Hie-BART : 階層型 BART による文書要約

秋山 和輝 [1]　田村 晃裕 [2]　二宮 崇 [1]
[1] 愛媛大学 大学院理工学研究科 電子情報工学専攻,
[2] 同志社大学 理工学部 情報システムデザイン学科
[1]{k_akiyama@ai., ninomiya@}cs.ehime-u.ac.jp, [2]aktamura@mail.doshisha.ac.jp

1 はじめに

近年，高い精度を報告している生成型の自動要約モデルは，ニューラルネットワークに基づくニューラル要約モデルであり，その多くが事前学習を導入している．これまで自動要約タスク用に拡張された事前学習モデルとして，BertSum モデル [1] が提案されている．また，複数タスクで利用することのできる汎用的な事前学習モデルとして，T5 モデル [2] や BART モデル [3] が提案されている．既存の事前学習モデルの中でも，BART モデルは高い精度での自動要約を可能にしている．しかし，BART モデルは要約生成時に文書の階層構造をとらえる構造にはなっていない．

一方，ニューラル要約モデル同様，系列変換モデルが適用されることが多いニューラル機械翻訳では，入力テキストの情報を「フレーズと単語」や「単語と文字」といった複数の粒度で捉えることで，翻訳精度の改善を実現している．Transformer[4] に基づくニューラル機械翻訳モデルでは，入力テキストを複数の粒度（単語やフレーズ）に分解し，それぞれの粒度を Multi-head Self-Attention Networks (Sans) の各ヘッドに割り当てることで，単語とフレーズの関係を考慮する Multi-Granularity Self-Attention (MG-SA)[5] を組み込むことで，翻訳精度を向上させている．このモデルでは，MG-SA により，単語間だけでなく単語とフレーズ間の相互作用を組み込むことが可能となる．

そこで，本研究では，MG-SA の概念を BART モデルに適用し，文と単語の階層的な関係を考慮した要約を可能とする手法として，Hierarchical-BART (Hie-BART) を提案する．具体的には，入力文書を単語レベルと文レベルの情報に分割し，BART における Encoder の Sans 層では，単語レベルの関連を計算するだけではなく，一部のヘッドにおいて文レベルの関連を計算する．そして，単語レベルと文レベルのマルチヘッドの出力を結合させることで，単語レベルと文レベルの双方を考慮した要約生成を行う．

提案モデルの評価は，CNN/DailyMail データセット [6] を用いた自動要約タスクで行った．既存の生成型自動要約モデルと比較した結果，提案モデルは非階層型の既存の BART モデルに比べて ROUGE-L[7] の F 値が 0.23 ポイント上回ることを確認した．

2 関連研究

2.1 BART

BART モデル [3] は，Transformer モデル [4] の構造をベースとした，汎用的な事前学習モデルである．事前学習の手法として，Token Masking，Sentence Permutation，Document Rotation，Token Deletion，Text Infilling の 5 つの手法を提案している．この 5 つの手法のうち，Sentence Permutation と Text Infilling を組み合わせたものが最高精度となっており，最終的な評価時に使用されている．Sentence Permutation は，文書内の文の場所をランダムに交換した文書を元の文書に復元する事前学習法である．Text Infilling は，複数の単語列を一つのマスクトークンに置き換えたり，文中にマスクトークンを挿入したりした文書を，元の文書に復元する事前学習法である．

2.2 Multi-Granularity Self-Attention (MG-SA)

MG-SA [5] は，入力テキストを複数の粒度の要素（単語とフレーズ）に分け，各粒度の要素を Sans の各ヘッドに割り当てることで異なる粒度の情報を捉える方法である．この手法ではまず，次のように，Sans の入力 H からフレーズレベルの情報を表す行列 H_g を生成する．

$$H_g = F_h(H) \tag{1}$$

ここで，$F_h(.)$ は h 番目のヘッドに対するフレーズレベルの入力を生成する関数である．具体的には，$F_h(.)$ は，単語レベルのベクトルに対して Max Pooling を適用することでフレーズレベルのベクトルを生成する．

この生成されたフレーズレベル入力を用いて，SANs を次のように実行する．

$$Q^h, K^h, V^h = HW_Q^h, H_g W_K^h, H_g W_V^h \quad (2)$$

$$O^h = \mathbf{ATT}(Q^h, K^h)V^h \quad (3)$$

ここで，$Q^h \in \mathbf{R}^{n\times d_h}, K^h \in \mathbf{R}^{p\times d_h}, V^h \in \mathbf{R}^{p\times d_h}$，$W_Q^h, W_K^h, W_V^h \in \mathbf{R}^{d\times d_h}$ であり，d は隠れ層の次元数，d_h は各ヘッドの次元数，n は単語レベルのベクトルの次元数，p はフレーズレベルのベクトルの次元数である．また，**ATT(X,Y)** は X と Y のアテンション重みを算出する関数である．この演算により，SANs における各ヘッドの出力 O^h が生成される．その後，全てのヘッドの出力を結合させたものが，入力 H に対する MG-SA の出力となる．

$$\mathbf{MG\text{-}SA(H)} = [O^1, ..., O^N], \quad (4)$$

各ヘッドの出力 O^h は，単語間の情報を含んでいるものや，単語とフレーズ間の情報を含んでいるものがある．したがって，MG-SA を使うことで，単語間の関連性に加えて，単語間とフレーズ間の関連性を捉えることができる．

