劣モジュラ最大化アルゴリズムを用いた文抽出と文圧縮に基づくクエリ指向要約

森田　一 † 　　笹野　遼平 ‡ 　　高村　大也 ‡ 　　奥村　学 ‡

† 東京工業大学 総合理工学研究科, ‡ 東京工業大学 精密工学研究所

morita@lr.pi.titech.ac.jp, {sasano, takamura, oku}@pi.titech.ac.jp

1　はじめに

近年, 文書要約を劣モジュラ関数の最大化として扱う研究が多く行われている. これは貪欲法によって劣モジュラ関数が高速に近似的な最大化が可能であることに加え, 整数計画法等を用いた場合と比べ自由に目的関数を選ぶことができるためである. 一方で, ナップサック制約を含む, 現在要約に用いられている単一のマトロイドによって制約された劣モジュラ最大化では, 本稿で扱う, 単純な文抽出以外の, 文圧縮と文抽出を同時に行う場合等を扱うことはできない.

一般的な文抽出要約では, 長い文の一部のみが重要な情報を含み, 他の部分は冗長あるいは重要でない場合にも, 不要な部分を含めて文を選択する必要がある. しかし, 文圧縮を要約の前あるいは後に行っても, すべての不要な箇所を選択する文から取り除くことはできない. 圧縮をあらかじめ行ってから要約をする場合には重要でない情報は要約前に捨てることができるが, 複数の文に共通した重要な情報は, あらかじめ圧縮をすることにより除くことはできず, 圧縮と選択を同時に行いすべての圧縮文候補から文を選択して要約を作る必要がある.

このため本稿では, 追加する要素のコストが既に被覆された集合によって減少する予算制約付き劣モジュラ最大化問題である, コスト漸減劣モジュラ最大化問題を新しく提案する. この最大化問題を利用し, 日本語において圧縮と選択を同時に行うための要約モデルの提案を行い, クエリ指向要約に対して適用する.

2　コスト漸減予算制約付き劣モジュラ最大化問題

2.1　問題定義

本稿では図 1 に示すように, 係り受け木から要約に必要な内容に応じて, 部分木 (a) や部分木 (b) のような

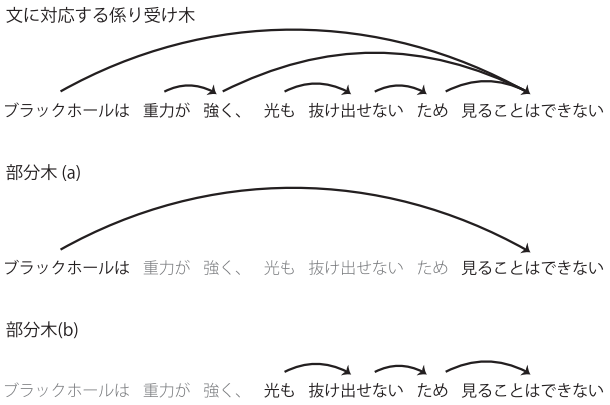

図 1: 部分木抽出による文圧縮および抽出.

に必要な部分木だけを抜き出すことで, 文圧縮と文抽出を行う.

文に対応する各ノードを文節とする係り受け木の部分木のうち, 自然な文に対応する部分木を有効な部分木とし, V をソース文書中の全ての有効な部分木の集合とする. 必須格および直前格を表す係り受けで結ばれた文節が分断されないようにするため, それらを表す係り受けで結ばれた文節は, 係り受け木上の 1 ノードとして表す. このため, 本稿では有効な部分木とは単に係り受け木の根を含む部分木を表すものとする. ここで, 有効な部分木を複数マージした部分木と等しい有効な部分木が必ず存在することに注意する. ソース文書中の各文 s_i ごとの係り受け木の部分木の集合を V_i とすると, 全体の部分木集合は $V = \bigcup_i V_i$ として表すことができる. $f(\cdot)$ を non-decreasing monotone submodular である目的関数, 生成する要約 S を有効な部分木集合の部分集合, $c(\cdot)$ を集合に対するコストを表す関数, L を予算の上限としたとき, S について目的関数を最大化する問題としてコスト漸減予算制約付き劣モジュラ最大化問題を次のように表す:

