社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－NL－186（11）
2008 / 7 / 18

構文木からの再帰構造の除去による文圧縮

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
〒 464-8601 名古屋市千種区不老町
E-mail: †{egawa,yosihide,matubara}@el.itc.nagoya-u.ac.jp

あらまし 文圧縮は，文のもつ情報をできるかぎり保持したまま，より短い文を生成するタスクである．既存の文圧縮手法の多くは，原文の構文木の構成素を取り除く単純な操作のみに基づいており，構文木の構造的変更に対応することはできない．本稿では，変更操作の対象として，構文木中の再帰構造に注目し，それを取り除く文圧縮手法を提案する．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ことで，文法性を保ったまま再帰構造を含む文を圧縮できる．評価実験の結果，圧縮された文の文法性について，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．

キーワード テキスト要約，句構造，最大エントロピー法，テキストコーパス

Sentence Compression by Removing Recursive Structure from Parse Tree

Seiji EGAWA†, Yoshihide KATO††, and Shigeki MATSUBARA†††

† Graduate School of Information Science, Nagoya University
†† Graduate School of International Development, Nagoya University
††† Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan
E-mail: †{egawa,yosihide,matubara}@el.itc.nagoya-u.ac.jp

Abstract Sentence compression is a task of generating a grammatical short sentence from an original sentence, retaining the most important information. The existing methods of only removing the constituents in the parse tree of an original sentence cannot emulate human compression which changes structures of the parse tree. This paper proposes a method to remove recursive structures, one of such structural conversions, and generate a grammatical short sentence. In order to remove a recursive structure, our method detects the constituents forming the structure and remove them as a unit. Compression experiments have shown that our method generates more grammatical compressed sentences than the previous method.

Key words text summarization, phrase structure, maximum entropy method, text corpus

1. はじめに

文圧縮は，文のもつ情報をできるかぎり保持したまま，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，表示できる文の長さに制限がある場面，例えば，ニュースのヘッドラインや字幕の自動生成などでも利用可能である．

文圧縮によって生成される文は**圧縮文**と呼ばれる．圧縮文は次の条件を満たすものでなければならない．

- 文法的である
- 原文の主要な意味を保持している

これまでに様々な文圧縮手法が提案されている [3]〜[6], [8], [9]. これらは，原文から単語や句，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ものであり，その多くは，構文木に基づき句や節などを認識し，これを削除する．

Knight らは，原文の構文木から付加的な構成素を取り除くことにより，文を圧縮する手法を提案している [5]. 構成素を取り除く確率を，原文とその圧縮文の対からなるコーパスから学

習し，それをもとに尤もらしい圧縮文を求める．

Knight らの手法は，単純な確率文脈自由文法に基づき圧縮過程をモデル化したが，これに対し Unno らは，最大エントロピー法により，文脈自由文法に含まれるノードだけでなく，その親や兄弟に位置するノードなどの複雑な素性を扱うことのできる文圧縮モデルを提案した [9]．

これらの手法に共通するのは，圧縮文を生成する方法として，句や節などの構成素を削除する単純な操作しか許していない点である．しかし，実際には，単純な構成素の削除では処理できない，構文木の構造の変更が必要な場合が存在する．

そこで本論文では，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，構文木に頻出する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，構成素の除去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．本稿で提案する手法では，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．この操作は，構文木の文法性を保ったまま再帰構造を除去できるため，再帰構造を含む文を自然な形で圧縮することができる．操作の適用には曖昧性があるため，本手法では，原文と圧縮文の対からなる圧縮文コーパスから削除操作が適用される確率を学習する．これにより，尤もらしい圧縮文を求めることができる．

評価実験の結果，圧縮文の文法性について，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．

本稿の構成は以下の通りである．次の 2. では，既存の手法とその問題点を述べる．3. では，構文木の再帰構造の除去による文圧縮手法について述べる．4. で実験について報告し，5. で本研究をまとめる．

2. 構成素の除去による文圧縮

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 s を返す．文中の語順は保ったままで，単語を別の単語に置き換えたり，新しく単語を加えることはしない．$2^{|l|}$ 個の圧縮文候補が存在するが，その多くは文法的でなかったり，重要な情報が欠けていたりするため，圧縮文として適さない．これらの候補のうち，圧縮文として適した文を見つけることがポイントとなる．

