短 報

大規模文献集合に対して階層的クラスタ分析法を適用するための単連結法アルゴリズム

A Single-Link Method Algorithm for Clustering Large Document Collections

岸 田 和 明
Kazuaki Kishida

Résumé

In the 1960s and 1970s, techniques for clustering a set of documents, in order to improve the effectiveness or efficiency of information retrieval systems, have been widely explored. Similar attempts have recently been made by many researchers to allow the visualisation of search results, to provide browsing based search modes or to enhance performance in searching very large collections. The purpose of this paper is to develop an algorithm for hierarchical clustering that can work for very large document collections. The algorithm is based on a combination of two ideas proposed by other researchers to save time and space in the process of hierarchical clustering; (1) the use of an inverted file for reducing the number of document pairs for which a similarity degree is calculated, and (2) a procedure for constructing a dendrogram based on single-link method from similarity data recorded on disk and not the main memory. In this paper, the algorithm is experimentally applied to a document set consisting of about 10,000 bibliographic records, and the processing time is analyzed empirically. In addition, the effects of removing words frequently appearing in documents are examined. As a result, we find that removing such words enable us to greatly reduce the processing time without significant change in the resulting set of clusters. Finally, an empirical comparison between the single-link method and the single-pass algorithm (leader-follower algorithm) is attempted.

岸田和明: 駿河台大学文化情報学部, 埼玉県飯能市阿須 698
Kazuaki Kishida: Surugadai University, 698 Azu, Hanno, Saitama
e-mail: kishida@surugadai.ac.jp
受付日: 2003年1月20日 受理日: 2003年3月14日

I. はじめに

情報検索の分野では，かつて1960年代後半から70年代にかけて検索性能や作業効率の改善の観点から，文献クラスタリング (document clustering) の応用が研究されていた（例えば，Jardine と van Rijsbergen[1] など)。最近，この種の試みに再び注目が集まりつつある。例えば Scatter/Gather[2] は，文献クラスタリングを応用して，利用者の要求に合わせて文献集合を絞り込んでいくシステムであり，また，WEBSOM[3] では，文献集合の構造が自己組織化マップによって可視化され，利用者に提示される。このほかにも数多くの研究がおこなわれている。

実際，文献クラスタリングの応用としては，

①クラスタリングの結果を検索に直接適用することによって検索性能を向上させる，

②検索結果としての文献集合をグループ化してわかりやすく提示する，

③キーワード検索とは異なった，ブラウジングに基づく検索様式を提供する，

などがあり，これらの目的に応じて，いくつかのクラスタリング技法が利用または提案されている。その中でも，一般的に普及している階層的クラスタ分析法は，文献が階層構造に組織化されるという利点があるものの，計算量が多く，大規模な文献集合に対しては不向きであるとみなされてきた。そのため，単一パス・アルゴリズム (single-pass algorithm) の応用が情報検索分野ではやくから検討されていた（例えば Crouch[4] など)。しかし，階層的クラスタ分析法によって生成される樹形図 (dendrogram) は検索にとって有用であり[5]，また近年のコンピュータの性能向上という状況もあるため，大規模な文献集合に対する階層的クラスタ分析法の適用を再び模索してみることには意義がある。

そこで，本稿は，階層的クラスタ分析法のなかでもアルゴリズムが比較的単純な単連結法 (single-link method) をとりあげ，それを大規模文献集合に適用するためのアルゴリズムを検討・提案する。そして，そのアルゴリズムを『図書館情報学文献目録』の約1万件のデータに適用して，実際の処理時間や結果の妥当性の分析を試みる。

以下，まず第II章では，単連結法アルゴリズムの問題について議論し，それに基づいて，転置索引ファイルを応用したアルゴリズムを提案する。次に，第III章において，そのアルゴリズムを実際のデータに適用し，処理時間の測定を試みる。そして第IV章では，クラスタの結果の妥当性について，平方誤差に基づく評価指標を使って検証する。

II. 単連結法の実行アルゴリズム

A. 単連結法とその計算量

2つの文献 d_i と d_k の類似度を s_{ik} と表記する。文献集合に対して単連結法を適用する場合，2つのクラスタ C_h と C_l との類似度 $\tilde{s}_{hl}$ は，

$$\tilde{s}_{hl}=\max\{s_{ik}|d_i\in C_h, d_k\in C_l\} \quad (1)$$

で定義される。つまり，それぞれのクラスタに含まれる文献間のうち最も近いものの類似度をクラスタ間の類似度として採用する。単連結法は，いわゆる “chain effect” による偏った樹形図を生成してしまうこともあるが，その結果が処理の順序に依存しないといった長所も併せ持っている[6]。

単連結法を実行するには，最初に文献間の類似度を計算しなければならない。そのため，処理する文献数が N である場合，計算時間はこの部分だけで $O(N^2)$ となる。例えば，1万件の文献があれば，文献の組数は $10{,}000\times(10{,}000-1)/2$＝約5,000万にもなる。そして，この約5,000万件のデータから，(1) 式に従ってクラスタの組ごとに最大の類似度を探索しなければならない。もしこのために，類似度データをすべて主記憶装置に置くとすれば，必要な主記憶領域の量はやはり $O(N^2)$ となってしまう。

このように単連結法を大規模な文献集合に適用するのは容易ではない。同様に，その他の階層的クラスタ分析法である完全連結法や群平均法も多量の資源を必要とし，その量は単連結法のそれよりも多い[7]。

