ニューラルヘッドライン生成における誤生成問題の改善

清野 舜 [1]　高瀬 翔 [2]　鈴木 潤 [2]　岡崎 直観 [3]　乾 健太郎 [1,4]　永田 昌明 [2]
[1] 東北大学　[2]NTT コミュニケーション科学基礎研究所　[3] 東京工業大学　[4] 理研 AIP
[1]{kiyono, inui}@ecei.tohoku.ac.jp　[3]okazaki@c.titech.ac.jp
[2]{takase.sho, suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

1 はじめに

アテンション付きエンコーダデコーダモデル（EncDec）[10] は，提案されて以来，機械翻訳 [1]，対話システム [9]，ヘッドライン生成 [7] など様々なタスクへと適用されてきた．現在，EncDec は強いベースライン手法としてこれらのタスクに用いられている．一方で，以下の例の様に，EncDec は同じフレーズ（単語）を繰り返し生成したり，全く関係のない単語を出力することが知られている．加えて，重要な語句が欠落した文を出力することもある．本研究では，この現象を「誤生成問題」と呼び，要約タスク上でこの問題の解決に取り組む．

(1) 繰り返し生成

Gold: duran duran group fashionable again
EncDec: duran duran duran duran

(2) 無関係な単語の生成

Gold: u.s. troops take first position in serb-held bosnia
EncDec: precede sarajevo

(3) 重要な語句の欠損

Gold: graf says goodbye to tennis due to injuries
EncDec: graf retires

機械翻訳タスクは「損失なし生成」タスクであり，入出力文間で情報が一致することを仮定している．この仮定にもとづいて誤生成問題を解決する Coverage モデル [5, 13] や Reconstruction モデル [12] が提案されている．一方で，要約は「損失あり圧縮生成」である [6, 11]．例えば，Rush ら [7] が提案したヘッドライン生成タスクでは，入力文から日時や人物名といった情報が省略されることが多い．このため，上記のモデルを要約タスクに適用することは難しい．そこで，本研究では，損失あり圧縮生成に適した，誤生成問題を解決するモデルを考案する．

損失あり圧縮生成においても，誤生成問題の解決に取り組んでいる研究はある．例えば，Zhou ら [14] は，エンコーダ側に選択式ゲート機構を追加し，入力文から適切な単語を選択するモデルを提案した．これは，重要な語句の欠損や無関係な単語の生成を低減すると期待される．また，Suzuki ら [11] は，与えられた入力文から，出力文の単語頻度の上限を予測することで，繰り返し生成に対処した．しかし，これらの対策手法は誤生成問題の一部の現象に着目して考案された方法であり，上記の誤生成問題の全ての現象に対応していない．

本研究では，全ての誤生成問題の統一的な解決に取り組む．具体的には，EncDec に拡張モジュールを用意し，入力文と出力文とのトークン単位の対応関係をモデル化することで，誤生成問題の解決を図る．このモジュールを Source-side Prediction Module（SPM）と呼ぶ．提案手法は，入出力間の対応関係を考慮しながら生成を行うため，同じ情報を出力し続けるという **(1)** 繰り返し生成問題を解消する．同様に，出力単語に対応する入力側の単語を考慮しているので，入力に無関係な単語を生成してしまうという **(2)** 無関係な単語の生成も防げる．また，入力から重要な情報を選択する機構を取り入れることで，**(3)** 重要な語句の欠損を解決する．Rush ら [7] によって提案されたヘッドライン生成データセット上で実験を行い，提案手法の有効性を検証する．結果から，提案手法は既存手法よりも性能が良く，また，今日までに提案されている手法の中で最高性能を達成できることを示す．また，提案手法はアライメントの正解データ無しに，入出力文のトークン単位の対応関係を学習できる．