Knight ら [5] の手法では，構文木をベースに文を圧縮する．まず与えられた文を構文解析し，得られた構文木から構成素を削除することにより圧縮文の構文木を生成する．その葉ノードの単語を並べたものが圧縮文となる．構成素の削除の方法はいく通りも存在するが，圧縮文に最も適しているものを noisy-channel モデルに基づき選択する．すなわち，文 l の圧縮文 s' を以下のように定義する．

$$s' = \underset{s}{\operatorname{argmax}}\, P(s|l) = \underset{s}{\operatorname{argmax}}\, P(s)P(l|s)$$

$P(s)$ は，文脈自由文法に基づいて計算され，s の文法性を評価する．$P(l|s)$ は，l を s に圧縮するときに削除される部分の冗長性を評価する．これは，原文と圧縮文の対からなる圧縮文コーパスから学習する．$P(s|l)$ を $P(s)$ と $P(l|s)$ に分解することで，文法性と意味の保持という二つの問題を独立に扱うことができる．以下では，Knight らの手法の問題点を議論するために，圧縮文コーパスからの学習に注目する．Knight らの手法では，圧縮文コーパスから $P(l|s)$ を推定する．コーパス中の文とその圧縮文を構文解析し，それぞれの構文木のルートから順にノード間の対応をつける．対応するノードをもたない原文の構文木中のノードを冗長なノードとみなし，その頻度をもとに $P(s|l)$ を推定する．

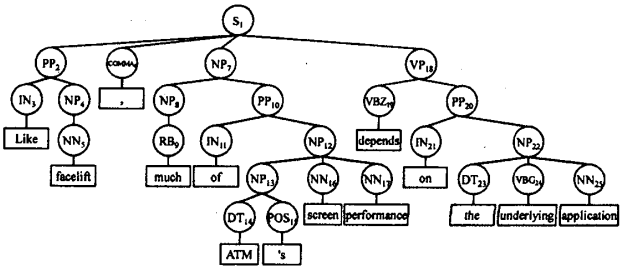

図 1　文 (1) の構文木

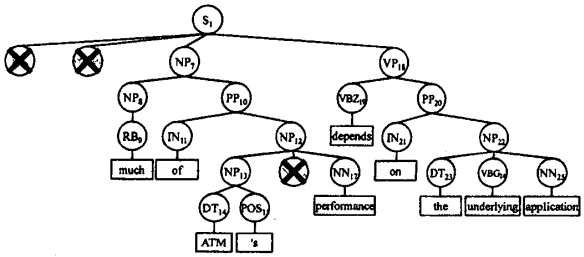

図 2　文 (2) の構文木

例として，原文 (1) と圧縮文 (2) の対について考える．なお，原文から削除される部分を太字で示している．

(1) **Like facelift,** much of ATM's **screen** performance depends on the underliying application.
(2) Much of ATM's performance depends on the underlying application.

これらを構文解析すると，図 1，及び図 2 のような構文木が得られる．これらをルートから順に対応付けていくと，結果としてノード "$\mathbf{PP_2}$", "$\mathbf{COMMA_6}$", "$\mathbf{NN_{16}}$" には対応するノードがなく，これらが冗長とみなされる．これを元にして，$P(s|l)$ が推定される．

このように，圧縮文コーパスから原文における冗長な構成素を求めることができる．しかし，文圧縮のプロセスが，単純な構成素の削除では捉えられない場合，そのプロセスを学習することはできない．例として，原文 (3) 及びその圧縮文 (4) について考える．

(3) The user can then abort the transmission, **he said.**
(4) The user can then abort the transmission.