B. 転置索引ファイルの利用

文献間の類似度の計算量は$O(N^2)$であるが，通常，2つの文献間の類似度は共有される語の数に基づいて算出されるため，語を共有しない文献間の類似度は自動的に0.0となる。したがって，これを改めて計算する必要はない。語句を共有しているかどうかは転置索引ファイルを調べればわかるので，その結果，類似度の計算量を減らすことが可能となる。

この着想を最初に用いたのはCroft[8]である。Croftは，転置索引ファイル中の各索引語に対応する個々の文献集合に対して，別々にクラスタ分析を実行した。例えば，転置索引ファイルにM個の索引語が登録されていたとすれば，それぞれの索引語を含む文献の集合もまたM個存在する。これらの集合ごとに類似度を計算したとすれば，各集合中の文献間では少なくとも1つの語（その索引語）が必ず共有されているから，類似度が0.0となる無駄な処理はなされないはずである（自動的に回避される）。そこで，文献集合ごとにクラスタ分析を実行し，その結果得られたM個の樹形図を最後に1つにまとめれば，計算量は全体として$O(N^2)$よりも小さくなることが期待できる。すなわち，

$$\sum_{j=1}^{M} O(n_j^2) < O(N^2) \qquad (2)$$

である。ここで，n_jは語t_jを含む文献の数を意味する（$j=1,\ldots,M$）。

しかし，文献1件あたりの索引語数が多く，M回のクラスタ分析において，1つの文献が何度も重複して出てくるような状況では，(2)式はうまく成立しない[9]。この問題に対しては，その後，Willett[10]によって以下のような改良法が考案された。例えば，「文献1」に「索引語1」「索引語2」「索引語3」の3つの語が含まれていたとする。さらに，転置索引ファイルからこれらの索引語のレコードを抽出したところ，次のような情報が得られたと仮定する（つまり，以下は，転置索引ファイルに3回アクセスした結果である）。

	文献1	文献2	文献3	文献4
索引語1	1	1	0	1
索引語2	1	0	0	1
索引語3	1	0	0	0

ここで「1」はその索引語が含まれていることを示し，「0」は含まれていないことを意味している。この情報から，文献1に対しては，文献3との類似度を計算しなくてもよいことが直ちにわかる。

本稿では，このWillettの方法を直接的に適用する。ただし，Willettでは，類似度の計算方法として，2値変数に基づく単純なものが使われているのに対して，ここでは，情報検索分野のベクトル空間モデル[11]に基づいて類似度を定義する。すなわち，まず，文献d_iの主題ベクトル中の重みを，

$$w_{ij} = (\log x_{ij} + 1.0) \log (N/n_j) \qquad (3)$$

とする。ここで，x_{ij}は文献d_iにおける語t_jの出現回数である（n_jとNはこれまでと同様）。そして，別の文献d_kについても，そのベクトル中の重みを(3)式を使って定義し，これらの文献間の類似度を，余弦係数により，

$$s_{ik} = \sum_{j=1}^{M} w_{ij} w_{kj} / \sqrt{\sum_{j=1}^{M} w_{ij}^2 \times \sum_{j=1}^{M} w_{kj}^2} \qquad (4)$$

で計算する。

実際に，転置索引ファイル中に，該当文献数n_jとともに，文献ごとにx_{ij}の値を記録しておけば，上記の文献1の場合，転置索引ファイルへの3回のアクセスだけで，この文献と語を共有するすべての文献に対する類似度(4)式の分子の計算が可能になる。つまり，ここでの方法では，上記の図中の「1」の代わりに，当該文献における索引語の出現回数を記憶しておくわけである。

あとは(4)式の分母を求めれば類似度が計算できる。分母はそれぞれの文献の長さ（主題ベクトルのノルム）を掛け合わせたものであるが，文献1自体の長さは，「1」の代わりに記録しておいた，索引語1〜3の出現回数を使って直ちに計算できる。一方，相手側の文献の長さはこの時点では不明である。したがって，ここでは，分子の値と文献1の長さだけをいったん別々のファイルに書

き出しておく。例えば，

ファイルA：0001　0002　12.3456

ファイルB：0001　23.4567

のようなレコードから成るファイルをそれぞれ作成する。ここで，「0001」「0002」は文献ID，ファイルAの「12.3456」は(4)式の分子の値，ファイルBの「23.4567」は「0001」のノルムの値である。

すべての文献に対してこの処理を施せば，ファイルAには，語を共有する文献のすべての組に対する(4)式の分子の値が格納される。また，ファイルBにはすべての文献のノルムが記録されることになる。したがって，次の処理として，ファイルAを順に読んでいき，必要に応じてファイルBを探索すれば，(4)式を計算できる。最後に，その結果を，

ファイルC：0001　0002　0.1234

のようにファイルに書き出す（「0.1234」は類似度の値）。ファイルBを2分探索木やハッシュ表の形式で主記憶領域に読み込んでおけば，この処理はかなり高速になる。

以上のように転置索引ファイルを利用することによって，類似度0.0の文献間の処理が自動的に回避される。したがって，計算量は必ず減ることになる。実際に計算量がどれだけ減少するかは文献集合の状態・性質に依存する。

C. 樹形図の構成

上記の方法では，中間結果（ファイルAとB）および最終結果（ファイルC）を外部記憶装置に書き出すので，主記憶領域をほとんど必要としない（もちろん，転置索引ファイルやファイルBの探索用の領域等は必要である）。

