E-035

文法性判定に基づくクエリ指向の文圧縮
Query-focused Sentence Compression based on Grammaticality Judgment

林 律希[1)]　加藤 芳秀[1)]　松原 茂樹[1)]
Ritsuki Hayashi　Yoshihide Kato　Shigeki Matsubara

1　はじめに

ユーザのクエリに焦点を当てた文圧縮は，Web 検索の結果の提示などにおいて有用である．従来の手法 [1] では，入力である原文，クエリ，圧縮文の長さの上限と出力である圧縮文の組からなる学習データを用いて文圧縮モデルを構築するが，学習データの作成やモデルの学習に労力を要するという問題がある．

そこで本論文では，そのような学習を必要としないクエリ指向の文圧縮手法を提案する．本手法では，依存構造に基づき圧縮文の候補を求め，既存の文法性判定手法を用いて文法的に正しい圧縮文を選択する．圧縮文の候補は，クエリを含むという制約のもとで選ばれるため，クエリに焦点を当てた文圧縮が実現できる．

2　クエリ指向の文圧縮

文圧縮は，原文の持つ情報をできる限り保持したまま，より短い文を生成するタスクである．その手法として，原文から重要でない単語を削除し，残りの断片をつなぎ合わせることで文を圧縮する抽出型文圧縮と，単語の削除以外の操作，例えば，並び替え，置換，挿入により，原文を作り直す抽象型文圧縮があり，既存の手法のほとんどが抽出型文圧縮に基づいている．抽出型文圧縮では，単語削除問題として扱う．すなわち，原文

$$S = w_1 w_2 ... w_n$$

が与えられたとき（w_i は文 S の i 番目の単語を表す），原文 S における単語の部分集合を削除して圧縮文 C を生成する．

一般的な文圧縮では，以下の 3 つの制約条件

- C は文法的に正しい
- C は指定の長さを超えない
- C は原文 S の主要な情報を保持する

を満たすように圧縮文 C を生成する．

クエリ指向の文圧縮では，原文に加えてクエリ $Q \subseteq S$ が与えられる．圧縮文 C は，上の 3 つの制約に加えて，次の制約を満たすことが求められる．

- C はクエリ Q の単語を保持する，すなわち $Q \subseteq C$

.

3　従来手法とその問題点

Handler らは，クエリ指向の文圧縮を実現するために，クエリのみからなる集合を初期状態とし，順次，原文の単語を付加する操作を繰り返すことで，圧縮文を生成する手法を提案している [1]．この手法では，付加する操作を適用するのかを，圧縮文に関する大規模なパラレルコーパスから学習する．そのための学習データを，通常の圧縮文コーパス [2] に疑似的にクエリ，及び圧縮文の長さの条件を付与することにより作成している．疑似的に作成されたコーパスを使用している理由として，クエリの情報が付与されたコーパスの作成が容易ではないことが挙げられる．

1）　名古屋大学

4　提案手法

本節では，学習データを必要としない，クエリ指向の文圧縮手法を提案する．本手法では Algorithm 1 に示す処理を実行し，圧縮文を生成する．以下では，提案手法を構成する要素技術について説明する．

Algorithm 1 Compress(S, Q, b)

```
D ← Parse(S)                                   ▷ 依存構造解析
R ← Operations(S, D, Q)                        ▷ 可能な削除操作
L ← {S}
while L ≠ ∅ do
    C ← pop(L)
    for all r ∈ R do
        C′ ← C − r
        push(C′, L)
        if isGrammatical(C′) ∧ length(C′) ≤ b then
                                               ▷ 文法性判定
            return C′
        end if
    end for
end while
fail
```

4.1　依存構造解析

提案手法における最初のステップ Parse(S) は，原文 S の依存構造解析である．本手法では，Universal Dependencies[3] に基づく依存構造解析器を使用する．例えば，“I know that the man who is running over there is his friend.”という原文を解析すると図 1 左に示す依存構造木が得られる．

4.2　削除操作

文圧縮は，単語の削除によるが，可能な削除は依存構造木に基づき決定する．

依存構造木に基づく従来の文圧縮手法 [4] では，圧縮文に対して依存構造木が割り当てられることを保証するために，部分木の削除による圧縮が行われる．これは原文における付加的な単語や句を除いたものを圧縮文とすることに相当する．しかし，圧縮文としては，従属節など原文の付加的な要素を中心に文を構成するのが適している場合がある．部分木の削除のみではこのような圧縮文の生成は困難である．そこで提案手法では，部分木の削除に加え，部分木の外側の削除という操作を導入する．この操作により，原文から従属節や付加的な名詞句を抽出することができる．具体的には，依存構造の各ノード $w_i (1 \leq i \leq n)$ に対して以下の 2 種類の削除操作を採用する．各削除操作の例を図 1 に示す．