図 3 に，これらの文の構文木を示す．構文木のルートから順に対応をとると，原文の "S → **S COMMA** NP VP" と圧縮文

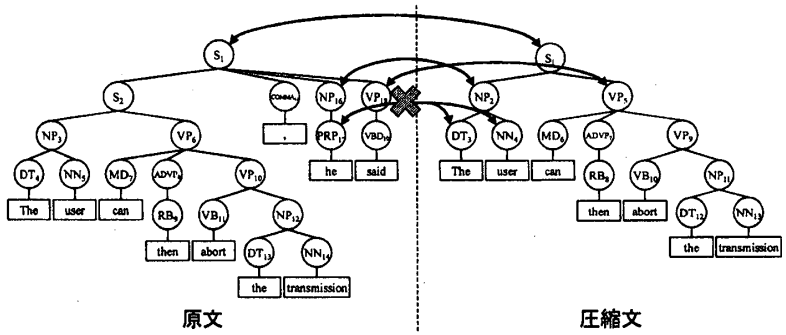

図 3　原文と圧縮文の構文木の対応が取れない例

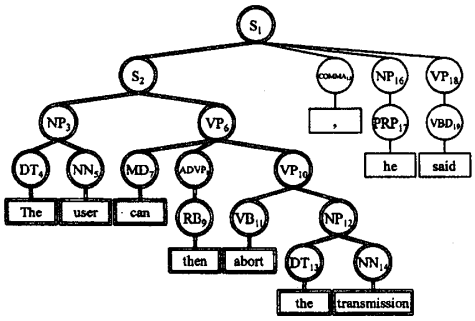

図 4　原文の構文木から圧縮文の構文木を抽出

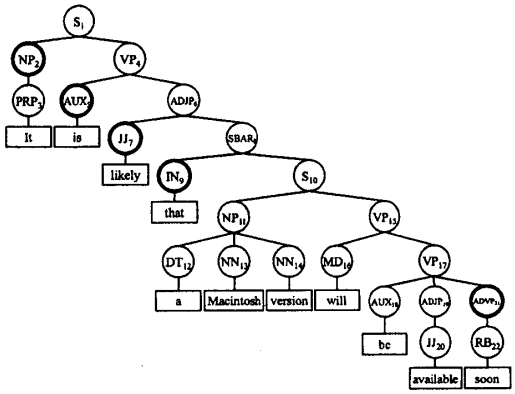

図 5　構成素の削除のみでは圧縮が難しい例

の “S → NP VP” が対応付けられる．続いて，原文の “NP → *PRP*” と圧縮文の “NP → *DT NN*” の対応を試みるが，マッチしないため対応付けに失敗する．このように，構成素の削除によって捉えられない文圧縮のプロセスは，対応付けに失敗するために学習することができない．

この問題に対して，Unno ら [9] は，原文のみを構文解析し，原文の構文木と圧縮文との対応をボトムアップに取る方法を提案した．原文の構文木上の単語のうち，圧縮文中に出現する単語をマークし，ボトムアップに構文木をたどることにより，圧縮文に対応する構文木を抽出する．

例として，文 (3) と文 (4) の対について考える．図 4 は文 (3) に対する構文木であり，二重線の四角で囲まれた単語が，圧縮文中に出現する単語である．これをボトムアップにたどっていくことにより，二重線の丸で囲まれたノードからなる構文木が得られる．これが，圧縮文に対する構文木となる．

この処理は，必ず成功する．すなわち，圧縮文に対応する構文木を必ず得ることができる．しかし，構成素の削除では捉えられない文圧縮の場合，得られる構文木は非文法的となる．その典型として，下記の例について考える．

(5) **It is likely that** a Macintosh version will be available **soon**.

(6) A Macintosh version will be available.

対応付けの結果，図 5 において太線で囲まれたノードが冗長なノードとして扱われる．結果として得られる圧縮文に対する構文木は非文法的である．もちろん，原文と圧縮文の構文木間の対応を付けることができるため，そこから文圧縮のプロセスを学習できるものの，それは，構成素 $\mathbf{NP_2}$, $\mathbf{AUX_5}$, $\mathbf{JJ_7}$, $\mathbf{IN_9}$ が各々独立に削除されるという不自然なプロセスであり，圧縮文の生成において，例えば $\mathbf{JJ_7}$ のみを削除して非文を生成してしまうという危険性を有している．