ファイルCを使って，単連結法を実行する方法は，クラスタ分析の基本的教科書[12]に記載されている。その手順は，大きく分けて，

①ファイルCを類似度の大きさの降順にソート

②ソートされた順にレコードを読み込み，樹形図を構成

の2つから成る。①については標準的なソート・アルゴリズム（ただし大規模データに対してのアルゴリズム）を使えばよい。

②については，例えば，ソート済みのファイルCが次のようになっていたとする。

0001　0002　0.9876　...(a)

0003　0004　0.8765　...(b)

0001　0003　0.7654　...(c)

0002　0003　0.6543　...(d)

このデータを上から順に読んでいき，(a)〜(d)の各レコードに対して，それぞれ次のように処理すれば樹形図を構成できる。

(a) 0001と0002を併合し，クラスタ1とする。

(b) 0003と0004を併合し，クラスタ2とする。

(c) 0001はクラスタ1，0003はクラスタ2に属しており，所属クラスタが異なっているので，それらを併合しクラスタ3とする。

(d) 0002はクラスタ1が合併されたクラスタ3に属しており，0003もまた同じクラスタ3に属しているので，何もしない。

この手順を繰り返していけば，最終的に，単連結法による樹形図を得ることができる[12]。この実行で問題となるのは，それぞれの文献が処理される時点でどのクラスタに属しているかという情報の探索である。この部分さえ，うまく実装すれば，問題なく上記の手順で樹形図を構成できる。

III. 処理時間に関する実験

A. 実験データ

以上述べた単連結法の実行アルゴリズムの効率を実際に調べるために，日本図書館情報学会文献目録委員会により作成された『図書館情報学文献目録』CD-ROM中の10,848件のデータを使って，実験を試みる。

この文献目録には抄録は含まれておらず，論文や記事の標題のみが収録されている。ここでは，標題から語を切り出し，それに基づいて(4)式の類似度を計算する。

語を切り出す方法としては，辞書による最長一致を基本とし，未知語・複合語に関する若干の発

見法的な規則を加えたものとした。辞書は『茶筌』[13]のものを使用し，そこに登録されていない未知語（未知の文字列）は字種の切れ目で分割した（なお，ひらがなのみから成る文字列は索引語として使用しない)。また，隣接する索引語を自動的に組み合わせて，複合語を構成した。ただし，2つの索引語の間にひらがなのみから成る文字列が挟まる場合には，複合語をつくらないことにした。例えば，ある文献の標題が「情報検索技術の実際」であり，辞書に「検索」のみ登録されているとすれば，この標題からは，「情報」「検索」「技術」「実際」「情報検索」「検索技術」の6つの索引語が切り出されることになる。

B. 実験環境

市販の標準的なパーソナルコンピュータ（CPU: 1.50 GHz, 主記憶: 256 MB, ハードディスク: 80 GB）を用い，OSはMS Windows XP, 言語はJavaを使った（C++によるDLLを作成して一部使用した)。この仕様からわかるように，今回の実験では，現時点で容易に入手可能な程度の性能しか持たないコンピュータを使用している。

C. 実験方法

全体のプログラムを，

①類似度の計算，

②ソート（上記「ファイルC」のソート)，

③樹形図の構成，

の3つに分け，それぞれの処理に要した経過時間（以下，単に「処理時間」とする）を計測する。なお，ソートについては，対象データが主記憶領域に収まりきらないので，プログラミングの教科書[14]を参考に，外部記憶装置を補助的に用いるプログラムを自作して使用した（ここでの実験状況では最大約30億レコードのソートが可能である)。

クラスタリングの対象文献数は10,848件であるが，文献数の影響を見るために，処理する文献レコードの数を5,000, 3,000, 1,000, 500件のように段階的に設定してクラスタリングを試みた。例えば5,000件の場合，ファイルの先頭からの5,000件と，5,001番目のレコードからの5,000件をそれぞれ調べれば排他的な2つの標本による結果が得られる。同様に，3,000件の場合には3つの標本をとることができる（開始位置はそれぞれ1, 3,001, 6,001とした)。1,000件の場合には5つの標本とし（開始位置は1, 2,001, 4,001, 6,001, 8,001)，500件の場合には10個の標本とした（開始位置は1, 1,001, 2,001, . . . , 9,001)。この方法では，データに含まれるすべての情報を完全に使ったことにはならないが，今回の実験の目的からは，それぞれの標本平均を調べれば十分であろう。

D. 実験結果

1. 処理時間の分析

実験結果を第1表(a)に示す。索引作成の結果，有効な語を持たないレコードを除いたため，表中の「実際のレコード件数」はそれぞれわずかに減少している。また，上で述べたように，「10,848件」の場合を除いて，表中の数値はすべて標本平均を意味している。

まず，表中で「率」と表記されているのは，文献のすべての組のうち，実際に語を共有していた組の割合である。例えば，「10,848件」の場合，実際の処理レコード件数は「10,826件」であり，単純にすべての組の類似度を求めてしまうと，おおよそ10,826の2乗分の計算が必要になるが，転置索引ファイルを利用することによって，約18.7%の組に対する処理で済んでいる。表が示すように，全体的には，この割合はおおよそ20%弱であり，今回の実験状況では，転置索引ファイルを利用することによって，約8割の余分な計算が回避されることがわかる。

次に，実際の処理時間を見ると，処理レコード件数が少ない場合には，類似度の計算に多くの時間が費やされているのに対して，処理件数が増えるに従って，次第にソートや樹形図の構成の割合が増加している。このことは，第1図からも明らかである。第1図は横軸に処理レコード件数をとり，縦軸にアルゴリズムの各部分および全体の処

