大規模索引語文書行列に対する潜在的意味インデキシング

執筆者：藤枝雅一

指導教官：新納浩幸

平成１５年３月３日

目次

図目次

表目次

第１章

序論

１．１はじめに

情報検索に用いられているベクトル空間モデルでは，一般に文書ベクトルと検索質問ベクトルとのコサイン尺度を取ることにより情報検索を行う．しかし，ベクトル空間モデルでは，例えば「茨城大学」というキーワードで検索しても，略語である「茨大」を含む文書を検索することはできない．この問題を解決するために，潜在的意味インデキシング（LSI）が提案されている．LSI は索引語文書行列に対して特異値分解を行い，文書ベクトルおよび検索質問ベクトルを低次元空間で表現する技術である．

しかし，実際に LSI を行う場合，特異値分解の対象の行列が非常に大規模なスパース行列となるので，通常の方法では特異値分解を行うことができない．ここでは，ランダムサンプリングと索引語の選別という手法を組み合わせて大規模な索引語文書行列に対する LSI を実現させる．

図１．１はベクトル空間モデルによる情報検索のイメージである．

１．２　本論文の構成

・第２章では情報検索する上でよく使用されているベクトル空間モデルについて説明する．
・第３章では本研究でのテーマとなる潜在的意味インデキシング（LSI）について説明する．
・第４章では特異値分解ツール SVDPACKC について説明する．
・第５章では LSI による実験と実験結果を説明する．
・第６章では考察を述べる．
・おわりにでは本研究のまとめを示す．
・プログラムリストでは，本研究で用いたプログラムを紹介する．

