抽出型要約の書き換えによる要約生成

高塚雅人
早稲田大学 理工学術院

小林 哲則
早稲田大学 理工学術院

林 良彦
早稲田大学 理工学術院

takatsuka@pcl.cs.waseda.ac.jp

1 はじめに

自動要約の手法は大きく抽出型要約と生成型要約の二つに分けられるが，近年，その二つの手法を組み合わせた手法 [1][2][3][4] も多く提案されている．抽出型と生成型を組み合わせた既存手法の多くは，抽出型要約モデルを使って原文書の重要な文を抜き出し，それらを生成型要約モデルに入力してパラフレーズや文圧縮を行う．これらの手法では，原文書中の文と人が書いた参照要約文のペアを作成し，生成型要約モデルにそのペアの書き換えを学習させるが，どのようなペアをどのようにして抽出し学習に用いるかが問題となる．既存手法 [1][2][3][4] では，人が作成した参照要約の各文に対して，最も ROUGE-L が高い文をペアとして抽出しているが，表 1 の上部に示すように，対応付けが適切でないペアが抽出されてしまう場合が多く存在し，このような場合，適切な書き換えを学習することは困難である．

そこで本研究では，参照要約の各文とこれに対応する原文書中の各文 (以下，ソース文) に対して編集距離を用いた対応付けにより学習データとして抽出するペアを制限すること，および，学習する生成型要約モデルを単に抽出型要約に対する後処理として用いることを提案する．前者により対応する文がない要約文や高い抽象化を行った要約文を排除し，抽出型要約を書き換えることによる要約生成の精度を向上させることを狙う．また，後者のシステム構成とすることで，抽出型要約の問題点である高い冗長性や代名詞的な共参照の欠損などを後処理によって修正する．

今回，我々が作成するソース文-要約文のペアは，編集距離の意味で類似性が高いペアに限定しているため，通常の機械翻訳のようにソース文-要約文のペアを source-target ペアとして用いると，入力を盲目的にコピーした出力が得られることがある．Dong ら [5] は Simplification Task において同様の問題を指摘し，complex-simple な文のペアを source-target ペアとして暗示的に編集を学習するより，編集操作の系列を target として用いて明示的に編集を学習することで精度が向上したと報告した．本研究では，Dong らの手法を基として，明示的な編集操作の系列を学習する抽出型要約書き換えモデルを提案する．

表 1 ソース文-要約文ペアの例

既存研究 [1]: ROUGE-L Recall: 0.273
ソース文:We reveal how to get the enviable physiques of the stars. 要約文:Her super-sculpted arms are envy of women the world over.
提案手法: 正規化した編集距離: 0.857
ソース文:His performances in such a role, however, have been somewhat mixed. 要約文:Midfielder 's previous performances in defense have been mixed.

代表的な要約のデータセットとして，CNN DailyMail のデータセット [6] を使用して実験を行った．我々の提案手法である抽出型要約に対する後処理を行うことで，後処理を行わない場合より高い ROUGE スコアと人間による評価が得られたことを確認した．

2 関連研究

抽出型と生成型を組み合わせた要約生成手法は多くの既存研究 [1][2][3][4] が存在する．Chen ら [1] は，抽出型要約モデルと生成型要約モデルを個別に学習した後に，生成型要約モデルで書き換えた文の ROUGE スコアを報酬として抽出型要約モデルを強化学習する手法を提案した．Bae ら [4] は，Chen ら [1] の手法では抽出型要約の精度が先頭の三文を抜き出すベースラインより劣っていることと，文レベルの ROUGE の最適化が必ずしも要約全体の ROUGE の最適化に繋がらないことを指摘し，BERT[7] を利用した抽出型要約モデルと，要約全体

の ROUGE を最適化するような強化学習の枠組みを提案した．本研究は，既存研究 [1][2][3][4] のようにソース文-要約文のペアを全て学習に用いるのではなく，うまく対応付けが出来ていないペアや高い抽象化を行っているペアをフィルタリングし，データセットから排除する．

