構文木の再帰構造除去に基づく文圧縮手法と複数正解に基づく評価

江川　誠二†　　加藤　芳秀‡　　松原　茂樹§

†名古屋大学大学院情報科学研究科　‡名古屋大学大学院国際開発研究科

§名古屋大学情報連携基盤センター

egawa@el.itc.nagoya-u.ac.jp

1　はじめに

文圧縮とは，文の主要な情報を保持したまま，より短い文を生成するタスクであり，テキスト自動要約の実現に必要不可欠な技術である．また，テレビの字幕やニュースのヘッドラインの自動生成など文の長さに制限がある場面で有用である．

文圧縮によって生成される文を**圧縮文**と呼ぶ．圧縮文は次の条件を満たす必要がある．

- 文法的である
- 原文の主要な情報を保持している

これまでに様々な文圧縮手法が提案されている．それらの多くは，構文木に基づき句や節などを認識し，原文から単語，句，あるいは節などを取り除くことにより圧縮文を生成する [6, 7, 9, 10, 11]．しかし，実際には，単純な構成素の削除では対処することが難しい，構文木の構造の変更が必要な場合が存在する．

そこで本稿では，構文木の構造的な変更による文圧縮手法を提案する．本手法では，構文木の再帰構造に注目する．再帰構造は，付加詞や等位構造，埋め込み文など，構文木中に頻出するが，構成素の削除による文圧縮手法では，これを文法的に自然な形で処理することができない．本手法では，構文木に出現する再帰的な構造を検出し，これを除去する操作を導入する．操作の適用には曖昧性があるため，操作が適用される確率を圧縮文コーパスから学習する．

圧縮文コーパスを用いて評価実験を実施した．被験者実験により，圧縮文の文法性において，提案手法が既存の手法に比べて優れていることを確認した．さらに，文圧縮の多様性という観点から，複数の正解を用意し，これを用いて評価実験を行った．その結果，提案手法が既存の手法と比べ，人手の圧縮に近い圧縮文を生成することを確認した．

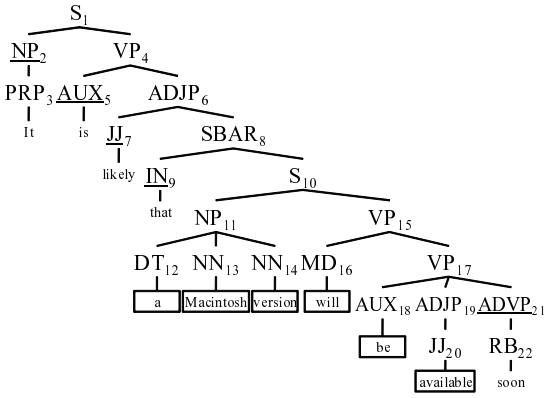

図 1: 構成素の削除のみでは圧縮が難しい例

2　構成素の除去に基づく文圧縮

従来の文圧縮手法の多くは，入力として文 l を受け取り，l からいくつかの単語を取り除いた文を返す [6, 7, 9, 10, 11]．

このうち，Knight ら [7] や Unno ら [11] は，構文木に基づく文圧縮手法を提案している．入力文 l を構文解析し，得られた構文木中の生成規則を縮約することにより，構成素を削除し，圧縮文を生成する．

しかし，生成規則の縮約では実現が難しい文圧縮が存在する．次のような場合である．

(1)　**It is likely that** a Macintosh version will be available **soon**.
(2)　A Macintosh version will be available.

(1) は原文で，(2) はその圧縮文である．この圧縮において，“It is likely that” は 1 つのまとまりとして削除されたと考えられる．“It is likely that” の一部を残すような削除のいずれも，圧縮文を不適格文にしてしまうからである．

ところが，従来の文圧縮手法によりこのような圧縮を行う場合，これをまとまりとして削除することはできな

い．図 1 は原文 (1) の構文木である．従来の文圧縮手法では，構文木中の下線を引いたノードをそれぞれ単独で削除することになる．すなわち，原文・圧縮文の対からなる圧縮文コーパスから圧縮のプロセスを学習するとき，それぞれの構成素を独立に削除するプロセスを学習することになり，その結果，不適格な圧縮文を生成してしまう危険性がある．