2 RNN エンコーダデコーダモデル

本節では，提案手法のベースライン手法である EncDec を定義する[*1]．EncDec への入力 $\boldsymbol{X}$ は 1-hot ベクトルから成る長さ I の系列である．$\boldsymbol{x}_i \in \{0,1\}^{V_s}$ は $\boldsymbol{X}$ 中 i 番目の 1-hot ベクトルとする．また，V_s は入力側語彙 $\mathcal{V}_s$ の語彙数とする．以降，$\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I)$ を $\boldsymbol{x}_{1:I}$ と略記する．同様に出力側について，$\boldsymbol{y}_j \in \{0,1\}^{V_t}$ は出力 $\boldsymbol{Y}$（長さ J）の j 番目の 1-hot ベクトルと定義する．V_t は出力側語彙 $\mathcal{V}_t$ の語彙数とする．ここで，$\boldsymbol{Y}$ 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，$\boldsymbol{y}_0$ が $\langle bos \rangle$ であり，$\boldsymbol{y}_{J+1}$ が $\langle eos \rangle$ である．また，Rush ら [7] の提案したヘッドライン生成タスクでは常に $I > J$ が成り立つ．

EncDec は以下の条件付き確率をモデル化する．

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J+1} p(\boldsymbol{y}_j|\boldsymbol{y}_{0:j-1}, \boldsymbol{X}) \quad (1)$$

EncDec は $\boldsymbol{x}_{1:I}$ を受け取り，隠れ層の系列 $\boldsymbol{h}_{1:I}$ へとエンコードする．ここで，$\boldsymbol{h}_i \in \mathbb{R}^H$ であり，H を隠れ層の次元数と定義する．その後，AttnDec をアテンション付きデコーダの全ての処理を表す関数だとする．各時刻において，以下の様に最終隠れ層 $\boldsymbol{z}_j \in \mathbb{R}^H$ が計算される．

$$\boldsymbol{z}_j = \mathrm{AttnDec}(\boldsymbol{y}_{j-1}, \boldsymbol{h}_{1:I}) \quad (2)$$

EncDec は単語確率分布 $\boldsymbol{o}_j \in \mathbb{R}^{V_t}$ から単語を生成する．

$$\boldsymbol{o}_j = \mathrm{softmax}(\boldsymbol{W}_o \boldsymbol{z}_j + \boldsymbol{b}_o), \quad (3)$$

ここで，$\boldsymbol{W}_o \in \mathbb{R}^{V_t \times H}$ はパラメータ行列であり，$\boldsymbol{b}_o \in \mathbb{R}^{V_t}$ はバイアス項である．

$\mathcal{D}$ 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 $\hat{\theta}$ を，次のコスト関数

[*1] EncDec モデルの詳細は Luong らのモデル [4] に従う．

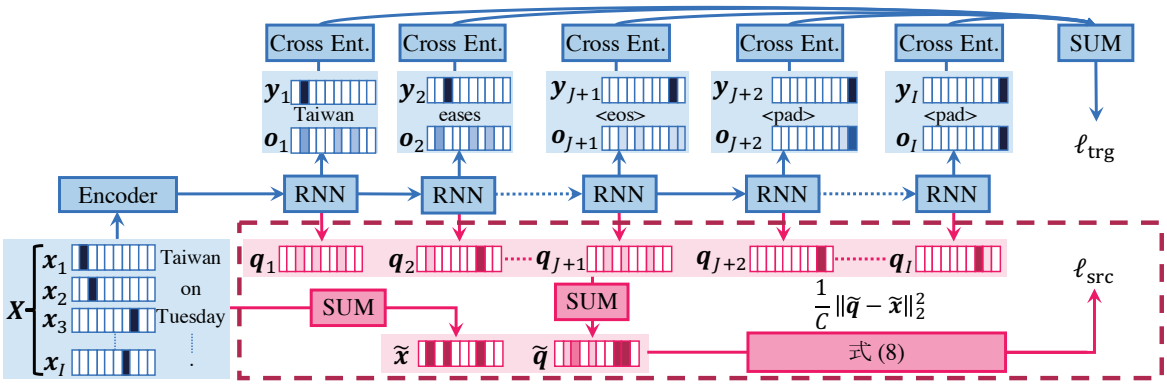


図 1: 提案手法（**EncDec+SPM**）の全体像：破線で区切られたモジュールが SPM を表す．

$G_1(\theta)$ を $\mathcal{D}$ 上で最小化することによって求める．

$$G_1(\theta) = \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \ell_{\mathrm{trg}}(\boldsymbol{Y},\boldsymbol{X},\theta),$$
$$\ell_{\mathrm{trg}}(\boldsymbol{Y},\boldsymbol{X},\theta) = -\log\Big(p(\boldsymbol{Y}|\boldsymbol{X},\theta)\Big) \quad (4)$$