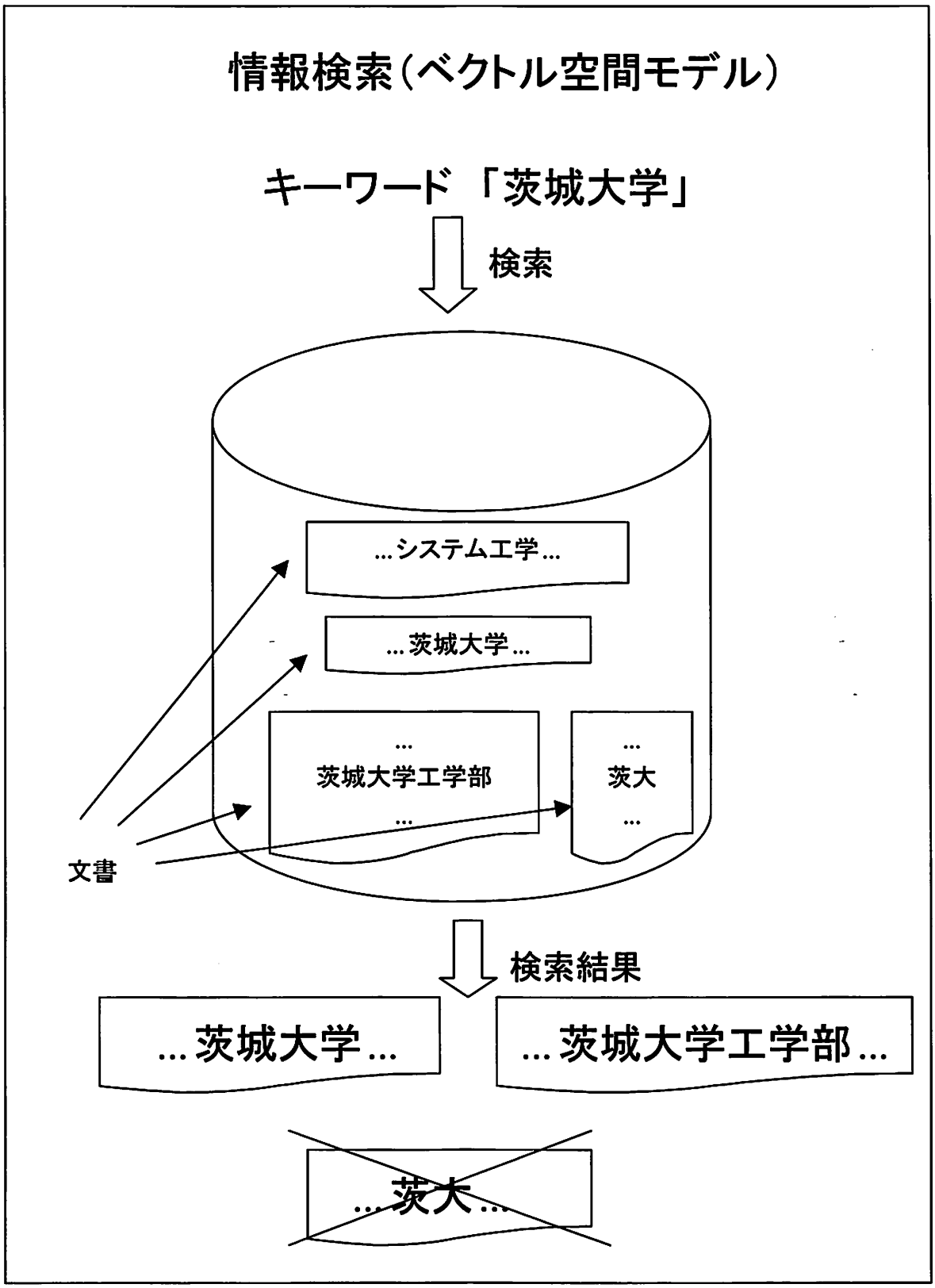

図 1．1　ベクトル空間モデルによる情報検索

第2章

ベクトル空間モデル

2．1　概要

ベクトル空間モデルは，１９７０年代にサルタンらによるSMARTシステムで採用されたモデルで，情報検索における代表的な検索モデルである．ベクトル空間モデルの最大の特徴は，文書および検索質問を多次元のベクトルである索引語ベクトルで表現し，ベクトル間の類似度を計算することにより文書の類似度検索を実現している点にある．ここでは，ベクトル空間モデルの仕組みについて説明する．

検索対象となるｎ個の文書を$d_1,d_2,\cdots,d_n$とし，n個の文書全体を通して全部でｍ個の索引語$w_1,w_2,\cdots,w_m$があるとする．このとき，文書 D_jは，次のようなベクトルで表現される．

$$d_j=\begin{bmatrix} d_{1j} \\ d_{2j} \\ \vdots \\ d_{mj} \end{bmatrix}$$

ここで，d_{ij}は索引語w_iの文書D_jにおける重みである．

また，文書集合全体では，次のような$m\times n$行列Dによって表現することができる．

$$D=\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}$$

これを索引語文書行列と呼ぶ．検索質問も文書と同様に索引語が要素とするベクトルで表現すると，検索質問文に含まれるw_iの重みをq_iとすると，次のように表すことができる．

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix}$$

２．２　ベクトル空間モデルの類似度計算

実際に文書検索を行うには，与えられた検索質問文と類似した文書を見つける必要がある．ベクトル空間モデルでは，これを検索質問ベクトルqと各文書ベクトルd_jの間の類似度を計算することによって行われる．類似度計算方法にはさまざまなものが提案されているが，文書検索においてよく用いられているのは，コサイン尺度や内積であり，次のような計算式である．

・ コサイン尺度

$$\cos(d_j, q) = \frac{d_j \cdot q}{\|d_j\| \ \|q\|} = \frac{\sum_{i=1}^{m} d_{ij} q_i}{\sqrt{\sum_{i=1}^{m} d_{ij}^2} \ \sqrt{\sum_{i=1}^{m} q_i^2}}$$

・ 内積

$$d_j \cdot q = \sum_{i=1}^{m} d_{ij} q_i$$

なお，ベクトルの長さが１に正規化されている場合には，コサイン尺度と内積は一致する．

2．3　ベクトル空間モデルによる検索例

では実際に例を挙げて説明する．この例では，索引語として 8 個の単語を用いる．また，文書dは6つ用意した．これらを以下の図 2.1 に示す．

索引語

w_1　茨城大学
w_2　茨大
w_3　私
w_4　友達
w_5　彼
w_6　茨城県
w_7　日立市
w_8　学生

文書

d_1　私は茨城大学の学生です．
d_2　友達は日立市に住んでいる．
d_3　茨城大学は茨城県日立市にある．
d_4　彼は日立市出身である．
d_5　茨城県は納豆が有名です．
d_6　茨城大学のことを茨大といいます．