第1表　単連結法アルゴリズムの実験結果

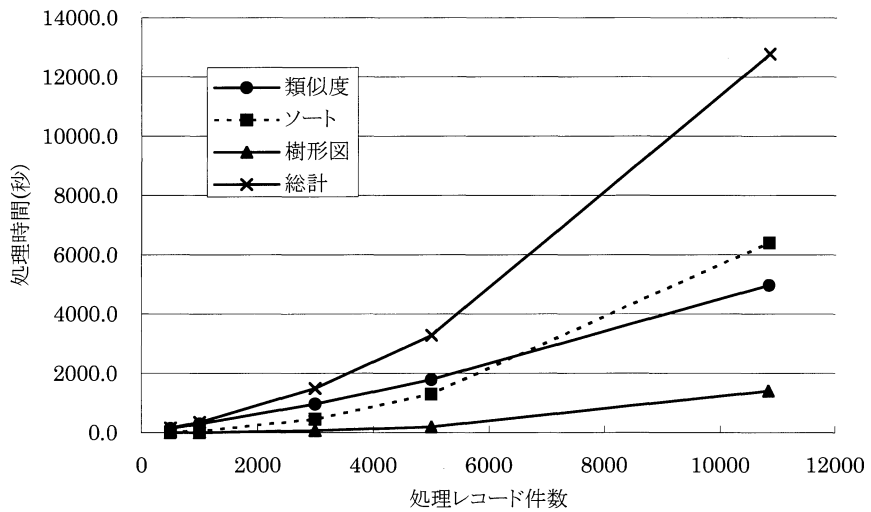
(a)　すべての語を使用した場合（標本平均）

レコード件数	標本数	実際のレコード件数	率*	処理時間（秒）			
				類似度	ソート	樹形図	合計
10,848	1	10,826	18.7	4,969 (38.9%)	6,408 (50.2%)	1,399 (11.0%)	12,776 (100.0%)
5,000	2	4,990.0	18.0	1,781.0 (54.4%)	1,293.7 (39.5%)	199.4 (6.1%)	3,274.2 (100.0%)
3,000	3	2,994.0	18.1	971.0 (65.0%)	460.8 (30.8%)	62.4 (4.2%)	1,494.2 (100.0%)
1,000	5	997.6	19.5	301.7 (84.4%)	49.0 (13.7%)	6.8 (1.9%)	357.5 (100.0%)
500	10	499.1	17.9	139.1 (91.5%)	11.3 (7.4%)	1.6 (1.0%)	152.0 (100.0%)

(b)　出現文献数の多い「図書館」「情報」を除去した場合（標本平均）

レコード件数	標本数	実際のレコード件数	率*	処理時間（秒）			
				類似度	ソート	樹形図	合計
10,848	1	10,808	5.2	1,999 (48.1%)	1,809 (43.5%)	351 (8.4%)	4,159 (100.0%)
5,000	2	4,984.5	5.5	787.6 (62.7%)	408.2 (32.5%)	59.7 (4.8%)	1,255.5 (100.0%)
3,000	3	2,991.0	5.6	436.7 (74.3%)	131.6 (22.4%)	19.1 (3.2%)	587.3 (100.0%)
1,000	5	996.6	6.5	142.2 (88.4%)	16.3 (10.1%)	2.4 (1.5%)	160.9 (100.0%)
500	10	498.7	6.6	68.9 (93.4%)	4.2 (5.7%)	0.7 (0.9%)	73.8 (100.0%)

注*：文献のすべての組数に対する，類似度 0.0 以外の組数の割合 (%)



第1図　単連結法アルゴリズムの処理時間

理時間を表示したものであるが，これを見れば，ソートや樹形図の構成の処理時間の増加は，類似度の計算のそれに比べて大きいことがわかる。ソートや樹形図の構成のための効率的なアルゴリズムの利用あるいは開発が今後必要となることがこの結果から示唆される。

2. 最頻出語の削除による処理時間の減少

各語の出現文献数を調べたところ,「図書館」と「情報」の2語が10,848件の10%以上に出現していた。具体的には,それぞれ3,935件,1,382件である。情報検索の理論に照らせば,これらの語の識別力は低く,実際,本稿で採用している(3)式でも,出現文献数の逆数(すなわちidf)が含まれているため,このような頻出語を無視しても結果に大きな変化は生じないと考えられる。しかも,「図書館」や「情報」の出現文献数の多さから考えて,これらのみを共有する文献の組がかなり存在することが予想されるので,「図書館」や「情報」の削除によって,処理時間の大幅な減少が期待できる。

その結果を第1表(b)に示す。確かに,処理時間はかなり減少しており,例えば,処理件数10,848件の場合,削除前では12,776秒であったのに対して,削除後は,約1/3の4,159秒で処理が終わっている。この減少幅についても,処理レコード件数に依存し,件数が少ない場合(500件や1,000件など)では,およそ1/2の短縮に過ぎないが,データを見る限り,逆に,処理件数の増加に従って,頻出語の削除はより大きな効果を持つことが予想される。

IV. クラスタリングの妥当性の検証

A. 最頻出語削除の影響の検証

1. 実験の目的と妥当性の指標

前章ではidfが(3)式に含まれることを根拠に,「図書館」「情報」という2つの最頻出語を削除してクラスタリングを実行し,処理時間を計測した。しかし,実際に,この処置がクラスタリングの結果にどの程度の影響を与えるかを確認しておく必要はあるだろう。理論的にはidfによって,これらの語の効果は縮小されているはずではあるが,現実的に頻出語の削除の影響が大きいようであれば,たとえ処理時間の短縮が果たされようとも,この方法は実用的とはいえない。