$$\max_{S \subseteq V} \{ f(S) : c(S) \leq L \}. \qquad (1)$$

ただし, $c(\cdot)$ は単純に被覆された部分木のコストの和

ではなく, 被覆された部分木の集合が表す文字列の長さを表す. これは, 文字数制限以下の要約長になるよう生成される要約のサイズを制約している. 同じ文中の複数の部分木を選択した場合には, 複数の部分木が表す文字列が重複するため, それらを被覆するコストは個々のコストの和よりも小さくなる. 例えば, 単語 $\{w_{\mathrm{a}}, w_{\mathrm{b}}, w_{\mathrm{c}}\}$ を含む部分木 t を, すでに同じ文から $\{w_{\mathrm{a}}, w_{\mathrm{b}}, w_{\mathrm{d}}\}$ を被覆している要約に追加するとき, t の追加に必要となるコストは w_{a} と w_{b} が既に要約に被覆されているため $c(\{w_{\mathrm{c}}\})$ のみとなる. 別の文からそれぞれ部分木を選択した場合には, それらの部分木がソース文書の同じ箇所を表すことはありえないため, 被覆するコストは単純にそれぞれのコストの和となる. それゆえ, 式 (1) は各文ごとのコストの和として,

$$\max_{S \subseteq V} \left\{ f(S) : \sum_{V_i \in V} c\left(V_i \cap S\right) \leq L \right\}, \qquad (2)$$

のように表すことができる.

2.2 貪欲法

この最大化問題を解くためのアルゴリズムを Algorithm 1 に示す. ここでは, G_i は部分木 s_i を部分木の集合 G_{i-1} に追加することで得られる要約を表し, U は 5 行目の選択においてその時点で選ばれていない部分木の集合を示す. パラメータ r は Lin et al. [6] によって提案された, 密度を計算する際のコストのスケーリングを行うパラメータである. 本稿では, Khuller ら [3] および Krause ら [4] のアルゴリズムを基とし, 要約のコストをコスト関数を用いて計算するように変更を行った. 5 行目では圧縮と文選択を同時に行うため, 一度も選択されていない部分木集合 U から, コストあたりの目的関数の増分が最も多い最密部分木 s_i を選択する. 予算制約を満たす場合には順に要約 G_{i-1} に加え, 追加できる部分木がなくなった時点で, G_i と全部分木の中で最も高いスコアをもつ $\{s^*\}$ を比較し, スコアの高い方を要約として出力する.

また, この Algorithm 1 はこれまでの予算制約付き劣モジュラ最大化問題の場合と同様に, 近似性能が保証されることを示す.

Theorem 1 *$f(\cdot)$ を non-decreasing normalized monotone submodular である関数, $r = 1$ とするとき, Algorithm 1 は以下の定数近似率をもつ:*

$$f(G_f) \geq \left(\frac{1}{2}(1 - e^{-1})\right) f(S^*). \qquad (3)$$

ただし, S^* を最適解, G_f をこのアルゴリズムによって得られる解とする.

Algorithm 1 貪欲アルゴリズム

1: $G_0 \leftarrow \phi$
2: $U \leftarrow V$
3: $i \leftarrow 1$
4: **while** $U \neq \phi$ **do**
5: $\quad s_i \leftarrow \underset{s \in U}{\operatorname{argmax}} \frac{f(G_{i-1} \cup \{s\}) - f(G_{i-1})}{(c(G_{i-1} \cup \{s\}) - c(G_{i-1}))^r}$
6: $\quad$ **if** $c(\{s_j\} \cup G_{j-1}) \leq L$ **then**
7: $\quad\quad G_i \leftarrow G_{i-1} \cup \{s_i\}$
8: $\quad\quad i \leftarrow i + 1$
9: $\quad$ **end if**
10: $\quad U \leftarrow U \backslash \{s_i\}$
11: **end while**
12: $\bar{s} \leftarrow \underset{s \in V, c(s) \leq L}{\operatorname{argmax}} f(\{s\})$
13: return $G_f = \underset{S \in \{\{\bar{s}\}, G_i\}}{\operatorname{argmax}} f(S)$

