Vol.2017-DBS-165 No.33
Vol.2017-IFAT-128 No.33
2017/9/20

TL;DR 3行要約に着目したニューラル文書要約

小平 知範[†1,a)]　小町 守[†1,b)]

概要：ニュースキュレーションアプリやブログまとめサービスでは，スマートフォンのような表示領域が限られた端末で読むことが多いため，出力をコンパクトにする必要がある．特に，3 点にポイントを絞り，かつ要約自体の流れも重要度順に並んでいる読みやすい要約を期待するユーザーが多い．これまで広く研究されてきた重要文抽出のような抽出型文書要約は，文数に制限を加えることで 3 点に絞ることは容易であるが，複数の文をまとめたりすることができないため，スペースの制約を満たすことが困難である．また，出力の構造も必ずしも考慮されていない．先行研究では要約の文書構造が明らかではないため，明示的に用いることができなかったが，3 行要約は文書構造の分析が容易なので，要約の文書構造を分析することで，要約の出力において文書構造を考慮することができる．そこで，我々は 3 文で構成された要約のみで新しい大規模データセットを構築し，生成する要約の構造を分析した．さらに，要約の文書構造を考慮したニューラルネットワークに基づいた抽象型文書要約モデルを提案する．

1. はじめに

要約を構築する主な目的は，読み手が文書すべてを読むことなくその文書を理解できるようにすることである．特にニュース要約では，スマートフォンユーザーは画面のサイズが限られているので，表示できる限られた量の要約を読みたい．これらの目的を達成するために，ポータブルデバイス向けの要約システムは重要な情報を含んだ要約を限られた要約長の中で生成しなければならない．

要約タスクには抽出型と抽象型の 2 つのアプローチがある．抽出型アプローチは要約を作るために文書の一部（文や句，単語など）を選ぶ．抽象型アプローチは文書に現れない単語も使って要約を生成する．抽出型アプローチ [1], [2] は元の文書から出力する表現を直接抽出するので，抽象型アプローチより文法的な要約を作ることができる．しかし，それでは元の文書に現れない単語を選ぶことができない．

抽象型要約は機械翻訳タスクとは異なり，おおよその出力は入力の文書から得ることができる．また，抽象型要約では主に Encoder-Decoder という機構を用いる．Encoder-Decoder モデルにおいて入力系列はソース，出力系列はターゲットと呼ばれる．Encoder-Decoder は，ソース（文書）の情報を読み取る RNN の Encoder と，その情報をもとにターゲット（要約）を生成していく Decoder を組み合わせたものである．入出力ともに系列の場合は sequence-to-sequence と呼ばれる．

Rush ら [3] は，Sutskever ら [4] が提案した sequence-to-sequence を基に，要約中に入力の文書に現れない単語を含む抽象型文要約タスクに取り組んだ．近年 Rush ら [3] の手法をもとに Nallapati らや See ら [5], [6] によってニューラルネットを用いた抽象型文書要約のアプローチが提案された．彼らの用いた CNN / Daily Mail データセットは様々な長さの文で構成された要約が含まれているので，構造化された要約を生成するために要約の構造情報の注釈を簡単につけることができない．そのため，彼らのモデルは構造的な要約の生成ができない．

そこで，本研究ではニュース要約のための構造的な要約（3 行要約）の生成に着目し，我々は CNN / Daily Mail データセットと同量の要約データセットを Livedoor News から構築した．Livedoor News は 3 行要約とニュースを公開しているので，このデータセットを用いた解析は容易である．

3 行要約の生成を解析するために，我々は See ら [6] のモデルを用いた．モデルを改善するために，我々は彼らのモデルを基に新しい機構を提案する．我々の貢献は以下である．

- 3 行要約のみを含む新しい日本語ニュースの要約データセットを構築した．
- データセットに対して，要約の構造の注釈付と解析を行った．

[†1] 首都大学東京
Tokyo Metropolitan University
[a)] kodaira-tomonori@ed.tmu.ac.jp
[b)] komachi@tmu.ac.jp

- このデータセットの特徴を基に 3 行要約に適応したモデルを提案した．

2. 関連研究

抽象型文要約では，Rush ら [3] が sequence-to-sequence モデルを使った抽象型要約を生成する新しい要約手法を提案した．彼らは Gigaword コーパスと DUC-2004 において，世界最高精度の精度を達成した．いずれのコーパスもニュース記事を含んでいるが，要約は構造化されていない．

