大規模データを用いた日本語文圧縮

長谷川 駿[1]，菊池 悠太[2]，高村 大也[1]，奥村 学[1]
[1] 東京工業大学，[2] 株式会社 Preferred Networks
{hasegawa.s, takamura, oku}@lr.pi.titech.ac.jp[1], kikuchi@preferred.jp[2]

1 はじめに

文圧縮とは，任意の一文からその大意をとらえた文法的な短い文を生成する研究課題である．頑健な文圧縮器は，デバイス幅に合わせた字幕表示に用いられたり，文抽出機構と組み合わせることにより文書要約器の中で用いられたりしている [1，8]．本稿では日本語を対象とした削除型文圧縮器の研究に取り組む．削除型文圧縮とは原文から特定の単位（ここでは単語）を削除することで圧縮文を生成する方法であり，原文中にない単語の出力を許す方法と比べて間違った情報を出してしまう可能性が低く，探索空間も小さいという利点がある．英語における高性能な削除型文圧縮器 [2, 3] は大規模な訓練データを用いて学習を行っている．

しかし，大規模な訓練データを人手で作成することは困難である．そのため英語においては新聞記事のヘッドライン（以降 H と表す）が記事の一文目（以降 S と表す）の要約となっていると考えられる記事を抽出し，H を手がかりに S の圧縮文を作成することで原文すなわち S と圧縮文の組を作り，大規模な訓練データを自動作成している [2]．この手法では単語の原形を用いて H と S で共有されている内容語[1] を同定し，それらを全て含む S の部分依存木を S の圧縮文としている．

日本語においても高性能な文圧縮器を構築するため大規模な訓練データを自動作成することが望まれるが，この部分依存木の抽出方法や記事を抽出する際の規則は英語に合わせて作られていることから，日本語特有の特徴に合わせ以下の四点の変更を行う必要がある．(1) 日本語は省略，動詞の名詞化が多用される．単語の原形を用いてこれらを同定することは困難であるため，文字による同定を行う．(2) 日本語では係り受けの単位が一般的に文節であるため，部分木は文節より細かい削除を表現できない．そこで，文節よりも細かい削除を表現するため，H 中の省略表現と S 中の元の表現を同定し，省略表現を圧縮文に用いる．(3) 主要部終端型である日本語では依存木の根が文末となる．そこで，部分木の根が文末としての文法上の制約を満たすよう圧縮文を抽出し S の依存木の根が削除される圧縮文を作成する．(4) 日本語では主語や目的語などの欠落が頻繁に起こる．これに合わせ，記事を抽出する際の制約を変更する．

次に作成した訓練データの有効性を確認するため文圧縮器の学習を行う．本稿では，Long short-term memory（LSTM）に基づく encoder-decoder モデルで各単語を “保持”するか “削除”するかのラベル列を推定する手法 [3] を用いる．要約文長制御が可能であれば，文長制約を満たしたより，情報量の多い圧縮文を作成できるが，Filippova らの手法 [3] では十分に文長制御が行えない．そこで，ヘッドライン生成において明示的な文長制御を可能としたモデル [4] によってラベル列の推定を行う．ただし，ヘッドライン生成では次に出力する単語を語彙中から推定しており，“保持”，“削除”の 2 ラベルを推定する本課題とは異なる．そのため，本課題においても明示的な文長制御が可能であるかを確認する．

2 日本語訓練データ自動構築

日本語の特徴に合わせて Filippova らの手法 [2] を改良する．Filippova らは H と S で共有されている内容語 (本稿では共有語と呼ぶ) を同定し，それらを全て含む S の部分依存木を圧縮文としている．この手法は，(1) 共有語の同定，(2) 依存木の改変，(3) 部分依存木の抽出，の 3 つの段階と，H が S の要約であると考えられる記事を抽出するための制約から成り立っている．

以上を踏まえ，日本語に適応するための変更を以下で述べる．

2.1 共有語の同定

英語に対する先行研究では，各単語を原形化し，代名詞は照応解析により先行詞に置き換え，内容語の同定を行っている．これらの同定では，日本語において多用される “省略”と “動詞の名詞化”を捉えられない．そこで，原形化による同定を行った後，以下の 2 つの方法による同定を行う．ただし，日本語は代名詞を省略することが多いため，代名詞の照応解析は行わない．