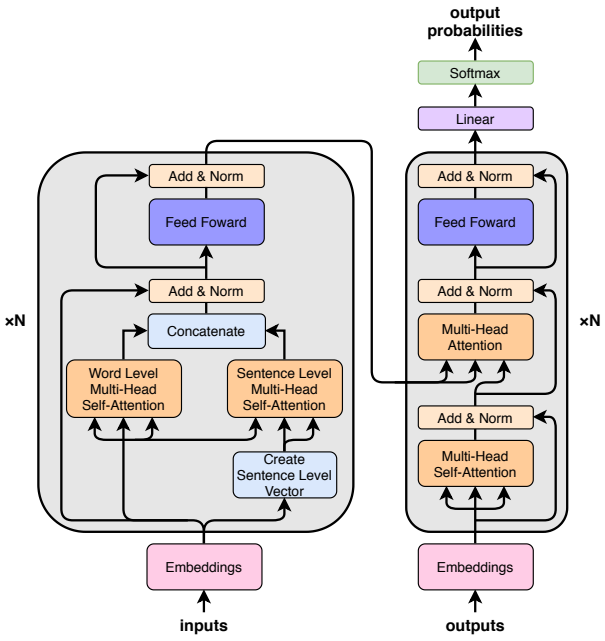

図 1　Hie-BART のモデル図

3　提案手法

Hie-BART（Hierarchical-BART）モデルでは，エンコーダは，通常の BART モデルが備えている単語同士の関連性を捉える単語レベルの Multi-head Self-Attention Networks （SANs）に加えて，文と単語間の関連性を捉える文レベルの SANs を備えている．Hie-BART の概要図を図 1 に示す．

エンコーダでは，入力テキストを Embedding 層に通した後，SANs での計算時に単語レベルと文レベルに分割して計算する．ここで，各文を区別するために，入力テキスト中の各文の先頭には $[BOS]$ トークンを付与している．文レベルの SANs では，Create Sentence Level Vector 層で単語レベルの入力から文レベルの入力を作成した後，アテンションの計算を行う．各ヘッドでアテンションの計算が終了したら，単語及び文レベルの出力を結合したものを，次層の Feed Forward 層への入力とする．

3.1　Create Sentence Level Vector 層

Create Sentence Level Vector 層の動作例を図 2 に示す．図 2 において，$E_{w_{ij}}$ 及び $E_{[BOS]}$ は単語 w_{ij}（i 番目の文中の j 番目の単語）及び $[BOS]$ トークンの埋め込みベクトルである．E_{s_i} は．単語レベルの埋め込みベクトルから生成された文 s_i（i 番目の文）の文レベルの埋め込みベクトルである．

Create Sentence Level Vector 層では，単語レベルの埋め込みベクトルから Average Pooling により文レベルの埋め込みベクトルを生成する．具体的には，単語系列 $W = (w_1, ..., w_N)$（各 w_i は単語）が入力として与えられた場合，まず，単語系列 W を文分割し，文系列 $S = (s_1, ..., s_M)$（各 s_i は文）を得る．ここで，N は総単語数，M は総文数を示す．そして，S の各要素（各文）に対して，次のように Average Pooling を適用する．

$$g_m = \mathbf{AVG}(s_m) \quad (5)$$

ここで，**AVG(・)** は Average Pooling を実行する関数である．これにより，各文に対する埋め込みベクトル $G = (g_1, ..., g_M)$ を生成する．実際には，W と S の各要素は埋め込みベクトルとなる．例えば，図 2 において，E_{s_1} を生成する場合は，$[E_{[BOS]}, E_{w_{11}}, E_{w_{12}}, E_{w_{13}}]$ に対して Average Pooling を適用する．

