検索エンジンを利用した
単語クラスタリング

執筆者：大城 亜里沙

指導教官：新納 浩幸

平成16年3月2日

目 次

図目次

表 目 次

第1章　はじめに

1.1　概要

コーパスを利用した統計的な自然言語処理はある程度の成功をおさめたが，いくつかの課題も残されている．その1つはコーパスのスパース性である．新聞記事10年分を調べても，日常的な単語「おにぎり」は数回しか出現しないという報告もある．一方，近年，Webが急速に普及し，Web上の情報は莫大な量である．そこでWebを巨大なコーパスとみなし．従来行われていた統計的な自然言語処理をWebを利用して行うことが期待できる．Webは情報が莫大であるために，コーパスのスパース性は回避できると考えられる．

Webのページは新聞記事のように単純なテキストではないので，単にページを収集しただけではコーパスとして利用することは難しい[3]．そこで本論文では検索エンジンを利用する．検索エンジンを利用した場合，ある単語群をクエリとして用いれば，その単語群を含むページ数を得ることができる．単語が他の単語とどの程度共起するかを表す共起分布を考えると，類似の単語同士の共起分布は類似しているという性質がある．ここで共起の定義として単語がどの程度まで離れていてもよいかは様々である．通常は文や周辺数単語内ということが多い．ここでは同じページ内に出現していれば共起し

ていると考える．すると，単語Aと単語Bが共起している頻度は検索エンジンに“A B”というクエリを渡すことで得ることができる．基本的に任意の2単語の共起の頻度を得ることができれば，単語を特徴ベクトルとして表現できるので，単語クラスタリングは実現できる．

本実験では5つのクラスを想定し，それぞれのクラスから5単語取り出し，合計25単語についてクラスタリングを行った．比較のために新聞記事1年間分のコーパスを利用したクラスタリングも行った．新聞記事を利用した場合，クラスタリングの結果は良好であったが，コーパスのスパース性の問題が確認できた．Webを利用したクラスタリングは，新聞記事を利用したものほどうまくはいかなかった．相関比が最大になるように基底の単語を設定し直すことで改良を試みたが，この場合も理想的なクラスタリング結果とは若干異っていた．うまくクラスタリングできない原因を考察する．

1.2 本論文の構成

第2章では単語のクラスタリングについて述べる．ここでは単語のベクトル空間モデルの作成について説明し，クラスタリング手法についても説明をする．クラスタリング手法は大きく，最短距離法などの階層的手法と，k-means法などの分割的手法に分けられるが，これらの基本的手法について説明する．各手法の特徴や傾向を無視すると，不適当な結果が導かれてしまうときがある．そこで，このような問題を回避するための，主な注意点をまとめ，クラスタリング

結果の解釈のセクションでそれについて述べる．

第3章では，R言語についての説明をする．k-means法のプログラムはRで作成した．

第4章では，階層的手法で代表的なWard法と分割的手法で代表的なk-means法の2つの手法を用いてクラスタリングを行う．まず，コーパスを利用した単語クラスタリングを行い，次にWebを利用した単語クラスタリングを行う．Webを利用したクラスタリング結果はコーパスを利用したクラスタリング結果よりうまくはいかなかった．そこで，Webを利用したクラスタリング結果の改良を試みるために最適な次元を求め，最適な次元でのクラスタリングも行った．

第2章　単語クラスタリング

2.1　単語のベクトル空間モデル

これは，テキストデータを知識源として単語を多次元空間中に配置し，空間中の2つの単語間の近さに基づいて単語化の類似度を計算するモデルである．テキストデータの種類や利用目的に応じて様々な技術が提案されているが，それぞれに共通する大まかな手順は，次の通りである．

(1) テキストデータから単語の特徴の重みを要素とするベクトル(「特徴ベクトル」と呼ぶ)を作成

(2) 特徴ベクトル中の特徴を互いに関連性の少ない特徴に変換した属性ベクトルを作成

(3) 2つの単語に対する属性ベクトルを用いて単語間の類似度を計算

本実験では，基底の単語として，95年度版毎日新聞1年間分の記事中の単語をクラスタリングすることから得た922単語を用いる．これは，論文[1]で抽出された922クラスの単語クラスタの各クラスタから代表的な単語を選んだ結果，選出された922単語である．この922単語を基底の単語と用いるため，手順2，3は行ってはいないが，

以下に，1から3までの手順および，それぞれにおける既存方式の特徴を説明する．

(1) 特徴ベクトルの作成

テキストデータから注目する単語に対応する表層的な特徴を抽出し，個々の特徴の重みを表す数値を要素とした単語の特徴ベクトルを作成する．

クラスタリングの対象であるn語を$w_1, \cdots, w_n$で表す．単語を表現する特徴数をmとし，単語w_iの特徴ベクトルw_iを下記のように表す．

$$w_i = (v_{i1}, \cdots, v_{im}) \tag{2.1}$$

$v_{ij}(j = 1, \cdots, m)$は，j番目の特徴と単語w_iの関係の強さを表す数値で，特徴の重みと呼ぶ．

$$G_o = \begin{pmatrix} w_1 \\ \cdots \\ w_i \\ \cdots \\ w_n \end{pmatrix} = \begin{pmatrix} v_{11} & \cdots & v_{ij} & \cdots & v_{im} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ v_{i1} & \cdots & v_{ij} & \cdots & v_{im} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ v_{n1} & \cdots & v_{nj} & \cdots & v_{nm} \end{pmatrix} \tag{2.2}$$

テキストより抽出する単語の特徴としては，新聞記事などのテキストコーパスを用いる場合には，主語や目的語に対する述語や名詞

と連接する名詞，注目する単語の近傍に存在する単語が提案されている．また，単語の類似性を考慮した情報検索技術では，検索対象の文書集合中で注目する単語を含む文書IDを特徴としている．これら特徴の出現傾向を数値化した重みを要素として特徴ベクトルが生成される．

一方，国語事典を用いた技術では，語義文中の単語を見出し語の特徴として用いている．全ての語義文が同じ詳細さで語の説明を記述していれば，語義文中の特徴となる単語の出現頻度のみから特徴の重みを計算すれば十分である．しかし電子化辞書の語義文は，人間の語の理解のために作成されているため，1語のみの簡潔な記述しかない語義文や，形式名詞や辞書に固有な言い回しなどの定義に直接関わらない単語が含まれる語義文が存在している．そのため，ニューラルネットワークの考えを用いたアプローチや，人間の辞書を利用する際のヒューリスティクスを適用するアプローチのように，見出し語と別の見出し語の語義などの間接的な関連性も考慮して特徴の重みを適切に表現する必要がある．

ここでは検索エンジンを利用して特徴の重みを設定する．単語の特徴としては，ある単語e_jと共起するかどうかを考え，単語w_iのe_jに関する重みv_{ij}を測る．単語e_jの設定方法は様々な手法があるが，ここでは空間上で基底をなすような単語群を直接選ぶことにした．論文[1]で抽出された922クラスの単語クラスタの各クラスタから代表的な単語を選んだ．結果922単語選ばれ，単語w_iは922次元の特徴ベクトルで表現されることになる．Googleを利用して，w_iとe_jの共

起頻度をカウントし，正規化することで，特徴ベクトルを作成した．

またクラスタリング対象の単語群を以下のように定めた．まず，互いに関連性の少ない5つの概念のクラスを設定し，そのそれぞれに対して関連性のある5つの単語を選出した計25単語を設定した．つまり理想的には，クラスタリングにより5つの単語からなる5つのクラスタリングができるはずである．

(2) 属性ベクトルの作成

単語の特徴をもれなく記述したテキストデータから特徴ベクトルを作成することができるならば，2つの単語の特徴ベクトル中の特徴ごとの重みの比較によって類似性判別が可能である．しかし，テキストデータから抽出される特徴には常に欠落やノイズが存在するため，適切な判別を行うことができない．そのため，特徴間の関連性を考慮して主要で相互に関連性の少ない特徴に変換して，テキストデータ中の特徴のノイズや欠落の影響を減らした属性ベクトルの作成が行われる．

特徴行列 G_o の m の特徴を $k(\leq m)$ の互いに関連性の少ない特徴(「属性」と呼ぶ)に変換する．特徴の線形変換によって特徴行列を個々の単語の属性の重みを要素とする属性行列 G へ変換できると仮定すると，その変換は m 行 k 列の変換行列を掛け合わせることに等しい．

$$G = G_o K \tag{2.3}$$

また，K を特徴ベクトルに掛け合わせることで属性ベクトルに変換できる．

$$\acute{w}_i = w_i K = (\acute{v}_{i1}, \cdots, \acute{v}_{ik}) \tag{2.4}$$

変換の方式としては主として，特徴行列の特異値分解や主成分分析による方式(数学モデル)と，大規模なシソーラスを利用し，特徴を表す語彙をシソーラスの分類に一般化する方式(シソーラスモデル)がある．最近，これらの2つの変換方式を併用する方式(併用モデル)が提案されている．特徴行列中の特徴をシソーラスの分類に一般化した後，特異値分解を適用するモデルであり，同じ次元数の属性行列では元となる2つのモデルより類似性判別の精度が高く，200以上の高い次元数まで単調に判別精度が高くなる属性行列が作成できる点で優れていることが報告されている．

(3) 類似度計算

属性行列 G を用いて，語彙 N 中の任意の2単語間の類似度を計算する．類似度とは，2つの単語の似ている度合を表す尺度であり，値が大きなほどその単語同士が類似していることを表す．単語 W_p と $W_q(\in N)$ の類似度 $sim(W_p, W_q)$ は，属性ベクトル w_p と w_q より計算される．代表的な類似度としては，2つの単語の属性ベクトルのなす角度の余弦が用いられている．

$$sim(W_p, W_q) = \frac{\acute{w}_p \acute{w}_q}{|\acute{w}_p||\acute{w}_q|} \tag{2.5}$$

これ以外の類似度の尺度としては，属性ベクトル中の重みの絶対値の比較に基づいて計算する方式や，重みを確率値とみなし，確率分布間の距離を測るダイバージェンスより計算する方式などがある．

類似度の計算に基づく単語の類似性判別例を示す．「「猪」と「馬」のどちらが「豚」に似ているか?」という質問に対する判別をするためには，まず以下の類似度を計算する．

$$sim(W_{豚}, W_{猪}) = 0.70 \qquad sim(W_{豚}, W_{馬}) = 0.53 \qquad (2.6)$$

この結果は，学研国語大事典の電子化辞書を用いて構築した属性行列を用い，(2.3) 式で類似度の計算をしたものである．両者の類似度を比較することにより，値がより高い「猪」の方が「豚」と類似していると判定される．

心理学の研究では，人間の単語の類似性判別には方向性が存在することが報告されている ($sim(W_p, W_q) \neq sim(W_q, W_p) \quad if \quad p \neq q$)．それに対して単語のベクトル空間モデルでは，2つの単語の類似度は (2.3) 式のように表現されるので方向性が無い．そのため，ユーザが指定する判別の観点を導入する方式や，判別の状況を表す部分空間を生成して類似度を計算する方式などが提案されている．

2.2 クラスタリング手法

クラスタリングとは，異質なもののまざり合っている対象の中で，互いに似たものを集めて集合 (クラスタ) をつくり，対象を分類しようという方法を総称したもので，数値分類法 (numerical classification,

numerical taxonomy) とも呼ばれる．症状や検査値にもとづく患者の分類，財務諸指標による企業の分類，形状や性質による細菌の分類，$\cdots$ といったさまざまな分野に応用されている．

クラスタリング手法は大きく，最短距離法などの階層的手法と，k-means 法などの分割的手法に分けられるが，これらの基本的手法を紹介する [2].