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サーチで探索する．

3 提案手法：入力側予測モデル

1 節では，入出力間の対応関係のモデル化が，誤生成問題の解決に重要であることを述べた．提案手法では，各出力トークンについて，それに対応する入力側のトークンを予測するという枠組みによって，モデル化を行う．

図 1 に提案手法の全体像を示す．EncDec のデコーダは，各時刻 j に出力側の単語確率分布 $\boldsymbol{o}_j \in \mathbb{R}^{V_t}$ と入力側の単語確率分布 $\boldsymbol{q}_j \in \mathbb{R}^{V_s}$ を同時に計算する．ここで，入力側の予測する単語は，出力側の予測する単語に対応することを期待する．また，SPM は出力系列長 J に関わらず，入力系列 I の長さまで確率分布を計算する．このとき，出力側では専用のトークンである $\langle pad\rangle$ を正解系列と定め，$\langle eos\rangle$ 後の $\boldsymbol{o}_j$ からは $\langle pad\rangle$ が出力されるとする．これにより，出力系列の長さは常に入力系列と一致するため，ヘッドライン生成のような損失あり圧縮生成タスクにおいても，トークン単位の対応関係を考慮できる．加えて，入力文から省略されるトークンについても，$\langle pad\rangle$ に対応させることができる．

ヘッドライン生成のベンチマークデータには，アライメント情報のような，入出力間のトークン単位の対応関係は付与されていない．そのため，毎時刻に各 $\boldsymbol{q}_j$ に対して教師データを用意することは困難である．そこで本研究では，トークン単位の対応関係を教師なし学習の枠組みで学習することを試みる．具体的には，トークン単位ではなく文単位の尤度関数を導入し最適化を行う．

以下では，モデルの詳細を定義する．

■モデルの定義　SPM は各時刻 j において，出力側の単語確率分布 $\boldsymbol{q}_j \in \mathbb{R}^{V_s}$ を以下の様に計算する．

$$\boldsymbol{q}_j = \mathrm{softmax}(\boldsymbol{W}_q\boldsymbol{z}_j + \boldsymbol{b}_q) \quad (5)$$

ここで，$\boldsymbol{W}_q \in \mathbb{R}^{V_s\times H}$ はパラメータ行列であり，$\boldsymbol{b}_q \in \mathbb{R}^{V_s}$ はバイアス項である．2 節で述べたように，EncDec は $\boldsymbol{z}_j$ から出力側の単語確率分布 $\boldsymbol{o}_j$ を計算する（式 3）．そのため，EncDec に SPM を組み合わせる場合，同じベクトル $\boldsymbol{z}_j$ から，入力と出力の単語確率分布が同時に予測される．

今，$\boldsymbol{Y}' = \boldsymbol{y}_{0:I}$ を $\boldsymbol{Y}$ と $\langle pad\rangle$ の 1-hot ベクトル系列を結合した系列として定義する．ここで，$\boldsymbol{y}_{J+1}$ は $\langle eos\rangle$ の 1-hot ベクトルであり，各 $j \in \{J+2,\ldots,I\}$ について $\boldsymbol{y}_j$ は $\langle pad\rangle$ に対応する 1-hot ベクトルである[*2]．

$\tilde{\boldsymbol{x}}$ と $\tilde{\boldsymbol{q}}$ をそれぞれ入力 1-hot ベクトル系列 $\boldsymbol{x}_{1:I}$ の和（$\tilde{\boldsymbol{x}} = \sum_{i=1}^{I}\boldsymbol{x}_i$）と SPM による予測 $\boldsymbol{q}_{1:I}$ の和（$\tilde{\boldsymbol{q}} = \sum_{j=1}^{I}\boldsymbol{q}_j$）とする．ここで，$\tilde{\boldsymbol{x}}$ は入力に登場する単語全てをまとめたベクトル表現（または，Bag-of-Words 表現）である．