3.　再帰構造の除去による文圧縮

前節では，構文木の構成素の削除による文圧縮モデルではうまく捉えることのできない文圧縮を，具体例を用いて説明した．この問題を解決するために，本節では，構成素を単純に削除するだけでなく，構文木の構造を変更する操作を導入し，文圧縮をモデル化する．本手法では，構文木の再帰構造を除去する操作を新たに導入する．本手法について説明する前に，その基本アイデアについて述べる．

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，文法的な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 6 に示すように，$\mathbf{S_1}$ に埋め込まれた $\mathbf{S_{10}}$ を取り出し，それ以外の部分を削除すればよい．このような処理を行う操作を導入できれば，自然な形で文圧縮の過程をモデル化できると考えられる．“It is likely that” をまとめて削除することが可能となり，$\mathbf{JJ_7}$ を単独で削除するような誤ったプロセスの学習も回避できる．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

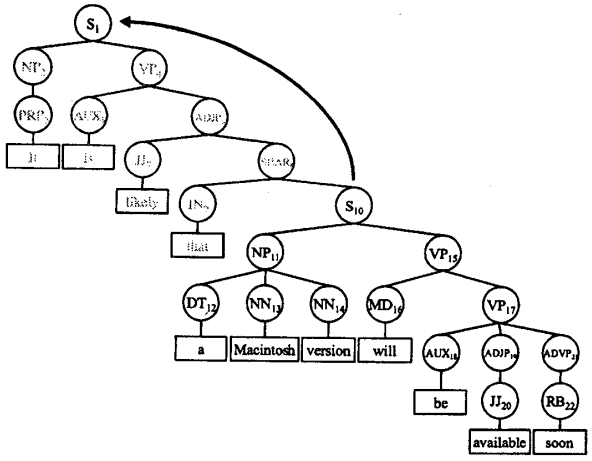

図 6　構文木の再帰構造の削除による文圧縮

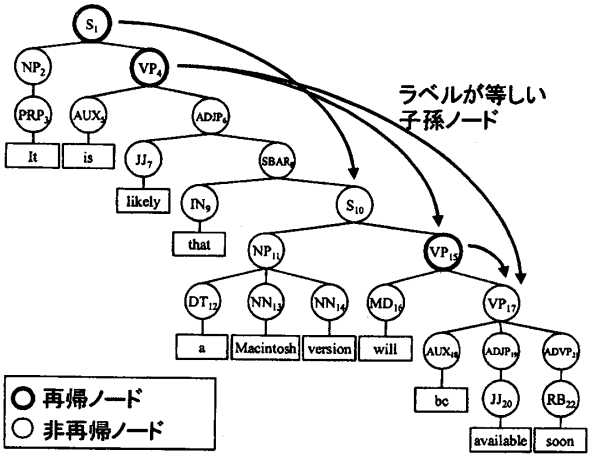

図 7　再帰ノードと非再帰ノード

である．

3.1　文圧縮のための基本単位

本節では，再帰構造を除去するための基本単位を新たに定義する．本手法では，基本単位ごとに，それを削除するか否かを選択する．基本単位を定義するための準備として，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．

1. η' は η の子孫ノードである
2. η' のラベルが X である

η が再帰ノードでないとき，η を**非再帰ノード**と呼ぶ．

例えば，図 7 中の再帰ノードは $\mathbf{S_1}$，$\mathbf{VP_4}$，$\mathbf{VP_{15}}$ である．

再帰ノード η 以下を幅優先で探索したときに最初に見つけられる η' を**足**と呼ぶ．η から足までのパスを **minimal recursive path (MRP)** と呼ぶ．η を MRP のルートと呼ぶ．ただし，足自身は MRP に含まれない．図 8 に，図 5 の $\mathbf{S_1}$ をルートとする MRP を示す．