(1) 階層的手法

階層的手法は，さらに分岐型 (divisive) と凝集型 (agglomeratibe) に分けられるが，ここでは後者のみについて紹介する．

いま n 個の対象 (個体でも変量でもより) $O_1, O_2, \cdots, O_n$ があり，対象 O_i との O_j との間の類似の度合を表す数値 d_{ij}，$(i, j = 1, 2, \cdots, n)$ が得られているとしよう．ただし，d_{ij} は対象的 $(d_{ij} = d_{ji})$ であるとする．

類似の度合を表す指標として，距離のように値の小さい方が類似性が高いことを表す場合と，相関係数のように値の大きい方が類似性が高いことを表す場合がある．両者を総称して類似度と呼ぶこともあるが，ここでは前者の指標を非類似度 (dissimilarity)，後者の指標を類似度 (similarity) と呼んで区別しておく．以下では，簡単のため，d_{ij} は値が小さいほど，類似性が高い非類似度を表すものとして記述する．

階層的手法では，このような対象間の非類似度 (d_{ij}) を手がかりに

して，樹形図あるいはデンドログラム(dendrogram)とよばれる樹状の分類構造を構成することを目標とする手法である．その樹形図を適当な断面で切ることにより，1～n個の任意個数のクラスタを得ることができる．このとき，枝の先端に近いところで切断してできる，少数の構成単位からなるクラスタは，その枝のついている，より大きい枝の根もとのところで切断してできる，多数の構成単位からなるクラスタに，そのまま含まれる．すなわち，樹形図のいろいろな断面で切ってできるクラスタは小分類-中分類-……-大分類という階層的構造をもっている．

凝集型のプロセスは，一般に次のようなアルゴリズムで行われる．凝集型のアルゴリズムを図2.1に示す．

1. 1つずつ対象を構成単位とするn個のクラスタから出発する
2. クラスタ間の非類似度行列(d_{ij})を参照して，もっとも類似性の高い2つのクラスタを融合して，1つのクラスタをつくる
3. クラスタ数が1になっていれば終了．そうでなければ，次のステップへ進む
4. ステップ2で新しくつくられたクラスタと，他のクラスタとの非類似度を計算して，非類似度行列(d_{ij})を更新して，ステップ2に戻る

図 2.1: 凝集型のアルゴリズム

図2.1のステップ4において計算される，クラスタ間の非類似度の定義の仕方にいろいろな考え方があり，それぞれに1つのクラスタリング手法が対応している．

(1.1) 最短距離法

クラスタ (p) とクラスタ (q) を融合して，新しくクラスタ (t) を作る段階を考えよう．それぞれのクラスタの大きさ，すなわちその中に含まれる構成単位の数を $n_p, n_q, n_t(=n_p+n_q)$ とする．

2つのクラスタを融合してつくられるクラスタ (t) と，別の任意のクラスタ (r) との間の非類似度 d_{tr} を，融合する前の段階のクラスタ (p)，クラスタ (q) とクラスタ (r) との非類似度 d_{pr}, d_{qr} を用いて

$$d_{tr} = \min(d_{pr}, d_{qr}) \tag{2.7}$$

と定義する．この定義を，クラスタの構成単位が1つのときから順次適用すると，結局2つのクラスタ間の非類似度は，それぞれに含まれる対象の対の中で，最も類似性の高い対の間の非類似度によって定義されることになる．その意味で，最短距離法とか，単連結法と呼ばれる．

(1.2) 最長距離法

クラスタ (p) とクラスタ (q) を融合してクラスタ (t) をつくるとき，クラスタ (t) と別の任意のクラスタ (r) との間の非類似度 d_{tr} を

$$d_{tr} = \max(d_{pr}, d_{qr}) \tag{2.8}$$

によって定義する．これはちょうど最短距離法の場合と対象的であり，2つのクラスタ間の非類似度は，対象間のもっとも類似性の低い

対の非類似度により定義されることになる．その意味で最長距離法あるいは完全連結法と呼ばれる．

(1.3) 群平均法

最短距離法や最長距離法では，クラスタ間の非類似度は，それらを構成する対象の対の非類似度の中の，最大または最小という極端な値にもとづいて定義された．これに対してクラスタ間の非類似度を，それらに含まれる対象間の類似度の平均的な値で定義しようという考え方がある．群平均法はそのような方法の1つで，クラスタ (t) に含まれる対象とクラスタ (r) に含まれる対象の，可能なすべての対の非類似度の平均により，両クラスタ間の比類似度を定義する．

融合する前と後の非類似度の間の関係は

$$d_{tr} = \frac{n_p d_{pr} + n_q d_{qr}}{n_p + n_q} \tag{2.9}$$

と表される．

(1.4) 重心法

重心法では，クラスタ間の非類似度はそれぞれのクラスタの重心(平均ベクトル)間の非類似度にもとづいて定義される．いま各対象について，m 次元の観測値 $(x_1, \cdots, x_m)$ が得られているとする．

クラスタ (p)，クラスタ (q) の重心を，それぞれ $(\bar{x}_1^{(p)}, \cdots, \bar{x}_m^{(p)})$，$(\bar{x}_1^{(q)}, \cdots, \bar{x}_m^{(q)})$

とすれば，クラスタ (t) の重心 $(\bar{x}_1^{(t)}, \cdots, \bar{x}_m^{(t)})$ は次のように表される．

$$\bar{x}_j^{(t)} = \frac{n_p \bar{x}_j^{(p)} + n_q \bar{x}_j^{(q)}}{n_p + n_q} \tag{2.10}$$

このとき，クラスタ (t) とクラスタ (r) の重心間のユークリッド平方距離 E_{tr}^2 は

$$E_{tr}^2 = \left(\sum_{j=1}^{m} \frac{n_p \bar{x}_j^{(p)} + n_q \bar{x}_j^{(q)}}{n_p + n_q} - \bar{x}_j^{(r)} \right)^2$$

$$= \frac{n_p}{n_p + n_q} E_{pr}^2 + \frac{n_q}{n_p + n_q} E_{qr}^2 - \frac{n_p n_q}{(n_p + n_q)^2} E_{pq}^2 \tag{2.11}$$

と表されるので，非類似度として $d_{ij} = E_{ij}^2$ と定義すると，次のような比類似度の更新の公式が得られる．

$$d_{tr} = \frac{n_p}{n_p + n_q} d_{pr}^2 + \frac{n_q}{n_p + n_q} d_{qr}^2 - \frac{n_p n_q}{(n_p + n_q)^2} d_{pq}^2 \tag{2.12}$$

形式的には，式 (2.12) はどのように定義された非類似度に対しても適用できるが，上の誘導より明らかなように，非類似度がユークリッドの平方距離で定義されることを前提としている．

(1.5) メジアン法

重心法を単純化した変法として，メジアン法がある．重心法では非類似度の更新の式 (2.12) において，クラスタの大きさ n_p, n_q で重みづけされていたのに対し，メジアン法では，クラスタの大きさによ

らず等しい重みをおいて

$$d_{tr} = \frac{1}{2}d_{pr} + \frac{1}{2}d_{qr} - \frac{1}{4}d_{pq} \tag{2.13}$$

とされる．これはちょうど融合してつくられたクラスタの代表点を，もとの2つのクラスタの代表点の中点にとって，クラスタ間の非類似度を代表点間のユークリッド平方距離で測ることに相当する．

(1.6) Ward法

クラスタ (p) に含まれる i 番目の対象を考え，その変量 x_j に関する観測値を $x_{ji}^{(p)}$ と表せば，クラスタ (p) 内の偏差平方和の合計は

$$\triangle S_{pq} = \frac{n_p n_q}{n_p + n_q} \sum_{j=1}^{m} \left(\bar{x}_j^{(p)} - \bar{x}_j^{(q)} \right)^2 \tag{2.14}$$

となる．Ward法では，クラスタ内平方和が，できるだけ小さいことを望ましいと考え，各段階でクラスタの融合による平方和の増分 $\triangle S_{pq}$ が，もっとも小さい (p) と (q) を融合しようとする．そのため，クラスタ (p) とクラスタ (q) の非類似度 d_{pq} として $\triangle S_{pq}$ を用いる．

2つのクラスタ (p) とクラスタ (q) を融合してつくられたクラスタ (t) と，別のクラスタ (r) を融合するときの平方和の増分 $\triangle S_{tr}$ は

$$\triangle S_{tr} = \frac{n_t n_r}{n_t + n_r} \sum_{j=1}^{m} \left(\bar{x}_j^{(t)} - \bar{x}_j^{(r)} \right)^2$$

$$= \frac{n_p + n_r}{n_t + n_r}\triangle S_{pr} + \frac{n_q + n_r}{n_t + n_r}\triangle S_{qr} - \frac{n_r}{n_t + n_r}\triangle S_{pq} \tag{2.15}$$

と表されるから，非類似度の更新の式は次のよになる．

$$d_{tr} = \frac{n_p + n_r}{n_t + n_r} d_{pr} + \frac{n_q + n_r}{n_t + n_r} d_{qr} - \frac{n_r}{n_t + n_r} d_{pq} \qquad (2.16)$$

すべてのクラスタが1つずつの対象からなる場合を考えると，もとの対象間の非類似度としては，ユークリッド平方距離の1/2を用いればよいことがわかる．

(1.7) 可変法

以上の(1.1)~(1.6)の各方法における非類似度の更新は，パラメータ $\alpha_p, \alpha_q, \beta, \gamma$ を用いて

$$d_{tr} = \alpha_p d_{pr} + \alpha_q dqr + \beta d_{pq} + \gamma |d_{pr} - dqr| \qquad (2.17)$$

のような，共通の1つの式で表される．Lance and Williams (1967) は式(2.17)を用いる方法を提唱し，この式にもとづく一群の方法を組合せ的方法と呼んだ．また，パラメータ $\alpha_p, \alpha_q, \beta, \gamma$ について，$\alpha_p + \alpha_q + \beta = 1; \alpha_p = \alpha_q; \beta < 1, \gamma = 0$ の条件をみたす範囲で，任意の値を用いる方法を可変法として提案している．上の条件より，自由なパラメータは β だけであるが，$-1/4 \leq \beta \leq 0$，とくに $\beta = -1/4$ が使われることが多い．以上，(1.1)~(1.7) の各方法について，組合せ法におけるパラメータの値ならびに性質をまとめたものを表2.1に示す．

表 2.1: 組合せ的方法のパラメータと階層的クラスタリング方法の対応

方法	組み合わせ的方法のパラメータ				樹形図の単調性	空間の収縮・膨張	その他
	α_p	α_q	β	γ			
(1) 最短距離法	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	○	収縮	どんな非類似度でも利用できる。(1), (2)では順序尺度でもよい。
(2) 最長距離	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	○	膨張	
(3)郡平均	$\frac{n_p}{n_t}$	$\frac{n_p}{n_t}$	0	0	×	保存	
(4)重心	$\frac{n_p}{n_t}$	$\frac{n_p}{n_t}$	$-\frac{n_p n_q}{n_t^2}$	0	×	保存	非類似度がユークリッド平方距離で与えられることを前提としている。
(5)メジアン	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0	×	保存	
(6)Ward	$\frac{n_p+n_r}{n_t+n_r}$	$\frac{n_p+n_r}{n_t+n_r}$	$-\frac{n_r}{n_t+n_r}$	0	○	膨張	
(7)可変	$\frac{1-\beta}{2}$	$\frac{1-\beta}{2}$	$\beta<1$	0	○	$\beta=0$ 保存 $\beta>0$ 収縮 $\beta<0$ 膨張	

注1) 樹形図の単調性：次々にクラスタを階層的に融合していくときの非類似度が大きくなること。そのとき，樹形図の枝は逆向きにのびることはない.

注2) 空間の収縮・膨張：2つのクラスタを融合して1つのクラスタをつくると，他のクラスタとの距離が近づく場合と遠ざかる場合がある．前者を空間の収縮(space-contracting)，後者を空間の膨張(space-dilating)，距離が変わらない場合を空間の保存(space-conserving)という．最短距離法のように空間が収縮する場合，融合するごとに距離が近づくため，長い鎖状のクラスタができやすいことがしられている.