以上のように生成された G を文レベルの SANs の入力とする．

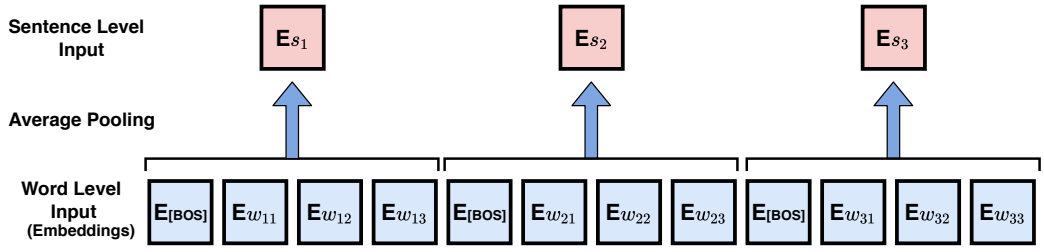

図 2　Create Sentence Level Vector 層の動作例

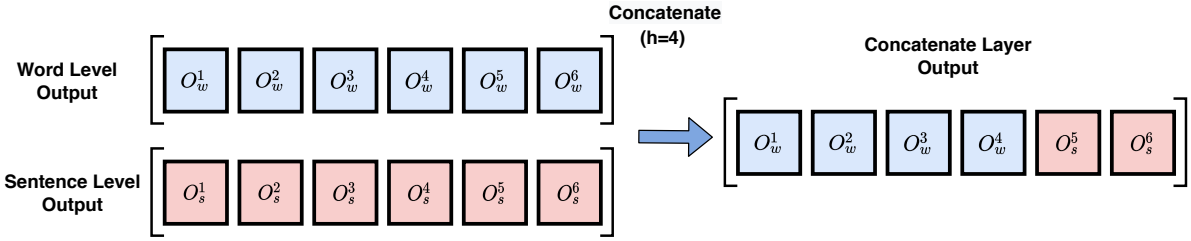

図 3　Concatenate 層の動作例 (ヘッド数が 6 の場合)

3.2　Concatenate 層

Concatenate 層では，単語レベル SANs と文レベルの SANs の出力を結合させる．Concatenate 層では，次のような単語レベル SANs と文レベルの SANs 層の出力が入力となる．

$$\textbf{MG-SANs(W)} = [O_w^1, ..., O_w^H] = O_w^{ALL} \tag{6}$$

$$\textbf{MG-SANs(G)} = [O_s^1, ..., O_s^H] = O_s^{ALL} \tag{7}$$

O_w^{ALL} は単語レベルの SANs の出力であり，O_w^i は単語レベルの SANs のヘッド i の出力である．一方で，O_s^{ALL} は文レベルの SANs の出力であり，O_s^i は文レベルの SANs のヘッド i の出力である．ここで，H は SANs のヘッド数を示す．Concatenate 層では，この単語及び文レベルの SANs の出力結果を次のように結合させる．

$$\textbf{CONCAT}(O_w^{ALL}, O_s^{ALL}, h) = [O_w^1, ..., O_w^h, O_s^{h+1}, ..., O_s^H] \tag{8}$$

ここで，h は結合点である．H 個の Multi-head の内，1 から h までのヘッドを単語レベルの出力とし，$h+1$ から H までのヘッドを文レベルの出力として結合する．

Hie-BART 内の Concatenate 層の動作例を図 3 に示す．図 3 は，Multi-head のヘッド数が 6 で，結合点 h が 4 の場合の動作を示している．単語レベルの SANs の出力である $[O_w^1, ..., O_w^6]$ と，文レベルの SANs の出力である $[O_s^1, ..., O_s^6]$ が結合点 $h=4$ で結合された $[O_w^1, O_w^2, O_w^3, O_w^4, O_s^5, O_s^6]$ が Concatenate 層の出力となる．

表 1　各モデルの要約性能（テストデータでの性能）

Model	ROUGE			
	1	2	L	平均
LEAD-3 [8]	40.42	17.62	36.67	31.57
BERTSUMEXTABS [1]	42.13	19.60	39.18	33.64
T5 [2]	43.52	21.55	40.69	35.25
BART [3]	44.16	21.28	40.90	35.45
BART (ours)	44.06	21.22	40.82	35.37
Hie-BART (ours)	**44.35**	**21.37**	**41.05**	**35.59**

以上のように生成された Concatenate 層の出力を次層の Feed Forward 層への入力とする．

4　実験

4.1　データセット

データセットは，英文ニュース記事の要約コーパスである CNN/DailyMail データセット[1] [6] を使用した．このデータセットは，学習データが 287,226 対，開発データが 13,368 対，テストデータが 11,490 対で構成されている．また，要約元文書は平均 781 トークン，要約文は平均 56 トークンである．データの前処理はトークン化は CNN/DailyMail ダウンロードページ[1]の通りに行い，BPE の使用などは fairseq による前処理に基づいて行った[2]．

1）CNN/DailyMail データセットのダウンロードページ: https://github.com/abisee/cnn-dailymail
2）fairseq による BART の利用法: https://github.com/pytorch/fairseq/tree/master/examples/bart