MRP を削除することは，すなわち，再帰構造を除去することを意味する．本手法では，構成素，及び MRP を，削除の基本単位とする．

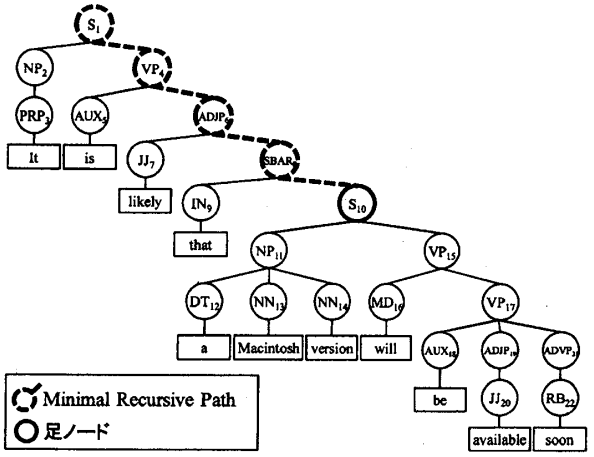

図 8　Minimal Recursive Path

3.2　構文木からの基本単位の削除

提案手法では，原文の構文木から，削除の基本単位である構成素と MRP を削除することにより圧縮文の構文木を生成する．以下に，2 種類の基本単位に対応する削除操作をそれぞれ定義する．

構成素の削除　非再帰ノード η および η のすべての子孫ノードを削除する

MRP の削除　再帰ノード η の位置を，対応する足 η' で置き換える

原文の構文木に対して，非再帰ノードであれば構成素の削除を，再帰ノードであればそのノードをルートとする MRP の削除を適用することにより，圧縮文の構文木を得ることができる．しかし，操作を適用するノードを適切に選択しなければ，得られる圧縮文が文法的でない，あるいは，重要な意味が失われるなどの問題が生じる．そこで本手法では，どのような場合に削除操作を適用すればよいかを，圧縮文コーパスから学習する．

本手法では，まず圧縮文コーパス中の原文を構文解析し，構文木を付与する [2]．次に付与された構文木と圧縮文から，どのような削除が行われたかを求め，その頻度を元に削除確率を推定する．得られた確率モデルにより，起こりやすい削除を求めることができ，圧縮文として尤もらしいものを得ることができる．

まず，原文の構文木とその圧縮文の対から，どのような削除が行われたかを求める方法について説明する．

本手法では，原文の構文木に対してトップダウンに削除操作の適用を試み，その削除が，圧縮文中の単語を削除しない場合に限り削除操作を適用する．

例として，文 (5) が原文として，文 (6) が圧縮文として与えられた場合を示す．図 9 に原文の構文木を再掲する．図中の太線でマークされた単語は圧縮文に含まれている単語である．$\mathbf{S_1}$ は，再帰ノードである．$\mathbf{S_1}$ に MRP の削除を適用すると，“It”，“is”，“likely”，“that” の 4 単語が削除される．これらの中には，圧縮文中の単語は含まれていないため，$\mathbf{S_1}$ に対してこの削除操作が適用される．この削除操作により，$\mathbf{NP_2}$, $\mathbf{PRP_3}$, ···,

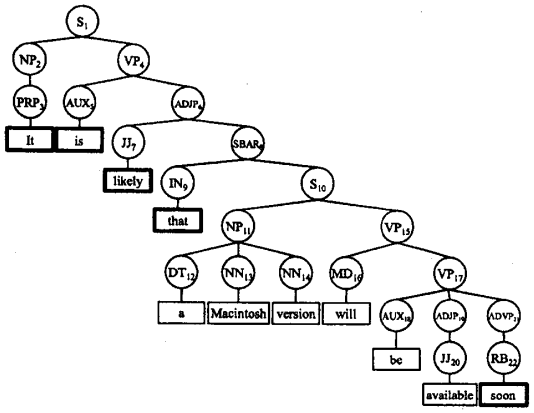

図 9　文 (5) の構文木

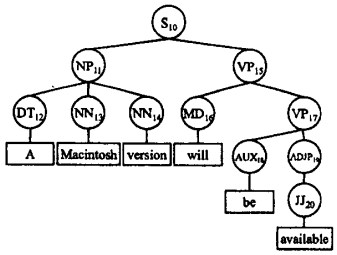

図 10　本手法によって得られる文 (6) の構文木

$\mathbf{IN_9}$ のノードが同時に削除される．

次に，$\mathbf{S_{10}}$ は，非再帰ノードである．$\mathbf{S_{10}}$ に構成素の削除を適用すると，残っている 7 単語すべてが削除されるが，これらの中には，圧縮文の単語が含まれるため，$\mathbf{S_{10}}$ に対して削除操作の適用は行われない．同様の理由により $\mathbf{NP_{11}}$ から $\mathbf{NN_{14}}$ には構成素の削除は適用されない．