図 2．1　索引語と文書の例

上記の索引語と文書から以下の索引語文書ベクトルを作成する．索引語の重みには索引語頻度を用いる．

索引語・文書行列　　$D =$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

検索キーワードを「茨城大学」「茨城県」とすると，検索質問ベクトルは以下のようになる．

検索質問ベクトル　　$q =$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

類似度計算にコサイン尺度を用いると，6 つの文書に対して以下のような図 2.2，表 2.2 となった．

類似度計算

$$\cos(d_1, q) = \frac{1}{\sqrt{1+1+1}\sqrt{1+1}} = 0.408$$

$$\cos(d_2, q) = \frac{0}{\sqrt{1+1}\sqrt{1+1}} = 0$$

$$\cos(d_3, q) = \frac{1+1}{\sqrt{1+1+1}\sqrt{1+1}} = 0.816$$

$$\cos(d_4, q) = \frac{0}{\sqrt{1+1}\sqrt{1+1}} = 0$$

$$\cos(d_5, q) = \frac{1}{\sqrt{1}\sqrt{1+1}} = 0.707$$

$$\cos(d_6, q) = \frac{1}{\sqrt{1+1}\sqrt{1+1}} = 0.5$$

図 2．2　コサイン尺度による類似度計算

検索順位	文書	類似度
1	d_3	0.816
2	d_5	0.707
3	d_6	0.5
4	d_1	0.408

表 2．2　検索結果

以上のことから，キーワード「茨城大学」「茨城県」を用いて検索を行うと，文書 d_3 が検索される（d_3 茨城大学は茨城県日立市にある．）．

第３章

潜在的意味インデキシング（LSI）

３．１　概要

情報検索においてベクトル空間モデルが主流である．しかし，ベクトル空間モデルでは，文書ベクトルの次元は索引語に対応するため，例えば「茨城大学」と「茨大」では意味は同じであるが，別次元ということになってしまう．よって，「茨城大学」を含む文書を検索しても「茨大」を含む文書は検索されず，また「茨大」で検索しても，「茨城大学」を含む文書は検索できないのである．

この問題を解決するために潜在的意味インデキシング（LSI）という手法が提案されている．LSI とは，高次元の空間にある文書ベクトルを低次元の空間へと射影することにより，検索精度の改善を図る技術である．ベクトル空間モデルでは別物と認識されていた「茨城大学」と「茨大」が，低次元の空間で１つの次元に縮退することが期待できる．そのため，「茨城大学」という検索質問によって「茨城大学」を含む文書ばかりでなく，「茨大」を含む文書も検索することが可能になる．図３．１では LSI による次元削減のイメージを示す．