Proof. 証明は Krause ら [4] の Theorem 1 と同様に行うことができる. Krause らの Lemma 2 をコストが漸減する場合に一般化するため, Lemma 2 における $PT \backslash \mathcal{G}_{i-1}$ を等価で各文一つの部分木に置き換える. 詳細は [13] を参照して欲しい. □

2.3 最密部分木探索

Algorithm 1 の 5 行目では, 文に対応する係り受け木のすべての部分木のなかで, 最も追加コストあたりのスコアの増分が大きい, 最密部分木を選択する. スコアが定数で表される場合に最密部分木を探索するアルゴリズムは, Hsieh ら [1] によって提案されており, 本稿でもこのアルゴリズムをベースとして, 各単語ごとのスコアが劣モジュラ関数によって与えられる場合に拡張する.

本稿では目的関数は単語ごとにスコアが独立で, かつ同一文中で二回目以降に出現した単語のスコアは数えないと仮定する. つまり, 同じ単語が複数回出現しなければ, 最密部分木を求める過程では各ノードのスコアは定数と考えることができる. このため, 予め文内で重複しうる単語を列挙して重複単語リストを用意しておき, リスト中の各単語を含むかどうかで場合分けを行うことで今回の目的関数に対して動的計画法を適用することができる.

このアルゴリズムではボトムアップに, あるノードを頂点とする部分木に対して, 図 2 のように部分木のコストごとおよび重複する単語ごとにスコアが最大となる部分木を計算する. 同じコストでかつ部分木に含まれるリスト中の単語が同じ場合, 必ずスコアの高い方を選択したほうがその部分木を含む部分木の密度は高くなる. より上位のノードを頂点とする最大スコア

の部分木は, 下位の最大スコアをとる部分木以外を含むことはないので, 動的計画法により効率的に求めることができる.

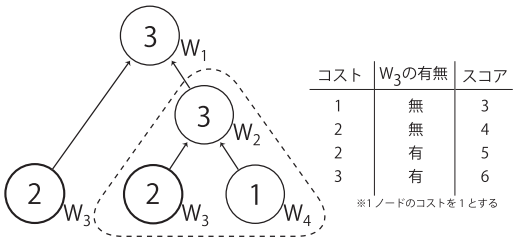

図 2: コストと重複リスト中の各単語を含むかどうかごとの, スコアが最大となる部分木の列挙. ノード内の数字はスコア, 右下の w の添字は単語の種類を表す.

2.4 目的関数

本稿では, 複数文書からクエリに従って文の抽出と圧縮を行う. このクエリ指向要約のための文圧縮を行うには, 圧縮する単位に応じたクエリとの関連スコアが必要となる. ソース文書中の各単語とクエリとの関連度スコアを計るため Query SnowBall (QSB) [12] を用いた. QSB はクエリ指向要約のための手法で, ソース文書中の単語の共起関係を用いて各単語にクエリとの関連度スコアを与える. この関連度スコアを基にして, 冗長でなくかつ断片的でない要約を生成するための目的関数を与える.

