小規模リソースにおける生成型要約のためのスタイル転移

人見 雄太[1]　田口 雄哉[1]　田森 秀明[1]　岡崎 直観[2]　乾 健太郎[3,4]

株式会社朝日新聞社[1]　東京工業大学[2]　東北大学[3]　理化学研究所 AIP センター[4]

{hitomi-y1, taguchi-y2, tamori-h}@asahi.com, okazaki@c.titech.ac.jp,
inui@ecei.tohoku.ac.jp

1　はじめに

Rush ら [1] の研究以降，新聞記事コーパスを用いた生成型要約の研究が盛んに行われている．現在の生成型要約の成功は，Transformer[2] などのエンコーダ・デコーダモデル (EncDec) の発展，および何百万事例という規模の大量の訓練データに支えられている．

ところが，大量の訓練データを利用できるドメインは限られている．新聞記事コーパスに関しては，記事本文と紙面見出しの組のデータは豊富に公開されているが，記事本文と要約の組のデータは入手しづらい．例えば，朝日新聞社が公開している Japanese News Corpus (JNC) は記事の先頭 3 文と見出しの組を 1,828,231 件収録している．一方，朝日新聞社の要約サービスの一つである，ニュース速報配信サービス (ANDES)[1]では 2019 年 6 月時点までに約 3 万件の記事と要約の組みしか蓄積されていない[2]．高々数万件の訓練データを用い要約モデルを学習するだけでは，高品質な要約が得られない．

そこで，本研究では記事本文と見出しの組の大規模な訓練データと，小規模な記事要約のデータを組み合わせて，高品質な要約器を構築する手法を探求する．具体的には，記事からの見出し生成と要約生成を同時に学習することを，生成スタイルの転移と見なし，転移学習により高いパフォーマンスの要約生成器を獲得する．要約生成をスタイル転移として捉えることで，異なる長さの要約を異なる部署・担当者が執筆している場合など，要約の「スタイル」の差も考慮できる．既に公開している JApanese MUlti-Length Summarization Corpus(JAMUL) の拡張版として，朝日新聞社が配信した新幹線向け短文要約を 30,656 件収録した JAMUL 2020 を構築し，見出し生成から要約生成へのスタイル転移の実験を行う．

結果から，短文要約のデータだけでモデルを学習した場合と比較すると，スタイル転移でモデルを学習することで，15.4 ポイントの ROUGE-1 スコアの改善が得られた．また，最善の 1 文をシステム要約としてみなす抽出型のオラクル要約に関して，ROUGE-1 で 4.8 ポイントの改善が見られた．

[1]http://www.asahi.com/information/haishin/service

[2]配信する記事を厳選していることと，人的コストの観点から要約を付与できる数に限りがある．

2　JAMUL 2020

2.1　コーパスについて

今回，人見ら [3] が提案した異なる長さの見出しが 1 記事につき 4 つ付与されたコーパス JAMUL について拡張した．具体的には，朝日新聞社が展開するニュース速報配信サービスから見出しと要約のデータを抽出し，JAMUL 2020 を構築・公開[3]する．

JAMUL 2020 は 2014 年 5 月から 2019 年 6 月までに朝日新聞社が展開する要約サービス ANDES で配信された 30,656 件のデータから構成される．それぞれの記事に対して最大で 5 種類の見出し・要約が付与されている．内訳は，新聞紙面向けの見出し (Print)，朝日新聞デジタル向け見出し (Large)，携帯端末向け見出し (Middle)，電光掲示板向け見出し (Short)，新幹線の電光掲示板などに配信される短文要約 (Sum) である．Print 以外に関しては，表示するデバイスやレイアウトの都合からそれぞれ異なる文字数の上限がある．上限はそれぞれ，Large が 26 文字，Middle が 13 文字，Short が 10 文字，Sum が 50 文字となっている．本研究では，Print・Large・Sum のみを用いる．