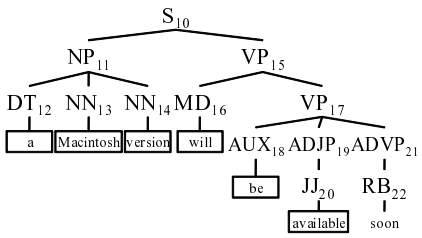

図 2: 圧縮操作の例

圧縮操作の種類，
注目しているノードのラベル，親ノードのラベル，子ノードのラベル，
左側の兄弟ノードのラベルとそれが削除されたかどうか(削除のみ)，
再帰ノードから足ノードまでのパス上のノードのラベル(置換のみ)，
再帰ノードから足ノードまでのパス上のノードの子ノードのラベル(置換のみ)，
足ノードのラベル(置換のみ)

図 3: 学習に使用した素性

3　再帰構造の除去に基づく文圧縮

前節の例において，原文の構文木から何らかのプロセスを経て，正しい圧縮文の構文木が得られるとすれば，それは自然な文圧縮プロセスであると考えられる．このような構文木を得るためには，図 1 の構文木において，$\mathbf{S_1}$ に埋め込まれた $\mathbf{S_{10}}$ を取り出し，$\mathbf{S_1}$ を $\mathbf{S_{10}}$ で置き換えればよい．このような処理を行う操作を導入できれば，“It is likely that” をまとめて削除することが可能となる．ここで注目すべきポイントは，$\mathbf{S_1}$ と $\mathbf{S_{10}}$ が同一の統語範疇であるということである．すなわち，あるノードを別の構文木で置き換えるとき，その範疇が同一であれば文法性が保たれる．これが本手法において再帰構造に注目する理由である．

3.1　再帰構造の除去

本節では，再帰構造の除去による文圧縮操作を導入する．初めに，再帰ノードを定義する．

定義 1 (再帰ノード) T を構文木，η を T 中のノード，X を η のラベルとする．以下の条件を満たすような η' が存在するとき，η を**再帰ノード**と呼ぶ．

- η' は η の子孫ノードである
- η' のラベルが X である

図 1 において，再帰ノードは，$\mathbf{S_1}$，$\mathbf{VP_4}$，$\mathbf{VP_{15}}$ の 3 つである．

再帰ノード η 以下を幅優先で探索し，最初に見つかる η' を足ノードと呼ぶ．図 1 において，$\mathbf{S_1}$ の足ノードは $\mathbf{S_{10}}$，$\mathbf{VP_4}$ の足ノードは $\mathbf{VP_{15}}$ である．

再帰ノードに対する圧縮操作を以下のように定義する．

置換 再帰ノード η 以下の部分木を，η に対する足ノード η' 以下の部分木で置き換える

この操作により，再帰構造を取り除いて文を圧縮できる．図 1 中の $\mathbf{S_1}$ に置換操作を適用して獲得される構文木を図 2 に示す．

一方，非再帰ノードに対しては，従来の手法と同様に，ノードを削除する操作を適用する．

3.2　確率的文圧縮モデル

本手法では，原文がある圧縮文に圧縮される確率を，原文の構文木に圧縮操作を適用して圧縮文の構文木が得られる確率と定義する．各圧縮操作が適用されるか否かは，置換操作に関しては他の操作の適用・非適用とは完全に独立であるとし，また，削除操作に関しては現在注目しているノードの兄弟ノードに対する操作の適用・非適用にのみ従属であると仮定する．これにより，圧縮文の構文木が得られる確率を，各圧縮操作の適用（非適用）確率の積として求めることができる．

本手法では，どのような場合に圧縮操作を適用すればよいかを，原文とその圧縮文の対からなる圧縮文コーパスから学習し，圧縮操作の適用確率を，最大エントロピー法 [1] によって推定する．使用した素性を図 3 に示す．

3.3　スコアの計算

前節のモデルをもとに，原文 l に対する各圧縮文候補 s のスコアを以下のように計算する．

$$Score(s) = length(s)^{\alpha} \cdot \log P(s|l)$$

このスコアは，確率モデルを我々のモデルに置き換えた点を除いて，Unno らの手法 [11] と同一である．圧縮文候補の生成確率に加えて，圧縮文の長さを考慮することによって，出力される圧縮文の平均文長を調節することができる．α はそのためのパラメータである．本手法は，このスコアを最大にするような s を圧縮文として出力する．