CNN / Daily Mail 要約タスクの中の記事から複数文で構成されている要約を出力するタスクにおいて，抽象型文書要約に取り組んだ研究がある．Nallapati ら [5] は Large Vocabulary Tric [7] や Switching Pointer-Generator，階層的ネットワークを Attention Encoder-Decoder モデルに取り入れ，このタスクにおいての改善モデルを提案した．Attention Encoder-Decoder は Encoder-Decoder モデルに，単語生成時にソース側のどの単語に注目してるかを入力系列長の長さの確率分布として与える機構を付与したものである（4.1 節）．Large Vocabulary Tric は要約側の単語のほとんどは入力文書からくるという特徴を利用し，ミニバッチごとに語彙を決めることで，低頻度の単語も含めた大きな語彙を使うことができる．Switching Pointer-Generator は Decoder が 1 つずつ単語を出力する各タイムステップで単語を生成するかソース側の単語をコピーするかを決める機構である．こうすることで，次に生成する単語が未知語の際に Pointer が選ばれることで，未知語の生成が可能なる．階層的ネットワークは，文単位での要約に含めるかの情報量や生成している際にどの文に注目しているかを捕らえるための文単位と単語単位の Attention を組み合わせたものである．彼らは複数文要約のための新しいデータセットを提案し，ベンチマークを構築した．この研究における出力は複数文であるが，出力は必ずしも文書構造を考慮したものではなく，出力も詳細に分析されていない．

See ら [6] は CNN / Daily Mail の要約タスクにおいて世界最高精度を達成した．彼らは Nallapati [5] らのモデルをベースに，Hybrid Pointer-Generator ネットワークと Coverage 機構を提案した．Hybrid Pointer-Generator は Attention と語彙の確率分布を重ね合わせて出力単語を決定するものである．2 つの分布を重ね合わせることで未知語を考慮でき，なおかつソース側の単語を出力しやすくなる．

Switching Pointer-Generator は Pointer が選ばれた際に確実にソース側から単語を選ぶことができる．Hybrid Pointer-Generator は同時に Decoder から出力された単語生成確率と Attention によるソース側の単語の確率分布を同時に考慮することができる．

Coverage 機構はニューラルネットワークを用いた言語生成に特有の，同じ内容の言語表現を繰り返し生成してしまう問題を解決するためのものである．我々のモデルは彼らのモデルをベースとしているため，詳細な内容は 4 節にて示す．

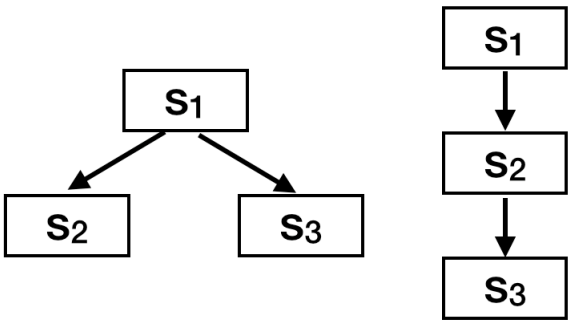

図 1 文書構造．左: 並列タイプ．右: 直列タイプ．

3. 日本語ニュース 3 行要約データセット

我々は Livedoor News [*1] から日本語の記事と要約のペアを収集した．この要約は人間の編集者によって書かれている．要約は 3 文で構成されている．詳細は後に示す．我々は 2014 年 1 月から 2016 年 12 月までの期間でデータの収集を行い，得られた記事と要約は計 215,560 ペアとなった．収集したデータを分割し，トレーニングデータとして 213,160 ペア，検証データとして 1,200 ペア，テストデータとして 1,200 ペアとした．検証データとテストデータは 2016 年 1 月から 2016 年 12 月の期間のものから毎月 100 件づつ抽出した．

3.1 記事の特徴

それぞれの記事に対して，9 つのカテゴリー（国内，海外，IT 経済，芸能，スポーツ，映画，グルメ，女子，トレンド）から 1 つのカテゴリーが選ばれ，そのカテゴリーに対するいくつかのサブカテゴリーから 1 つのサブカテゴリーが選ばれている．さらに，特定のタグ（キーワードやキーフレーズ，より詳細なカテゴリー）が付与されている．収集したデータはニュースの記事とタイトル，より短いタイトルと抽象型要約が存在する．