LSI では，特異値分解と呼ばれる技術を用いて高次元ベクトルの次元削減を行う．次の項では特異値分解について説明する．

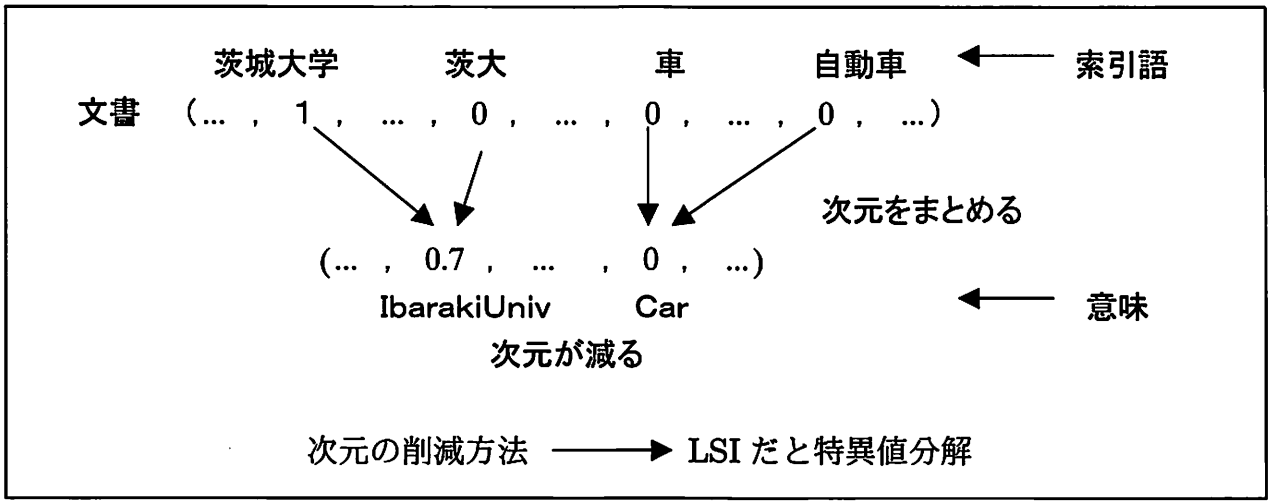

図 3．1　LSI のイメージ

3．2 特異値分解

特異値分解とは索引語文書行列 D を以下のような行列の積で表現することである．

$$D = U \Sigma V^T$$

行列 Uは $m \times m$直交行列，Vは $n \times n$直交行列，Σはm×n行列である．特に行列 Σは，$rank\ (D) = r$とすると，対角線上 r個の要素$\sigma_1, \sigma_2, \cdots, \sigma_r$が並んだ行列で$\sigma_i$は Dの特異値となっている．例として索引語文書行列 Dを

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

とすると，Dを特異値分解された時の U，Σ，V^Tを図 3．2 に示す．

$$U = \begin{bmatrix} 0.1628 & -0.0361 & 0.6739 & .02649 & 0.5774 & -0.1569 & -0.1316 & -0.2694 \\ 0.3988 & -0.1727 & -0.5208 & -0.1292 & 0.5774 & 0.4091 & -0.0658 & -0.1347 \\ 0.1379 & -0.3506 & -0.1267 & -0.2145 & -0.0000 & -0.5389 & 0.6024 & -0.3777 \\ 0.1379 & -0.3506 & -0.1267 & -0.2145 & -0.0000 & -0.5389 & -0.6682 & 0.2430 \\ 0.1827 & 0.0984 & -0.3712 & 0.8451 & 0.0000 & -0.2872 & 0.0658 & 0.1347 \\ 0.5616 & -0.2088 & 0.1531 & 0.1356 & -0.5774 & 0.2522 & -0.1974 & -0.4041 \\ 0.6131 & 0.6339 & 0.0862 & -0.2902 & -0.0000 & -0.2019 & 0.1316 & 0.2694 \\ 0.2276 & -0.5176 & 0.2717 & 0.0656 & -0.0000 & 0.2008 & 0.3290 & 0.6735 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 3.2577 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.1366 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.6608 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.2900 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6326 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.2922 & -0.3569 & 0.6616 & 0.3613 & 0.0000 & 0.4680 \\ 0.4494 & -0.7490 & -0.2103 & -0.2767 & -0.0000 & -0.3409 \\ 0.4167 & 0.2450 & -0.0794 & 0.5353 & -0.5774 & -0.3745 \\ 0.6712 & 0.4148 & -0.1175 & -0.4450 & -0.0000 & 0.4070 \\ 0.2382 & 0.2798 & 0.4577 & -0.0197 & 0.5774 & -0.5672 \\ 0.1785 & -0.0348 & -0.5371 & 0.5549 & 0.5774 & 0.1928 \end{bmatrix}$$

図３．２　索引語文書行列の特異値分解例

次に LSI で必要な知識を説明する．

・ 直交行列

直交行列とは，正方行列を A とすると，

$$A^TA=AA^T=I \quad \text{（単位ベクトル）}$$

となる行列のことである．

・行列のランク

行列 A のランク（rank）あるいは階数とは，A の列ベクトルの中から選びうる 1 次独立なベクトルの最大個数のことである．これはまた，A の行ベクトルの中から選びうる 1 次独立なベクトルの最大個数に等しい．行列 A のランクを rank(A)で表す．A がm×n行列ならば，

$$rank(A) \leq \min\{m,n\}$$

である．

・固有値，固有ベクトル

正方行列 A に対し，

$$Ax=\lambda x \quad (x\neq 0)$$

を満たす実数λを A の固有値，また x を固有値λに属する固有ベクトルという．x が固有ベクトルならば x のスカラー倍も固有ベクトルになるが，通常は $\|x\|=1$ となるものだけを考える．

一般に，行列 A の相違なる固有値に属する固有ベクトルは 1 次独立である．特に，A が $n\times n$ 対称行列である場合には，n 組の固有値と固有ベクトルを持ち，これら n 個の固有ベクトルは互いに直交する．