2.2　見出しと要約の比較

一般的に，見出しと要約は共に記事の重要な部分の抜粋である．このことから着目する単語や文がある程度共通していることが想定される．そこで JAMUL 2020 における Print と Sum についていくつかの分析を行なった．

まず，後述する検証データ (4.1 節) における Print と Sum でどれだけ単語が重複しているのかを確認していく．具体的には，Print をシステム出力，Sum を参照要約とした際の ROUGE の F1 スコアを求めた．結果は ROUGE-1 が 42.8，ROUGE-2 が 20.8，ROUGE-L が 35.1 となり，ある程度単語が重複していることが伺え

[3]https://cl.asahi.com/api_data/jnc-jamul.html

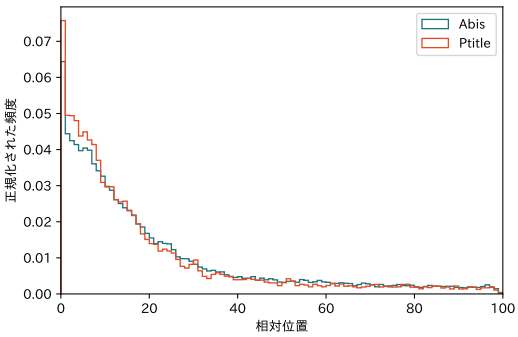

図 1: 正解要約に含まれる bi-gram に関する記事中の相対位置のヒストグラム

記事: サッカーのスペイン 1 部リーグで 13 日、バルセロナの FW リオネル・メッシ (31)=アルゼンチン=が本拠でのエイバル戦で得意の左足でゴールを決め、リーグ史上初の通算 400 ゴールの偉業を成し遂げた...		
Print(19 文字): メッシがスペインリーグ 400 点の金字塔		
Sum(48 文字): サッカー・スペイン 1 部リーグで、バルセロナのメッシが通算 400 ゴール。リーグ史上初の歴代最高記録		
c	ドメイン	入力
タスク	Print	@Print サッカー の スペイン...
	Sum	@Sum サッカー の スペイン ...
長さ	Print	@Len19 サッカー の スペイン...
	Sum	@Len48 サッカー の スペイン...

表 1: コントロールコード (c) を導入した際の入力例.

る．次に，重複している単語の品詞について MeCab[4] を用いて調べていく．頻度順に名詞が 68 %，助詞が 19 %，記号が 6 %となり，それ外については 3 %未満であった．重複する 7 割近くの単語が名詞であり，キーワード選択の手がかりとして効果的であることが伺える．

最後に，Print と Sum の位置バイアスについて確認する．具体的には，見出しと要約における各 bi-gram が記事中のどの位置に存在するかの頻度について，図 1 にヒストグラムで可視化した．図からは位置バイアスがほぼ一致していることが分かる．つまり，見出しと要約で着目する部分が殆ど同じであり，転移学習に効果的であることが期待できる．

3 提案手法

3.1 タスク定義

入力文 X を One-hot ベクトルからなる長さ I の系列とする．ここで，$x_i \in 0,1^{|V_s|}$ は，入力 X の i 番目のトークン（単語）を表す．また，V_s は語彙であり，$|V_s|$ は語彙 V_s に含まれる単語数を表す．以降，$X = (x_1, \ldots, x_I)$ の略記法として $x_{1:I}$ を用いる．同様に出力側の要約について，$y_j \in 0,1^{|V_t|}$ は長さが J の出力 Y における，j 番目の 1-hot ベクトルと定義する．$|V_t|$ は出力側語彙 V_t の語彙数とする．ここで，Y は常に専用のトークンを 2 つ含むと仮定する．具体的には，y_0 が<bos> であり，y_{J+1} が<eos>である．また，Rush ら [1] の提案したヘッドライン生成タスクに倣い，$I > J$ とする．また，短文要約と見出しのデータを区別して学習するためにコントロールコード c を導入する．具体的には，x_0 に c が与えられるものとする．EncDec は以下の条件付き確率をモデル化する．