表2.2のようなm変量観測値が得られているときに，個体間の非類似度を表す量として，ユークリッド平方距離(squared Euclidean distance)と標準ユークリッド平方距離(standardized squared Euclidean distance)を用いる場合を考える．

表 2.2: n 個体の m 変量観測値

個体＼変量	x_1	x_2	$\cdots$	x_m
1	x_{11}	x_{21}	$\cdots$	x_{m1}
2	x_{12}	x_{22}	$\cdots$	x_{m2}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
n	x_{1n}	x_{2n}	$\cdots$	x_{mn}

ユークリッド平方距離

個体iと個体jとの非類似度d_{ij}を

$$d_{ij} = \sum_{k=1}^{m} (x_{ki} - x_{kj})^2 \tag{2.18}$$

により定義する．表2.2に示したように重心法，メジアン法，Ward法を利用する場合には，非類似度としてユークリッド平方距離あるいは標準ユークリッド平方距離を用いるのがよい．

標準ユークリッド平方距離

個体iと個体jとの非類似度d_{ij}を

$$d_{ij} = \sum_{k=1}^{m} \frac{(x_{ki} - x_{kj})^2}{s_k^2} \tag{2.19}$$

により定義する．ここにs_k^2は変量のx_kの分散を表す．これは各変量の分散を1に標準化しておいて，式(2.18)により平方距離を求めることと同じである．

(2) 分割的手法

分割的手法は，分割の良さの評価関数を定め，その評価関数を最適にする分割を探索する．可能な分割の総数は単語数に対して指数的なので，実際は準最適解を求める．代表的なk-means法では，セントロイドをクラスタの代表点とし

$$\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i}(\boldsymbol{D}(\boldsymbol{x},\boldsymbol{c_i}))^2 \tag{2.20}$$

の評価関数を最大化する．最適解の探索のアルゴリズムを図2.2に示す．対象のクラスタへの割当てと代表点の再計算を交互に繰り返して行っている．この手法は山登り法で，局所最適解しか求められないため，ランダムに初期値を変更して，評価関数を最大にする結果を選択する．

1. k個の代表点$c_1, \cdots, c_k$をランダムに選択
2. $\forall x \in \boldsymbol{X}$ を $min_i \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{c_i})$ なる代表点に割当て
3. ***if*** 代表点への割当が変化しない ***then*** 終了 ***else*** 各クラスタのセントロイドを代表点にしてステップ2.へ

図 2.2: k-means法アルゴリズム

本実験では，階層的手法で代表的なWard法と，分割的手法で代表的なk-means法を用いてクラスタリングを行う．

2.3 クラスタリング結果の解釈

[2]

クラスタリングは探索的なデータ解析手法であって，分割は必ず何らかの主観や観点に基づいている．よって，クラスタリングした結果は，データの要約などの知見を得るために用い，客観的な証拠として用いてはならない．

データベースから明確な目的に適合する文書を検索する場合，キーワードを用いた文書検索手法は有効である．しかし，明確な目的がなく，データベース全滝野傾向を知りたい場合はどうであろうか?この場合，具体的なキーワードを示すことは困難なので，文書検索手法の利用は不適当である．そこで，クラスタリングによって，その要約，すなわち，データベース中の主な話題を表すカテゴリの一覧を取り出す．Cuttingらは，ニューヨーク・タイムズ紙1990年8月の約5000件の記事のデータベースの傾向を抽出する問題にクラスタリングを用いた．話題が類似している文書をまとめたクラスタを生成した結果，以下の話題を含むクラスタが発見された．

教育，国内，イラク，芸術，スポーツ，石油，ドイツ統合，裁判

利用者は，内容が全く不明であった新聞記事のデータベースのおおまかな内容を，これらのクラスタからしることができるであろう．この要約は，一つのクラスタ，例えば慰楽をさらに分割して，パキス

タンやアフリカといったより詳細な要約を得たり，文書検索のためのキーワードを決める目的にも利用できる．クラスタリングは，このように未知のデータベースの内容に見当をつける目的で利用できるため探索的であるといえる．

ここで注意すべき点は，この例では8個のクラスタに記事を分類し，データベースの「正しい」要約を得ることができた．しかし，イラクと石油やどちらも湾岸戦争に関する話題なので，これらをまとめても，データベースの「正しい」要約といえる．すなわち，どちらにも，それを正当化する視点が存在する．このように，クラスタリングの結果は絶対的でも，普遍的でも，客観的でもないので，分割結果は結論を導く証拠にはなり得ない．例えば，教育と国内が違うクラスタに分類されているが，これは実社会で二つの問題に関連性が皆無であることを意味しない．クラスタリング結果の妥当性は，その分割の利用目的など，外的な知識によって判断するしかない．例えば，新聞記事の話題の抽出という目的であれば，国内とドイツ統合を同じクラスタに分類していれば，妥当な要約とはいえないであろう．だが，同じ週に起きた事件をまとめるという目的ならば，これらをまとめるのも妥当かもしれない．クラスタリングの結果は，その利用目的などに応じて，妥当性を常に検証する必要がある．

2.4 クラスタリングを行う25単語

本実験では25単語に対してクラスタリングを行う．その25単語の選出手順としては，まず互いに関連性の少ない5つの概念を定めた．

その5つの概念とは,「動物」,「食べ物」,「感情や人生観」,「繁栄しているもの」,「地名」である．次に，定めた5つの概念別に，その概念と関連性のある単語をそれぞれ5つ抽出した．クラスタリングを行う25単語はこのようにして設定した．以下にその25単語を示す．

動物 プードル，チワワ，犬，猿，ゴリラ

食べ物 カレー，ラーメン，スパゲッティー，焼きそば，ハンバーグ

感情や人生観 幸福，満足，愛情，結婚，運命

繁栄しているもの 情報，知識，手段，交通，設備

地名 今帰仁，沖縄，ブリスベン，オーストラリア，オーストリア

2.5 基底の単語選出方法

基底の単語は，95年度版毎日新聞1年間分の記事中の単語をクラスタリングすることから得たもので，論文[1]で抽出された922クラスの単語クラスタの各クラスタから代表的な単語を選んだ結果，選ばれた922単語である．

2.5.1 共起データの抽出と共起関係行列の作成

[4]

単語のクラスタリングを行うために，新聞記事から共起関係を持つ名詞単語の組を抽出し，それを特徴要素として利用する．抽出する共起関係としては,「～の～」といった表現となる「名詞」＋「格助詞」＋「名詞」，複合名詞など名詞が組み合わさってまとまりをなし

た「名詞」＋「名詞」の2つのパターンを持つ名詞単語の組とする．

この共起関係を抽出する際，いくつかの例外を定義してより正確な共起関係が抽出できるようにしている．まず，名詞間に副詞や形容詞が挿入されている場合は，これらの単語は無視して上記の形に従っていれば単語の組を抽出する．また，ここでは名詞単語のみを考慮しているため，代名詞は共起データの候補として残さず，簡単化のため「1月」,「一月」のように月日を表す名詞，ひらがな，カタカナ，アルファベット語は無視している．さらに，名詞単語の中で「名詞-数」や「名詞-接尾-助数詞」と判定される名詞単語については,「1本の鉛筆を買った」や「鉛筆を1本買った」というように「1本」という語が使われる位置が前後するために共起データの候補としては考慮しなかった．検索対象のデータに対して，茶筅を用いて形態素解析を行い，上述のパターンに従い共起データを抽出し，このデータをもとに統計的な重みを与えて共起関係行列を作成する．このとき，名詞単語の組でどちらの名詞単語を対象としてクラスタリングを行うかが問題となる．例えば「AのB」というパターンを考えた場合，名詞単語AとBのどちらを用いるかということであるが，修飾される単語の方が重要性が高くなると考える．そのため，名詞単語Bを対象にクラスタリングを行い，以降これを対象単語と呼ぶこととする．この対象単語を数値的に表現するために，修飾単語Aの部分に出現する単語を要素とする単語ベクトルとしてベクトル空間内の点とみなす．

これらの単語を要素とする文書ベクトルを作成するとき，単語の

頻度に重みを加えた数値をベクトルの要素とする．数多く提案されている重みづけ手法の中で一般的に文書検索で用いられる対数エントロピー重みを用いて重みづけを行った．この重みづけは，局所的重みL_{ij}を第j番目の対象単語に対する，i番目の修飾単語への重み，大域的重みG_iを全共起データに出現するi番目の修飾単語への重みとしてそれぞれ以下のように表す．

$$L_{ij} = \begin{cases} 1 + \log f_{ij} & (f_{ij} > 0) \\ 0 & (f_{ij} = 0) \end{cases} \tag{2.21}$$

ここで，nは対象単語数，f_{ij}はj番目の対象単語に共起するi番目の単語の頻度，F_iは全体の共起データにおけるi番目の修飾単語の頻度を表す．これにより，j番目の対象単語を表す単語ベクトルにおけるi番目の修飾単語を表現した．

2.5.2 基底の922単語

論文[1]で抽出された922単語を以下に示す．本実験では，この922単語を基底の単語として用いる．

制裁，甘口，決定，仮谷，村山，制度，伊勢，新社屋，表明，固定，帝京，常任，学識，任意，東条，勢力，場合，場所，結果，手当，間違い，信用組合，出場，東大，福祉，丸太町，病院，能力，地域，全身，目標，夏季，玉沢，行方，地下，尼崎，高原，週間，改革，確信，産業，幹部，参謀，登録，規制，大使，問題，羽田，表面，依然，規

模，移転，握手，地元，代行，来年，羽生，治療，衛星，資料，記者，衛生，清輝，継投，出版，巨額，記念，一時，大手，製薬，調査，比例，正常，大好き，海外，紹介，梅田，地震，調達，動燃，拒否，区間，拠点，富士山，促進，混乱，放送，寄稿，暫定，自ら，残り，合意，神戸大，実感，保険，金融，気候，海上，件数，老人，情報，地方，泳三，化粧，指摘，発生，条約，注意，礼拝，方法，選挙，民事，方針，環境，広島，役割，補助，政界，円滑，葬式，予定，政権，仕事，退席，合同，五木，財政，下車，法案，国税局，長男，価格，政策，首相，抵抗，首都，逮捕，特訓，政治，健康，審査，野球，救援，一切，三月，監督，千葉，部分，国道，拉致，法人，価値，孝弘，北海道，戦時，少数，選手，故郷，新王子製紙，一挙，原子，一体，使用，政府，自宅，判決，三郎，管理，台東，根本，追及，等身，自動車，期待，投稿，弾圧，敦子，信義，太郎，緩やか，制球，回顧，断層，相当，職員，分野，自分，吸収，整備，周囲，神戸，自民党，模様，一致，自由，国家，江藤，批判，千代田，国選，国会，史上，弟子，半分，伊奈，歌集，存在，震度，快挙，全壊，青山，震源，閣議，観客，五輪，観光，未知数，一般，行為，自覚，一番，開始，崩壊，作家，異臭，開催，不安，練習，午後，定例，低め，国際，一部，殺人，出馬，密集，製品，入学，勝利，慰安，援助，生