続く $\mathbf{VP_{15}}$ は再帰ノードである．$\mathbf{VP_{15}}$ に MRP の削除を適用すると，"will" が削除されるが，これは圧縮文に含まれる単語であるため，MRP の削除は適用されない．

$\mathbf{MD_{16}}$ から $\mathbf{JJ_{20}}$ には，構成素の削除は適用されない．非再帰ノード $\mathbf{ADVP_{21}}$ に対して構成素の削除を適用し，圧縮文に含まれない "soon" を削除する．

このように，$\mathbf{S_1}$ と $\mathbf{ADVP_{21}}$ に対して削除操作を適用することで，圧縮文の構文木が得られる（図 10 参照）．この構文木は文法的に正しい．

3.3　最大エントロピー法による学習

前節の方法により，コーパス中の原文と圧縮文の対において，どのような削除操作が適用されているのかを検出した後，本手法では，削除操作適用の確率を最大エントロピー法 [1] により学習する．学習に使用した素性は以下の通りである．

（a）　削除操作の種類（構成素の削除，または，MRP の削除）

（b）　注目しているノードのラベル

（c）　親ノードのラベル

（d）　子ノードのラベル

（e）　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（構成素の削除の場合のみ）

（f）　MRP 上のノードのラベル（MRP の削除の場合のみ）

（g）　MRP 上のノードの子ノードのラベル（MRP の削除の場合のみ）

（h）　足ノードのラベル（MRP の削除の場合のみ）

3.4　確率的文圧縮モデル

この節では，圧縮文の生成確率の計算方法について述べる．本手法では，原文 l が圧縮文 s に圧縮される確率を，l の構文木に削除操作を適用して s の構文木が得られる確率と定義する．各基本単位の削除は独立であると考え，この確率を，基本単位の削除確率の積により求める．すなわち，次のように定義する．

$$P(s|l) = \prod_{\eta \in N} P(a_\eta | \eta, l)$$

ここで，N は l の構文木のノードの部分集合で，削除操作が適用されなかったノード（= s の構文木に残ったノード），及び削除操作が適用されたノードからなる．祖先ノードに削除操作が適用された結果として削除されたノードは含まない．a_η は，η に削除操作が適用されたとき 1 となり，そうでないとき 0 となる．

このモデルに従えば，文 (5) が文 (6) に圧縮される確率は以下のように求められる．なお，簡単のため，l は省略している．

$$P(1|\mathbf{S_1})\,P(0|\mathbf{S_{10}})\,P(0|\mathbf{NP_{11}})\,P(0|\mathbf{DT_{12}})\,P(0|\mathbf{NN_{13}})$$
$$P(0\,|\,\mathbf{NN_{14}})\,P(0\,|\,\mathbf{VP_{15}})\,P(0\,|\,\mathbf{MD_{16}})\,P(0\,|\,\mathbf{VP_{17}})$$
$$P(0|\mathbf{AUX_{18}})P(0|\mathbf{ADJP_{19}})P(0|\mathbf{JJ_{20}})P(1|\mathbf{ADVP_{21}})$$

3.5　スコアの計算

前節のモデルをもとに，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 らの手法と同一である．一般に，短い圧縮文の確率が過剰に高くなる傾向があるため，対数確率を圧縮文の長さによって正規化する必要がある．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．

4.　評価実験

実験と評価には Knight ら [5] が Ziff-Davis コーパスから抽出した，原文と圧縮文の対応の取れたコーパスを使用した．原文と圧縮文の間で，単語の対応が一意に決定できる 943 文をトレーニングデータとし，Knight らと同じテストデータ 32 文に対して文圧縮を適用し評価を行った．

人手の評価によって，提案手法を Knight らの Noisy-channel モデルによる手法と比較した．また，再帰構造除去の純粋な有効性を確かめるため，本手法において再帰構造の除去を行わない場合（以下，比較手法）と比較した．提案手法において，スコアを調整するためのパラメータ α は，トレーニングデータから無作為に抽出した 50 文に対してオープンな実験を行い，圧