・特異値

階数 r の $m\times n$ 行列 A に対し，AA^T あるいは A^TA の 0 でない固有値 λ_i $(i=1,\ldots,r)$ の正の平方根

$$\mu_i=\sqrt{\lambda_i}\ (i=1,\ldots,r)$$

を A の特異値という．また，対応する固有ベクトル $u_1,\ldots,u_r$ および $v_1,\ldots,v_r$ を特異ベクトルという．

3．3　LSIによる検索

索引語文書行列 D の特異値分解により作成された行列 U の列ベクトルは，D の列ベクトルの張る正規直行形になっていて，左側にあるものほど重要度が高い．U の最初の k 個の列ベクトルから構成される $m\times k$ の行列を U_k とおく．U_k を用いて m 次元の文書ベクトル d，検索質問ベクトル q を，k 次元のベクトル U_k^Td, U_k^Tq に射影することができる．そして，U_k^Td, U_k^Tq のコサイン尺度を取ることにより d と q の類似度が計算される．図３．３に U の次元削減を示す．

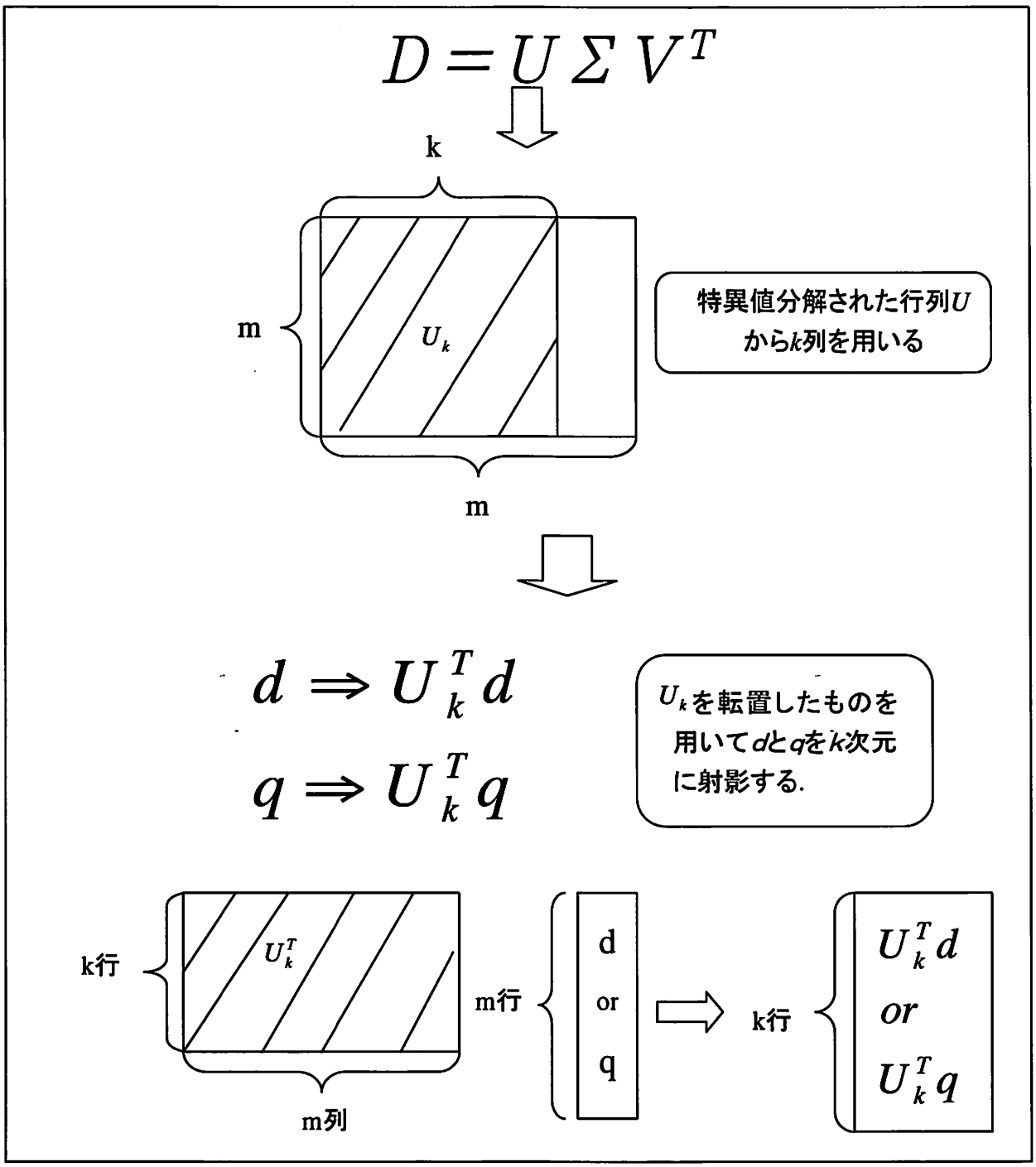
$D = U \Sigma V^T$
k
m
U_k
m
特異値分解された行列U
からk列を用いる
$d \Rightarrow U_k^T d$
$q \Rightarrow U_k^T q$
U_kを転置したものを
用いてdとqをk次元
に射影する.
k行
U_k^T
m列
m行
d
or
q
k行
$U_k^T d$
or
$U_k^T q$