息，　需要，　再生，　胃がん，　年度，　建設，　写真，　開発，　還流，　漢字，　建築，　展示，　大統領，　屈指，　予想，　競技，　文化，　技術，　文化財，　競争，　民間，　余裕，　反省，　当時，　可能，　物資，　直径，　安心，　昨年，　一方，　富市，　責任，　後期，　疑問，　詰め，　疑惑，　歴史，　栗山，　再販，　永田町，　相手，　英明，　周辺，　国籍，　初め，　当面，　憲法，　民主，　特徴，　共同，　多く，　看板，　宗教，　感染，　疑い，　関係，　道路，　長銀，　成長，　回帰線，　勤務，　外交，　公的，　外国，　一郎，　国内，　豊か，　赤字，　昭二，　関西，　外資，　統一，　子供，　協会，　検討，　投与，　訂正，　処理，　不十分，　負担，　貢献，　利用，　明治，　構成，　旗印，　協力，　正念場，　上村，　同士，　外科，　首席，　高橋，　自衛隊，　竜太郎，　国民，　多数，　税金，　同紙，　平和，　人間，　複雑，　江戸，　直接，　人気，　山田，　伸び，　官房，　近畿，　国立，　野茂，　死亡，　分け，　全国，　人権，　松本，　国連，　廃絶，　出資，　預金保険機構，　連続，　川崎，　拍手，　積極，　家族，　研究，　関連，　大リーガー，　安全，　伊達，　必然，　全体，　韓国，　株式会社，　南西，　連邦，　思い，　推測，　本塁打，　福岡，　電気，　連絡，　番組，　議員，　知事，　容疑，　全日本，　議会，　本件，　市内，　関ケ原，　親族，　議席，　問い，　一層，　科学，　注文，　黒人，　次郎，　不審，　世論，　広三，　岐部，　出身，　出頭，　販売，　明美，　中央，　総額，　出家，　体験，　反撃，　政調，　選択，　説明，　各国，　同様，　大使館，　教育，　彼女，　年間，　安保，　電話，　復興，　優勝，　日本一，　満州，　通用，　大

切，　実施，　余震，　保守，　賢治，　恵子，　医師，　貿易，　登山，
教授，　防衛，　教祖，　実績，　遺伝子，　拡大，　少年，　幼稚園，
喜び，　負け，　記憶，　状況，　補給，　設置，　保安，　貸し手，　相
撲，　坂東，　幸い，　突進，　協議，　喫煙，　飲料，　具体，　慎重，
交通，　心配，　清水，　実験，　案内，　応募，　追加，　今回，　満
載，　核兵器，　今季，　上九一色，　持参，　流動，　文藝春秋，　教
室，　動物，　今後，　弁護，　引き金，　一つ，　政党，　大阪ガス，
以上，　新た，　中国，　逃避，　以来，　番号，　今春，　生産，　体
制，　軍事，　不通，　平安，　高齢，　刺激，　商店，　指定，　明日，
就任，　指導，　被害，　東証，　二郎，　組織，　被告，　四月，　対
応，　都市，　支援，　適用，　今年，　今年度，　連結，　提出，　幸
男，　対策，　生活，　事務所，　敗北，　対象，　通産，　予知，　自治
体，　墨田，　首脳，　状態，　太平洋，　新規，　女子，　通常，　通
信，　米国，　中心，　施設，　天理教，　事態，　革新，　田嶌，　招
致，　焦点，　御影，　橋本，　女性，　快勝，　会館，　姿勢，　同
社，　社員，　専攻，　社会，　証人，　兵庫，　正面，　社会党，　須
美子，　死去，　航空，　右足，　本拠地，　推進，　学校，　同志社
大，　不服，　協同，　奥田，　編集，　末尾，　犯罪，　無事，　木
津，　連勝，　心理，　北朝鮮，　教団，　放棄，　学生，　集中，　企
画，　企業，　建物，　社長，　欧州，　危機，　一見，　延べ，　危
険，　助手，　公開，　水俣病，　同日，　昭和，　半面，　契約，　復
旧，　熱帯，　浮島，　報告，　気味，　抗議，　辺り，　住宅，　肺炎，
初期，　手帳，　在日，　予感，　主婦，　委員，　住民，　有力，　生

まれ，　安田，　捜査，　明大，　紘一，　生駒，　私立，　改編，　喪主，　三浦，　国防，　意識，　自主，　嗅覚，　破防法，　宝塚，　脚光，　信組，　水面，　導入，　年齢，　意見，　懸念，　画面，　井口，　証券，　正式，　佐久，　早大，　行政，　原則，　個性，　貸し，　損害，　影響，　落札，　浮世絵，　幻想，　行動，　公証，　義援金，　流通，　賞金，　原爆，　強制，　公示，　河野，　松下，　和平，　新党，　日本，　為替，　共産党，　公衆，　病死，　基準，　富士銀行，　競合，　宇宙，　課税，　課題，　作り，　九段，　時刻，　国有，　購入，　専用，　陸海空，　関心，　迅速，　大江，　基本，　十分，　勝浦，　勝ち，　日常，　景気，　信頼，　楽しみ，　戦い，　休み，　本格，　居住，　労災，　新聞，　労働，　厚生省，　従来，　製造，　不定，　単独，　実業，　重視，　視聴，　増加，　移植，　英雄，　代表，　活動，　総務庁，　運転，　項目，　移民，　維持，　試合，　救急，　香港，　東芝，　経営，　分析，　横浜マ，　伊豆，　相談，　合間，　差別，　経済，　学院，　佐賀，　運営，　本社，　推薦，　資金，　歌舞伎，　書士，　実勢，　福利，　応援，　違反，　主義，　費用，　将来，　人事，　禁止，　下山，　問い合わせ，　預金，　株主，　現金，　現行，　参加，　現在，　遺産，　言葉，　章一郎，　一帯，　劇場，　犯人，　現場，　現職，　計画，　学部，　緊急，　現代，　総合，　主演，　一緒，　現地，　営業，　専門，　部落，　三宮，　期限，　警察，　上部，　下敷き，　事業，　敏夫，　事件，　災害，　近く，　大会，　体育館，　保護，　医療，　強調，　事実，　野菜，　各州，　旅行，　機会，　大学，　本部，　機関，　好調，　事務，　保障，　機構，　井戸，

団体，　商品，　掛け声，　山口，　個人，　竜彦，　小型，　軽水炉，　取引，　取材，　重要，　高校，　機能，　予算，　東京，　急激，　介護，　必要，　浴室，　融資，　請求，　補正，　映画，　会員，　朝鮮民主主義人民共和国，　会期，　会議，　会計，　高温，　予防，　帰国，　仰木，　日曜，　特定，　会社，　西宮，　収穫，　特別，　健在，　最後，　被災，　最高，　会見，　会長，　付き，　準備，　最終，　震災，　無党派，　世界，　救済，　会派，　杉並，　葬儀，　時代，　予備，　映像，　実現，　阪神，　解散，　手足，　延長，　与党，　安打，　最大，　得意，　新宿，　戦後，　料金，　合理，　香り，　装置，　投打，　障害，　大卒，　実習，　年代，　解放，　受賞，　入院，　駐車，　仮設，　加入，　搭載，　戦争，　地下鉄，　従事，　私有，　全面，　耐震，　時間，　独立，　手作り，　時期，　変更，　人柄，　青葉，　毎日，　工事，　学期，　毎日新聞，　回復，　避難，　健二，　入り，　時事，　通り，　舞台，　洗濯，　東亜，　左手，　出演，　決議，　発見，　大阪，　決算，　日本電信電話，　決勝，　制限，　化学

第3章　R言語の説明

クラスタリング手法のプログラムとして，R言語でk-means法のプログラムを作成した．ここではR言語について説明する．

Rとは統計計算とグラフィックスのための言語・環境である．Rは多様な統計手法(線形・非線形モデル，古典的統計検定，時系列解析，判別分析，クラスタリング，その他)とグラフィックスを提供し，広汎な拡張が可能である．Rの強みの一つは，うまくデザインされた出版物並のプロットを容易に作成できる点であり，これには必要なら数学記号や式を含めることもできる．グラフィックスにおける細かなデザインの標準設定に，これまで周到な配慮が払われてきているが，ユーザが完全に制御することもできる．RはFree Software FoundationのGNU General Public Licenseの条項の下で，ソースコードの形で入手できる．様々なUnixプラットフォームや．類似のシステム(FreeBSDやLinuxを含む)では，ダウンロードすればそのままでコンパイルでき，稼働する．また，Windows 9X/NT/2000やMacOSでもコンパイルでき，稼働する．

Rはデータ操作，計算，そしてグラフィックス表示のためのソフト機能の統合されたまとまりである．それには以下のものが含まれる．

- データを効率的に操作し，保管する機能

- 配列，とくに行列の計算のための演算子のセット
- データ解析の媒介に使う道具の大規模で一貫した集り
- データ解析のためのグラフィカルな機能と，画面または印刷物への出力
- 条件分岐，ループ，ユーザー定義の再帰的関数や入出力機能を含む，開発の進んだ，簡潔で効率的なプログラミング言語

「環境」という言葉が使われているのは，Rは完全に計画され一貫性を持ったシステムであり，固有の目的のみを持って融通のきかない道具の積み重ねではない (他のデータ解析ソフトウェアはしばしばそうだが) という特徴を示すためである．

第4章　実験

4.1　コーパスを利用した単語クラスタリング

比較実験のためにコーパスを利用したクラスタリングを行なう．

まずコーパスを利用して特徴ベクトルを作成する．単語の特徴としては，単語e_jと1文中で共起するかどうかを考え，単語w_iのe_jに関する重みv_{ij}を，コーパス中でw_iとe_jが共起した文の数で測る．単語e_jの設定方法は，検索エンジンを用いたものと同じ単語を利用した．

コーパスとしては，95年度版毎日新聞1年間分の記事を用いた．次にw_iとe_jの共起頻度をカウントし，正規化することで，特徴ベクトルを作成した．

ward法とk-means法でクラスタリングを行った結果を図4.1，図4.2に示す．

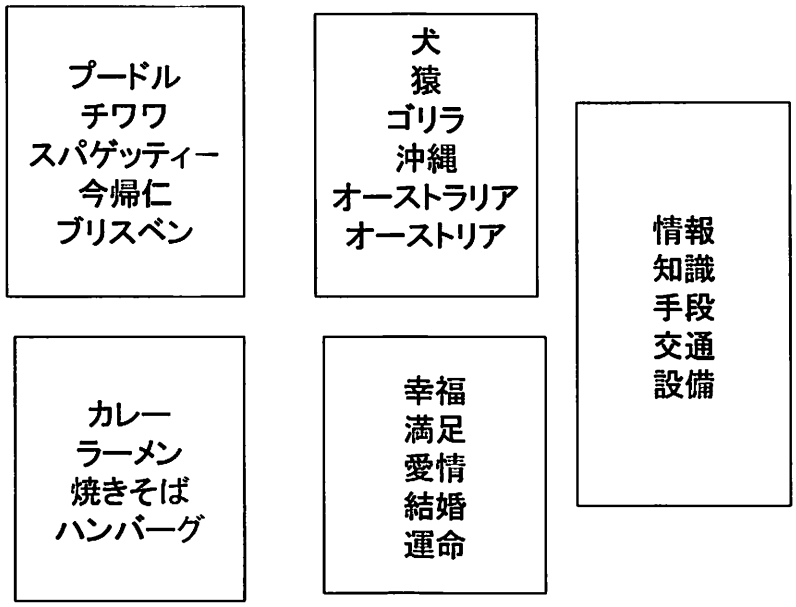

図 4.1: ward 法 (コーパス)

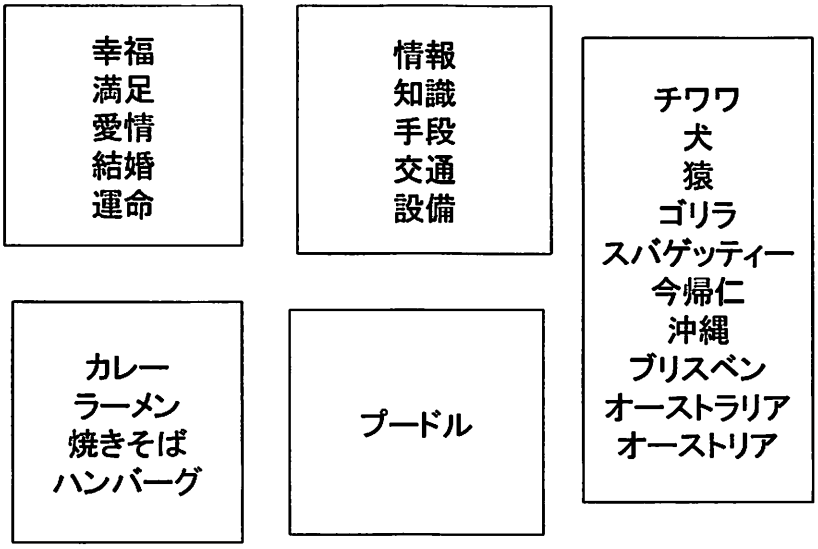

図 4.2: k-means 法 (コーパス)