このデータセットは上記のような多くの有用な情報を持つが，本研究では記事と要約のみを用いる．

3.2 3 行要約に対する文書構造アノテーション

Livedoor News の要約は 3 文で構成されているため，出力の構造の解析が容易である．そこで，本研究では要約の一部である検証データとテストデータに対して要約の文書構造に対して注釈付けを行った．

それぞれの要約に対して話の流れに対応する一つのタグを付与した．多くの要約は**並列**タイプと**直列**タイプの 2 つの種類に分けられる（**図 1**）．例文を**表 1** に示す．最初の 2

[*1] http://news.livedoor.com/

並列	マクドナルド HP の、各国の違いを紹介している 日本はアメリカと似ているが、より情報を多く載せようという意図がみえる ドイツはバランスよく整理整頓され、フランスはモダンさが感じとれるという
直列	ソニーの VR ゴーグルが即完売し、生産が追いつかない人気ぶりだという 米投資銀行は 25 年に、VR・AR 関連の世界市場が約 9 兆 5000 億円になると予測 10 兆円市場はコンビニ全体の売上高と同規模で、投資家も注目しているそう

表 1 各文書構造タイプにおける要約例.

列挙型並列	コンビニ 3 社のボジョレー・ヌーヴォーを飲み比べている セブン-イレブンは多少渋味が強く、ファミリーマートは多少酸味が強いそう ローソンは後味がスッキリし人気が高く、予約分が完売した店舗もあるという
文分割型直列	山本昌氏が 23 日の「ジョブチューン」で星野仙一氏に殴られた話を明かした 「投げ終わってベンチ裏に来いと言われて多少かわいがられまして」と暴露 「顔が腫れ過ぎて降板した」と驚きの事実を伝えた

表 2 列挙型並列と文分割型直列の要約例.

	検証	テスト	全て
直列	278	320	598
文分割型直列	10	4	14
並列	836	808	1,644
列挙型並列	76	68	144

表 3 3 行要約に対する文書構造アノテーションの結果.

文は 2 種類とも特徴が似ており，最初の文では主な出来事について記載され，2 文目は 1 文目に対する追加情報が記述されている．“並列” タイプは 3 文目が 2 文目とは異なる 1 文目に対する追加情報が書かれている．一方，“直列” タイプは 3 文目が 2 文目に対する追加情報が書かれている．つまり，“並列” タイプは 2 文目と 3 文目には特に順序はなく，“直列” タイプは 2 文目と 3 文目は順序を持っている．

2 つのタグをアノテーションする中で，特徴的な構造をしている “列挙型並列”，“文分割型直列” を追加して，最終的には 4 タイプに分けた．追加したタグの例を**表 2** に示す．列挙はあるものを紹介する時に要約の中に含まれることが多い．文分割は元々の文が長い場合に要約の中に現れる．これらは，主にスマートフォンで閲覧されることを想定してコンパクトに情報を提示する必要がある Livedoor News に特徴的な要約の例である．

3.3 アノテーションの結果と分析

表 3 に示すように，検証データとテストデータのいずれにおいても，約 70%の要約は “並列”，残りは “直列” のタグが振られる結果になった．表 3 では “列挙型並列” にタグづけされた中には単に例を並べるだけの文ではないものが存在した．

“並列” と “直列” に共通する特徴として，最初の文は主な出来事が記されている．おおよそ 2 文目は 1 文目の内容に対して結果の説明，詳細な情報，例などが書かれている．

一方，このデータセットでは 3 文目は様々な役割をしている．“直列” にタグ付けされた要約では，3 文目の内容は 2 文目に依存している．一方で “並列” にタグ付けされた要約では，3 文目は 1 文目に依存している．つまり，要約システムは 3 文目を生成する際にタグによって 1 文目か 2 文目のどちらに注意を向けるか決めなければならない．

4. 3 行要約モデル

我々のモデルは Attention Encoder-Decoder と Hybrid Pointer-Generator，Coverage 機構を組み合わせた See ら [6] のモデルをもとに構築した．まず，これらの機構について説明を行う．次に 3 行要約のための提案手法を説明する．