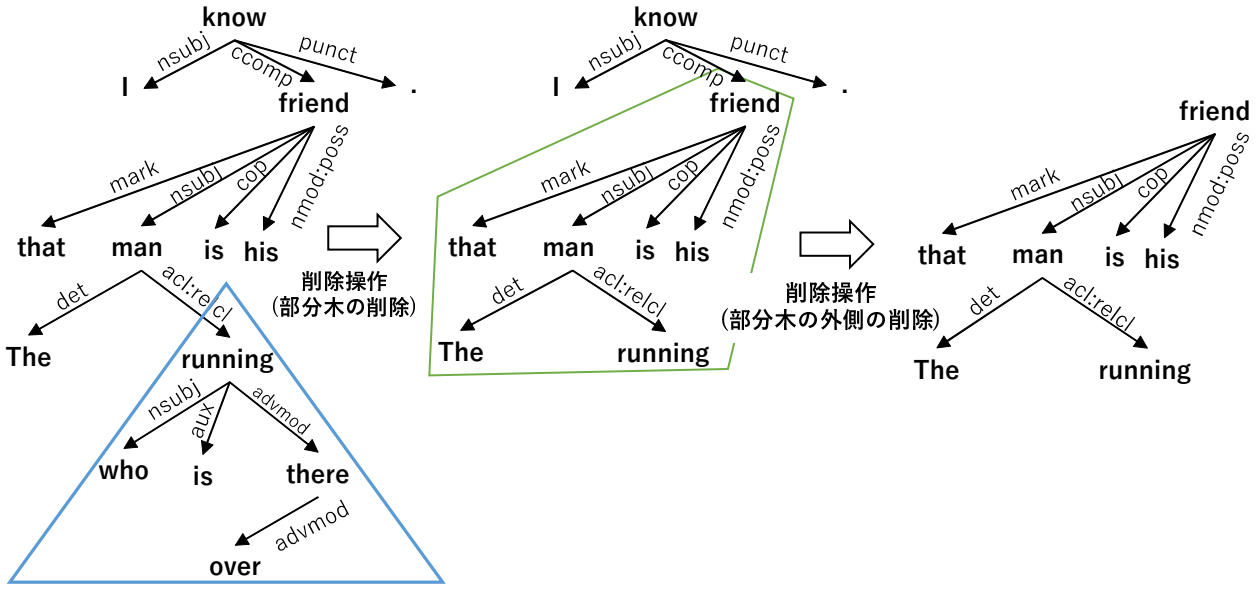

図 1　削除操作

部分木の削除　ノード w_i が支配するすべてのノード $\mathrm{dom}(w_i)$ を削除する（図 1 中）．

部分木の外側の削除　ノード w_i が支配しないノード，すなわち $S - \mathrm{dom}(w_i)$ を削除する．これは，ノード w_i をルートとする部分木の外側の単語をすべて削除する操作である（図 1 右）．

クエリ指向の文圧縮では，クエリ内の単語が圧縮文に含まれる必要がある．提案手法では，上述の削除操作において，削除される箇所にクエリ内の単語が含まれているとき，その削除操作を除外する．

Algorithm 1 の Operations(S, D, Q) は可能な削除操作の集合である．これは Algorithm 2 のように整理できる．

Algorithm 2 Operation(S, D, Q)

```
R ← ∅
for all w_i ∈ S do
    if dom(w_i) ∩ Q = ∅ then
        R ← R ∪ {dom(w_i)}
    end if
    if (S − dom(w_i)) ∩ Q = ∅ then
        R ← R ∪ {S − dom(w_i)}
    end if
end for
return R
```

4.3　削除操作の組み合わせ

前節で述べた削除操作を組み合わせ，それらを文に適用することにより，圧縮文の候補を生成する．削除操作を適用するたびに，圧縮文の候補が，文法的に正しいかどうかを判定する（後述）．正しいと判定され，長さの制約も満たしている場合は，これを出力して終了する．あらゆる削除操作の組み合わせを行っても，文法的に正しく，かつ長さの制約を満たす圧縮文を生成できない場合は，圧縮不可能として処理を終了する．

4.4　文法性判定

提案手法では，生成された圧縮文の候補について，それが文法的に正しいか否かを判定する（Algorithm1 における isGrammatical）．これは，文法誤り訂正モデルを活用して実現する．大規模言語モデルに基づく文法誤り訂正モデル（例えば，[5]）により，高い精度で文法性を判定できる．

5　おわりに

本論文では，クエリ指向の文圧縮手法を提案した．本手法は，依存構造に基づき圧縮文の候補を求め，既存の文法誤り訂正モデルを用いて文法的に正しい圧縮文を選択するものであり，学習データを必要としないという利点がある．今後，文圧縮実験により提案手法を評価する予定である．

謝辞　本研究は，一部，科学研究費補助金（基盤研究（B））（No.21H03773）により実施したものである．

参考文献

[1] Hander, A. and O' Connor, B. "Query-focused Sentence Compression in Linear Time", In Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), pp. 5969–5975 (2019).

[2] Filippova, K. and Altun, Y. "Overcoming the Lack of Parallel Data in Sentence Compression", In Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-2013), pp. 1481–1491 (2013).

[3] De Marneffe, M. C., Manning, C. D., Nivre, J., and Zeman, D. "Universal Dependencies.", Computational Linguistics, Vol. 47, No. 2, pp. 255–308 (2021)

[4] Filippova, K. and Strube, M. "Dependency Tree Based Sentence Compression", In Proceedings of the Fifth International Natural Language Generation Conference (INLG 2008), pp. 25–32 (2008).

[5] Yasunaga, M. Leskovec, J. and Liang, P. "LM-Critic: Language Models for Unsupervised Grammatical Error Correction", In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP-2021), pp. 7752–7763 (2021).