コーパスを利用したクラスタリング結果をみると，どちらの手法とも「感情や人生観」の概念と「繁栄しているもの」の概念は適切にクラスタリングされている．また，「食べ物」の概念も「スパゲッティー」を除けば適切にクラスタリングされている．そこで，これら以外の2つのクラスに着目してみると，どちらの手法も「犬」，「猿」，「ゴリラ」は同クラスに分類され，「沖縄」，「オーストラリア」，「オーストリア」も同クラスになっている．ここでは，コーパスのスパース性の影響に着目しているので，Ward法とk-means法で異った結果となった，「プードル」，「チワワ」，「スパゲッティー」，「今帰仁」，「ブリスベン」は誤ってクラスタリングされているとみなすことができる．誤りの原因を探るため各単語の頻度を調べる．

Ward法とk-means法でクラスタリングを行った結果に各単語の頻度を調べたものを図4.3，図4.4に示す．各単語の頻度を（ ）に示す．

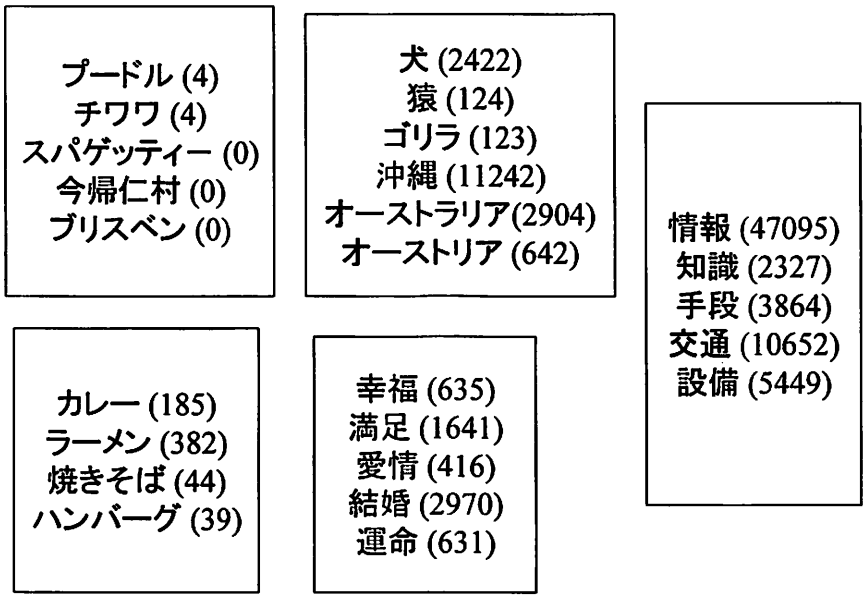

図 4.3: ward 法 (コーパス)

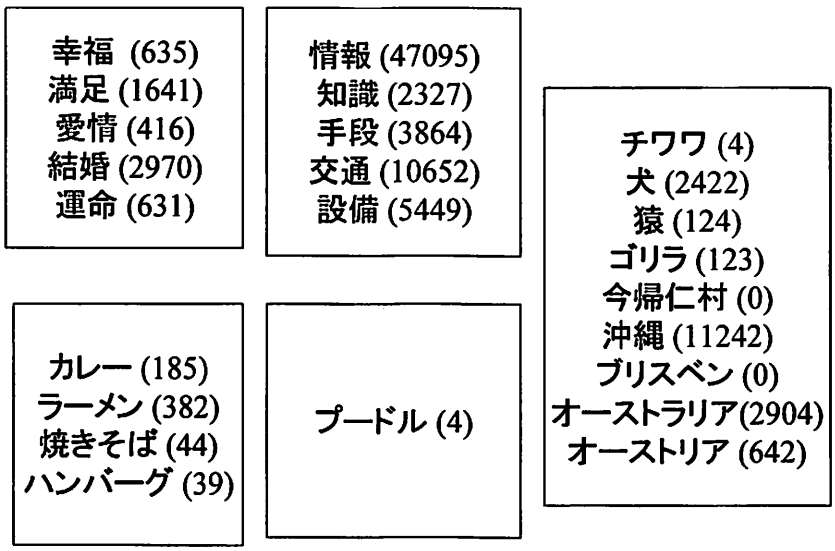

図 4.4: k-means 法 (コーパス)

図4.3と図4.4をみると，頻度が高い単語は適切にクラスタリングされている．誤ってクラスタリングされているとみなした「プードル」,「チワワ」は頻度が4で,「スパゲッティー」,「今帰仁」,「ブリスベン」は頻度が0で，頻度が低いと誤分類されていることがわかる．よって，コーパスを利用したクラスタリングを行うと，コーパスのスパース性の影響があることが確認できる．

4.2 Webを利用した単語クラスタリング

Googleを利用して，単語w_iと単語e_jの共起頻度をカウントし，正規化することで，特徴ベクトルを作成する．

ward法とk-means法でクラスタリングを行った結果を図4.5，図4.6に示す．

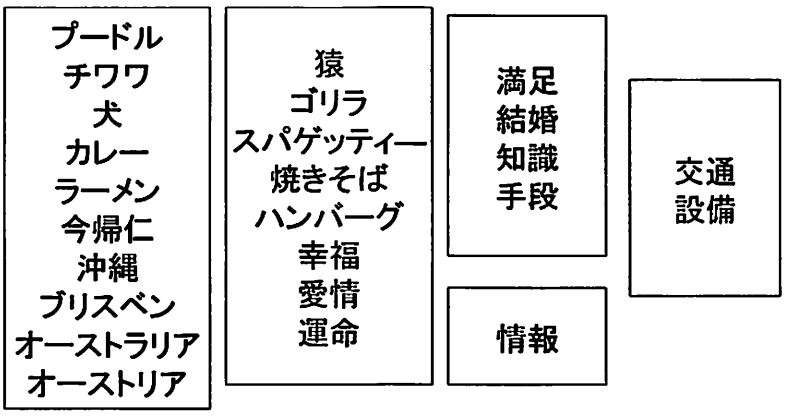

図 4.5: ward法(Web)

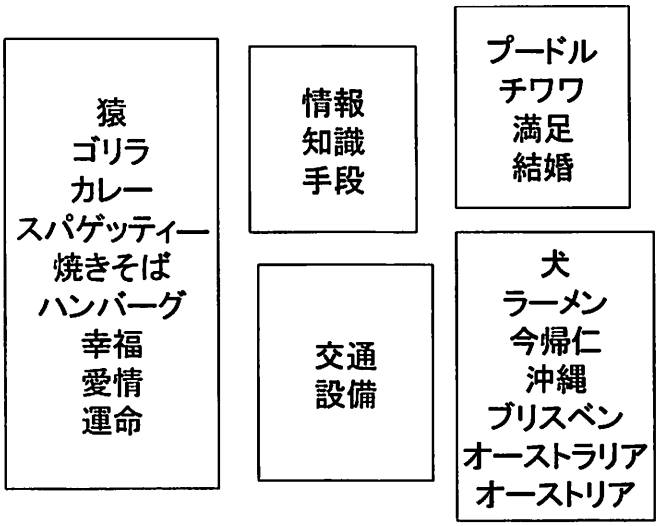

図 4.6: k-means 法 (Web)

Ward法で行ったクラスタリング結果をみてみると,「地名」の概念の単語は皆同じクラスに分類されてはいるが，同じクラス内に「動物」の概念の単語と「食べ物」の概念の単語とが入り混じっている.

k-means法で行ったクラスタリング結果をみてみても，Ward法と同じように他の概念の単語とが入り混じっている.

Webを利用したクラスタリングでは，理想的なクラスタリングとは異った結果となっている．どの手法とも概念別に適切にクラスタリングされているものは一つもなく，コーパスを用いたものほどうまくクラスタリングされていない.

4.3 最適な次元の選出

Webを利用したクラスタリング結果を改良するために，最適な次元の選出を行う．最適な次元は相関比計算より求める．K個のクラスが，どの程度よく判別されているかを表す指標として，相関比η^2(=クラス間分散(S_B)/クラス内分散(S_T))を用いることができる．相関比は$0 \leq \eta^2 \leq 1$の範囲の値をとり，定義より明らかなように，1に近いほどよく判別され，0に近いほど，あまり判別されていないことを表している．

まず，各次元の相関比を求めた．手順は以下のように行った．

1. 25単語を概念別に5つのクラスに分ける
2. 各クラスの最初の要素を削除
3. 各クラスを正規化して，それぞれの相関比を求める
4. 各クラス間の相関比の平均を求める
5. 順に次の要素を削除し，手順3と手順4を繰り返す
6. 求めた各クラス間の相関比平均の中で，最大値をとるものが，その次元の相関比．その次元の相関比となる要素を最終的に削除

このような手順2~5を行って各次元の相関比を求めた．これにより求まった922個の相関比の中で，最大値をとるものが最適な次元となる．

このようにして，相関比が最大になるように基底の単語を設けると，384次元が最適な次元となった．

4.4 最適な次元での単語クラスタリング

相関比が最大になるように基底の単語を設けると，384次元が最適な次元となった．384次元でward法とk-means法でクラスタリングを行った結果を図4.7と図4.8に示す．

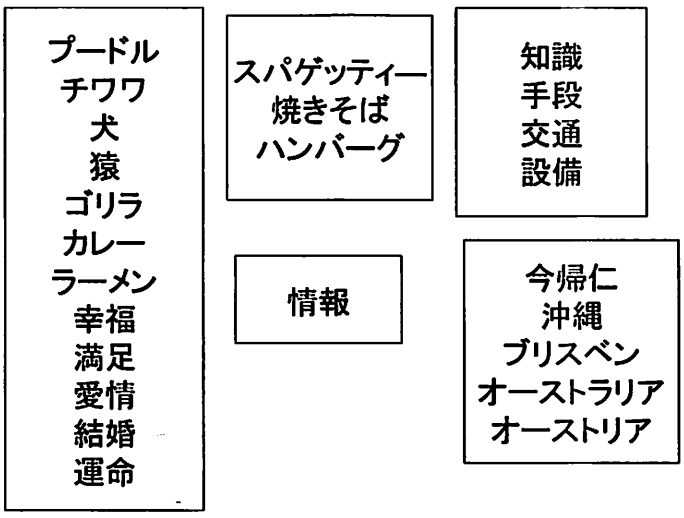

図 4.7: ward法 (改良 Web)

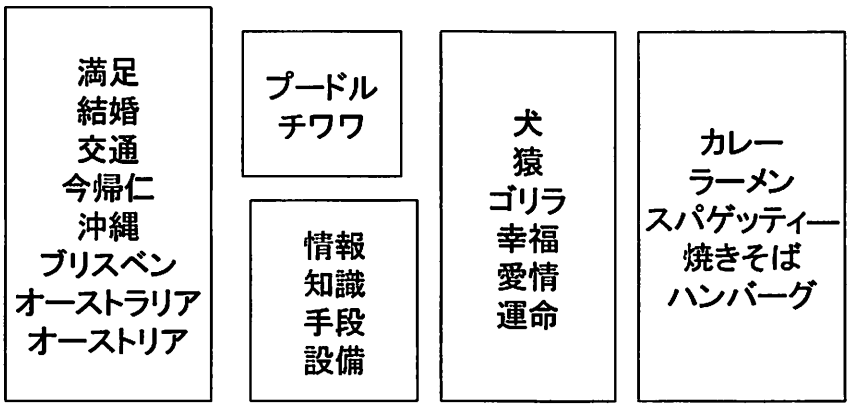

図 4.8: k-means法 (改良 Web)

Ward法で行ったクラスタリング結果をみてみると,「動物」の概念の単語と「感情や人生観」の概念の単語は皆同じクラスに分類されてはいるが，概念別に適切にクラスタリングされてはいない.「繁栄しているもの」の概念の単語は,「情報」を除けば適切にクラスタリングされている．Ward法で適切にクラスタリングされているものは,「地名」の概念だけである.

k-means法で行ったクラスタリング結果をみると,「地名」の概念の単語は皆同じクラスに分類されているが，同じクラス内に他の概念の単語とが入り混じっている.「繁栄しているもの」の概念の単語は,「交通」を除けば適切にクラスタリングされている．k-means法で適切にクラスタリングされているものは,「食べ物」の概念だけである.