抽出型要約の後処理として，文圧縮や共参照のエラーの修正などを行うことは古くから研究されている．Mendes ら [8] は，各文に対して文圧縮を施した CNN DailyMail データセットとオラクルサマリーのセットを作成し，それらを用いて文の抽出と文圧縮を学習するモデルを提案した．Antunes ら [9] は，共参照解決器を用いて，抽出型要約内の共参照関係のエラーの修正を行い，17 個の文のスコアリング手法と 4 つの抽出型要約システムを用いてその影響を調査した． Antunes らは，共参照のエラーの修正によって要約の一貫性，流暢さ，可読性を向上させることができたと報告した．本研究では，既存研究のように，文圧縮や共参照のエラーの修正をそれぞれ単体で扱うのではなく，抽出型要約に対する後処理をソース文と参照要約文のペアからデータドリブンに学習を行う．

3 提案手法

本研究では，編集距離に基づくソース文-要約文間の対応づけとフィルタリングによるデータセット構築と，明示的な編集操作を学習する抽出型要約の書き換えモデルを提案する．

N 文からなる原文書を $D = [d_1, ..., d_N]$ とし，原文書の i 番目の文を 単語系列 $d_i = [w_1, ..., w_K]$ で表す．また，人が作成した M 文からなる参照要約を $S = [s_1, ..., s_M]$ とし，ソース文-要約文間の編集操作の系列を $[z_1, ..., z_L]$ とおく．

3.1 データセット構築

既存研究 [1][2][4] では ROUGE-L を用いてソース文-要約文間の対応付けを行い，生成型要約モデルを学習している．本研究では，ソース文を書き換えることによって生成することが難しい要約文や，高い抽象化を行っているソース文-要約文ペアをフィルタリングするために，ROUGE-L ではなく，二つの文間の編集度合いを測ることができる編集距離を用いる．本研究では，以下の式で各要約文 s_t に対するソース文 d_{j_t} を選ぶ．

$$j_t = \mathrm{argmin}_i\left(\frac{lev(d_i, s_t)}{|d_i|}\right) \quad (1)$$

$lev(d_i, s_t)$ は (d_i, s_t) 間の編集距離を，$|d_i|$ はソース文 d_i の単語数を表す．その後，正規化した編集距離が閾値以下のペアのみをデータセットに加える．閾値はデータからいくつかの候補を設定し，ベースラインモデルの精度で決定をした．

3.2 提案モデル

本研究で提案するモデルを図 1 に示す．このモデルは，Dong ら [5] のモデルを基に，作成したソース文-要約文のペア間の明示的な編集操作を学習する．Dong ら [5] が提案したモデルは，入力テキストからある単語 w に対する編集操作を予測する programmer と，programmer が過去に予測した編集操作によって部分的に編集された文を入力とし，programmer の予測を補助する interpreter からなる．本研究で提案するモデルも Dong らのモデルと同様に programmer と interpreter から構成されるが，Transformer[10] ベースのモデルを programmer と interpreter に用いる点が異なる．

本研究では編集操作として **[ADD(W)，KEEP，DEL]** を定義する．ADD(W) は単語 W を追加，KEEP は現在の単語を保持，DEL は現在の単語を削除する操作である．作成したソース文-要約文ペアに対して編集操作の系列を生成し，モデルに学習させる．また，各ペアに対する編集操作の系列を一意にするために，KEEP，DEL の前に ADD(W) が来るという制約を加える．