一般に,本稿が対象とするような大規模な樹形図の場合,それらを直接的に比較して,その変化を把握することは難しい。ひとつの方法として,クラスタリングの結果の妥当性を検証する指標として用いられている,平方誤差の総和[6),15)]を利用することが考えられる。つまり,頻出語の削除前と後のそれぞれの樹形図に対して平方誤差の総和を計算し,その値の差が小さければ,頻出語の削除は大きな影響を与えないと判断すればよい。

実際には,本稿では,文献間の類似度の測定にはユークリッド距離ではなく,余弦係数を用いているから,通常の平方誤差の総和に相当する指標は,

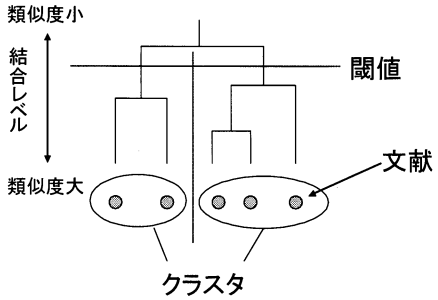
$$J_e = \frac{1}{2}\sum_{h=1}^{L}\left(\frac{1}{\tilde{n}_h}\sum_{d_i,d_k\in C_h;\, i\neq k} s_{ik}\right) \qquad (5)$$

となる(付録参照)。ここで,Lはクラスタの個数,$\tilde{n}_h$はh番目のクラスタに含まれる文献数である。また,( )内の総和は1つのクラスタに属するすべての文献の組に対して類似度を合計することを意味する。ただし,自分自身との類似度s_{ii}は除かれる。

(5)式の値は大きいほどよい。(5)式中のs_{ik}は2つの文献間の類似度なので,クラスタリングの結果,高い類似度を持つ文献が1つのクラスタとしてうまくまとまるほど(さらに,類似度が低い2つの文献が異なるクラスタにうまく分離するほど),(5)式のJ_eの値は大きくなる。したがって,(5)式によって計測される「妥当性」は,いわゆる「教師つきの(supervised)」自動分類における妥当性とは異なり,現実世界に照らしての分類結果の妥当性を意味しない。つまり,本稿で対象とする階層的クラスタ分析は「教師なしの(unsupervised)」分類であって,(5)式は,元の類似度行列を出発点として,それらの類似度に従ってどれだけうまく文献がクラスタにまとめられたかを測っているのにすぎない。ただし,頻出語削除の影響の検証というここでの目的からは,(5)式は充分な指標であると考えられる。

2. 実験の方法

(5)式を使うには,階層的な樹形図から,「平面的なクラスタ」を切り出さなければならない。このためには,閾値を定めて,それ以上の結合レベルを持つ「枝」を1つのクラスタと見なせばよい



第2図　樹形図からのクラスタの構成

(第2図参照)。ここではこの閾値を，0.1, 0.2, 0.3, 0.4, 0.5, 0.6 に設定する。処理対象文献数は 500 に固定し，標本は第1表と同じものを使用する。したがって標本数は 10 である。

3. 実験の結果

実験の結果を第2表に示す。第2表には，樹形図から切り出されたクラスタの個数 (L) と，(5) 式の指標の値 J_e を示してある。ただし，いずれも，大きさ 10 の標本平均の値である。また，表中の「単連結法1」は通常のクラスタリング，「単連結法2」は「図書館」「情報」の2つの頻出語を削除した結果を意味する。

例えば，「単連結法1」の「閾値 0.1」では，クラスタ個数 71.2，評価指標は 3.5 である。そして，

第2表　クラスタリングの妥当性検証の結果

閾値	単連結法1		単連結法2		単一パス	
	L	J_e	L	J_e	L	J_e
0.1	71.2	3.5	73.8	3.6	130.9	29.2
0.2	235.5	15.4	238.6	15.6	222.4	35.4
0.3	336.1	24.8	336.5	24.9	293.1	35.6
0.4	393.2	26.3	393.4	26.3	342.7	33.6
0.5	427.2	23.5	427.3	23.5	375.3	30.9
0.6	449.3	19.3	449.1	19.3	401.5	27.5

注：「単連結法1」はすべての語を使用，「単連結法2」は「図書館」「情報」の2語を削除。また，L はクラスタ数，J_e は本文 (5) 式の値であり，いずれも標本平均である。

閾値を上げていくと，クラスタ個数は単調に増加する。この個数は「閾値 0.6」においては 449.3 であり，全体の文献数は 500 であるから，このレベルでは，多くの文献が「singleton」として単独で存在していることになる。一方，評価指標は閾値を増やしていく途中でピークを持つことを第2表は示している（このピークは閾値 0.4 の近くにある)。

これらのクラスタ個数および評価指標について，「単連結法1」と「単連結法2」との間にほとんど相違がないのは第2表から明らかである。このことは，「図書館」「情報」の2語を削除しても，クラスタリングの結果はほぼ変わらないことを示している。すでに述べているように，(3) 式に idf が含まれていることから，この結果は容易に予想することができるが，(5) 式を使った実験によって，改めて実証的に確認されたといえる。この結果から，最頻出語の削除は，処理の実行時間を短縮するための便利かつ妥当な方法であると結論できる[16]。