改良を試みるまえのWebを利用したクラスタリング結果では，どの手法をみても概念別に適切にクラスタリングされているものは一つもなかったが，最適な次元でクラスタリングを行うことにより，概念別に適切にクラスタリングされているものが一つできた．相関比が最大になるように基底の単語を設けたことで，全体的に改良することができた．しかし，理想的なクラスタリング結果とは若干異った結果となった.

第5章　考察

Webを利用したクラスタリングがあまりうまくいかなかった原因は2つ考えられる．

第1の原因は，基底の単語の選出方法である．ここでの基底の単語選出は，コーパス中の単語をクラスタリングすることから行なわれている[1]．このために，利用した基底はコーパスに対しては有効であったが，Webに対しては有効でなかった可能性がある．最適な次元でクラスタリングを行っても理想的な結果を得ることができなかったのも同じ理由からだと考えられる．

第2の原因は，共起の定義である．Webを利用したクラスタリングでは共起の定義を同じページ内に出現することとしている．しかしこれは共起の定義としては荒い可能性がある．

ちなみに，以下のような実験を行なってみた．

最適な次元となる384次元のデータから，同じクラスで距離が最も離れている単語の対を1つ選んだ．ここでは「情報」と「交通」であった．次にこの2単語で最も距離が離れた次元を選んだ．この次元に対応する単語は『浴室』であった．“情報 浴室”と“交通 浴室”をクエリにしてGoogleで検索し，提示された上位20ページの内容を調べてみた(実験A)．括弧内の数値は上位20ページ中のページ数

である.

「情報」と『浴室』の検索結果

住関連用品 (7)

リフォーム (6)

宿・旅館施設 (3)

住宅設備と建材 (2)

生活情報 (2)

「交通」と『浴室』の検索結果

宿・旅館施設 (7)

賃貸物件 (4)

住宅設備 (4)

旅情報 (1)

(中国語のため内容不明 (4))

次に，同じクラスで距離が最も近い単語の対を1つ選んだ．ここでは「プードル」と「チワワ」であった．次にこの2単語で最も距離が近い次元を選んだ．この次元に対応する単語は『暫定』であった．“プードル 暫定” と “チワワ 暫定” をクエリにして Google で検索し，提示された上位20個のページの内容を調べてみた (実験B).

「プードル」と「暫定」の検索結果

エッセイ・日記等 (15)

動物の画像 (1)

クイズ (1)

動物の病状 (1)

書籍 (2)

「チワワ」と「暫定」の検索結果

エッセイ・日記等 (15)

ペットグッズ (3)

メキシコ年表 (1)

ボクシング (1)

実験 A ではページの内容が異なり「情報」と「交通」の類似性が反映されていない．実験 B ではページの内容が似通っており「プードル」と「チワワ」の類似性が反映されている．つまりここでの共起の定義はある部分では有効に機能しているが，別の部分ではまったく機能していない．

今後は基底の単語の選び方や有効な共起の定義及びその定義に基づく共起頻度を Web から得る方法を考えたい．

第6章　おわりに

本研究では Web をコーパスとして利用するための研究の一貫として，検索エンジンを利用したクラスタリングを行なった．

比較のために新聞記事を利用したクラスタリングも行った．クラスタリング結果は良好であったが，コーパスのスパース性の問題があることが確認できた．

Web のページは新聞記事のように単純なテキストではないので，単にページを収集しただけではコーパスとして利用することは難しい．そこで，Web をコーパス代わりにするために，検索エンジンを利用してクラスタリングを行った．Web 上の情報は莫大な量なので，コーパスのスパース性の影響は回避できると考えられたため，新聞記事をコーパスとしてクラスタリングを行った結果よりも，うまくクラスタリングされることを予想していた．しかし，結果はコーパスを利用したものほどうまくはいかなかった．

原因は基底の単語の選出方法と，ここでの共起の定義が荒いためであると考えられる．今後は基底の単語の選出方法や有効な共起の定義及びその定義に基づく共起頻度を Web から得る方法を考えたい．

単語の選出方法の方向性としては，Web のデータを用いてクラスタリングし，そこからまた新たな基底の単語を選出することを考え

ている．

共起頻度を得る方法の方向性としては，検索エンジンのクエリをうまく使うことや，検索オプションを使うことを考えている．検索エンジンのクエリを2重引用符で囲んで検索することにより，検索範囲を狭めることができる．また，検索オプションを用いることで，ドメインを指定してそのドメイン内だけから検索したり，そのドメイン以外だけから検索したりすることができるので，高度な検索が可能となる．

第7章　謝辞

本研究の遂行及び論文の作成に多大なご助言及び指導を賜わった新納浩幸教官 (茨城大学工学部システム工学科) と岩崎唯史教官 (茨城大学工学部システム工学科) に深い感謝の意を表します．また，本研究でコーパスとして利用した文書データは，95年度版の毎日新聞です．利用を許可して頂きました毎日新聞社に深く感謝します．最後に，本研究を進めるにあたり助言，協力を頂きました，同研究室の紺野憲一 氏 (茨城大学大学院修士課程)，結城隆 氏 (茨城大学工学部システム工学科4回生)，藤井丈明 氏 (茨城大学工学部システム工学科4回生)，脇坂恭志郎 氏 (茨城大学工学部システム工学科4回生)，ならびに，岩崎研究室の時田陽一 氏 (茨城大学工学部システム工学科4回生) にも深く感謝します．