つまり，SPM は以下のような，入力文に対するヘッドライン及び入力文の単語集合の条件付き確率をモデル化する．

$$p(\boldsymbol{Y}',\tilde{\boldsymbol{x}}|\boldsymbol{X}) = p(\tilde{\boldsymbol{x}}|\boldsymbol{Y}',\boldsymbol{X})p(\boldsymbol{Y}'|\boldsymbol{X}) \quad (6)$$

ここで，$p(\boldsymbol{Y}'|\boldsymbol{X})$ は式 1 の $J+1$ を I に置換したものである．

また，$p(\tilde{\boldsymbol{x}}|\boldsymbol{Y}',\boldsymbol{X})$ を次の様に定義する．

$$p(\tilde{\boldsymbol{x}}|\boldsymbol{Y}',\boldsymbol{X}) = \frac{1}{Z}\exp\left(\frac{-\|\tilde{\boldsymbol{q}}-\tilde{\boldsymbol{x}}\|_2^2}{C}\right) \quad (7)$$

ここで Z は正規化項であり，C は確率分布 $p(\tilde{\boldsymbol{x}}|\boldsymbol{Y}',\boldsymbol{X})$ の影響を制御するハイパーパラメータである．$\tilde{x}$ が文中の単語全てを含むことから，式 7 は文単位の尤度関数とみなせる．

■モデルの訓練　今，γ を SPM によって新しく追加されたパラメータとすると，SPM のコスト関数は次のように定義される．

$$\ell_{\mathrm{src}}(\tilde{\boldsymbol{x}},\boldsymbol{X},\boldsymbol{Y}',\gamma,\theta) = -\log\Big(p(\tilde{\boldsymbol{x}}|\boldsymbol{Y}',\boldsymbol{X},\gamma,\theta)\Big).$$

式 7 から，ℓ_{src} は次のように展開できる．

$$\ell_{\mathrm{src}}(\tilde{\boldsymbol{x}},\boldsymbol{X},\boldsymbol{Y}',\gamma,\theta) = \frac{1}{C}\|\tilde{\boldsymbol{q}}-\tilde{\boldsymbol{x}}\|_2^2 + \log(Z). \quad (8)$$

ここで，右辺の第 2 項は，γ と θ に依存しない定数のため，学習時の目的関数としては無視できる．

今回，SPM と EncDec を同時に最適化することを考える．そのため，SPM のコスト（ℓ_{src}）と EncDec のコスト（ℓ_{trg}）の和によって全体のコスト関数 G_2 を定義する．

$$G_2(\theta,\gamma) = \tfrac{1}{|\mathcal{D}|}\textstyle\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}}\Big(\ell_{\mathrm{trg}}(\boldsymbol{Y}',\boldsymbol{X},\theta) + \ell_{\mathrm{src}}(\tilde{\boldsymbol{x}},\boldsymbol{X},\boldsymbol{Y}',\gamma,\theta)\Big) \quad (9)$$

直観的には，この学習の枠組みを 2 つのコスト $\ell_{\mathrm{trg}},\ell_{\mathrm{src}}$ を考慮するマルチタスク学習として解釈できる．

[*2] $J+1=I$ のときに限り $\boldsymbol{Y}'=\boldsymbol{Y}$ である．

Source Vocab. Size V_s	5131
Target Vocab. Size V_t	5131
Word Embedding Size D	200
Hidden State Size H	400
RNN Cell	Long Short-Term Memory（LSTM）[2]
Encoder RNN Unit	2 層双方向 LSTM
Decoder RNN Unit	アテンション付き 2 層 [4]
Optimizer	Adam
初期学習率	0.001
学習率減衰	Epoch9 後，各 Epoch で 0.5 倍
ℓ_{src} の重み C	10
ミニバッチサイズ	256（各 Epoch でシャッフル）
Gradient Clipping	5
終了条件	max 15 epochs with early stopping
正則化	Dropout（rate 0.3）
ビームサーチ	ビーム幅 20 & length normalization

表 1: モデルの詳細及びハイパーパラメータ

■推論　推論時には，通常の EncDec モデルの様に，⟨*eos*⟩ を出力するまで探索を行えば良いため，⟨*pad*⟩ を出力する必要がない．そのため，SPM を用いる場合も，推論における計算コストは EncDec と同じである．

