia「Natural language processing」の記事の本文の一部．**(a) 段落の末尾の一行が次のページに分かれてしまっている（ウィドウ）． (b) 本手法を適用しいくつかの言い換えによって文章を短くした結果，段落がページ内に収まるようになった．