ダブル配列上のシングル節点に着目した更新法の提案

重越 秀美† 蔵満 琢麻† 望月 久稔†
†大阪教育大学

1 はじめに

自然言語処理システムの辞書を中心に広く用いられるトライのデータ構造として，記憶領域のコンパクト性と探索処理の高速性をあわせ持つダブル配列がある[1]．しかし，ダブル配列は，キーの追加や削除を繰り返すと，更新速度が低下し，記憶領域が増加する．記憶領域の増加を抑制するために，ガベージコレクションを実行する手法が提案されている[2]が，ガベージコレクションに要する時間は無視できる程度ではない．

そこで本論文では，シングル節点に着目し，ダブル配列におけるガベージコレクションの回数を抑制することで，更新速度を高速化する手法を提案する．

2 ダブル配列における更新処理の問題点

ダブル配列は，2本の配列 Base と Check を用いて節点 n から遷移種 a による節点 m への遷移を式 (1) により定義する[1]．節点の Base 値には遷移集合を定義するための基底値，Check 値には遷移元の節点を格納する．以下，ダブル配列上の要素 n における Base 値を $\mathrm{B}[n]$，Check 値を $\mathrm{C}[n]$，節点 n からの遷移における遷移種の集合を $\mathrm{L}(n)$ と表記する．また，遷移元が同じ節点を兄弟節点，兄弟節点を持たない節点をシングル節点，トライの節点として使用されていないダブル配列上の要素を未使用要素と呼ぶ．

$$m = \mathrm{B}[n] + a, \quad \mathrm{C}[m] = n \qquad (1)$$

ダブル配列は，節点 n の遷移集合を定義する際，式 (1) を満たすために，$\{\mathrm{B}[n] + x; x \in \mathrm{L}(n)\}$ の各要素がすべて未使用要素である基底値 $\mathrm{B}[n]$ を求める必要がある[1]．以下，節点 n の遷移集合を定義するために，ダブル配列上から取得する未使用要素の集合を節点 n の新規節点集合と呼ぶ．

新規節点集合を効率的に取得するために，ダブル配列上の未使用要素をリスト構造（以下，未使用要素リスト）により管理する手法が提案されている[2][3]．未使用要素リストでは，ダブル配列の要素0をリストのヘッダとして使用し，未使用要素における Base 値をダブル配列後方へのリンク，Check 値を前方へのリンクとして，双方向のリスト構造を実現できる[3]．

未使用要素リストを実装したダブル配列では，キーの削除によりトライ上から削除された要素（以下，削除要素）を，未使用要素リストの先頭に追加して管理する．しかし，削除要素が未使用要素リスト前方に蓄積すると，新規節点集合の取得に要する未使用要素リスト上の遷移回数が増加し，更新速度が低下する．これは，トライ上の節点が削除要素の周辺に多く存在し，削除要素が新規節点集合になりづらい特徴を持つためである[3]．よって，削除要素を未使用要素リスト上から除去することで，未使用要素リスト上の遷移回数を抑制でき，更新速度を高速化できるといえる．

3 シングル節点を利用した削除要素の除去法

削除要素を未使用要素リスト上から除去して，シングル節点に利用することで，削除要素を除去するガベージコレクションの実行回数を抑制し，更新速度を高速化する手法を提案する．

新たに定義する遷移集合の要素数が1つの場合，未使用要素リスト先頭の削除要素 d を新規節点集合として一意に取得できるため，容易に既存のシングル節点 s を削除要素 d の位置に定義できる．そこで，提案手法では，未使用要素 e に加えて，シングル節点 s も新規節点集合における要素の候補とし，新規節点集合にシングル節点 s が含まれる場合，削除要素 d を未使用要素リスト上から除去して，シングル節点 s を要素 d の位置に定義する．

新規節点集合 $\{s, e_1\}$ に含まれるシングル節点 s を，未使用要素リスト先頭の削除要素 d=B[0] の位置に定義し，削除要素 d を未使用要素リスト上から除去する手順を図1に示す．図1中，削除要素を除去する前後のダブル配列と未使用要素リストの状態をあわせて示す．配列の添字を四角枠で囲んでいる要素は未使用要素リスト上の要素を表し，丸枠で囲んでいる要素はトライの節点を表す．図1では，step1により新規節点集合に含まれる未使用要素 e_1 を未使用要素リスト上から除去し，step2により削除要素 d を未使用要素リスト上から除去し，step3によりシングル節点 s を削除要

Proposal of Dynamic Update Method Focused on the Single Node on a Double-Array Structure
Hidemi SHIGEKOSHI†, Takuma KURAMITSU† and Hisatoshi MOCHIZUKI†
†Osaka Kyoiku University

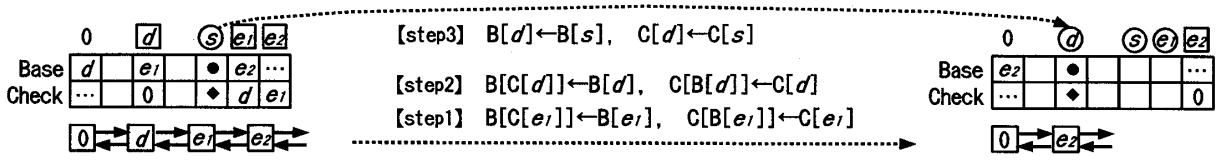

図 1: 削除要素 d の除去

素 d の位置に定義する．要素 d は，未使用要素リスト上の削除要素からトライ上のシングル節点へ変更され，$\{s, e_1\}$ はトライの節点として使用される．

4 実験による評価

提案手法の有効性を示すため，矢田らのガベージコレクション[2]を実装した比較手法との実験を Intel Core2 Duo 2.93GHz，Fedora8 上で行った．各手法には，記憶領域を抑制するためにトライ上の最終分岐以降の遷移を格納する配列 Tail[1]，追加・削除処理を高速化するために遷移集合を管理する配列 Sibling と Child[3]を実装した．また，各手法に対して，削除処理の度にガベージコレクションを実行する場合（以下，提案手法 P と比較手法 Q）と，削除処理を 1000 回行う度にガベージコレクションを実行する場合（以下，P_K と Q_K）について実験した．

実験では，wikipedia の英語タイトルから 600 万語をランダムに抽出して母集団とし，そのうち 500 万語を登録したダブル配列に対して，キーの追加もしくは削除をランダムに 100 万回繰り返した．表 1 に，100 万回の処理に要した更新時間，追加処理における未使用要素リスト上の遷移回数（以下，回数 I），ガベージコレクションの実行時における未使用要素リスト上の遷移回数（以下，回数 G），記憶領域，ダブル配列の先頭から最後方の節点までの区間に存在する節点の割合（表中，空間効率）を示す．

表 1 に示すように，各手法ともガベージコレクションにより削除要素を除去するため，空間効率は約 100%であり，記憶領域はほぼ同等である．

表 1: 実験結果

	提案手法		比較手法	
	P	P_K	Q	Q_K
更新時間 (s)	3.08	2.43	3.18	3.88
回数 I (M)	0.69	2.55	1.93	64.32
回数 G (M)	24.51	0.60	22.55	1.47
記憶領域 (M)	164.39	164.39	164.39	164.39
空間効率 (%)	99.997	99.996	99.998	99.996

Q_K は，削除処理の度に削除要素を除去する Q に比べ，更新時間が 22.0%増加した．これは，比較手法においてガベージコレクションの実行回数を抑制した場合，回数 G は抑制できるが，未使用要素リスト上の削除要素を除去しきれず，回数 I が 1.93M から 64.32M へと大幅に増加し，回数 I と G の合計が 24.48M から 66.25M に増加するためである．

これに対し，P_K は，Q_K に比べ，回数 I を 64.32M から 2.55M へと大幅に抑制した．これは，提案手法が，シングル節点を利用した削除要素の除去により，ガベージコレクションの実行回数を抑制した場合も未使用要素リスト上の削除要素を除去できるためである．また，P_K は，ガベージコレクションの実行回数を抑制することで，P に比べて回数 I と G の合計を 25.20M から 3.15M に抑制し，更新速度を高速化した．比較手法に対しては，更新時間を Q より 23.6%，Q_K より 37.4%削減した．

5 おわりに

提案手法は，シングル節点を利用して未使用要素リスト上から削除要素を除去することで，ガベージコレクションの実行回数を抑制した．その結果，提案手法は記憶領域を保ちつつ，更新時間を高速化した．今後の課題として，削除要素を除去するために用いるシングル節点の数が少ないデータを用いた検証が挙げられる．

参考文献

[1] Aoe, J.: An Efficient Digital Search Algorithm by Using a Double-Array Structure, *IEEE Transactions on Software Engineering*, Vol. 15, No. 9, pp. 1066–1077 (1989).

[2] 矢田 普，大野将樹，森田和宏，泓田正雄，吉成友子，青江順一：接頭辞ダブル配列における空間効率を低下させないキー削除法，情報処理学会論文誌，Vol. 47, No. 6, pp. 1894–1902 (2006).

[3] 重越秀美，蔵満琢麻，望月久稔：ダブル配列の遷移集合管理による追加・削除処理の高速化，第 8 回情報科学技術フォーラム (FIT2009)，RD-001，pp. 1–6 (2009).