4　実験

4.1　実験設定

■データセット　ヘッドライン生成タスクの訓練・評価には，Rush ら [7] が提案したベンチマークデータを用いるのが一般的である．同ベンチマークデータは，約 380 万文の訓練データ，20 万文の開発データ，40 万文のテストデータで構成されている．ここで，低頻度語は ⟨*unk*⟩ で置換されているため，評価時に ⟨*unk*⟩ を出力することが正解として扱われる．本稿では，より現実的な設定で実験を行うため，公開されているデータ生成・前処理スクリプト[*3]を変更し，⟨*unk*⟩ による置換をせずにデータセットを構築した．その後，開発データ 20 万文から 8000 文，1 万文をサンプリングし，それぞれ新しい開発データとテストデータとした．また，既存手法との比較のため，Rush ら [7] と同じテストデータ上での評価も行った．以降，我々のテストデータを Gigaword Test（Ours），Rush ら [7] のテストデータを Gigaword Test（Rush）と表記する．

■比較手法　提案手法の有効性を検証するため，SPM をそれぞれ (1) EncDec (2) 現在の最高性能のモデル (EncDec+sGate) に組み合わせて実験を行った．具体的には，以下の 4 モデル間での実験を行った．

EncDec 2 節で定義したエンコーダデコーダモデル

EncDec+sGate Zhou ら [14] が提案した選択式ゲート機構 (sGate) と EncDec を組み合わせたモデル

EncDec+SPM 3 節で定義した SPM と EncDec を組み合わせたモデル

EncDec+sGate+SPM EncDec+sGate モデルと SPM を組み合わせたモデル

■実装の詳細　表 1 に本稿で用いた実験の設定をまとめた．モデルの次元数，最適化手法などについては，文献 [6, 7, 11] などで一般的に用いられているものを採用した．

語彙については，近年のニューラル機械翻訳の慣習にならい，Byte-Pair-Encoding（BPE）[8] を用いて構築した[*4]．BPE の merge operation を 5000 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．入力側語彙 $\mathcal{V}_s$ と出力側語彙 $\mathcal{V}_t$ の両方で共通の語彙を用いた．

[*3] https://github.com/facebookarchive/NAMAS

[*4] https://github.com/rsennrich/subword-nmt

	Gigaword Test (Ours)			Gigaword Test (Rush)		
	RG-1	RG-2	RG-L	RG-1	RG-2	RG-L
EncDec	45.74	23.80	42.95	34.52	16.77	32.19
EncDec+sGate	45.98	24.17	43.16	35.00	17.24	32.72
EncDec+SPM†	46.18	24.34	43.35	35.17	17.07	32.75
EncDec+sGate+SPM†	**46.41**	**24.58**	**43.59**	**35.79**	**17.84**	**33.34**
EncDec+sGate+SPM (5 Ensemble)†	**47.16**	**25.34**	**44.34**	**36.23**	**18.11**	**33.79**
ABS [7]	-	-	-	29.55	11.32	26.42
SEASS [14]	-	-	-	36.15	17.54	33.63
DRGD [3]	-	-	-	36.27	**17.57**	33.62
WFE [11]	-	-	-	**36.30**	17.31	**33.88**

表 2: 実験結果: † は提案手法を示す．

4.2　実験結果

表 2 に比較実験の結果を示す．ここで，RG-1，RG-2，RG-L は ROUGE-1，ROUGE-2，ROUGE-L の F1 スコアである．表は横二重線で上部と下部に分割されている．上部は提案手法とベースライン手法を比較するため，また，下部は既存研究の値を参考にするために示す．上部は 4.1 節で述べたように前処理が異なることに加えて，BPE を用いて語彙の構築を行っている．これらの実験条件の違いから，上部と下部の値の比較によって手法間の優劣を議論することはできない．

表 2 の上部より，両方のテストデータで EncDec+SPM が EncDec と EncDec+sGate を上回る性能を示していることがわかる．このことから，SPM が EncDec の性能を改善することがわかる．また，EncDec+sGate+SPM が全体の中でも最も良い性能を示したことから，SPM は既存の最高性能のモデル（EncDec+sGate）の性能を向上させられることがわかる．