B. 単一パス・アルゴリズムとの比較

すでに議論しているように，(5) 式を使えばクラスタリングの妥当性の検証が可能である。そこで，本稿では最後に，これを利用して，単連結法による階層的クラスタ分析と単一パス・アルゴリズムとの比較を試みる。

単一パス・アルゴリズムは，少ない計算量でクラスタリングを可能とするために研究・開発されてきた方法で，情報検索の分野でも，その応用が 1960 年代から 70 年代にかけて検討されてきた[17]。その過程においてさまざまな方法が提案されたが，ここでは最も基本的な leader-follower アルゴリズムをとりあげる。アルゴリズムの概要は以下のとおりである[15]。

①文献をクラスタに併合するかどうかを決めるための閾値を設定する。
②ファイルから文献を1件読む。もしすべての文献が読み終われば，処理を終了する。
③読み出した文献と，その時点で存在するすべ

てのクラスタとの類似度を計算する。

④類似度が最も大きなクラスタにその文献を併合し，クラスタ中の語の重みの値を更新する。ただし，その最大の類似度が閾値を超えなければ，どのクラスタにも併合せず，その文献を1つの単独のクラスタとして独立させる。

⑤手順②に戻る。

このアルゴリズムはk-means法とよく似ているが，k-means法のように，クラスタの個数は先験的に固定せず，またしたがって種子(seeds)も使わない。

本稿では，ここでの単連結法における(3)式との整合性を考え，leader-followerアルゴリズムにおける，クラスタ C_h 中の語 t_j の重みを

$$\tilde{w}_{hj}=\log \sum_{d_i \in C_h} x_{ij}+1.0 \qquad (6)$$

で定義する。(6)式が示すように，ここではクラスタを，それに含まれる文献を単純に併合した「巨大な」文献として捉えることになる。

一方，文献中における語の重みは(3)式，クラスタとの類似度は余弦係数(4)式をそのまま使用する（w_{kj} を $\tilde{w}_{kj}$ で置き換える）。したがって，上記手順③における語の重みの更新では，(6)式により，新たに併合される文献における語の出現回数 x_{ij} を，クラスタのそれに単純加算することになる。

以上の単一パス・アルゴリズム（leader-followerアルゴリズム）を使ってクラスタリングを実行し，(5)式の評価指標の値を求めた結果は，比較のため，すでに第2表中に示してある（第2表の右側の縦列参照）。当然のことながら，単連結法と同じデータを使っており，第2表の「単一パス」の見出しの下に示された数値はやはり標本平均である。ただし，この場合の閾値は第2図に示されたものではなく，leader-followerアルゴリズムにおいて，文献をクラスタに併合する場合の境界値である（上記手順①および④参照）。

このように閾値の意味合いが異なるので，単連結法との直接的な比較は難しいが，第2表を見る限り，単一パス・アルゴリズムに分があるのは明らかであろう。すべての閾値にわたって単一パス・アルゴリズムの J_e の値が上回っており，単連結法の閾値の設定をいくら工夫してみても，単一パスの値を超えられそうにない。ここではあくまで(5)式のみに基づく限定的な評価ではあるが，単連結法の妥当性にはやや問題があることが明らかになった。

この結果から単連結法の代わりに単一パス・アルゴリズムを使えばよいようにも思われるが，話はそれほど単純ではない。なぜなら，ここで使用したleader-followerアルゴリズムを実行するには，単連結法と同様に莫大な資源が必要になるためである。k-means法のようにクラスタの個数を固定して，しかもその数が少なければ，この方法は大規模な文献集合に比較的容易に適用できる[18]。しかし，ここで用いているleader-followerアルゴリズムでは，クラスタの個数は可変であり，しかも，第2表が示すように，その数はかなり多い。これは，(6)式で計算されるクラスタ中の語の重みの値を記録しておくのに，莫大な記憶容量が要求されることを意味している。当然，クラスタ個数が多くなれば，主記憶装置上にデータが収まらなくなる。実際，本稿のために開発したleader-followerアルゴリズムのプログラムでは，主記憶領域が枯渇するため，1万件のデータは処理できない。この問題を解決するには，外部記憶装置との併用が必要であるが，B木などの高速探索アルゴリズムを使ったとしても，アクセス回数が多いため，実用的な実行速度を得るのは難しいかもしれない。

また，本来的に，単一パス・アルゴリズムは文献を階層的に構成できず，この点，情報検索への応用からみれば，単連結法に劣る。このような事情から，大規模文献集合に対する検索への適用という観点からは，単純に単一パス・アルゴリズムに乗り換えることはできない。むしろ，単連結法の欠点を補うため，大規模文献集合に適用可能な完全連結法や群平均法のアルゴリズムの開発を追究していくべきであろう。

V. おわりに

本稿では大規模文献集合に適用可能な単連結法アルゴリズムを開発し，実際に，約1万件の文献集合に適用した。その結果，この規模のデータに対しては合理的な時間内で処理が可能ではあるものの，さらにレコード件数が増えた場合には，ソートや樹形図の構成の部分で，アルゴリズムの改良が必要なことがわかった。

また，多くの文献に出現する頻出語を削除すると，処理時間は大幅に短縮されるにも関わらず，クラスタリングの結果には大きな影響が生じないことが明らかになった。この方法は，機械学習を応用したテキスト分類の研究などですでに採用されているが，階層的クラスタ分析の場合にも有用であることが改めて実証された。

本稿では最後に，単連結法によるクラスタリングの結果の妥当性を，単一パス・アルゴリズムのそれと比較した。その結果，本稿で採用した評価指標の範囲内では単連結法は相対的に妥当性に乏しいことが明らかとなった。このことは，大規模文献集合に適用可能な完全連結法や群平均法に対するアルゴリズムの開発に進むべきことを示唆している。