まず, $d < 1$ を減衰率, $count_S(w)$ を要約 S に単語 w が含まれる文の数, $n(s)$ を要約に含まれる文の数, $qsb(w)$ を単語 w の関連度スコアとして,

$$f(S) = \sum_w \left\{ qsb(w) \sum_{i=0}^{count_S(w)} d^i \right\} + \gamma(c(S) - n(S)). \quad (4)$$

第一項は各単語についてのスコアを表し, 同じ単語については追加するたびにスコアの増分が d にしたがって減衰することで, 冗長性に対するペナルティを含めたスコアとしている. 第二項は, 断片化していないことに対する positive reward となっており, 断片化し要約に含まれる文数が増えると高くなりにくい値となっている. この目的関数は単語ごとのスコアの和として計算できるため, 動的計画法を用いて効率的に最密部分木を探索することができ, Algorithm 1 を用いて最大化を行うことができる.

3 実験設定

提案手法を評価するため, 日本語質問応答テストコレクションである NTCIR-7 ACLIA1 [10] および NTCIR-8 ACLIA2 [11] を用いて実験を行った. 各コレクションには質問と重み付けられた解答ナゲットが含まれている. 実験設定は要約長の設定を除き, Morita ら [12] の設定に従い, 質問をクエリとして指定された新聞記事から 140 文字を上限とする要約を生成し解答ナゲットがどれだけ含まれるかを判定することで評価を行う. 形態素解析には JUMAN [9] を, 係り受け解析には KNP[5] を利用している. *idf* の計算には毎日新聞の 1991 年度から 2005 年度の記事を用いた. また, 必須格および直前格の判定は KNP を通して, 京大格フレーム [2] を利用している.

最終的な評価に必要となる人手によるナゲットのマッチングは高コストであるため, 自動評価のための指標として, POURPRE [8] を用いた. 断片的なテキストに対してスコアを与えることを防ぐ目的で, Mitamura ら [11] の設定に従いナゲットのマッチングは閾値 0.5 を用いて二値化している. ベースラインおよび提案手法のパラメータ調節は開発データ上でこの POURPRE に従って決定した. ベースラインとしては, Lin ら [7] による劣モジュラ最大化を用いたクエリ指向要約の手法を用いた. この手法は DUC 04 から 07 において, 文圧縮と組み合わせた手法を含めそれ以前の手法を上回る性能を示している. 手法に用いる文書のクラスタリングでは, N をソース文書中の文数として, K={0.2N,0.15N,0.05N} として事前に k-means で求めたクラスタを用いている.

4 結果

表 1 に実験結果を示す. 表中の “Subtree extraction (SE)” は提案手法を表す. “No compression (NC)” は提案手法と比較するため, 同じ目的関数で文抽出のみを行った場合の結果である. また, 圧縮によって対応する箇所が失われ, 被覆できなくなってしまったナゲットを調べるため, “Subtree extraction*” では提案手法の文圧縮で除かれた箇所を復元し, 仮に同じ圧縮率で理想的に圧縮が行われた場合の Recall を調べた. 提案手法の F1 値と F3 値はそれぞれ 0.159 と 0.190 となり, ベースラインの 0.135 と 0.174 を上回った. ベースラインの手法では, 多様なクラスタから文を抽出しやすくなるように目的関数を与えることにより要約に多様性を持たせているが, 提案手法ではパラメータ d によって直接重複する単語に対するペナルティを設定することで, 類似したナゲットを要約に含める際に避けられない単語の重複を許容できるようになっている. 実際に, 開発データで調整された減衰率 d の値は圧縮を行わない NC では比較的大きい 0.8 が, 文圧縮によって冗長な箇所を自由に除くことができる提案手法

表 1: ACLIA2 テストセットでの結果

	POURPRE	Precision	Recall	F1	F3
Lin et al.[7]	0.215	0.126	0.201	0.135	0.174
Subtree extraction(SE)	0.268	0.156	0.213	**0.159**	**0.190**
No compression(NC)	0.278	0.131	0.215	0.139	0.183
Subtree extraction*	—	—	0.228	—	—

では比較的小さな 0.2 が選ばれている.