提案手法と表 2 の下部を比較すると，一部の既存手法は提案手法に比べて高い ROUGE スコアを示しているとわかる．この差は，語彙の構築手法の差によるものと考えられる．我々の実験設定では，BPE を用いて語彙を構築しており，語彙に ⟨*unk*⟩ を含まないため，⟨*unk*⟩ を出力することはできない．そのため，Gigaword Test（Rush）で ⟨*unk*⟩ が正解とされる箇所は，仮に正しい生成をしていたとしても必ず推定誤りと判定されるため，ROUGE スコアが不当に減点された可能性がある．

5　分析

■システムの生成例　図 2 にシステムの実際の生成例を示す．図から，EncDec と比較すると，SPM は同じフレーズの出力，関係の無い単語の出力，重要な語句の欠落といった全ての誤生成問題を減らせていることが定性的にわかる．

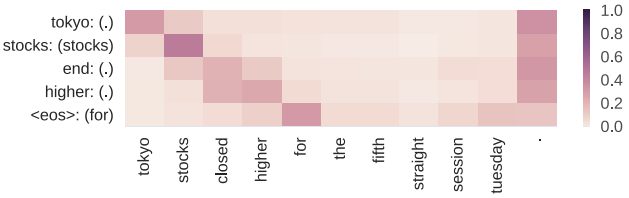
■SPM とアテンションの可視化　SPM が入出力間のトークン単位の対応関係を学習できているかを確かめるため，SPM の予測を可視化する．その際，一般にアライメントのように振る舞うとされる [1] アテンション分布との比較を行う．

入出力ペア $(\boldsymbol{X}, \boldsymbol{Y})$ を各 EncDec と EncDec+SPM に入力し，EncDec+SPM の入力側予測 $\boldsymbol{q}_{1:I}$ と EncDec のアテンション分布 $\boldsymbol{\alpha}_{1:J}$ を得る．アテンション分布は次式で計算する．

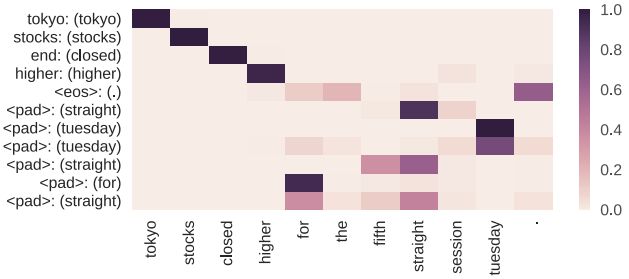
$$\boldsymbol{\alpha}_j[i] = \frac{\exp(\boldsymbol{h}_i^\top \boldsymbol{W}_\alpha \vec{\boldsymbol{z}}_j)}{\sum_{i=1}^{I} \exp(\boldsymbol{h}_i^\top \boldsymbol{W}_\alpha \vec{\boldsymbol{z}}_j)} \tag{10}$$

(1) 繰り返し生成			
Gold:	duran duran group fashionable again	Gold:	community college considers building $ ## million technology
EncDec:	duran duran duran duran	EncDec:	college college colleges learn to get ideas for tech center
EncDec+SPM:	duran duran fashionably cool once again	EncDec+SPM:	l.a. community college officials say they 'll get ideas
(2) 無関係な単語の生成			
Gold:	u.s. troops take first position in serb-held bosnia	Gold:	northridge hopes confidence does n't wane
EncDec:	precede sarajevo	EncDec:	csun 's csun
EncDec+SPM:	u.s. troops set up first post in bosnian countryside	EncDec+SPM:	northridge tries to win northridge men 's basketball team
(3) 重要な語句の欠損			
Gold:	graf says goodbye to tennis due to injuries	Gold:	new york 's primary is most suspenseful of super tuesday races
EncDec:	graf retires	EncDec:	n.y.
EncDec+SPM:	german tennis legend steffi graf retires	EncDec+SPM:	new york primary enters most suspenseful of super tuesday contests

図 2: ベースライン手法 **EncDec** と提案手法 **EncDec+SPM** の生成文の比較：“Gold” は正解のヘッドラインを表す．



(a) EncDec のアテンション分布



(b) EncDec+SPM の入力側予測

図 3: **EncDec** と **EncDec+SPM** の可視化：x 軸と y 軸はそれぞれ入力文と出力文に対応する．y 軸の括弧内のトークンは，アラインされた入力側のトークンを表す．