表 1　人手による評価の結果

	圧縮率	文法性	意味
Knight	70.4%	4.05	3.80
提案手法	50.7%	4.20	3.41
比較手法	50.5%	2.65	2.95
人手	53.3%	4.44	3.67

縮率が人手による圧縮と等しくなるように定めた．その値は $\alpha = -0.43$ であった．同様に比較手法については $\alpha = -0.21$ とした．

テストデータ中の文を，比較する 4 種類の圧縮文（人手，Knight らの手法，提案手法，比較手法によるもの）と併せて，4 人の被験者に提示し，文法性と意味の保持の観点から，各圧縮文を 1 から 5 の 5 段階で評価した．被験者は，4 種類の圧縮文が，すべて異なるシステムによって自動生成されたものだと伝えられている．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かえた．

実験結果を表 1 に示す．圧縮文の文法性に関して，我々の手法は，Knight らの手法に比べ，より良い評価を得ている．主要な意味の保持に関しては，Knight らの手法が，人手，及び我々の手法を上回る結果となったが，これには圧縮率の差が強く影響しており，単純な比較はできない．また，提案手法と比較手法の評価値より，再帰構造の除去の有効性を確認した．

表 2 に，人手，Kight らの手法，Unno らの手法，我々の手法による文圧縮の例を示す．これらの例文は，Unno らが文献 [9] で文圧縮の例として提示した文である．1 つ目の例は，Kight らが圧縮に失敗し，Unno らがボトムアップ手法によって成功した例である．提案手法でも正しく圧縮されている．2 つ目の例は，Unno らのボトムアップ手法でも正しく圧縮できない例であるが，提案手法では，構文木の再帰構造を除去することによって，正しく圧縮を行っている．3 つ目の例では，人手による圧縮を含むすべての圧縮文が異なっているが，提案手法による圧縮が，既存の手法によるものよりも，文法性と意味の保持という観点から優れているといえる．

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を行う確率的モデルを提案した．本モデルでは，既存の手法で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．

実験により，提案手法が，構成素の削除のみを行う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．本実験では，提案手法の圧縮率を人手による圧縮と等しくなるように定めたが，圧縮率は意味の保持の程度にも影響を与える．その関係について調べる実験を今後実施したい．

謝辞　圧縮文コーパス，及び実験データを提供していただいた，南カリフォルニア大学の Kevin Knight 氏，ならびに，Daniel Marcu 氏に深く感謝いたします．

表 2　文圧縮の例

原文	The user can then abort the transmission, he said.
人手	The user can then abort the transmission.
Knight	The user can abort the transmission said.
Unno	The user can then abort the transmission.
提案手法	The user can then abort the transmission.
原文	It is likely that both companies will work on integrating multimedia with database technologies.
人手	Both companies will work on integrating multimedia with database technologies.
Knight	It is likely that both companies will work on integrating.
Unno	It is will work on integrating multimedia with database technologies.
提案手法	Both companies will work on integrating multimedia with database
原文	A file or application "alias" similar in effect to the MS-DOS path statement provides a visible icon in folders where an aliased application does not actually reside.
人手	A file or application alias provides a visible icon in folders where an aliased application does not actually reside.
Knight	A similar in effect to MS-DOS statement provides a visible icon in folders where an aliased application does reside.
Unno	A file or application statement provides a visible icon in folders where an aliased application does not actually reside.
提案手法	A file or application "alias" similar in effect to the MS-DOS path statement provides a visible icon in folders.

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] E. Charniak, "A Maximum-Entropy-Inspired Parser, " Proc. 6th ANLP, pp.132-139, 2000.

[3] 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.

[4] H. Jing, "Sentence Reduction for Automatic Text Summarization, " Proc. 6th ANLP, pp.310-315, 2000.

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

[6] K. Knight, and D. Marcu, "Summarization beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression, " Artificial Intelligence 139, pp.91-107, 2002.

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

[8] J. Turner, and E. Charniak, "Supervised and Unsupervised Learning for Sentence Compression, " Proc. 43rd ACL, pp.290-297, 2005.

[9] 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.