4.1 Attention Encoder-Decoder

入力単語列は記事のトークン，出力単語列は要約のトークンである．Encoder 側は 1 層の双方向 LSTM [8] を用い，Decoder 側には順方向の LSTM を用いる．Encoder によって生成される Encoder の隠れ層を h_i とする．それぞれのステップ t で，Decoder は出力の前単語の単語埋め込みベクトル（学習時は正解の要約を前単語として用い，テスト時は Decoder によって生成された前単語を用いる）と Decoder の状態 s^t を渡す．Attention の分布 a^t は Bahdanau ら [9] と同様に計算される．

$$e_i^t = v^T \tanh(W_h h_i + W_s s^t + b_a) \quad (1)$$

$$a^t = \mathrm{softmax}(e^t) \quad (2)$$

ここで v は重みベクトルであり，W_h を W_s は重み行列，b_a はバイアスベクトルであり，それぞれ学習可能パラメータである．Attention の分布はタイムステップ t における Encoder の隠れ層の重要度を示す確率分布として表される．文脈ベクトル h_*^t は以下によって計算される．

$$h_*^t = \sum_i a_i^t h_i \quad (3)$$

この文脈ベクトルを Decoder の状態 s^t と連結し，2 つの線形変換を用い語彙分布 P_{vocab} が生成される．

$$P_{vocab} = \mathrm{softmax}(V'(V([s^t, h_*^t] + b) + b')) \quad (4)$$

ここで V と V' は重み行列，b と b' はバイアスベクトル

であり，それぞれ学習可能なパラメータである．得られた P_{vocab} の確率分布から一番確率の高い単語 w^t が出力単語として選ばれる．各タイムステップ t におけるロスはターゲット側の単語 w_*^t の負の log likelihood である．

$$\text{loss}_t = -\log P_{vocab}(w_*^t) \tag{5}$$

また，全体の系列に対してのロスは以下である．

$$\text{loss} = \frac{1}{T}\sum_{t=0}^{T}\text{loss}_t \tag{6}$$

ここで，T はシステム出力の単語数である．

4.2 Hybrid Pointer-Generator Network

Hybrid Pointer-Generator は See ら [6] によって提案され，Attention と語彙分布を組み合わせるものである．彼らの Pointer-Generator は Attention モデル（4.1 節）と Pointer Network [10] を組み合わせたものである．これはソース側の単語分布をターゲット側の単語分布と同様に考慮する．こうすることで Switching Pointer-Generator 同様，ソース側にある見たことのない単語を考慮することができ，未知語の生成問題に対応できる．さらに，ソース側の単語の生成確率が高くなり，ソース側と同じ単語を使うことが多い要約タスクでは有用である，各タイムステップ t で Pointer-Generator モデルの生成確率 $p_{gen} \in [0,1]$ は文脈ベクトル h_*^t と Decoder の状態 s^t，Decoder の入力である単語埋め込みベクトル x_t によって計算される．

$$p_{gen} = \sigma(v_{h^*}^T h_*^t + v_s^T s^t + v_x^T x_t + b_g) \tag{7}$$

ここでベクトル v_{h^*} と v_s，v_x は重みベクトル，b_g はスカラーのバイアスであり，それぞれ学習可能パラメータであり，σ はシグモイド関数である．p_{gen} は単語分布 P_{vocab} か Attention の分布 a^t のどちらを用いるかの soft switch として用いる．各文書において，これらは拡張語彙を作り，それはターゲットの語彙とソースの語彙との和集合である．拡張語彙の生成確率は以下で計算される．

$$P(w) = p_{gen}P_{vocab}(w) + (1-p_{gen})\sum_{i:w_i=w} a_i^t \tag{8}$$

もし w が out-of-vocabulary（OOV）ならば，$P_{vocab}(w)$ は 0 であり，また w がソース側の単語に存在しなければ $\sum_{i:w_i=w} a_i^t$ は 0 である．

4.3 Coverage Mechanism

See ら [6] は Encoder-Decoder モデルにおける繰り返しの問題を解決するために Coverage モデル [11] を改善させた．彼らのモデルでは，各 Decoder のタイムステップまでの Attention の分布の合計が Coverage ベクトル c^t として保存される．