第8章　プログラムソースリスト

```
/* ================================================
 25単語と基底の922単語の共起頻度を求める
================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 256
#define WORDSIZE 100
#define TANGO 25
#define DUMMY 922

void quit(char *);

int main(int argc,char *argv[])
{
 FILE *f1,*f2,*f3;
```

```
 char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE]
buf1[WORDSIZE],tango[TANGO][WORDSIZE],tango1[TANGO][WORDSIZE],d,
buf2[WORDSIZE], dummy[DUMMY][WORDSIZE],dummy1[DUMMY][WORDSIZE],f;
 int count1[TANGO],count2[DUMMY],comb[TANGO][DUMMY];
 int r1,r2,r3,i=0,j=0,k,a,b;

 if(argc!=4) quit("引数の数が違う。prog datafile\n");
 if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けない。
\n");
 if((f2=fopen(argv[2],"r"))==NULL) quit("ファイルが開けない。
\n");
 if((f3=fopen(argv[3],"r"))==NULL) quit("ファイルが開けない。
\n");

 /* 25単語のデータを取り出す*/
 while(fgets(buf1,WORDSIZE,f1)!=NULL)
 {
   strcpy(tango[i],buf1);
   i++;
 }

 for(i=0;i<TANGO;i++)
 {
```

```
  a=0; d=tango[i][a];
  while(d!='\n'){
   tango1[i][a]=tango[i][a]; a++;
   d=tango[i][a];
  }
 tango1[i][a]='\0';
}

/* 922単語のデータを取り出す */
while(fgets(buf2,WORDSIZE,f2)!=NULL)
{
  strcpy(dummy[j],buf2);
  j++;
}

for(j=0;j<DUMMY;j++)
{
  b=0; f=dummy[j][b];
  while(f!='\n'){
   dummy1[j][b]=dummy[j][b]; b++;
   f=dummy[j][b];
  }
 dummy1[j][b]='\0';
```

```
}

for(i=0;i<TANGO;i++)  count1[i]=0;
for(j=0;j<DUMMY;j++)  count2[j]=0;

for(i=0;i<TANGO;i++){
 for(j=0;j<DUMMY;j++){
   comb[i][j]=0;
 }
}

 /* コーパスのデータを取り出す */
 while(fgets(buf,LINESIZE,f3)!=NULL)
{
  sscanf(buf,"%s %s %s %s",word1,word2,word3,word4);
  k=0; c=word1[k];
  while(c!='\0'){
   word5[k]=word1[k]; k++;
   c=word1[k];
  }
  word5[k]='\0';

  /* 1文中に25単語中の単語が現れたらカウント */
```

```
for(i=0;i<TANGO;i++){
 if(strcmp(word5, tango1[i])==0)
 {
  if(count1[i]==0) count1[i]=1;
  else if(count1[i]>=1) count1[i]++;
 }
}

/* 1文中に922単語中の単語が現れたらカウント*/
 for(j=0;j<DUMMY;j++){
  if(strcmp(word5,dummy1[j])==0)
  {
   if(count2[j]==0) count2[j]=1;
   else if(count2[j]>=1) count2[j]++;
  }
 }

 /* 25単語と922単語の共起頻度 */
 if(strcmp(word5,"EOS")==0){
  for(i=0;i<TANGO;i++){
   for(j=0;j<DUMMY;j++){
    if(count1[i]>=1 && count2[j]>=1)
    {
```

```
            if(comb[i][j]==0) comb[i][j]=1;
            else if(comb[i][j]>=1) comb[i][j]++;
           }
          }
         }

         for(i=0;i<TANGO;i++)  count1[i]=0;
         for(j=0;j<DUMMY;j++)  count2[j]=0;
      }
   }
      for(i=0;i<TANGO;i++){
        for(j=0;j<DUMMY;j++){
          if (comb[i][j]){
            printf("%d  %d  %d\n",i,j,comb[i][j]);
          }
        }
       }

   if(((r1=fclose(f1))==-1) && ((r2=fclose(f2))==-1) && ((r3=fclose(
   {
     quit("ファイルが閉じれない。\n");
   }
}
```

```
void quit(char *s)
{
 puts(s);
 exit(1);
}

/* ========================================================
 共起頻度の整理 (1)
========================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 60
#define WORDSIZE 20
#define ARYSIZE 100000

void quit(char *);
int main(int argc,char *argv[])
{
  FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE
      str1[WORDSIZE],str2[WORDSIZE],str3[WORDSIZE],ary[ARYSIZE][WOR
```

```
  int count[ARYSIZE];
  int r,i,j,k,l,lastid=0,a,b;

  if(argc!=2) quit("引数の数が違う。prog datafile\n");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けな
い。\n");

  while(fgets(buf,LINESIZE,f1)!=NULL)
  {
   sscanf(buf,"%s %s %s",word1,word2,word3);

   while(c!='\0'){
    str1[i]=word1[i]; i++;
    c=word1[i];
   }
    str1[i]=' '; str1[i+1]='\0';

    j=0; d=word2[j];
    while(d!='\0'){
     str2[j]=word2[j]; j++;
     d=word2[j];
    }
    str2[j]='\0';
```

```
k=0; e=word3[k];
while(e!='\0'){
 str3[k]=word3[k]; k++;
 e=word3[k];
}
str3[k]='\0';

strcpy(word,str1);
strcat(word,str2);

for(l=0;l<lastid;l++){
 if(strcmp(word,ary[l])==0){
  a=count[l];  b=atoi(str3);
  count[l]=a+b;
  break;
 }
}

if(l==lastid){
 strcpy(ary[l],word);
 count[l]=atoi(str3);
 lastid++;
}
```

```
    word[0]='\0';
   }
   for(l=0;l<lastid;l++) printf("%s %d\n",ary[l],count[l]);
   if((r=fclose(f1))==-1) quit("ファイルが閉じれない。");
}

void quit(char *s)
{
 puts(s);
 exit(1);
}

/* ============================================================
 共起頻度の整理 (2)
============================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define LINESIZE 60
#define WORDSIZE 20
#define ARYSIZE 100000
```

```
void quit(char *);
int main(int argc,char *argv[])
{
  FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE
       word[LINESIZE],
      str1[WORDSIZE],str2[WORDSIZE],str3[WORDSIZE],ary[ARYSIZE][WOR
  int count[ARYSIZE];
  int r,i,j,k,l,lastid=0,a,b;

  if(argc!=2) quit("引数の数が違う。prog datafile\n");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けな
い。\n");

  while(fgets(buf,LINESIZE,f1)!=NULL)
  {
   sscanf(buf,"%s %s %s",word1,word2,word3);

   while(c!='\0'){
    str1[i]=word1[i]; i++;
    c=word1[i];
   }
    str1[i]=' '; str1[i+1]='\0';
```

```
j=0; d=word2[j];
while(d!='\0'){
 str2[j]=word2[j]; j++;
 d=word2[j];
}
str2[j]='\0';

k=0; e=word3[k];
while(e!='\0'){
 str3[k]=word3[k]; k++;
 e=word3[k];
}
str3[k]='\0';

strcpy(word,str1);
strcat(word,str2);

for(l=0;l<lastid;l++){
 if(strcmp(word,ary[l])==0){
  a=count[l];  b=atoi(str3);
  count[l]=a+b;
  break;
 }
```

```
    }

    if(l==lastid){
     strcpy(ary[l],word);
     count[l]=atoi(str3);
     lastid++;
    }
   word[0]='\0';
  }
  for(l=0;l<lastid;l++) printf("%s %d\n",ary[l],count[l]);
  if((r=fclose(f1))==-1) quit("ファイルが閉じれない。");
}

void quit(char *s)
{
 puts(s);
 exit(1);
}

/* ============================================================
共起頻度の整理 (3)
============================================================*/
#include<stdio.h>
```

```
#include<stdlib.h>
#include<string.h>

#define LINESIZE 60
#define WORDSIZE 20
#define ARYSIZE 100000

void quit(char *);
int main(int argc,char *argv[])
{
  FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE
       word[LINESIZE],
      str1[WORDSIZE],str2[WORDSIZE],str3[WORDSIZE],ary[ARYSIZE][WOR
  int count[WORDSIZE];
  int r,i,j,k,l,lastid=0,a,b;

  if(argc!=2) quit("引数の数が違う。prog datafile\n");
  if((f1=fopen(argv[1],"r"))==NULL) quit("ファイルが開けな
い。\n");

  while(fgets(buf,LINESIZE,f1)!=NULL)
  {
    sscanf(buf,"%s %s %s",word1,word2,word3);
```

```
    while(c!='\0'){
     str3[i]=word3[i]; i++;
     c=word3[i];
    }
    str3[i]='\0';

    a=atoi(str3);
    printf("a=%d\n",a);
   }

   if((r=fclose(f1))==-1) quit("ファイルが閉じれない。");
}

void quit(char *s)
{
 puts(s);
 exit(1);
}

/* ============================================================
k-means法のプログラム
============================================================*/
```

```
k.means <- function(data,class,maxloop=10){
 attach(data)
 x <- class
 gyou <- nrow(data)
 retsu <- ncol(data)
 rep <- matrix(,nr=x,nc=retsu)
 dis <- c(1:x)

  random.data <- data[sample(1:gyou),]
  random.matrix <- data.matrix(random.data)

 for(i in 1:x){
  rep[i,] <- as.matrix(random.matrix[i,])
 }

  for(loop in 1:maxloop ){
   group <- array(,c((gyou+1),retsu,x))

   for(i in 1:x){
    group[1,,i] <- as.matrix(rep[i,])
   }

   for( i in 1:gyou ){
```

```
    for( j in 1:x ){
      dis[j] <- sqrt(sum((data.matrix(data[i,])-rep[j,])^2))
    }
    if(min(dis)!=0){
      mindis.no <- which(dis[]==min(dis))
      group[(i+1),,(mindis.no)] <- as.matrix(data.matrix(data[i,]))
    }
   }

for( j in 1:x ){
 for( i in 1:retsu ){
  a <- group[,i,j]
  b <- a[which(a!="NA")]
  rep[j,i] <- sum(b)/length(b)
 }
}

 flag <- 0
 for(j in 1:x ){
  for(i in 1:retsu ){
   if(group[1,i,j]!=rep[j,i]){
    flag <- 1
   }
```

```
  }
 }

 if(flag==0) break
}

g <- group[2:(gyou+1),1,]
cluster <- c[1:gyou]
for(i in 1:gyou){
 d <- g[i,]
 if(length(which(d!="NA"))==0){
  for(j in 1:x){
    if(sum(data[i,]==rep[j,])==retsu){
      cluster[i] <- j
    }
  }
 }
 if(length(which(d!="NA"))>=1){
  cluster[i] <- which(g[i,]!="NA")
 }
}
 Lst <- list(centroid=rep[(1:x),],cluster1=group[(2:(gyou+1)),,(1:
 #Lst <- list(cluster=cluster)
```

```
  detach(data)
  Lst
}
/* ========================================================
922次元中での最適な次元を求めるプログラム
(*不備あり．次元を削除した際に基底の単語を正規化する
  箇所がない)
========================================================*/
#include<stdio.h>
#include<stdlib.h>

void soukanhikeisan(void);

#define W 25
#define K 922
#define LINE 10000
#define CLS 5
#define COM 10 /* combination 5C2=10 */

double data[W][K];
double cls[CLS][K];
double juushin[CLS][K];
double soukanhi[COM],mean_soukanhi;
```

```
double mean_whole,mean_class[CLS],souwa; /* mean_whole:全
体平均,mean_class：クラス別平均 */
int i,j,k,n,zigen=K,a;
double within_class[10],between_class[10]; /* クラス内分散,
クラス間分散 */
double max,max_com; /* max_com：その次元の最大相関比 */
int optimal_zigen,max_line,line; /* max_line:最大相関比と
なる列 */
int flag[K];  /*減らす次元用の判断配列*/

int main(int argc,char *argv[])
{
 FILE *fp;
 char buf[LINE],*p;
 for(i=0;i<K;i++)flag[i]=0;

 fp = fopen(argv[1],"r");

 fgets(buf,LINE,fp);  /*1行目は要らない*/

 for(i=0;i<W;i++){
  fgets(buf,LINE,fp);
  for(j=0;buf[j]!=' ';j++);
```

```
  for(   ;buf[j]==' ';j++);
  k= 0;
  while(k < K){
   p=(char *)(buf+j);
   data[i][k] = atof(p);
   k++;
   for( ;(buf[j]!=' ') && (buf[j]!='\n');j++);
   for( ;buf[j] == ' ';j++);
  }
 }

/*****クラス分け****/
  for(i=0;i<CLS;i++){
   for(k=0;k<5;k++){
    for(j=0;j<K;j++){
     cls[i][j]+=data[5*i+k][j]/5;
    }
   }
  }

  line=-1;
  soukanhikeisan();
  max=mean_soukanhi; optimal_zigen=zigen; /*922次元の時の
```

```
相関比をmaxに代入*/

 for(zigen=K-1;zigen>0;zigen--){

  for(line=0;line<K;line++){
   if(flag[line]==0){
     soukanhikeisan();
     max_com=mean_soukanhi; max_line=line;
     a=line;
     break;
   }
  }

  for(line=a+1;line<K;line++){

  /*921次元以下の相関比を求めるプログラム*/

   soukanhikeisan();

  /*次元の中の最適な組み合わせ判断*/
   if((max_com < mean_soukanhi) && flag[line]==0){
    max_com=mean_soukanhi; max_line=line;
   }
```

```
  }
  flag[max_line]=1;

 /*相関比が最大となる次元の判断*/
   printf("%d次元の時、相関比%lf……現在のTOPは%d次元\n",zigen,max_

  if( max < max_com){
   max=max_com; optimal_zigen=zigen;
  }
 }

 printf("最大値%lfをとる%d次元が最適次元となる。\n",max,optimal_zigen)
 fclose(fp);
}

/*相関比を求める関数*/
void soukanhikeisan(void)
{
/*mean_class  クラス別平均*/
 mean_class[0]='\0';
 for(i=0;i<CLS;i++){
  for(j=0;j<K;j++){
   if(flag[j]==0 && line!=j)
```

```
   mean_class[i]+=cls[i][j]/zigen;
 }
}

/*クラス間分散とクラス内分散を求める*/
n=0; between_class[0]='\0'; within_class[0]='\0'; mean_soukanhi=0.
for(i=0;i<CLS;i++){
 for(j=i+1;j<CLS;j++){

/*mean_wholeはcls[i] と cls[j] の全体の平均値*/