$$p(Y|X,c) = \prod_{j=1}^{J+1} p(y_j|y_{0:j-1}, X, c) \quad (1)$$

$\mathcal{D}$ を訓練データの集合，θ を EncDec で訓練するパラメータの集合だとする．最適なパラメータ θ' を，次の損失関数 $G(\theta)$ を $\mathcal{D}$ 上で最小化することによって求める．

$$G(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(X,Y)\in\mathcal{D}} \ell_{trg}(Y, X, c, \theta), \quad (2)$$

$$\ell_{trg}(Y, X, c, \theta) = -\log(p(Y|X, c, \theta))$$

推論時には，訓練したパラメータのもと，式 1 で定義した条件付き確率の積を最大化する系列をビーム探索する．

3.2 モデル

EncDec として生成型要約で高いパフォーマンスが報告されている Transformer[2] と BertSumAbs[5] を用いる．その上で，見出し生成と要約生成のスタイルを分離して学習するために，2 つの観点からコントロールコードを導入する．一つ目はタスクについてのコントロールコード [6]，2 つ目は長さについてのコントロールコード [3] である．コントロールコードの具体例については表 1 にまとめた．最終的に，EncDec とコントロールコードの組み合わせとして以下の 4 つを用いる．

Conditional Transformer(Task) Keskar ら [6] が提案したコントロールコードを Transformer に適用したモデル．コントロールコードは@Print と@Sum の 2 つを用いる．

Conditional Transformer(Length) 人見ら [3] が提案した手法で，コントロールコードは 7〜50 文字を表す計 44 個を用い．

Conditional BertSumAbs(Task) Liu ら [5] が提案した要約モデルにコントロールコードを導入したモデル．コントロールコードは@Print と@Sum の 2 つを用いる．

Conditional BertSumAbs(Length) Liu ら [5] が

提案した要約モデルに，コントロールコードは 7〜50 文字を表す計 44 個を用いる．

4 実験

4.1 データセット

JAMUL 2020 訓練データとして，2017 年以前のものを用い，比較実験の都合から同一記事に Sum･Large の両方が付与されたものを抽出した．検証・評価データについては，JNC との重複を避けるため 2018 年以降の組みから構成した．その後，比較実験と第 2.2 節の分析の都合から同一記事に Sum・Large・Print の 3 つの要約が付与されている組みのみを抽出した．その後，ランダムに半分を抽出し検証データとし，残りを評価データとした．最終的に，訓練データは 23,049 件，検証データは 2,321 件，評価データは 2,322 件となった．

JNC 人見ら [3] が提案した見出しと記事の対からなる大規模なコーパス．JNC は朝日新聞の記事と Print の対を 2007 年から 2016 年にかけて収集したコーパスで，1,932,398 件からなる．人見ら [3] の用いた前処理スクリプト[4]によって得られた 1,523,468 件を転移学習を目的とした訓練データとして用いる．

4.2 ベースライン

ベースラインとして，以下の 6 手法と提案手法を比較する．

Extractive Oracle 入力記事の各文から ROUGE-1 と ROUGE-2 の合計が最大になる 1 文[5]をシステム出力とみなす．

LEAD-1 入力記事の第 1 文をシステム出力とする．

Transformer(Sum) Sum のみで訓練された Transformer．

Transformer(Sum+Large) Paulus ら [7] が導入した設定で，入力として見出し・特殊トークン・記事を結合したものを使用し，訓練された Transformer．今回は見出しとして Large を用いる．