ここで $\boldsymbol{W}_\alpha \in \mathbb{R}^{H\times H}$ はパラメータ行列，$\boldsymbol{\alpha}_j[i]$ は $\boldsymbol{\alpha}_j$ の i 番目の要素を表す．また，$\tilde{\boldsymbol{z}} \in \mathbb{R}^H$ はデコーダの隠れ層である．入力側の予測に関しては，各 j について $\boldsymbol{q}_j$ から $\boldsymbol{x}_i \in \boldsymbol{X}$ に対応する値を抽出した．

図 3 にヒートマップを示した．テストデータには Gigaword Test (Ours) を用いた．ここで，y 軸括弧内のトークンは，その時刻で出力側のトークンにアラインされた入力側のトークンを表す．トークンのアライン先については，アテンション分布（図 3a）に基づく方法では，アテンション確率が最も高かったトークンを選択し，SPM の予測（図 3b）では，全入力側語彙 $\mathcal{V}_s$ の中から最も出力確率が高いトークンを選択した．

アテンション分布（図 3a）では，値のほとんどが文末に集中している．このことから，アテンション分布は，入出力間のトークン単位の対応関係を正確に表していないと想定できる．例えば，出力側の “tokyo” や “end” は入力側のピリオドにアラインされている．一方で，SPM の予測（図 3b）は対応関係を捉えることに成功している．例えば，入力側の “tokyo stocks closed higher” は出力側の “tokyo stocks end higher” にアラインされている．各値がほぼ離散的な値を示すことから，SPM は高い確度を持ってアラインする先を決めていると分かる．また，SPM は “tuesday” や “straight” など，ヘッドラインで重要度の低い単語を $\langle pad \rangle$ に対応させている．これらの事実から，SPM はアテンション分布と比較して，より良いトークン単位の対応関係を予測しているとわかる．SPM の訓練時にアライメントの正解データを与えていないにも関わらず，対応関係を学習できているのは，特筆すべき結果である．

6 おわりに

本稿では，ヘッドライン生成のような損失あり圧縮生成タスクに対して，誤生成問題を減らす手法を提案した．提案手法である SPM は，入力と出力間のトークン単位の対応を予測するモデルである．実験では，ヘッドライン生成のベンチマークデータ上で，既存の最高性能の手法を上回る性能を達成し，誤生成問題の減少を確認した．また，SPM はアライメント情報を与えられることなく，入出力間のトークン単位の対応関係を学習できることを示した．

謝辞

株式会社 Preferred Networks の小林颯介氏には研究の遂行に有益なアドバイスを頂いた．また本研究は，文部科学省科研費 15H01702，15H05318 の支援を受けたものである．

参考文献

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*, 2015.
[2] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.
[3] Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. Deep Recurrent Generative Decoder for Abstractive Text Summarization. In *EMNLP*, pp. 2081–2090, 2017.
[4] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*, pp. 1412–1421, 2015.
[5] Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. Coverage Embedding Models for Neural Machine Translation. In *EMNLP*, pp. 955–960, 2016.
[6] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In *SIGNLL*, pp. 280–290, 2016.
[7] Alexander M. Rush, Sumit Chopra, and Jason Weston. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*, pp. 379–389, 2015.
[8] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *ACL*, pp. 1715–1725, 2016.
[9] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural Responding Machine for Short-Text Conversation. In *ACL & IJCNLP*, pp. 1577–1586, July 2015.
[10] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*, pp. 3104–3112, 2014.
[11] Jun Suzuki and Masaaki Nagata. Cutting-off Redundant Repeating Generations for Neural Abstractive Summarization. In *EACL*, pp. 291–297, 2017.
[12] Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. Neural Machine Translation with Reconstruction. In *AAAI*, pp. 3097–3103, 2017.
[13] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling Coverage for Neural Machine Translation. In *ACL*, pp. 76–85, 2016.
[14] Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. Selective Encoding for Abstractive Sentence Summarization. In *ACL*, pp. 1095–1104, 2017.