3.2.1 programmer

programmer には，事前学習済みの encoder decoder model の BART[11] を用いる．BART は Transformer[10] ベースのアーキテクチャを採用しており，生成型要約において高い精度を報告している．提案モデルは，推論時に抽出型要約を一文づつ書き換えていくので，過去の文に対してどのような編集を行ったかを知る必要がある．そこで，encoder には原文書に加えて，要約の先行文脈を入力する．要約の先行文脈について詳しくは 4.2 で述べる．また，encoder は入力された原文書と要約の先行文脈をそのままでは見分けることができないため，encoder の入力ベクトルには，PositionalEncoding に加えて，各文の役割を伝える SentenceID ベクトル

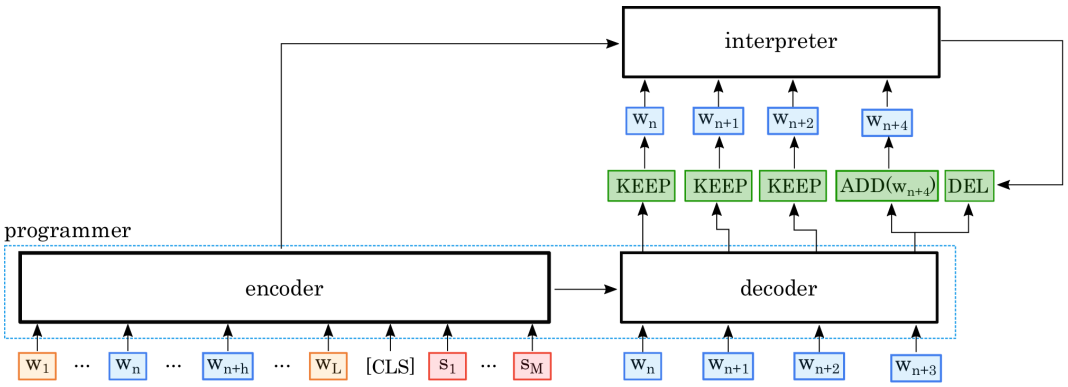

図 1　提案モデルの概要図: 青字は書き換え対象のソース文を，赤字は要約の先行文脈を示す．

を追加する．SentenceID は，書き換え対象のソース文，書き換え対象以外の本文，要約の先行文脈の三つからなる．入力テキストの i 番目の単語ベクトルを e_i とすると,encoder への入力ベクトル x_i^e は以下のように計算される．

$$x_i^e = e_i + PositionalEncoding(i) + SentenceID(i) \quad (2)$$

PositionalEncoding と SentenceID はそれぞれ学習中に更新される．また encoder の出力 h^e は以下のように計算される．

$$h^e = Trans_l^e(h_{l-1}^e) \quad (3)$$

$Trans_l^e$ は l 層目の Transformer 層，h_{l-1}^e は l-1 層目の Transformer 層の出力を示す．

時刻 t において，書き換えを行うソース文の k_t 番目の単語 w_{k_t} に対する編集操作 z_t を予測する場合，decoder には単語系列 $w_{1:k_t}$ を入力する．i 番目の単語ベクトルを e_i とすると，decoder への入力ベクトル x_i^d は以下のように計算される．

$$x_i^d = e_i + PositionalEncoding(i) \quad (4)$$

また decoder の出力 h^d は以下のように計算される．

$$h^d = Trans_l^d(h_{l-1}^d, h^e) \quad (5)$$

$Trans_l^d$ は l 層目の Transformer 層，h_{l-1}^d は l-1 層目の Transformer 層の出力，h^e は encoder の出力を示す．

3.2.2　interpreter

interpreter は，decoder と全く同じアーキテクチャを用いて，decoder と重みを共有する．時刻 t において，過去にモデルが出力した編集操作系列 $z_{1:t-1}$ に従って，ソース文を部分的に編集した単語列 $y_{1:j_{t-1}}$ を interpreter に入力する．interpreter は，decoder と同一の構造なので，interpreter の出力 h^{int} は，式 4, 5 を用いて計算される．

最終的な編集操作のラベルの確率分布 P_{edit} は以下のように計算される．

$$P_{edit} = \text{softmax}(W_{out}(\text{relu}(W_d h_{k_t}^d + W_{int} h_{j_{t-1}}^{int}))) \quad (6)$$