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{9}$$

c^t はタイムステップ t までにそれぞれの単語に対してどれだけ注目したかを示す，ソース文書の単語に対する分布である coverage ベクトルは以下のように用いられる．

$$e_i^t = v^T \tanh(W_h h_i + W_s s^t + w_c c_i^t + b_a) \tag{10}$$

ここで，w_c は重みベクトルであり，学習可能パラメータである．彼らは同じ場所への繰り返しの Attention に対してペナルティを与える目的で，Coverage のロスを取り入れた新しいロス関数を構築した．

$$\text{loss}_t = -\log P(w_t^*) + \lambda\sum_i \min(a_i^t, c_i^t) \tag{11}$$

λ は同じ場所への繰り返しの Attention をどれだけ許容するかのパラメータである．これにより，$w_c c_i^t$ では Attention が同じ場所を繰り返し指すことを防ぐので，同じ内容を出力することを防ぐことに繋がる．

4.4 先行研究のモデルの改善

Livedoor News のデータセットに適応させるために，先行研究のモデルを拡張する．このデータセットでは要約の構造によって 3 文目の役割が変わってくるので，提案手法は Coverage 機構および Decoder において文書構造の情報を利用する．

具体的には，要約を生成する際に以下の変更を行う．

(1) 3 文目生成時に，Coverage ベクトルを 1 文目生成終了時のものに戻す（CovReset）．
(2) 3 文目生成時に，Decoder の状態を 1 文目生成終了時の Decoder の状態に戻す（DecReset）．
(3) 各文の生成を終了するごとに Coverage ベクトルを初期化する（DecAllReset）．

(1) では，“並列” タイプでは同じ主語をゼロ主語として用いることがあり，Coverage ベクトルをそのまま利用すると Attention が同じ場所を指せず，2 文目で利用した主語を見れなくなる場合があるので，1 文目の終了時の Coverage ベクトルの状態に戻す．また，“直列” タイプにおいても 2 文目で利用した Encoder 側の情報は必要なためでもある．(2) においては，“並列” タイプでは 2 文目の情報は 1 文目の情報のみが必要かつ，2 文目と違う情報を用いるので，Coverage の情報は残しつつデコードの状態のみを戻す．(3) では，Coverage の状態のみをを保持することで繰り返し同じ内容を出力することなく，文のはじめに Encoder 側の情報を持つことで情報の消失を防ぐ．

（1），（2）では文構造を考慮した手法であり，（3）では長文生成で Coverage や Decoder の状態の情報が曖昧にならないようにするためのものである．これらの手法は追加学習を行わずに用いる．

Vol.2017-DBS-165 No.33
Vol.2017-IFAT-128 No.33
2017/9/20

	ALL			直列			並列		
	ROUGE								
	1	2	L	1	2	L	1	2	L
PG	21.54	**4.84**	21.27	19.76	**4.20**	19.51	26.38	**6.46**	26.01
PG+Cov	**22.60**	4.42	**22.35**	**21.03**	4.11	**20.83**	26.87	5.25	26.50
PG+Cov+CovReset	22.23	4.01	22.04	20.65	3.94	20.45	26.77	4.94	26.40
PG+Cov+DecReset	22.41	4.21	22.17	20.93	3.95	20.72	26.51	4.88	26.16
PG+Cov+DecAllReset	22.01	3.97	21.77	20.22	3.56	20.01	**26.98**	5.08	**26.63**
Combination	22.62	4.37	22.38	21.03	4.11	20.83	26.98	5.08	26.63

表 4 実験結果

	ALL			直列			並列		
	ROUGE								
	1	2	L	1	2	L	1	2	L
PG	2.70	0.87	2.70	2.45	0.76	2.45	3.27	1.11	3.27
PG+Cov	**3.78**	1.08	**3.73**	**3.58**	1.24	**3.54**	**4.33**	0.61	**4.20**
PG+Cov+CovReset	2.73	0.90	2.71	2.52	0.80	2.49	3.29	**1.18**	3.29
PG+Cov+DecReset	3.20	0.55	3.20	3.26	0.60	3.26	3.10	0.44	3.10
PG+Cov+DecAllReset	3.41	0.72	3.39	3.41	0.79	3.38	3.32	0.51	3.32

表 5 3 文目のみを評価した場合の実験結果.