省略の同定　図 1 の「東工大」は「東京工業大学」の省略表現である．原形化ではこの組を同定できないが，この省略表現は元の表現の部分文字列である．ただし，省略表現と元の表現のどちらかまたは両方が複

[1] 名詞，動詞，副詞，形容詞

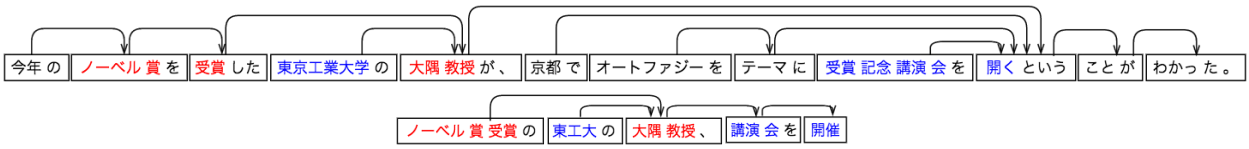

図 1: S（上）と H（下）の依存木

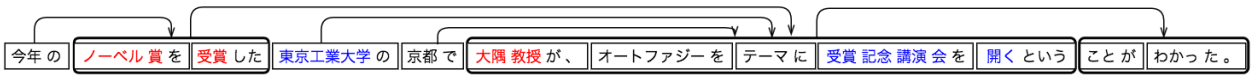

図 2: 改変された S の依存木：赤字は単語の原形を用いた同定，青字は追加した方法による同定

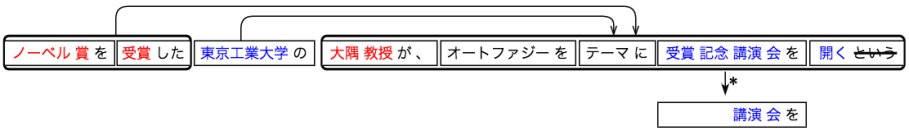

図 3: 圧縮文として抽出した部分依存木

数の単語から成ることもある．そこで，文節内の名詞の連続[2]を 1 ブロックとし，H 中のブロックが S 中のブロックの部分文字列であれば，そのブロックの組を同定したとみなす．図 1 では例えば「ノーベル」と「賞」の 2 単語で,「東京工業大学」は 1 単語でブロックとなる．結果として「東工大」と「東京工業大学」,「講演 会」と「受賞 記念 講演 会」の 2 組が同定される．この方法により前者の略語と元の表現の組，後者の形容詞的な役割をしている表現を削除する省略表現と元の表現の組を同定することができる．

動詞の名詞化の同定　図 1 にある「開催」と「開く」は品詞の異なる非常に意味の近い単語である．原形化ではこの組を同定できないが，この組には「開」という漢字が共通している．そこで表意文字である漢字を用い，H 中の名詞と S 中の動詞[3]が漢字 1 字を共有していればその組を同定する．

2.2　依存木の改変

英語に対する先行研究同様，部分依存木の文法性を高めるために S の依存木を改変する．ただし，同様の理由により文節依存木を用いる．日本語では機能語が「を，に，が，は」である文節は，係り先の文節の必須格となることが多い．そこで，機能語が「を，に，が，は」である文節は，係り先の文節と合わせて 1 つのノードとみなす．この変更により S の依存木は図 2 のように変わる．

2.3　根付き部分木の抽出

英語に対する先行研究同様，改変した依存木から共有語が全て含まれる最小根付き部分木を抽出し，元の順番に単語を並べ圧縮文を作成する．

この方法だけで日本語でも確かに圧縮文は作成できるが，1 節の (2)，(3) で述べたように，S の依存木の根の削除，依存木のノードである文節よりも細かい削除が含まれた圧縮文が作成できることが望ましい．そこで以下の 2 つの方法によって圧縮文を作成する．

S の依存木の根の削除　英語に対する先行研究では全動詞を部分木の根の候補とし，S の依存木の根が削除される圧縮文を作成している．ただし，日本語は主要部終端型であるため文末の文節が根となる．つまり，部分木の根の候補は文末としての文法上の制約を満たす必要がある．そこで，以下の方法で圧縮文を作成する．

1. 文末となりうる単語[4]が含まれるノードと S の依存木の根を部分木の根の候補とする．
2. 改変した依存木から全ての共有語が含まれる最小根付き部分木を抽出する．
3. S の依存木の根以外が根となった場合，文末となりうる単語より後の単語を削除する．

よって図 2 では文末となりうる「受賞」,「開く」を含むノードと S の依存木の根である「わかった」の 3 つのノードが部分木の根の候補である．最小根付き部分木を抽出すると「開く」を含むノードが部分木の根となるため，図 3 において「という」が削除される．

文節より細かい削除　文節より細かい削除が含まれる圧縮文を作成するため，2.1 節で同定した省略表現を元の表現の代わりに圧縮文に用いる．ただし，重要な単語である主辞が削除される省略表現，単語の削除では再現できない省略表現は用いない．よって圧縮文を作成する際，図 3 の*のように「受賞 記念 講演 会」を「講演 会」に置き換えるが，単語の削除では再現できない「東工大」は「東京工業大学」と置き換えない．

最終的に「ノーベル賞を受賞した東京工業大学の大隅教授が，オートファジーをテーマに講演会を開く」という圧縮文が作成される．

[2]先頭には接頭詞がついてもよい．
[3]漢字が 1 字で残りの文字が平仮名の動詞を対象とする．
[4]文末となりうる単語は，次の単語が助動詞，名詞，連体化の「の」ではない原形の動詞，助動詞，名詞としている．

表 1: 閾値ごとの訓練データ

閾値	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
文数 (万)	202	189	172	154	134	106	81	59	35	27
語彙 (千)	105	102	98	94	90	81	72	62	48	42

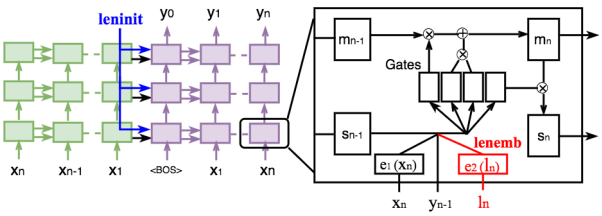

図 4: LSTM に基づく Encoder-Decoder モデル

2.4 使用する新聞記事の制約

制約の目的は H が S の要約であると考えられる記事を選ぶことである．英語に対する先行研究では 8 つの制約を設けている．まず，日本語は英語に比べて語順が自由であるため,「H の内容語が同じ順序で S に現れている」という制約を削除する．また，日本語では単語が省略されやすく，必ずしも H 中の内容語が全て同定できなくても H が S の要約となっていることは多い．そこで,「H 中の内容語が S にすべて含まれている」という制約を緩め,「H 中の内容語のうち S に含まれているものの割合 (内容語一致率) が閾値を超える」という制約に変更する．また残りの制約は，2 つを削除 [5]，4 つ [6] をそのまま残している．

3 LSTM を用いた削除型文圧縮

LSTM に基づく encoder-decoder モデルを用いて各単語を “保持”するか “削除”するかのラベル列を推定し文圧縮を行う [3]．入力列を $\mathbf{x} = (x_1, x_2, \ldots, x_n)$，出力ラベル列を $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ として全体の流れを式にすると

$$
\begin{aligned}
s_0 &= encoder(\mathbf{x}) \\
(m_t, s_t) &= decoder(s_{t-1}, m_{t-1}, e_1(x_t) \bigoplus e_l(y_{t-1})) \\
y_t &= softmax(W * s_t + b)
\end{aligned}
$$

である．この時，$\bigoplus$ はベクトルの連結，s_t，m_t は t 回目の出力時の隠れ層とメモリーセル，$e_1(word)$ は単語埋め込み，$e_l(label)$ は $label$ が “保持”なら $(1, 0)$，“削除”なら $(0, 1)$ である．m_0 には零ベクトルを用いる．

また，本稿ではヘッドライン生成タスクにおいて直接的な文長制御を可能としたモデル [4] でも学習を行う（図 4）．彼らは *leninit* と *lenemb* という 2 種類の方法を提案している．*leninit* はメモリーセルを文長の操作に用いる．そのため，$tarlen$ を目的文長とすると i 層目のメモリーセルの初期値は $m_{i,0} = tarlen * b_{i,len}$ となる．$b_{i,len}$ はモデルと同時に学習する．*lenemb* では，長さ埋め込み $e_2(length)$ を導入し，各ラベルを推定する際に入力として残り文長の長さ埋め込みも用いる．そのため，l_t [7] を t 回目の出力時の残り文長とすると，*decoder* への入力として $e_1(x_t) \bigoplus e_l(y_{t-1}) \bigoplus e_2(l_t)$ を用いる．

[5] H に動詞がある，H が動詞で始まっていない．

[6] H が質問ではない，H と S が 4 単語以上である，S が H の 1.5 倍より長い，S が圧縮文の 1.5 倍より長い．

4 実験

提案法により自動作成した訓練データの有効性と 2 種類の文長制御機構の有効性を確認する．

実験設定　表 1 のように 10 段階の内容語一致率に関する閾値で訓練データを作成し，それぞれで文圧縮器を学習する．そのうち，評価には開発データにおいて ROUGE-2 を最良とする文圧縮器 [8] を用いた．訓練データの作成には 3 社 [9] 合計 35 年分の新聞記事を用いた．また，各文に人手で作成した 5 個の圧縮文がある 1000 文 [6] のうち，100 文を開発データに，900 文を評価データに用いた．語彙（表 1）は訓練データ中で頻度 5 以上の単語で構成し，未知語は<unk>に置き換えた．モデルは LSTM を 3 層 [10] とし，学習には adam[11]，評価時には beam-search[12] を用いて出力した．ただし，不当に長い圧縮文が生成されることを防ぐため文長制約を超える文を beam-search の候補から取り除いた．文長制約には 5 文ある参照文から無作為に選んだ 1 文の文長を用いた．

学習を行わない文圧縮器　大規模な訓練データを用いた日本語文圧縮器はこれまでにないため，学習を行わずに依存木を刈り込む手法 [7] と比較する（*tree-base*）．この手法は，次の目的関数を最大化する根付き部分木を ILP を用いて抽出する：$f(X) = \sum_x x^l_{h,w} \cdot P(l|h) \cdot I(w)$．$x$ はあるノード w を部分木に含むなら 1，含まないなら 0 となり，h は w の係り先の主辞，l は w から h が含まれるノードへの係り受けラベルを表す．また，$P(l|h)$ は言語コーパスにおける条件付き確率，$I(w)$ は w に含まれる内容語の生起確率と依存木での位置の情報を用いて算出されるノードの重要度スコアである．ただし，本稿で用いた構文解析器 [5] は係り受けラベルを付与しないため l として w の機能語を用いた．言語コーパスには閾値が 0.1 の場合の文圧縮訓練データの原文を用いた．

4.1 自動評価

表 2 に自動評価結果を示す．ROUGE-1，2，L（R-1, 2，L）で評価を行う．圧縮率は，圧縮文の原文に対す

[7] 目的文長を越えた場合は残り長さ 0 文字として扱う

[8] 最良の閾値は表 2 の上から 0.4，0.5，0.5，0.6，0.4，0.2 である．

[9] 読売新聞，産経新聞，毎日新聞

[10] 単語埋め込み:256 次元，隠れ層:256 次元，dropout:20%

[11] alpha:0.001，beta1:0.9，beta2:0.999，eps:1e-8，batchsize:180

[12] beam 幅:20

表 2: 自動評価結果

訓練データ	モデル	R-1	R-2	R-L	圧縮率
	tree-base	68.9	52.4	34.4	59.1
rooted	*lstm*	65.1	49.6	35.7	50.9
rooted	*lstm+leninit*	70.4	54.2	36.0	56.8
rooted	*lstm+lenemb*	70.6	54.4	36.0	57.6
multi-root+	*lstm*	60.2	49.5	33.7	47.9
multi-root+	*lstm+leninit*	70.9	54.5	36.3	57.8
multi-root+	*lstm+lenemb*	72.2	55.2	35.8	59.4