  mean_whole=(mean_class[i]+mean_class[j])/2;

/*クラス間分散の計算*/
  between_class[n]=(zigen*((mean_class[i]-mean_whole)*(mean_class[
  between_class[n]+=(zigen*((mean_class[j]-mean_whole)*(mean_class
  between_class[n]=between_class[n]/(zigen*2);

/*クラス内分散の計算*/
  for(k=0;k<zigen;k++){
   within_class[n]+=(cls[i][k]-mean_whole)*(cls[i][k]-mean_whole);
   within_class[n]+=(cls[j][k]-mean_whole)*(cls[j][k]-mean_whole);
  }
  within_class[n]=within_class[n]/(zigen*2);
```

```
/*相関比の計算*/
    soukanhi[n]=between_class[n]/within_class[n];
    mean_soukanhi+=soukanhi[n];

    n++;
   }
  }
 mean_soukanhi=mean_soukanhi/(double)COM;
 return;
}

/* ========================================================
最適な次元を求めるにあたって削除された基底の単語
========================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

void soukanhikeisan(void);
void quit(char *);

#define W 25
#define K 922
```

```
#define LINE 10000
#define CLS 5
#define COM 10 /* combination 5C2=10 */
#define WORDSIZE 100

double data[W][K];
double cls[CLS][K];
double juushin[CLS][K];
double soukanhi[COM],mean_soukanhi;
double mean_whole,mean_class[CLS],souwa; /* mean_whole:全
体平均,mean_class：クラス別平均 */
int i,j,k,n,zigen=K,a,r;
double within_class[10],between_class[10]; /* クラス内分散,
クラス間分散 */
double max,max_com; /* max_com：その次元の最大相関比 */
int optimal_zigen,max_line,line; /* max_line:最大相関比と
なる列 */
int flag[K],del[K],del_no=0;  /* flag:減らす次元用の判断配
列*/

int main(int argc,char *argv[])
{
 FILE *f1,*f2; /* f1：web.dat ,f2：key922.dat */
 char buf[LINE],buf2[WORDSIZE],*p;
```

```
char word[K][WORDSIZE],word1[K-optimal_zigen][WORDSIZE],c;
char sakugen;
for(i=0;i<K;i++) flag[i]=0;

f1 = fopen(argv[1],"r");
f2 = fopen(argv[2],"r");

fgets(buf,LINE,f1);  /*1行目は要らない*/

for(i=0;i<W;i++){
 fgets(buf,LINE,f1);
 for(j=0;buf[j]!=' ';j++);
 for(   ;buf[j]==' ';j++);
 k= 0;
 while(k < K){
  p=(char *)(buf+j);
  data[i][k] = atof(p);
  k++;
  for( ;(buf[j]!=' ') && (buf[j]!='\n');j++);
  for( ;buf[j] == ' ';j++);
 }
}
```

```
/*****クラス分け****/
  for(i=0;i<CLS;i++){
   for(k=0;k<5;k++){
    for(j=0;j<K;j++){
     cls[i][j]+=data[5*i+k][j]/5;
    }
   }
  }

  line=-1;
  soukanhikeisan();
  max=mean_soukanhi; optimal_zigen=zigen; /*922次元の時の
相関比をmaxに代入*/

 for(zigen=K-1;zigen>0;zigen--){

  for(line=0;line<K;line++){
   if(flag[line]==0){
     soukanhikeisan();
     max_com=mean_soukanhi; max_line=line;
     a=line;
     break;
   }
```

```
 }

 for(line=a+1;line<K;line++){

/*921次元以下の相関比を求めるプログラム*/

  soukanhikeisan();

 /*次元の中の最適な組み合わせ判断*/
  if((max_com < mean_soukanhi) && flag[line]==0){
   max_com=mean_soukanhi; max_line=line;
  }

 }
 flag[max_line]=1; del[del_no]=max_line; del_no++;

/*相関比が最大となる次元の判断*/

 if( max < max_com){
  max=max_com; optimal_zigen=zigen;
 }
}
```

```
printf("削減された次元 =>");
for(i=0;i<(K-optimal_zigen);i++) printf("%d ",del[i]);
printf("\n");

/* 922次元のデータを挿入 */
while(fgets(buf2,WORDSIZE,f2)!=NULL){
  strcpy(word[i],buf2);
  i++;
}
for(i=0;i<(K-optimal_zigen);i++){
 sakugen = del[i];
 j=0; c=word[sakugen][j];
  while(c!='\n'){
   word1[i][j]=word[sakugen][j]; j++;
   c=word[sakugen][j];
  }
 word1[i][j]='\0';
}
for(i=0;i<(K-optimal_zigen);i++) printf("%s\n",word1[i]);

fclose(f1);
fclose(f2);
}
```

```
/*相関比を求める関数*/
void soukanhikeisan(void)
{
/*mean_class　クラス別平均*/
 mean_class[0]='\0';
 for(i=0;i<CLS;i++){
  for(j=0;j<K;j++){
   if(flag[j]==0 && line!=j)
    mean_class[i]+=cls[i][j]/zigen;
  }
 }

 /*クラス間分散とクラス内分散を求める*/
 n=0; between_class[0]='\0'; within_class[0]='\0'; mean_soukanhi=0.
 for(i=0;i<CLS;i++){
  for(j=i+1;j<CLS;j++){

 /*mean_wholeはcls[i]とcls[j]の全体の平均値*/
   mean_whole=(mean_class[i]+mean_class[j])/2;

 /*クラス間分散の計算*/
   between_class[n]=(zigen*((mean_class[i]-mean_whole)*(mean_class|
   between_class[n]+=(zigen*((mean_class[j]-mean_whole)*(mean_class
```

```
    between_class[n]=between_class[n]/(zigen*2);

  /*クラス内分散の計算*/
    for(k=0;k<zigen;k++){
     within_class[n]+=(cls[i][k]-mean_whole)*(cls[i][k]-mean_whole);
     within_class[n]+=(cls[j][k]-mean_whole)*(cls[j][k]-mean_whole);
    }
    within_class[n]=within_class[n]/(zigen*2);

/*相関比の計算*/
    soukanhi[n]=between_class[n]/within_class[n];
    mean_soukanhi+=soukanhi[n];

    n++;
   }
  }
  mean_soukanhi=mean_soukanhi/(double)COM;
  return;
}

void quit(char *s)
{
  puts(s);
```

```
 exit(1);
}

/* ============================================================
各単語間の距離を測る
============================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

void soukanhikeisan(void);

#define W 25
#define K 922
#define LINE 10000
#define CLS 5
#define COM 300 /* combination 25C2=300 */

double data[W][K];
double dis[COM];

int i,j,k,n;
```

```
int main(int argc,char *argv[])
{
 FILE *fp;
 char buf[LINE],*p;

 fp = fopen(argv[1],"r");

 fgets(buf,LINE,fp);  /*1行目は要らない*/

 for(i=0;i<W;i++){
  fgets(buf,LINE,fp);
  for(j=0;buf[j]!=' ';j++);
  for(   ;buf[j]==' ';j++);
  k= 0;
  while(k < K){
   p=(char *)(buf+j);
   data[i][k] = atof(p);
   k++;
   for( ;(buf[j]!=' ') && (buf[j]!='\n');j++);
   for( ;buf[j] == ' ';j++);
  }
 }
```

```
 n=0;
 for(i=0;i<(W-1);i++){
  for(j=i+1;j<W;j++){
   for(k=0;k<K;k++){
    dis[n]+=(data[i][k]-data[j][k])*(data[i][k]-data[j][k]);
   }
   dis[n]=sqrt(dis[n]);
   n++;
  }
 }

 for(i=0;i<COM;i++) printf("%lf ",dis[i]);

 fclose(fp);
}

/* ========================================================
クラス内で単語間の距離が最大となる次元
========================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
```

```
#define W 25
#define K 384
#define LINE 10000
#define CLS 5
#define WORDSIZE 100

double data[W][K];
double dis,max_dis;
char data2[K][WORDSIZE],word[WORDSIZE],c;
int i,j,k,a,b,zigen,max_zigen,n;

int main(int argc,char *argv[])
{
 FILE *f1,*f2;
 char buf[LINE],buf2[LINE],*p;

 f1 = fopen(argv[1],"r");
 f2 = fopen(argv[2],"r");

 fgets(buf,LINE,f1);  /*1行目は要らない*/

 for(i=0;i<W;i++){
  fgets(buf,LINE,f1);
  for(j=0;buf[j]!=' ';j++);
```

```
 for(   ;buf[j]==' ';j++);
 k= 0;
 while(k < K){
  p=(char *)(buf+j);
  data[i][k] = atof(p);
  k++;
  for( ;(buf[j]!=' ') && (buf[j]!='\n');j++);
  for( ;buf[j] == ' ';j++);
 }
}

/* data2にkye384.datのデータを挿入 */
 n=0;
 while(fgets(buf2,LINE,f2)!=NULL)
 {
     sscanf(buf2,"%s",word);
     i=0; c=word[i];
     while(c!='\0'){
      data2[n][i]=word[i]; i++;
      c=word[i];
     }
     data2[n][i]='\0';
     n++;
```

```
}

/* for(i=0;i<K;i++) printf("%s\n",data2[i]); */

/* data1 と data5 */
i=0; a=0; b=4;
dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
dis=fabs(dis);
max_dis = dis; max_zigen=i;

for(i=i+1;i<K;i++){
 dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
 dis=fabs(dis);

 if(max_dis < dis){
   max_dis=dis; max_zigen=i;
 }
}
printf("data%d と data%d…\n",a+1,b+1);
printf("max_dis=%lf,max_zigen=%d,%s\n",max_dis,max_zigen,data2[ma
printf("\n");

/* data7 と data8 */
```

```
i=0; a=6; b=7;
dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
dis=fabs(dis);
max_dis = dis; max_zigen=i;

for(i=i+1;i<K;i++){
 dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
 dis=fabs(dis);

 if(max_dis < dis){
   max_dis=dis; max_zigen=i;
 }
}
 printf("data%dとdata%d…\n",a+1,b+1);
printf("max_dis=%lf,max_zigen=%d,%s\n",max_dis,max_zigen,data2[ma
printf("\n");

/* data12とdata13 */
i=0; a=11; b=12;
dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
dis=fabs(dis);
max_dis = dis; max_zigen=i;

for(i=i+1;i<K;i++){
```

```
 dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
 dis=fabs(dis);

 if(max_dis < dis){
   max_dis=dis; max_zigen=i;
 }
}
 printf("data%dとdata%d…\n",a+1,b+1);
printf("max_dis=%lf,max_zigen=%d,%s\n",max_dis,max_zigen,data2[ma
printf("\n");

/* data16とdata19 */
i=0; a=15; b=18;
dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
dis=fabs(dis);
max_dis = dis; max_zigen=i;

for(i=i+1;i<K;i++){
 dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
 dis=fabs(dis);

 if(max_dis < dis){
   max_dis=dis; max_zigen=i;
```

```
  }
 }
printf("data%d と data%d…\n",a+1,b+1);
 printf("max_dis=%lf,max_zigen=%d,%s\n",max_dis,max_zigen,data2[ma
 printf("\n");

/* data21 と data23 */
 i=0; a=20; b=22;
 dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
 dis=fabs(dis);
 max_dis = dis; max_zigen=i;

 for(i=i+1;i<K;i++){
  dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
  dis=fabs(dis);

  if(max_dis < dis){
    max_dis=dis; max_zigen=i;
  }
 }
printf("data%d と data%d…\n",a+1,b+1);
 printf("max_dis=%lf,max_zigen=%d,%s\n",max_dis,max_zigen,data2[ma
 printf("\n");
```

```
 fclose(f1); fclose(f2);
}
```

```
/* ========================================================
25単語中の「プードル」と「チワワ」の
                    各次元で最も差が小さい次元
========================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>

#define W 25
#define K 384
#define LINE 10000
#define CLS 5
#define WORDSIZE 100

double data[W][K];
double dis,min_dis;
char data2[K][WORDSIZE],word[WORDSIZE],c;
int i,j,k,a,b,zigen,min_zigen,n;

int main(int argc,char *argv[])
{
 FILE *f1,*f2;
```

```
char buf[LINE],buf2[LINE],*p;

f1 = fopen(argv[1],"r");
f2 = fopen(argv[2],"r");

fgets(buf,LINE,f1);  /*1行目は要らない*/

for(i=0;i<W;i++){
 fgets(buf,LINE,f1);
 for(j=0;buf[j]!=' ';j++);
 for(   ;buf[j]==' ';j++);
 k= 0;
 while(k < K){
  p=(char *)(buf+j);
  data[i][k] = atof(p);
  k++;
  for( ;(buf[j]!=' ') && (buf[j]!='\n');j++);
  for( ;buf[j] == ' ';j++);
 }
}

/* data2にkye384.datのデータを挿入 */
 n=0;
```

```
while(fgets(buf2,LINE,f2)!=NULL)
{
    sscanf(buf2,"%s",word);
    i=0; c=word[i];
    while(c!='\0'){
     data2[n][i]=word[i]; i++;
     c=word[i];
    }
    data2[n][i]='\0';
    n++;
}

/* for(i=0;i<K;i++) printf("%s\n",data2[i]); */

/* data1 と data2*/
i=0; a=0; b=1;
dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
dis=fabs(dis);
min_dis = dis; min_zigen=i;

for(i=i+1;i<K;i++){
 dis=(data[a][i]-data[b][i])*(data[a][i]-data[b][i]);
 dis=fabs(dis);
```

```
    if(min_dis > dis){
      min_dis=dis; min_zigen=i;
    }
   }
   printf("data%dとdata%d…\n",a+1,b+1);
   printf("min_dis=%lf,min_zigen=%d,%s\n",min_dis,min_zigen,data2[mi
   printf("\n");

 fclose(f1); fclose(f2);
}
```

関連図書

[1] M. Sasaki and H. Shinnou: "Automatic thesaurus construction using word clustering", PACLING-03, pp.55-62, 2003.

[2] 神嶌敏弘："データマイニング分野のクラスタリング手法(1) クラスタリングを使ってみよう! ", 人工知能学会誌, Vol.18, No.1, pp.59–65, 2003.

[3] 関口洋一, 山本和英："Web コーパスの提案", 情報処理学会自然言語処理研究会, NL-157-17, pp.123–130, 2003.

[4] 佐々木稔, 新納浩幸："単語クラスタリングの語義判別問題への応用", 情報処理学会自然言語処理研究会, NL-154-21, pp.145–152, 2003.