データの規模に関して，「1万件」という大きさ自体は1980年代にすでにVoorhees[11]による計算例があるので特に目新しいものとはいえない。最近でも，topic detectionの研究において大規模なデータに対する階層的クラスタ分析が適用されており，この場合の規模も1万件を超える[19]。

しかしながら，「1万件」という数字自体には目新しさはないとはいえ，本稿の方法は外部記憶装置を有効利用しており，この点で，さらに大きな文献集合への適用可能性を残している。コンピュータのハードウェアの進歩はこれからも続き，さらに優れたCPUや大規模な主記憶装置が登場するであろう。これに伴って，当然，処理可能なレコード件数は増加していく。何年かの後には，市販の標準的なコンピュータの主記憶領域だけで，1万件を超えるデータに対して単連結法を適用できるかもしれない。しかし，その場合でも，本稿の外部記憶装置併用型のアルゴリズムは常に，主記憶領域のみを使う方法よりも，大きなデータを扱える可能性を持っている。この点に本研究の価値がある。

しかし，実際には，実用規模のデータベースを扱うにはまだ工夫の余地がある。例えば，100万件の規模の全文データベースを扱うには，本稿で使用したものよりも高い性能を持つコンピュータを利用するとともに，アルゴリズム上の工夫をいくつか加える必要がある。例えば，

(1) ごく少数の語しか共有しない文献間の類似度は0.0に近いと想定して，計算からはずす（このことは転置索引ファイルの確認時にわかる），

(2) 第一段階として，適当にクラスタ個数を設定してk-means法によりデータベースを分割しておき，個々のクラスタについてさらに階層的クラスタ分析法を適用する，

ことなどは容易に思いつく。

本稿の冒頭で述べたように，特にWWWの普及後，情報検索において，以前よりも大規模なデータを扱う必要が生じた。このような状況においては，データを何らかの局所的性質を持つ部分に分割することが，さまざまな点で便利である。その分割が，情報検索の性能向上に利用されるのか，あるいは文献提示などのインタフェース部分で使用されるのかは，それぞれの状況に依存するが，このための基本的技術として，より有用なクラスタリングのアルゴリズムや手法を探究していくことは重要であろう。

謝　辞

『図書館情報学文献目録』データの研究目的での使用を認めていただいた，日本図書館情報学会文献目録委員会に感謝いたします。

注・引用文献

1) Jardine, N.; van Rijsbergen, C. J. "Document clustering: an evaluation of some experiments with the Cranfield 1400 collection". Information Processing & Management. Vol. 17, No. 5/7, p. 171–182 (1975)

2) Cutting, D. R., et al. "Scatter/gather: a cluster-based approach to browsing large document collections". Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 318–329 (1992)
3) Kohonen, T., et al. "Self organization of a massive document colletion". IEEE Transactions on Neural Networks. Vol. 11, p. 574–585 (2000)
4) Crouch, D. B. "A file organization and maintenance procedure for dynamic document collections". Information Processing & Management. Vol. 11, No. 1/2, p. 11–21 (1975)
5) 例えば，階層構造の上下を適当に動くことによって，出力文献数に関する利用者の要求に応じることができる。さらには，これをうまく利用すれば，精度志向や再現率志向といった，意図の異なる検索が実現できる。詳しくは文献 1) などを参照。
6) 宮本定明. クラスター分析入門: ファジィクラスタリングの理論と応用. 森北出版, 1999. p. 140–145.
7) Voorhees, E. M. "Implementing agglomerative hierarchic clustering algorithms for use in document retrieval". Information Processing & Management. Vol. 22, No. 6, p. 465–476 (1986)
8) Croft, W. B. "Clustering large files of documents using the single-link method". Journal of the American Society for Information Science. Vol. 28, No. 6, p. 341–344 (1977)
9) Harding, A. F.; Willett, P. "Indexing exhaustivity and the computation of similarity measures". Journal of the American Society for Information Science. Vol. 31, No. 4, p. 341–344 (1977)
10) Willett, P. "A fast procedure for the calculation of similarity coefficients in automatic classification". Information Processing & Management. Vol. 17, p. 53–60 (1981)
11) Buckley, C.; Allan, J.; Salton, G. "Automatic routing and ad-hoc retrieval using SMART: TREC2," The Second Text Retrieval Conference (TREC2). D. K. Harman ed. National Institute of Standards and Technology, Gaithersberg, MD, 1994, p. 45–55.
12) Anderberg, M. R. 西田英郎ほか訳. クラスター分析とその応用. 内田老鶴圃, 1983. p. 191–193.
13) 松本裕治，北内啓，山下達雄，平野善隆，松田寛，高岡一馬，浅原正幸. 日本語形態素解析システム『茶筌』version 2.2.1 使用説明書. 2000. URL: http://chasen.aist-nara.ac.jp/chasen/bib.html.ja
14) Sedgewick, R. Algorithms in C++: Parts 1–4. 3rd ed. Addison-Wesley, 1998. p. 466–472.
15) Duda, R., et al. Pattern Classification. 2nd ed. John Wiley & Sons. 2001. 654 p.
16) もちろん，idf を含めて計算された類似度が，実際の文献間の類似性を測る尺度として妥当であるかは別問題である。これに関しては，本稿は，情報検索研究におけるベクトル空間モデルのこれまでの研究成果に依拠して，本文 (3) および (4) 式を使用しているのにすぎない。これらの類似度自体の妥当性の問題はもちろん探究されるべきであろうが，本稿の範囲外である。
17) 岸田和明. 情報検索の理論と技術. 勁草書房, 1998. p. 139–140.
18) 文献データベースへの k-means 法の適用例として次の文献がある。Xu, Jinxi; Croft, W. Bruce. Topic-based language models for distributed retrieval. Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval. W. Bruce Croft ed. Kluwer Academic Publishers, 2000. p. 151–172.
19) Franz, M.; McCarley, J. S.; Ward T., Zhu, W. J. Unsupervised and supervised clustering for topic tracking. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 310–317 (2001)