Transformer+SP-token 人見ら [3] の手法で，入力の先頭に長さの情報を持ったスペシャルトークを追加する．なお，JNC ので訓練したケース．

BertSumAbs Liu ら [5] が提案した生成型要約モデル．Sum のみで Fine-tuning する．

4.3 実装の詳細

Transformer の実装として Fairseq[6] [8] を用いた．各種パラメータについては学習率を除き，Vaswani ら [2] の WMT 2014 の English-German における base model のものを用いた．学習率については 0.0007[7]に変更した．また，単語埋め込みベクトルの学習に用いる重み行列については入力側・出力側で別々のものを用いた．BERT の訓練済みモデルとしては日本語の Wikipedia で事前学習を行っている Japanese BERT[8]を用いた．BERTSUM の実装は著者実装[9]を用い，コントロールコードの追加には 🤗 Transformers[10]を用いた[11]．また，BERTSUMABS 以外の実験における語彙については，SentencePiece[12]で構築した．merge operation を 8000 と定め，入力文，出力文を結合したコーパスから語彙の構築を行った．長さについてのコントロールコードを用いたモデルについては，評価時に 48 文字を表現したコントロールコードを用いる．推論時のビーム幅は 5 とした．また，生成結果・正解要約共に MeCab [4] により分かち書きをして ROUGE を求めた．

4.4 実験結果

単語の被覆率による評価 F 値ベースの ROUGE-1, ROUGE-2，ROUGE-L を用いた．それぞれの略記法として，R-1，R-2，R-L を用いる．表 2 が実験結果である．ROUGE のスコアにおいて，提案手法である Conditional Transformer が，最も高いパフォーマンスとなった．

そこで Conditional Transformer とその他手法について比較して行く．まず，抽出型のアプローチである Extractive Oracle･LEAD-1 の 2 手法と比べていく．提案手法が Extractive Oracle に対して R-1 で 4.8 ポイント，LEAD-1 に対して R-1 で 10 ポイントの改善が確認された．次に，Sum だけで訓練されたモデルと比較すると，提案手法が R-1 で 25.4 ポイントの改善が見られた．ヘッドラインを追加の入力として与えるケースについてもほぼ同様の結果となった．Transformer+SP-token と比較していくと，R-1 で 15 ポイントの改善が確認された．このことから，単純に見出しのみを学習データに用いるだけでは不十分であることが分かる．続いて，BERTSUMABS と比較すると，提案手法は R-1 で 12.5 ポイントの改善が見られた．

長さの制約についての評価 生成された要約が長さの制約をどれくらい正確に守れているかを評価するために以下の式から *Average Length Difference(ALD)* を求めた．

$$ALD = \frac{1}{n}\sum_{i=1}^{n} |l_i - len| \quad (3)$$

[4] https://github.com/asahi-research/Gingo
[5] https://github.com/nlpyang/BertSum を参考にした．
[6] https://github.com/pytorch/fairseq
[7] WMT 2014 の English-German における Vaswani らのスコアを Fairseq 上で再現した時の学習率．
[8] https://github.com/cl-tohoku/bert-japanese
[9] https://github.com/nlpyang/PreSumm
[10] https://github.com/huggingface/transformers
[11] 追加されたコントロールコードの重みベクトル・バイアスベクトルの初期化については Devlin ら [9] に従った．
[12] https://github.com/google/sentencepiece

	事前学習データ	訓練データ	R-1	R-2	R-L	*ALD*
Conditional Transformer(Task)	-	JNC, Sum	57.5	32.9	46.3	2.6
Conditional Transformer(Length)	-	JNC, Sum	57.5	32.8	46.4	1.4
Conditional BertSumAbs(Task)	日本語 Wikipedia	JNC, Sum	52.5	28.7	37.3	19.2
Conditional BertSumAbs(Length)	日本語 Wikipedia	JNC, Sum	53.4	28.6	38.3	8.4
Extractive Oracle	-	-	52.7	28.7	40.4	24.0
LEAD-1	-	-	47.5	23.8	36.3	28.9
Transformer(Sum)	-	Sum	32.1	7.7	23.9	5.7
Transformer(Sum+Large)	-	Sum, Large	32.9	7.7	23.8	3.8
Transformer+SP-token	-	JNC	42.5	18.0	30.7	2.1
BertSumAbs	日本語 Wikipedia	Sum	45.0	21.8	32.8	24.0

表 2: 実験結果．上から 4 段目までが提案手法．

ここで，n は生成された要約の数を，l_i は生成された要約の文字数を，len は正解要約の文字数を表す．スコアの解釈としては，正解要約から平均何文字ずれているかを表現している．