5. 日本語の 3 行要約実験

我々はベースラインのモデルとして See ら [6] のモデル，提案手法として 4.4 節で提案したモデルを用いて実験を行った．See ら [6] と同様に，モデルの隠れ層は 256 次元，単語埋め込みベクトルは 128 次元に設定した．語彙サイズもまたソース側とターゲット側ともに 50,000 に設定した．バッチサイズは 16 に設定した．記事と要約の単語分割には MeCab v0.996 [*2] (IPAdic v2.7.0) を用いた．

See ら [6] ら同様に，いくつかの制約を付与した．学習時，テスト時には記事の単語数を 400 単語に制限し，デコードの際は出力単語数を 100 単語と 120 単語に学習時とテスト時にそれぞれ制限した．モデルの学習時には Adagrad [12] を学習率 0.15 で用い，gradient clipping の最大勾配ノルムを 2 にした．モデルの学習では，はじめに Attention Encoder-Decoder を用いた Hybrid Pointer-Generator の学習を行い，追加学習で Coverage ベクトルの学習を行った．

評価指標には ROUGE [13] の ROUGE-1，ROUGE-2，ROUGE-L の F_1 スコアを用いた．ROUGE は要約システムの自動評価法のひとつであり，システム要約と参照要約間での N グラム一致の割合を示す指標である．ROUGE-1 および ROUGE-2 はそれぞれ，uni-gram と bi-gram に着目した評価指標である．ROUGE-L は最長一致部分文字列に着目した評価指標である．

評価には 3 節で行ったアノテーション結果より得られた 2 タイプのタグごとにも分けて行った．

5.1 結果

実験結果を**表 4** に示す．Combination は 2 つのタグそれぞれのタイプにおけるベストなモデルを組み合わせたものである．ここでは，PG+Cov と PG+Cov+DecAllReset のモデルを用い，“直列” タイプと “並列” タイプのテストデータをそれぞれで生成した結果である．テストデータ全体ではベースラインの Coverage ベクトルを用いるモデルが良い結果となった．“直列” タイプのみでのテストデータでは提案手法すべてに対して，ベースラインの Coverage ベクトルを用いたモデルを下回る結果となった．“並列” タイプのみでのテストデータにおいては提案手法のデコードの状態を戻す手法において精度の向上が見られた．

システム要約と参照要約の 3 文目のみでの評価結果を**表 5** に示す．表 4 の要約全体を見たときの結果とは異なり，3 文目のみに着目した場合，ベースラインの PG+Cov が全体的に良い結果が得られた．

6. 考察

“直列” タイプの文書構造を持つデータは今回収集したデータ中の 30 %であるため，“並列” タイプに比べて低い精度となっている．

CovReset の手法の場合は，Coverage の情報がなくなるので，同じ文を出力することが多くなり，精度が悪くなったと考えられる．もともと Coverage は Encoder-Decoder における繰り返しを防ぐために用いられているので，Coverage の情報を戻すと同じ文が出るようになる．また，その他の出力でも 2，3 文目に似たような出力が観測されため，Decoder が持つ出力単語の情報は出力単語が長くなるほど曖昧になっていくと考えられる．

表 6 に Decoder の状態を戻す手法の出力例を示す．3 つのシステムの出力は，3 文目では正解と関係ないものを出力してい，人名が違う人の名前となっている．DecReset と

[*2] https://github.com/taku910/mecab

PG+Cov	日本人 が 米国 で 優勝 を 達成 し た 丸山 茂樹 日本 中 の ゴルフ ファン を 大いに 喜ば せ た の は 時間 の 問題 だ という 「 リオデジャネイロ 五輪 も 期待 が 持て ます し 、 これから が 楽しみ 」 と 述べ た
DecReset	日本人 が 米国 で 優勝 を 達成 し た 丸山 茂樹 日本 中 の ゴルフ ファン を 大いに 喜ば せ た の は 時間 の 問題 だ という 2 人 目 の 快挙 に 、 複数 回 優勝 を 達成 し た こと が 勝因 だ という
DecAllReset	日本人 が 米国 で 優勝 を 達成 し た 丸山 茂樹 丸山 茂樹 と の 熾烈 な プレーオフ を 制し 、 米 ツアー 2 勝 目 を 挙げ た 松山 英樹 が 複数 回 優勝 を 達成 し て い た 丸山 さん
正解	松山 英樹 が 熾烈 な プレーオフ を 制し 、 米 ツアー 2 勝 目 を 挙げ た 丸山 茂樹 は 「 米 ツアー 2 勝 目 、 本当に おめでとう ござい ます 」 と 祝福 「 プレッシャー が かかる 中 で どう 集中 を 高めれ ば いい か 知っ てる 」 と 称賛 し た

表 6 Decoder の状態を変更した場合の例文.