表 1: 被験者による評価の結果

	圧縮率	文法性	意味
Knight	70.4%	4.29	4.09
提案手法	68.1%	4.45	4.04

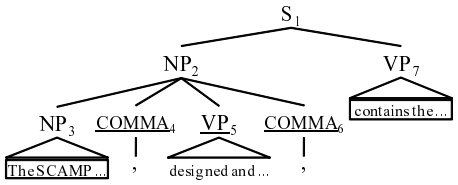

図 4: 原文 (4) の構文木

4 評価実験

4.1 実験の設定

提案手法の有効性を確認するために評価実験を行った．Knight らの手法 [7] との比較のため，実験データとして，Knight らが Ziff-Davis コーパスから作成した，原文と圧縮文の対からなるコーパスを使用した．943 対を学習データとして使用し，32 対をテストデータとした．テストデータは，Knight ら [7] の実験と同一であり，学習データとは異なる対から構成される．学習・文圧縮の処理において構文木が必要であるため，Charniak Parser[3] を使用し構文木を付与した．文長を制御するパラメータ α は，Knight らの手法の圧縮率と等しくなるように定めた．その値は $\alpha = -0.50$ である．

4.2 被験者実験による評価

被験者実験により，提案手法と Knight らの手法を文法性と意味の保持の観点から評価した．英語母語話者である被験者 4 名に対し，原文と，それに対する Knight らの手法の圧縮文，及び提案手法の圧縮文を併せて提示した．被験者は，圧縮文が文法的にどれだけに正しいか（**文法性**）と，原文の主要な意味をどれだけ保持しているか（**意味保持度**）の 2 つの観点で，1 から 5 の 5 段階で評価した．5 が最も良い評価である．圧縮文の順序は，テストセットの文ごとにランダムに並び替えられている．

表 1 に，2 種類の文圧縮手法それぞれの圧縮率，及び被験者が評価した文法性と意味保持度を示す．提案手法は Knight らの手法と圧縮率，意味の保持に関してはほぼ同等でありながら，文法性に関して高い評価を受けていることから，提案手法がより高い文圧縮性能を備えていることがわかった．

4.3 再帰構造の除去の効果

表 2 に提案手法，Knight らの手法，及びコーパス中の圧縮文の例を示す．2 つの原文 (3), (4) は，いずれもテストセットに含まれる文である．

原文 (3) に対して，コーパスの圧縮文では “Another slight downside is that” が削除されている．提案手法では，置換操作によって “Another slight downside is that” を 1 つのまとまりとして削除し，コーパスの圧縮文と同様の圧縮文を生成している．それに対して，Knight らの手法では原文をそのまま圧縮文としている．Knight らの手法でコーパスの圧縮文と同じ圧縮文を生成するには，原文 (1) と圧縮文 (2) の例と同様，“Another slight downside”，“is”，及び “that” をそれぞれ単独で削除する必要があるが，このような削除が行われる確率は極めて低いと考えられる．

原文 (4) の構文木を図 4 に示す．この原文に対して，コーパスの圧縮文では “, designed and ... Intel process,” が削除されている．提案手法では，$\mathbf{NP_2}$ に置換操作を適用することによって同様の箇所を 1 つのまとまりとして削除している．それに対して，Knight らの手法では，一方の “,” と “designed and ... Intel process” を削除し，もう一方の “,” を残しており，その結果，非文法的な圧縮文が生成されている．

4.4 複数の正解に基づく評価

文圧縮手法の評価は被験者実験によることが望ましいが，その都度，被験者実験を行うのは大きなコストを要するため，何らかの方法で自動評価できることが望ましい．本節では，文圧縮手法の自動評価法について議論し，提案手法の自動評価について述べる．

これまでの文圧縮手法の自動評価では，正解圧縮文をただ 1 つだけ用意し，手法が生成する圧縮文との一致の度合により評価している [6, 9, 11]．しかし，一般には，原文に対して正しい圧縮文は必ずしも 1 つとは限らない[1]．そこで我々は，原文に対して複数の圧縮文を用意し，文圧縮手法を評価した．

正解圧縮文は，3 人の作業者（英語母語話者）が作成した．テストデータの各文に対する圧縮文の上限を 10 文とし，作成した圧縮文には，圧縮文の文法性，意味保持度，圧縮率などの観点から，作業者が総合的に判断した順位付けがなされている．

正解圧縮文と完全一致した文の数を表 3 に示す．n-best の列は，上位 n 位以内の正解圧縮文のいずれかに圧縮された文の数を示す．コーパスの圧縮文でも，1 位

[1] テキスト要約に関して同様の事柄が，石川らにより指摘されている [5].