表 2 を *ALD* について見てみると，抽出型要約である Extractive Oracle・LEAD-1 については平均 24〜29 文字ずれているのに比べ，Conditional Transformer は平均 1〜3 文字程度のずれであった．厳密な長さの制約がある場合，制約に合わせて生成ができる生成型要約の方が本タスクにおいて有効であるといえる．

次に，コントロールコードの種類による影響を見ていく．具体的に，タスクについて制御するケースと長さについて制御するケースについて比較する．結果から，長さについて制御した方がより厳密に制約を守れていることが分かった．これは，タスクについてのコントロールコードが長さについて幅を持って学習しているのに対し，長さについてのコントロールコードは 1 文字ずつ学習できるためである．一方，タスクについて制御を行うと，見出しにしか出現しないスペースなどの表現が生成された要約に全く現れなくなるなど，ライティングスタイルについての改善が確認された．

また，Conditional BertSumAbs は Conditional Transformer に比べ正確に出力長をコントロールできないことが分かった．これは事前学習時に長さの情報について学習できていないことが原因であると考えられる．

5 関連研究

要約において，長さの制約を考慮したコーパスは JAMUL[3] と DUC 2004[13]がある．しかし，JAMUL は 1,489 件，DUC 2004 は 500 件のデータで構成されており，EncDec の枠組みで訓練・検証・評価の 3 つに分割することが困難であった．また生成型要約の小規模データに対する取り組みとして，Tilk ら [10] は見出し生成タスクに取り組んだ．約 300,000 件のデータサイズを小規模データと定義し，事前学習によるパフォーマンスの改善を提案した．Parida ら [11] は単一文書要約において，90,000 件のデータサイズを小規模データと定義し，擬似データによるデータ拡張でパフォーマンスの改善に取り組んだ．ただし，追加の要約データが豊富にあることが前提であるため，今回の実験設定とは異なる．これらの先行研究に比べると，我々が扱う小規模データは約 23,000 件であるという点でより小規模なデータに取り組んでいる．

データサイズについて，我々より小規模でで実験を行なった Zhang ら [12] の研究では，英語の単一文書要約のコーパス CNN/Dailymail[13] において 1,000 件程度で高いパフォーマンスを出せることを示している．しかし，事前学習に 1.5B のデータが必要であり，我々は 1.5M を Style transfer に用いているため，追加で用いるデータが大幅に小規模である．

[13]https://duc.nist.gov/duc2004/

6 おわりに

本研究では，小規模リソースを用いた要約生成におけるパフォーマンスの改善に取り組んだ．提案手法は，コントロールコードによるスタイル分離することで，見出し生成と要約生成を同時に学習し，スタイル転移によるパフォーマンスの改善を行った．今回構築したコーパス，JAMUL 2020 の評価データにおいては抽出型のオラクル要約に対して R-1 で 4.8 ポイント，その他既存手法に対しても大幅な改善が確認された．

参考文献

[1] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *In EMNLP*, pp. 379–389, 2015.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

[3] Yuta Hitomi, Yuya Taguchi, Hideaki Tamori, Ko Kikuta, Jiro Nishitoba, Naoaki Okazaki, Kentaro Inui, and Manabu Okumura. A large-scale multi-length headline corpus for analyzing length-constrained headline generation model evaluation. In *INLG*, 2019.

[4] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *EMNLP*, pp. 230–237, 2004.

[5] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *EMNLP-IJCNLP*, pp. 3728–3738, 2019.

[6] Nitish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. In *arXiv preprint*, 2019.

[7] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *ICLR*, 2018.

[8] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT*, pp. 48–53, 2019.

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

[10] Ottokar Tilk and Tanel Alumäe. Low-resource neural headline generation. In *NEWSUM*, pp. 20–26, 2017.

[11] Shantipriya Parida and Petr Motlicek. Abstract text summarization: A low resource challenge. In *EMNLP-IJCNLP*, pp. 5993–5997, 2019.

[12] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *ArXiv*, 2019.

[13] Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom.