人工知能学会研究会資料
SIG-FPAI-B501-07

ストリーム中の頻出アイテム発見のための Simple Algorithmの高速化

Faster Simple Algorithm for Finding Frequent Items in Streams

岩崎暁[1*]　坂本比呂志[1,2]
Satoru Iwasaki[1]　Hiroshi Sakamoto[1,2]

[1] 九州工業大学　情報工学府
[1] Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Japan

Abstract: We implement the simple algorithm for counting frequent items in stream data by a constant space, and compare the performance with a naive algorithm.

1　はじめに

本研究では頻出アイテム発見のための Simple Algorithm を改良して高速化を図った．近年，頻度計算を用いた文法圧縮アルゴリズムが提案されており，高速な圧縮を実現するために頻度計算の部分の高速化が求められる．文法圧縮とは，入力文字列から文脈自由文法 (context-free grammar，CFG) を構築する圧縮方法の一つである．CFG の構築方法には，出現頻度に基づいて生成規則を決定する方法がある．頻度計算を用いた文法圧縮アルゴリズムは，圧縮率は高いが頻度計算のためのメモリ使用量が多い．データマイニング等に用いられる頻度計算のアルゴリズムには，頻出アイテム発見のための Simple Algorithm のほかに Space-Saving[2], Randomized Algorithm[3] などがある．そこで本研究では，定数領域で頻度計算を行うことが可能な頻出アイテム発見のための Simple Algorithm に注目した．item counting を用いた応用として，定数領域圧縮 [4] がある．

本論文の構成は次のようになっている．まず第 2 章で必要となる基礎知識について説明する．第 3 章では今回実装した手法について説明を行う．第 4 章では計算機を用いて提案アルゴリズムの性能を，Simple Algorithm と比較する．

2　準備

準備として，頻出アイテム発見のための Simple Algorithm[1] について説明する．

*連絡先：九州工業大学　情報工学府　先端情報工学専攻
〒 820-8502　福岡県飯塚市川津 680-4
E-mail: s_iwasaki@donald.ai.kyutech.ac.jp

$|X|$ は X に含まれるアイテムの個数とする．
Simple Algorithm とは，定数領域でアイテムをカウントを行うアルゴリズムである．入力するアイテム列の長さを N，閾値を $\theta(0 < \theta < 1)$ とすると，$\frac{1}{\theta}$ 個のアイテムを最大で保持でき，最終的に θN より大きいものを必ず含むことが保証される．

Simple Algorithm では，保持しているアイテムの集合を K として，入力されるアイテム列の先頭から順にカウントしていく．

- 入力されたアイテムが既に K に含まれている場合，

  そのアイテムの頻度を 1 を増加させる．例を図 1 に示す．

- 入力されたアイテムが K に含まれていない場合，

  そのアイテムの頻度を 1 として新しく K に加える．$|K| = \frac{1}{\theta}$ の場合に K に存在しないアイテムが入力された場合，K に含まれるすべてのアイテムの頻度を 1 減少させ，頻度が θ となったアイテムを K から削除する．このときの操作に $O(\frac{1}{\theta})$ 時間かかる．例を図 2 に示す．

3　提案手法

Simple Algorithm で削除する操作に $O(\frac{1}{\theta})$ 時間かかるのに対して，$O(1)$ 時間で処理できるアルゴリズムとデータ構造を提案する．

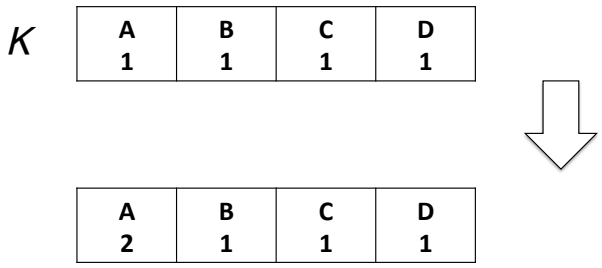

図 1: K に存在するアイテム A が入力されたときの例

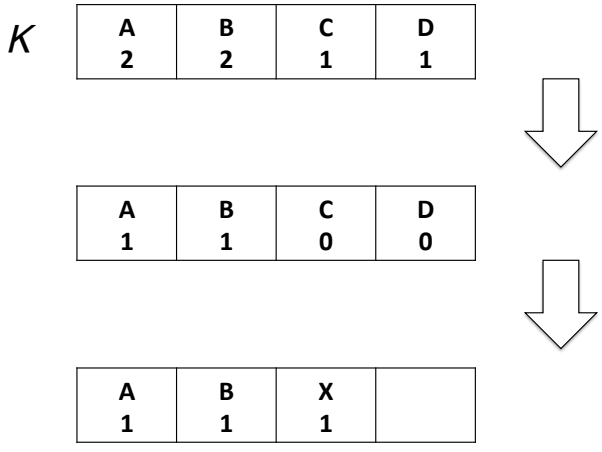

図 2: K に存在しないアイテム X が入力されたときの例

3.1 データ構造

提案するデータ構造を以下に示す.
A : 数え上げたアイテムが頻度の降順にソートされ，格納されている配列
x : A のアイテムが格納されている部分の末尾を示す配列番号
H_{item} : A 中のアイテムの有無と格納されている A の配列番号を返すハッシュテーブル
$H_{interval}$: 頻度が f である配列 A 上の区間 $[l_f, r_f]$ を返すハッシュテーブル
例を図 3 に示す.

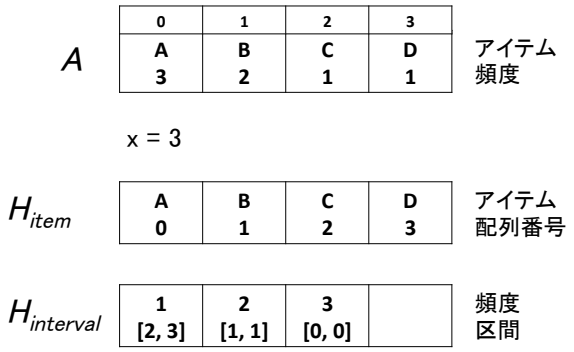

図 3: データ構造

3.2 アルゴリズム

次に，これらのデータ構造を用いたアルゴリズムを示す.

- H_{item} で A 中にアイテムが含まれていた場合,

  そのアイテム頻度を 1 増加させる．増加前の頻度を f として，$H_{interval}$ で返ってきた頻度が f である配列 A 上の区間 $[l_f, r_f]$ の l_f と現在の位置のアイテムを入れ替える．例を図 4 に示す.

- H_{item} で A 中にアイテムが含まれていなかった場合,

  $|A| = \frac{1}{\theta}$ となるまでは A の先頭からアイテムを格納する．例を図 5 に示す．$|A| = \frac{1}{\theta}$ の場合は，$H_{interval}$ で返ってきた頻度が最小である配列 A 上の区間 $[l_f, r_f]$ の l_f の位置に x を移動させる．その位置に新しく入力されたアイテムを格納する．頻度はもともとその位置に格納されていたアイテムの頻度を 1 増加させたものとする．x よりも大きい配列番号に格納されているアイテムは A に存

在しないものとみなす．このようにすることで，Simple Algorithm では $O(\frac{1}{\theta})$ かかる操作を $O(1)$ 時間ででき，Simple Algorithm と同じ挙動になる．その後新しいアイテムが入力されたときは，x のアイテムの頻度と同じ頻度で A にアイテムを格納する．また，$|A| = \frac{1}{\theta}$ の場合は同じ動作を繰り返す．例を図 6 に示す．

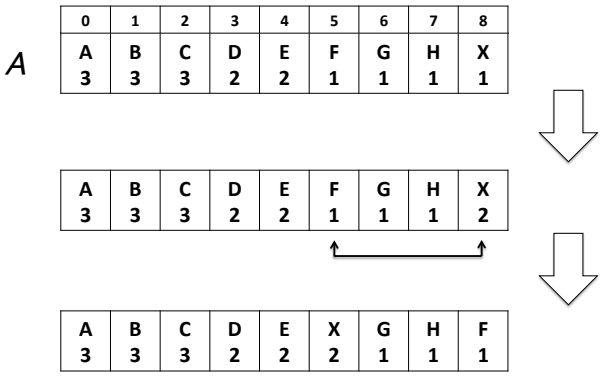

図 4: A に存在するアイテム X が入力されたときの例

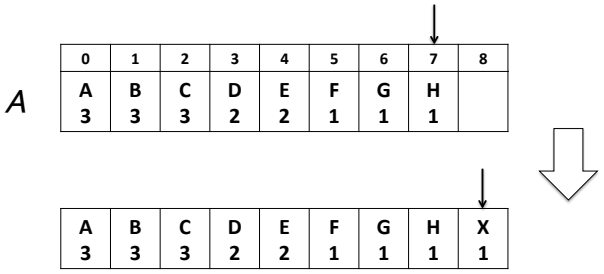

図 5: $|A| < \frac{1}{\theta}$ で A に存在しないアイテム X が入力されたときの例

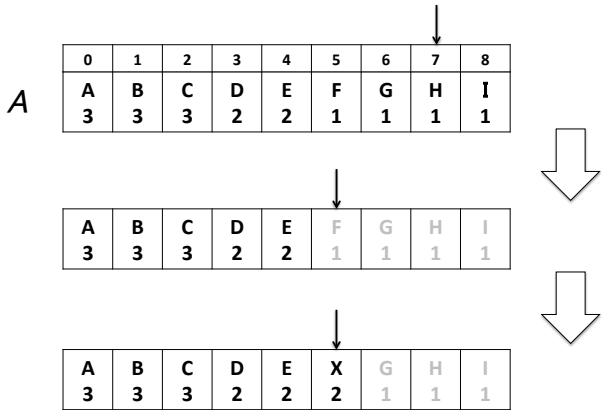

図 6: $|A| = \frac{1}{\theta}$ で A に存在しないアイテム X が入力されたときの例

4 実験

4.1 実験方法と入力データ

提案手法を用いて比較実験を行った．隣り合う二文字の頻度計算時間を比較した．実験の指標はファイルサイズに対する計算時間とし，比較対象は提案アルゴリズム (proposed)，Simple Algorithm(original) の 2 つである．実験データには English(1024MB) を用いた．

4.2 実験結果

$\frac{1}{\theta} = 100$ としたときの結果を図 7 に示す．メモリ使用量は，提案アルゴリズムが 6.88KB，Simple Algorithm が 3.7KB となった．提案アルゴリズムでは，よりデータ構造が増えているので，メモリ使用量も約 2 倍増えている．計算時間は提案アルゴリズムが Simple Algorithm に比べて 88〜90 % に短縮された．

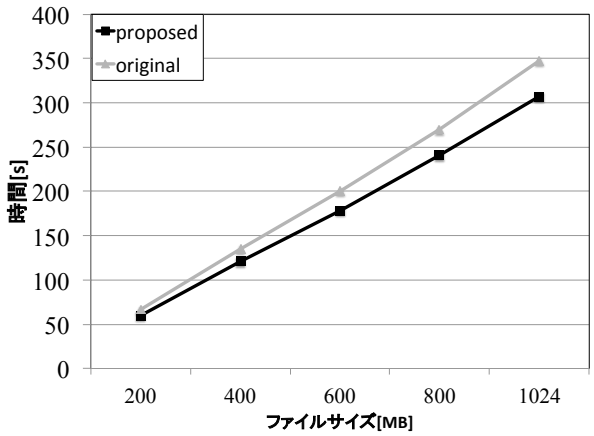

図 7: 実行時間

4.3 考察

今回提案した手法では削除操作を改善しているため，この操作が多く行われるときに有効である．したがって，アイテムの種類数が多い入力データに対して有効であると考えられる．

参考文献

[1] Richard M. Karp, Scott Shenker and Christos H. Papadimitriou. : A Simple Algorithm for Finding Frequent Elements in Streams and Bags. *ACM*

Transactions on Database System(TODS) Vol. 28 No. 1, pp. 51-55 (2003).

[2] Ahmed Metwally, Divykant Agrawal and Amr Abbadi. : Efficient Computation of Frequent and Top-k Elements in Data Streams. *Database Theory - ICDT 2005 Lecture Notes in Computer Science* Vol. 3363, pp. 398-412 (2005).

[3] Masatora Ogata, Yukiko Yamauchi, Shuji Kijima and Masafumi Yamashita. : A Randomized Algorithm for Finding Frequent Elements Streams Using O(loglogN) Space. *ISAAC 2011* LNCS 7074, pp. 514-523 (2011).

[4] Shirou Maruyama, Yasuo Tabei : Fully Online Grammar Compression in Constant Space. *DCC 2014* 173-182.