る文長の割合であり，文長制限を圧縮率で表すと平均61.0%となる．*rooted* は 2.3 節で根付き部分木を抽出し作成した訓練データ，*multi-root+*は 2.3 節で S の依存木の根の削除，文節よりも細かい削除のための改良も含めて作成した訓練データである．*lstm* は LSTM に基づく encoder-decoder モデル [3] を示す．

tree-base と訓練データを用いたモデルを比べると，基本的に訓練データを用いた場合のほうがスコアが高い．これは提案法で自動作成した訓練データが有効であることを示す．例外として *lstm* では R-1，2 のスコアが低下しているが，これは文長が短いことが原因であると思われる．圧縮率を見ると文長制御機構を取り入れた *lstm+leninit* と *lstm+lenemb* で目的文長に近い圧縮文が作成できていることがわかる．それに伴い，R-1，2 も向上している．よって，これらの文長制御機構が本課題でも有効 [13] であることがわかる．次に，訓練データに *multi-root+*を用いた効果を見ると，*lstm+leninit* と *lstm+lenemb* はより柔軟な文長制御とそれに伴う R-1，2 の向上が確認できる．

表 2 の結果では *leninit* と *lenemb* に大きな違いは見られない．しかし，*multi-root+*で学習した *lstm+leninit* と *lstm+lenemb* の圧縮文で文末に文法上の明らかな誤りがある [14] 文数を数えると，100 文中 *lstm+leninit* では 2 文であったのに対し，*lstm+lenemb* では 24 文であった．これは，*lstm+lenemb* は各ラベルを出力する際に入力として残り文長の情報を受け取るため，出力が残り文長に強く影響を受けたと考えられる．

4.2 人手評価

*multi-root+*を用いて学習した *lstm+leninit* によって生成した圧縮文の人手評価を行う．*lstm+lenemb* は文法上の誤りが多くみられたため用いない．比較のため，参照文 [15]，tree-base も評価する．人手評価にはクラウドソーシングを用いた．評価者は原文と順序を無作為に変えた 3 つの圧縮文を提示され，各圧縮文に対し可読性と情報量の観点で 5 段階評価を行った．1 点が最低点，5 点が最高点となる．結果を表 3 に示す．

表 3: 人手評価結果: スコアの平均

モデル	可読性	情報量
参照文	4.20	3.64
lstm+leninit	3.90	3.15
tree-base	3.18	2.54

可読性，情報量の両方の観点において，訓練データを用いた文圧縮器が tree-base より高いスコアである．また，情報量では依然として参照文には及ばないものの，可読性は訓練データを用いない場合に比べてかなり参照文に近い評価値を得ることができている．よって提案手法により自動作成された訓練データの有効性が確認できる．

5 まとめ

本稿では，英語のための訓練データ自動構築手法を日本語の特徴に合わせて改良し，日本語の文圧縮のための訓練データ自動構築手法を示した．次に，提案手法によって自動作成した訓練データを用いて LSTM を用いた文圧縮器の学習を行い自動作成された訓練データの有効性を示した．また，Kikuchi ら [4] が提案した 2 つの文長制御機構のうち *leninit* が本課題でも有効であることがわかった．一方，*lenemb* は残り文長の強い影響を受け，文末において文法上の誤りが起こりやすいことが確認された．

謝辞 本研究は JSPS 科研費 26280080 の助成を受けたものである．

参考文献

[1] T. Berg-Kirkpatrick et al., Jointly Learning to Extract and Compress. *ACL*, pp 481-490, 2011

[2] K. Filippova and Y. Altun, Overcoming the Lack of Parallel Data in Sentence Compression. *EMNLP*, pp 1481-1491, 2013.

[3] K. Filippova et al., Sentence Compression by Deletion with LSTMs. *EMNLP*, pp 360-368, 2015.

[4] Y. Kikuchi et al., Controlling Output Length in Neural Encoder-Decoders. *EMNLP*, pp 1328-1338, 2016.

[5] T. Kudo and Y. Matsumoto, Japanese Dependency Analysis using Cascaded Chunking. *CoNLL*, pp 63-69, 2002

[6] 平尾 努ら, 識別学習による組合せ最適化問題としての文短縮手法. 人工知能学会論文誌, pp. 574-584, 2007.

[7] K. Filippova and M. Strube, Dependency Tree Based Sentence Compression. *INLG*, pp 25-32, 2008.

[8] K. Thadani and K. McKeown, Sentence Compression with Joint Structural Inference. *CoNLL*, pp 65-74, 2013.

[13] 本稿には載せていないが，目的文長を変えるとそれに合わせ出力文長が変わることも確認した．

[14] 例としては,「わかった.」という文が「わかっ」で終わっていれば明らかに文法上の誤りがある．

[15] 5 つの参照文のうち 1 つを無作為に選んでいる