提案手法 SE を圧縮を除いたモデル NC と比較すると, POURPRE では NC が提案手法を上回っているが, 人手の評価である F1 値 及び F3 値の両方において性能の向上が見られる. しかし, 文圧縮を組み合わせたことにより, precision は向上している一方で, recall はあまり変わっていない. また, 選択した文に対して理想的に圧縮が行われた場合との比較を行い, 以下の結果を得た. 提案手法において選択された文が含む合計 108 の解答ナゲットのうち, 8 は文圧縮によって削除され要約には含められていない. このため, 理想的に圧縮が行われ, 選択した文の全てのナゲットが要約に含まれた場合と比較すると, 提案手法の recall は 7% 低下している. しかし, 文圧縮によって 19% の文字列が削減された一方で recall の低下は 7% に留まるため, 圧縮を組み合わせることでよりクエリに関連した箇所を要約として出力できている.

5 まとめと今後の課題

劣モジュラ最大化問題の一種として文圧縮と文抽出を同時に行う要約モデルを提案し, 実際にこのモデルを扱うための新しい劣モジュラ最大化問題である, コスト漸減予算制約付き劣モジュラ最大化問題を提案した. また, この問題が貪欲法により定数近似が可能であることを示し, そのアルゴリズムを用いて F3 値 0.19 を ACLIA2 日本語テストコレクションにおいて達成した. 今後の課題としては, 単語ごとにスコア関数が独立でなければいけないという目的関数に対して課せられている制約の緩和および, 日本語以外の言語への適用を考えている.

参考文献

[1] Sun-Yuan Hsieh and Ting-Yu Chou. The weight-constrained maximum-density subtree problem and related problems in trees. *J. Supercomput.*, 54(3):366–380, December 2010.

[2] Daisuke Kawahara and Sadao Kurohashi. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of HLT-NAACL '06*, pages 176–183.

[3] Samir. Khuller, Anna Moss, and Joseph S. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[4] Andreas Krause and Carlos Guestrin. A note on the budgeted maximization on submodular functions. Technical Report CMU-CALD-05-103, Carnegie Mellon University, 2005.

[5] Sadao Kurohashi and Daisuke Kawahara. Kn parser (kurohashi-nagao parser) 3.0 users manual. `http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP`.

[6] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of HLT-NAACL '10*, pages 912–920.

[7] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of ACL-HLT '11 - Volume 1*, pages 510–520.

[8] Jimmy Lin and Dina Demner-Fushman. Methods for automatically evaluating answers to complex questions. *Inf. Retr.*, 9(5):565–587, November 2006.

[9] Yuji Matsumoto, Sadao Kurohashi, Yutaka Nyoki, Hitoshi Shinho, and Makoto Nagao. User's guide for the JUMAN system, a user-extensible morphological analyzer for Japanese. `http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN`.

[10] Teruko Mitamura, Eric Nyberg, Hideki Shima, Tsuneaki Kato, Tatsunori Mori, Chin-Yew Lin, Ruihua Song, Chuan-Jie Lin, Tetsuya Sakai, Donghong Ji, and Noriko Kando. Overview of the ntcir-7 aclia tasks: Advanced cross-lingual information access. In *Proceedings of the 7th NTCIR Workshop*, 2008.

[11] Teruko Mitamura, Hideki Shima, Tetsuya Sakai, Noriko Kando, Tatsunori Mori, Koichi Takeda, Chin-Yew Lin, Ruihua Song, Chuan-Jie Lin, and Cheng-Wei Lee. Overview of the ntcir-8 aclia tasks: Advanced cross-lingual information access. In *Proceedings of the 8th NTCIR Workshop*, 2010.

[12] Hajime Morita, Tetsuya Sakai, and Manabu Okumura. Query snowball: a co-occurrence-based approach to multi-document summarization for question answering. In *Proceedings of ACL-HLT'11 - Volume 2*, pages 223–229.

[13] Hajime Morita, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. Subtree extractive summarization via submodular maximization. *to be submitted to ACL2013*.