表 2　単語レベルと文レベルのヘッド数の割合による性能比較（開発データでの性能）

単語：文	ROUGE			
	1	2	L	平均
16:0	44.72	21.73	41.43	35.96
15:1	44.95	**21.92**	41.68	36.18
14:2	**45.01**	**21.92**	**41.75**	**36.23**
13:3	44.91	21.87	41.64	36.14
12:4	44.74	21.66	41.49	35.96
11:5	44.88	21.81	41.62	36.10
10:6	44.78	21.75	41.51	36.01
9:7	44.70	21.71	41.46	35.96
8:8	44.79	21.77	41.58	36.05

4.2　実験設定

Hie-BART モデルの実装は，fairseq による BART モデルのコードを基に改良することで実装した．また，事前学習済みのモデルは fairseq のモデル “bart.large”[2] を使用した．BART を fine-tuning する際は，勾配累積のハイパーパラメータ update-freq を 10，最大エポック数 max-epoch を 6，総学習ステップ数を 23,916 とした．また，提案手法 Hie-BART のハイパーパラメータは，Multi-head のヘッド数を 16，単語レベルと文レベルの SANs における出力の結合ヘッド数の割合を “単語:文 = 14:2” とした．結合ヘッド数の割合のハイパーパラメータは開発データで調整した．表 2 に示す通り，開発データにおいて最も性能がよかった割合を採用している．このハイパーパラメータに関しては，5 章にて考察する．その他のハイパーパラメータは fairseq による設定[2] を用いた．

4.3　結果

実験結果を表 1 に示す．ここでは，CNN/DailyMail のテストデータによる結果を示している．評価指標として，ROUGE-1，ROUGE-2，ROUGE-L の F 値 [7] とそれらの平均値を用いた．ROUGE スコアの算出には，files2rouge[3] を利用した．実験では，提案手法の Hie-BART モデルを，非階層の通常の BART モデルに加えて，LEAD-3[8]，BERTSUMEXTABS[1]，T5[2] と比較した．ベースラインの BART モデルとして，我々の計算機環境で再現した BART モデルの結果（BART (ours)）に加えて，Lewis ら [3] で報告されている結果も示している．

表 1 より，提案手法 Hie-BART は ROUGE-1/2/L の F 値において，Lewis らが報告している BART の性能から平均約 0.1 ポイント改善していることが分かる．また，我々が再現した BART からは，平均約 0.2 ポイント改善していることが分かる．これらの結果より，提案手法が有効であることを確認できた．

5　考察

提案手法は，Concatenate 層で単語レベルと文レベルのヘッドの出力を特定の割合で結合する．この割合の変化で提案手法の性能がどう変わるかを考察する．表 2 に，結合時の単語レベルと文レベルのヘッドの割合を変えた時の提案手法の開発データにおける性能を示す．ヘッド数の割合はハイパーパラメータとして手動で調節しており，表 2 の最左列に示されている．最左列が “単語:文 = x:y” の場合，単語レベルのヘッド数が x，文レベルのヘッド数が y を意味する．

表 2 より，“単語:文 = 14:2” の場合が ROUGE-1/2/L スコアの平均が最大になっていることが分かる．また，単語レベルに比べて文レベルのヘッド数の割合が少ない方が，ROUGE スコアが高くなる傾向にあることが分かる．ただし，文レベルのヘッド数が 0，つまり通常の非階層の BART による精度は，Hie-BART よりも性能が低い．また，単語レベルのヘッド数を 8 以下に減らしていく方向は，精度の改善が見られなかったので表 2 には掲載していない．

6　おわりに

本研究では，BART におけるエンコーダの Self-Attention 層を単語レベルと文レベルに分割し計算することで，単語と文間の関係性を考慮する Hie-BART を提案した．BART を階層化することにより，CNN/DailyMail データセットにおいて，ROUGE-L の F 値が 0.23 ポイント改善することを確認した．今後は，単語間や単語と文間の情報に加え，文同士の関係を取り入れる手法に拡張予定である．また，Xsum データセットなどの CNN/DailyMail データセット以外のデータセットで提案モデルの有効性を検証予定である．

参考文献

[1] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and*

3）files2rouge 利用法: https://github.com/pltrdy/files2rouge

the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3730–3740, Hong Kong, China, November 2019. Association for Computational Linguistics.

[2] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, Vol. 21, No. 140, pp. 1–67, 2020.

[3] 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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880. Association for Computational Linguistics, July 2020.

[4] 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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008. Curran Associates, Inc., 2017.

[5] Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang, and Zhaopeng Tu. Multi-granularity self-attention for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 887–897, Hong Kong, China, November 2019. Association for Computational Linguistics.

[6] Karl Moritz Hermann, Tomas Kocisky, 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] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[8] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. SummaRuNNer: A recurrent neural network based sequence model for extractive. In *Proceedings of In Association for the Advancement of Artificial Intelligence*, 2017.