W_{out}, W_d, W_{int} は学習パラメータ，$h_{k_t}^d$ は k_t 番目の単語に対する decoder の出力，$h_{j_{t-1}}^{int}$ は j_{t-1} 番目の単語に対する interpreter の出力である．k_t, j_{t-1} はそれぞれ過去に出力した編集操作系列 $z_{1:t-1}$ から決定される．P_{edit} の次元数は，BART の語彙数 V に対して，KEEP と DEL のラベルを加えた $V+2$ となる．

4　実験

4.1　データセット

本研究では，ニュース記事の要約のデータセットである CNN DailyMail (CNN-DM) のデータセット [6] を用いる． CNN-DM データセットのニュース記事と参照要約をそれぞれ文分割し，3.1 の手法を用いてソース文-要約文のペアを作成した．作成した 1,070,203 ペアのうち，正規化した編集距離の閾値が 0.95 以下のペアのみを採用し，それぞれ学習/validation/テストデータとして，240,918/14,814/11,806 ペアを作成した．その後，各ペアに対して編集操作系列を生成した．

4.2　実験設定

本研究では，抽出型要約を入力として書き換えを行う要約生成実験と人手による評価を行った．なお，書き換え処理単体での機能を確認するため，ソース文-要約文ペアのテストデータを用いた事前実験を行っている．その詳細については，付録 A.1 に示す．

抽出型要約を入力として書き換えを行う要約生成実験では，4.1 で作成したデータセットを用いて

提案モデルを学習させ，推論時には，抽出型要約を先頭から一文ずつ書き換えた．学習時に，参照要約の j 番目の文を生成する際は，j-1 番目までの文を要約の先行文脈として使用し，推論時には，過去に書き換えた要約文を先行文脈として使用した．また，抽出型要約として，lead3 と matchsum[12] を実験に用いた．ベースラインモデルとして，提案モデル (programmer のみ) において，編集操作系列ではなく，要約文を target に用いたモデルを使用した．評価指標として ROUGE-1，ROUGE-2，ROUGE-L を使用した．詳しい学習条件は付録 A.2 に示す．

また，抽出型要約とモデルが書き換えた要約を比較した人手による評価を行った．評価基準として要約の冗長性，理解のしやすさ，全体的な評価の三つを設定した．冗長性は，要約内に同じ表現の繰り返しや，無駄な情報が含まれていないかを表す．理解のしやすさは，要約のみを読んだ際に要約が読みやすく，要約の内容を簡単に理解できるかを表す．全体的な評価では，上記の二つの観点を踏まえて，要約として好ましいかどうかを表す．CNN-DM のテストデータから，ランダムに 100 サンプル選び，どちらの要約の方が冗長性が少ないか/理解しやすいか/要約として好ましいかを評価した．また，各指標について，どちらの要約も同じくらい良い/悪い場合は fair を選択する．実験は Amazon MTurk 上で行い，各サンプルについて 5 人のワーカーが評価を行った．各ワーカーはまず参照要約を読み，ニュースの概要を理解したうえで，各要約を読み，評価を行った．

4.3 結果・考察

4.3.1 抽出型要約の書き換えを行う要約生成実験

抽出型要約の書き換えを行った実験結果を表 2 に示す．表 2 の R-1，R-2，R-L は ROUGE-1，ROUGE-2，ROUGE-L を示す．lead3 は，記事の先頭三文を抜き出した抽出型要約の結果，matchsum (ours) は Zhong ら [12] が提案したモデルを再現した場合の結果を示す．lead3+提案モデル，matchsum+ベースライン，matchsum+提案モデルは，それぞれ抽出型要約をモデルに入力した場合の結果を示す．表 2 に示すように，matchsum+提案モデルが最も高い ROUGE スコアとなり，提案モデルで抽出型要約を書き換えることで精度が向上することがわかった．また，lead3 を提案モデルに入力した際もスコアが向上しており，提案モデルは，抽出型モデルに依存しないことが分かる．ベースラインモデルと比較すると，提案モデルは，特に ROUGE-2 が高くなった．要約の生成例は，付録の表 4 に示す．

表 2 抽出型要約の書き換え実験結果

モデル	R-1	R-2	R-L
lead3	39.94	17.47	36.08
lead3+提案モデル	40.44	17.66	36.77
matchsum (ours)	43.96	20.72	39.99
matchsum+ベースライン	44.04	20.74	40.35
matchsum+提案モデル	**44.10**	**20.93**	**40.38**

表 3 人手による評価実験結果

	冗長性	理解しやすさ	全体
matchsum	110	175	201
matchsum+提案モデル	**231****	**225***	**248***
fair	159	100	51

4.3.2 要約の人手評価

要約の人手評価結果を表 3 に示す．抽出型要約 (matchsum) と，それを提案モデルで書き換えた要約を評価に使用した．表の数値は，各評価指標において，matchsum，matchsum+提案モデル，fair の選択肢を選んだ人数を表す．表 3 から分かるように，全ての指標で，提案モデルを用いて抽出型要約を書き換えた要約が高いスコアとなり，抽出型要約を書き換えることで，要約の冗長性が減り，より理解しやすくなり，要約としての全体の評価が向上することが分かる．特に，冗長性の観点でよりスコアが向上した．表 3 の**は，有意水準 1%で，*は，有意水準 5%で有意差があることを示す．

5 おわりに

本研究では，抽出型と生成型要約を組み合わせた手法として，編集距離に基づくソース文-要約文間の対応付けとフィルタリングを行い，抽出型要約の後処理を学習することを提案した．また，作成したソース文-要約文間の明示的な編集操作を学習するモデルを提案した．自動評価と人手による評価の両方において，提案したモデルによって抽出型要約を後処理することの有効性を確認した．今後の方針として，事実関係のミスなどの観点で，生成型要約と提案手法の要約の比較を行っていきたい．また，Martin ら [13] の研究のように，出力をコントロールできるような手法も検討していきたい．

参考文献

[1]Yen Chun Chen and Mohit Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL 2018*, pp. 675–686, 2018.

[2]Edward Moroshko, Guy Feigenblat, Haggai Roitman, and David Konopnicki. An editorial network for enhanced document summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 57–63, 2019.

[3]Liqiang Xiao, Lu Wang, Hao He, and Yaohui Jin. Copy or Rewrite: Hybrid Summarization with Hierarchical Reinforcement Learning. *AAAI 2020*, pp. 9306–9313, 2020.

[4]Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sang goo Lee. Summary level training of sentence rewriting for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 10–20, 2019.

[5]Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *ACL 2019*, pp. 3393–3402, 2019.

[6]Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

[7]Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL 2019*, pp. 4171–4186, 2019.

[8]Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André F.T. Martins, and Shay B. Cohen. Jointly extracting and compressing documents with summary state representations. In *NAACL 2019*, pp. 3955–3966, 2019.

[9]Jamilson Antunes, Rafael Dueire Lins, Rinaldo Lima, Hilário Oliveira, Marcelo Riss, and Steven J Simske. Automatic cohesive summarization with pronominal anaphora resolution. *Computer Speech and Language*, pp. 141–164, 2018.

[10]Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[11]Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL 2020*, pp. 7871–7880, 2020.

[12]Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching. In *ACL 2020*, pp. 6197–6208, 2020.

[13]Louis Martin, Éric Villemonte de la Clergerie, Benoît Sagot, and Antoine Bordes. Controllable sentence simplification. In *LREC 2020*, pp. 4689–4698, 2020.

[14]Chin-Yew Lin. Rouge: A package for auto- matic evaluation of summaries. In *Text summarization branches out: ACL workshop*, 2004.

[15]Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, pp. 401–415, 2016.

表 4　要約の生成例: 赤文字は，モデルが削除した部分，青文字は，モデルが追加した部分，マゼンタは，モデルが置換した部分を表す．

参照要約	Major General James Post III was fired for saying that the retirement of the A-10 Warthog amounted to ‘treason’ The incident added fuel to a controversy over efforts to retire the low-flying, tank-killer plane highly regarded by ground troops. Post said he told the group the Air Force didn’t want to get rid of the plane but needed to because of budget constraints.
抽出型要約	Fired : Major General James Post III was fired on Friday after making a treason comment. An Air Force major general has been formally reprimanded and removed from his job for telling a group of officers that talking to Congress in a bid to block retirement of the A-10 Warthog amounted to ‘treason,’ the Air Force said on Friday.
提案モデル	Major General James Post III was formally reprimanded on Friday after making a treason comment to 300 airmen. He told a group of officers that talking to Congress in a bid to block retirement of the A-10 Warthog amounted to ‘treason’
参照要約	Half of hospitals in England let patients jump queues if they pay for surgery. Charging up to #2,700 for cataract surgery on one eye - treble cost to NHS. Campaigners have accused hospitals of profiting from elderly patients.
抽出型要約	Some are charging up to a ‘shameful’ #2,700 for one eye – treble what it costs the health service – raising suspicions that they are ripping off elderly patients. Cataract treatment is being rationed in England to save money, even though the NHS recently announced it would fund weight-loss surgery for 15,000 obese adults every year at a cost of #6,000 each.
提案モデル	Some hospitals are charging up to #2,700 for one eye – treble what it costs the NHS. Cataract treatment is being rationed in England to save money, even though the NHS recently announced it would fund weight-loss surgery for 15,000 obese adults every year.

A　付録

A.1　ソース文-要約文ペアの書き換え実験

ソース文-要約文ペアの書き換え実験では，ソース文-要約文ペアのテストデータを用いて，書き換え単体の精度を実験した．ソース文をモデルが書き換えた文と，ソース文とペアになっている要約文間で自動評価を行った．encoder に入力する要約の先行文脈は学習時と同じ設定とした．評価指標として，ROUGE-1, ROUGE-2, ROUGE-L[14] と SARI スコア [15] を使用した．

ソース文-要約文ペアの書き換え実験の結果を表 5 に示す．source は，編集前のソース文の結果を示し，ベースラインは，提案モデル (programmer のみ) において，編集操作系列ではなく要約文を target に用いた場合の結果を示す．また表 5 の% unc. は，無編集の割合を示し，R-1，R-2，R-L は ROUGE-1，ROUGE-2，ROUGE-L を示す．表 5 から，ベースライン，提案モデル共に書き換えによって，ROUGE スコアが向上していることがわかる．また，ベースラインと提案モデルの ROUGE スコアに大きな差はなかったが，SARI スコアでは，1.69 ポイントと大きく提案モデルが低い結果となった．これは，提案モデルがベースラインと比べて，単語の削除の予測精度が悪く，DEL のスコアが低くなったからである．また，ベースライン，提案モデルの無編集の割合は，ソース文-要約文間の無編集の割合よりも高くなった．

表 5　ソース文-要約文ペアの書き換え実験結果

モデル	R-1	R-2	R-L	SARI	% unc.
source	69.09	55.09	67.39	-	**5.3**
ベースライン	**72.09**	57.94	**70.24**	**48.66**	15.8
提案モデル	71.92	**57.99**	70.21	46.97	22.5

A.2　学習条件

事前学習済みモデルとして BART-large[11] を用いた．初期学習率は 6e-05 とし，500 step の warmup の後，最終的に 20000 step 学習させた．訓練時には Label Smoothing を導入し，ε =0.1 に設定した．バッチサイズは 16 に設定した．最適化関数には Adam を用いた．