付録: クラスタの妥当性を測定する指標について

本稿で使用した評価指標 (5) 式は宮本[6] の p. 71 に掲載されている式に本質的に等しいが，少なくとも図書館・情報学分野では，使用例がほとんどないため，若干，補足説明を加えておく。

クラスタリングの各対象が m 次元のベクトルで表現されるとして，その 1 つを $\mathbf{x}_i$ と表記する。また，ここでは表記を簡単にするために，クラスタ C_h $(h=1, 2, \ldots, L)$ を，それに属する $\mathbf{x}_i$ の添字の集合と規定しなおす。この結果，その重心ベクトルは

$$\mathbf{m}_h = \frac{1}{\tilde{n}_h} \sum_{i \in C_h} \mathbf{x}_i \qquad \text{(A.1)}$$

となる。ここで，$\tilde{n}_h$ はクラスタ C_h に含まれる対象の個数とする。一般にクラスタリングの結果についての平方誤差は，

$$J_e = \sum_{h=1}^{L} \sum_{i \in C_h} \|\mathbf{x}_i - \mathbf{m}_h\|^2 \qquad \text{(A.2)}$$

で定義される[15]。ここで，$\|\cdot\|$ はノルムである。(A.2) で定義される値が小さいほど，全体的に，各クラスタがそれぞれの重心の周りに密集していることになる。後で示すようにこの式は，

$$J_e=\sum_{h=1}^{L}\left(\frac{1}{\tilde{n}_h}\sum_{i,k\in C_h;\,i<k}\|\mathbf{x}_i-\mathbf{x}_k\|^2\right) \qquad (A.3)$$

に変形される。ここで，$\|\mathbf{x}_i-\mathbf{x}_k\|^2$ は 2 つの対象間の「距離」を意味する。この部分を一般化して，この「距離」の代わりに類似度 s_{ik} を使うことができる。この結果，本文 (5) 式を得る。ただし，距離を類似度に置き換えたので，この場合には，J_e が大きいほど，クラスタは密集していることになる。

(A.2) から (A.3) が導かれることを示す。(A.1) 式を使えば，各クラスタにおける平方誤差は，

$$\sum_{i\in C_h}\|\mathbf{x}_i-\mathbf{m}_h\|^2=\sum_{i\in C_h}\|\mathbf{x}_i\|^2-\frac{1}{\tilde{n}_h}\left\|\sum_{i\in C_h}\mathbf{x}_i\right\|^2$$

と書ける。ここで，

$$\left\|\sum_{i\in C_h}\mathbf{x}_i\right\|^2=\sum_{i\in C_h}\|\mathbf{x}_i\|^2+2\sum_{i,k\in C_h;\,i<k}\langle\mathbf{x}_i,\mathbf{x}_k\rangle$$

に注意すると（$\langle\cdot,\cdot\rangle$ は内積を示す），結局，

$$\sum_{i\in C_h}\|\mathbf{x}_i-\mathbf{m}_h\|^2$$

$$=\frac{1}{\tilde{n}_h}\left[(\tilde{n}_h-1)\sum_{i\in C_h}\|\mathbf{x}_i\|^2-2\sum_{i,k\in C_h;\,i<k}\langle\mathbf{x}_i,\mathbf{x}_k\rangle\right] \qquad (A.4)$$

となる。

一方，(A.3) 式中の（　）中の総和部分だけ取り出して変形してみると，

$$\sum_{i,k\in C_h;\,i<k}\|\mathbf{x}_i-\mathbf{x}_k\|^2$$

$$=\sum_{i,k\in C_h;\,i<k}(\|\mathbf{x}_i\|^2-2\langle\mathbf{x}_i,\mathbf{x}_k\rangle+\|\mathbf{x}_k\|^2) \qquad (A.5)$$

であるが，この右辺の総和の中に各 $\|\mathbf{x}_i\|^2$ ($i\in C_h$) が何回出てくるかを数えてみると，それぞれ $\tilde{n}_h-1$ 回であるから，(A.5) 式は，

$$\sum_{i,k\in C_h;\,i<k}\|\mathbf{x}_i-\mathbf{x}_k\|^2$$

$$=(\tilde{n}_h-1)\sum_{i\in C_h}\|\mathbf{x}_i\|^2-2\sum_{i,k\in C_h;\,i<k}\langle\mathbf{x}_i,\mathbf{x}_k\rangle \qquad (A.6)$$

となる。このことから，(A.4) 式を (A.2) 式に代入すれば，(A.6) 式を経由して，(A.3) 式が得られることがわかる。

Duda ら[15]が注意しているように，本文 (5) 式の J_e は万能の尺度ではなく，クラスタに含まれる対象の数（本稿では文献数）に大きな差があると，(5) 式は不適切となる可能性がある（Duda ら[15]の p. 543 によい例が掲載されている）。