DecAllReset では 3 文目に正解の内容と関連のある内容を出力できている．Decoder の状態を戻すことにより，ソース側の情報と関連のある文を 3 文目でも出力できてはいるが，正解要約の 3 文目とは異なった内容である．

表 4 における実験結果では，DecAllReset が “並列” タイプにおいて精度の向上が見られた．長文を生成する際は Decoder の状態を文ごとにリセットすることで，Encoder の情報を再度明示的に与えられるので精度が向上したと考えられる．Coverage ベクトルの存在が，今まで注目した単語情報を保持しているのでこのようなことが可能である．“並列” タイプの場合は，今まで出していない情報を出せば良いので全体的な精度が向上したと考えられる．しかし，表 5 の結果から，Decoder の状態をリセットしたことにより，3 文目のみで評価した際にはベースラインの PG+Cov より精度が低い結果となっている．毎文ごとに Decoder の情報をリセットすることで，本来保持しているはずの今まで出力した単語情報から得られる話の流れがないため，精度が低下していると考えられる．

今回提案した手法において，“直列” タイプにおける精度の向上が見られなかった．“直列” タイプは 1 文目と 2 文目の話の流れを掴まなければならないが，提案法では Decoder の状態を戻すのでこのような情報が考慮できなくなるためだと考えられる．よって，ベースラインである PG+Cov が一番良い精度となっている．

3 文のような長い系列を Decoder を用いて出力する際は，1 文ごとに Decoder に次にどのような文を出力すれば良いのか明示的に渡す必要があると考えられる．

7. 終わりに

本研究では Livedoor News からデータを収集し，3 行要約に注目したニュース文書要約データセットの構築を行った．さらに，3 行要約の文書構造に着目したアノテーションを行なった．本研究で構築したデータセットは公開する予定である．また，アノテーションで付与されたタグの文構造に適応したモデルの検討を行った．

本研究では各文書構造を考慮した学習を行わなかった．今まで出力した文構造を考慮しつつ学習するなどの必要がある．これから，追加情報（カテゴリーとサブカテゴリー）を用い，自動で “直列”’ と “並列” タグを認識し，文書構造を考慮しながら学習を行うことを検討する．

参考文献

[1] Nallapati, R., Zhai, F. and Zhou, B.: SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents, *Proceedings of AAAI 2016*, pp. 3075–3081 (2016).
[2] Li, C., Qian, X. and Liu, Y.: Using Supervised Bigram-based ILP for Extractive Summarization, *Proceedings of ACL 2013*, pp. 1004–1013 (2013).
[3] Rush, A. M., Chopra, S. and Weston, J.: A Neural Attention Model for Abstractive Sentence Summarization, *Proceedings of EMNLP 2015*, pp. 379–389 (2015).
[4] Sutskever, I., Vinyals, O. and Le, Q. V.: Sequence to Sequence Learning with Neural Networks, *Proceedings of NIPS 2014*, pp. 3104–3112 (2014).
[5] Nallapati, R., Xiang, B. and Zhou, B.: Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond, *Proceedings of CoNLL 2016*, pp. 280–290 (2016).
[6] See, A., Liu, P. J. and Manning, C. D.: Get To The Point: Summarization with Pointer-Generator Networks, *Proceedings ACL 2017*, pp. 1073–1083 (2017).
[7] Jean, S., Cho, K., Memisevic, R. and Bengio, Y.: On Using Very Large Target Vocabulary for Neural Machine Translation, *Proceedings of ACL 2015*, pp. 1–10 (2015).
[8] Hochreiter, S. and Schmidhuber, J.: Long Short-Term Memory, *Neural Comput.*, Vol. 9, No. 8, pp. 1735–1780 (1997).
[9] Bahdanau, D., Cho, K. and Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate, *Proceedings of ICLR 2014* (2014).
[10] Vinyals, O., Fortunato, M. and Jaitly, N.: Pointer Networks, *Proceedings of NIPS 2014*, pp. 2692–2700 (2015).
[11] Tu, Z., Lu, Z., Liu, Y., Liu, X. and Li, H.: Modeling Coverage for Neural Machine Translation, *Proceedings of ACL 2016*, pp. 76–85 (2016).
[12] Duchi, J., Hazan, E. and Singer, Y.: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *J. Mach. Learn. Res.*, pp. 2121–2159 (2011).
[13] Lin, C.-Y.: ROUGE: A Package for Automatic Evaluation of Summaries, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74–81 (2004).