表 2: 圧縮文の例

原文 (3)	Another slight downside is that envelopes must be fed manually.
Knight	Another slight downside is that envelopes must be fed manually.
提案手法	Envelopes must be fed manually.
コーパス	Envelopes must be fed manually.
原文 (4)	The SCAMP module, designed and built by Unisys and based on an Intel process, contains the entire 48-bit A-Series processor.
Knight	The SCAMP module, contains the entire 48-bit A-Series processor.
提案手法	The SCAMP module contains the entire 48-bit A-Series processor.
コーパス	The SCAMP module contains the entire 48-bit A-Series processor.

表 3: 複数の正解に基づく評価の結果

	圧縮率	1-best	3-best	10-best
Knight	70.4%	3	3	6
提案手法	68.1%	6	7	8
コーパス	53.3%	11	14	22

のみを正解とした場合，合致するものは 11 文に過ぎず，全体の 34%でしかない．一方，10 位までのすべての正解圧縮文を用いた場合では，22 文と 69%を占める．このことは，正解の圧縮文が複数存在していることを意味しており，複数の正解を用いての評価は有用であると考えられる．

提案手法と Knight らの手法について見てみると，提案手法のほうが完全一致する文の割合が多い．このことから，提案手法がより正解圧縮文に近い圧縮を行っていると考えられる．

5 おわりに

本稿では，構文木の再帰構造を除去することによって文圧縮する手法を提案した．本手法では，構成素の削除のみを行う既存の手法ではうまく処理できなかった付加詞，等位構造，埋め込み文などを，再帰構造の除去によって，文法的に自然な形で削除できる．

被験者による評価実験から，提案手法が，既存の手法と比べて，文法的に正しく圧縮できることを確認した．また，複数の正解に基づく評価実験により，提案手法が既存の手法と比べ，人手の圧縮に近い圧縮文を生成できることを示した．

今後は，再帰構造除去以外の構文木の構造的変更についても検討し，より柔軟な文圧縮手法を提案したい．また，複数の正解に基づく評価の妥当性を，より詳細に検証する予定である．

参考文献

[1] A.L. Berger, V.J. Della Pietra, and S.A. Della Pietra, “A Maximum Entropy Approach to Natural Language Processing, ” Computational Linguistics, vol. 22, num. 1, pp. 39–71, 1996.

[2] T. Briscoe and J. Carroll, “Robust Accurate Statistical Annotation of General Text,” Proc. 3rd LREC, pp. 1499–1504 2002.

[3] E. Charniak, “A Maximum-Entropy-Inspired Parser, ” Proc. 6th ANLP, pp. 132–139, 2000.

[4] J. Clarke, and M. Lapata, “Models for Sentence Compression: A Comparison across Domains, Training Requirements and Evaluation Measures, ” Proc. 44th ACL, pp. 377–384, 2006.

[5] 石川開, 安藤真一, 奥村明俊, “テキスト要約の複数の正解に基づいた評価, ” 自然言語処理，vol. 9，no. 4, pp. 33–53, 2002.

[6] H. Jing, “Sentence Reduction for Automatic Text Summarization, ” Proc. 6th ANLP, pp. 310–315, 2000.

[7] K. Knight, and D. Marcu, “Statistics-Based Summarization – Step One: Sentence Compression, ” Proc. 17th AAAI, pp. 703–710, 2000.

[8] I. Mani, Automatic Summarization, John Benjamins, 2001.

[9] S. Riezler, T. H. King, R. Crouch, and A. Zaenen, “Statistical Sentence Condensation using Ambiguity Packing and Stochastic Disambiguation Methods for Lexical-Functional Grammar, ” Proc. HLT-NAACL-2003, pp. 118–125, 2003.

[10] J. Turner, and E. Charniak, “Supervised and Unsupervised Learning for Sentence Compression, ” Proc. 43rd ACL, pp. 290–297, 2005.

[11] Y. Unno, T. Ninomiya, Y. Miyao, and J. Tsujii, “Trimming CFG Parse Trees for Sentence Compression Using Machine Learning Approaches, ” Proc. 44th ACL, pp. 850–857, 2006.