図 3．3　Uの次元削減

特異値分解ツール SVDPACKC

４．１　概要

本研究では,索引語文書行列を特異値分解するのに SVDPACKC というツールを用いる．SVDPACKC はフリーで配布されており，以下の URL から入手できる．

http://www.netlib.org/svdpack/svdpackc.tgz

SVDPACKC には特異値分解を行う C 言語のプログラムが 8 つ入っている．この中で最も計算速度が優れているのは las2 と名付けられているランチョス法を使ったプログラムである．本研究ではこの las2 を利用する．実験では las2 を Windows2000 上の cygwin の環境で gcc を用いてコンパイルした．

利用方法

今回の実験では，特異値分解の結果だけを得たいので，las2 だけをコンパイルした．las2.c をコンパイルする前に，las2.c の中でコメントアウトされているマクロ定数 UNIX_CREAT を有効にしておく．

/* #define　UNIX_CREAT */　→　#define　UNIX_CREAT

これによって las2 による特異値分解の結果がファイルに保存される．次に，las.h のマクロ定数 LMTNW，NMAX，NZMAX の値を適当に調整する．また，las2.c の中では random 関数が事前に用意してあるため，stdlib.h で定義されている random 関数と競合する場合がある．ここでは，las2.c の中の stdlib.h を include しないことにした．

#include <stdlib.h>　→　/*　#include <stdlib.h>　*/

las2 は内部で 2 つのファイルを読み込む．1 つは特異値分解を行うための行列が記述された matrix というファイルで，もう 1 つはパラメータを記述したファイル lap2 である．この 2 つのファイルを用いることでから las2 を実行することができる．

配布されたキットには,サンプルの行列が belladit.Z という名前のファイルが提供されている．このファイルから，以下のコマンドにより，matrix ファイルを作成し，las2 を実行してみる．lap2 はサンプル用にキット内に用意されている．

zcat belladit.Z > matrix

実行方法は以下のとおりである．

./las2

実行結果は lav2 と lao2 というファイルに保存される．lav2 には特異値分解した時の行列 Uや Vの配列が保存される．ただし，バイナリファイルなので直接見ることはできない．lao2 には特異値分解した時の行列 Σ，つまり特異地の列や行列の大きさ，実行時間等が保存される．これはテキストファイルなので中身を確認できる．

4．3　matrix ファイルの記述方法

特異値分解の対象となる行列 D は，ハーウェル・ボーイング形式(Harwell-Boeing format)と呼ばれる列方向の圧縮形式を用いて matrix ファイルに記述する．これはスパース行列を少ない記述量で表現することができる．

注意として行列 D の大きさを $m \times n$ とした場合，SVDPACKC では $m \geqq n$ を仮定しているため，特異値分解したい行列 D の列数のほうが行数よりも大きい場合には，行列 D の転置行列 D^T に対して特異値分解を行う必要がある．この場合，行列 Uは行列 Vと置き換える必要もある．

では実際に例を挙げて matrix ファイルの記述方法を説明する．例として以下のような行列 D を考える．

$$D = \begin{bmatrix} 1.0 & 0 & 0 & 1.5 \\ 0 & 0 & 2.0 & 0 \\ 0 & 1.3 & 0 & 1.2 \\ 0 & 0 & 0 & 1.0 \\ 1.7 & 1.1 & 0 & 0 \\ 0 & 2.1 & 1.3 & 0 \\ 1.1 & 0 & 0 & 0 \end{bmatrix}$$

この行列に対して,1 列目から順に非ゼロ要素を取り出し，以下に示す．

データ番号	1	2	3	4	5	6	7	8	9	10	11
位置	(1,1)	(5,1)	(7,1)	(3,2)	(5,2)	(6,2)	(2,3)	(6,3)	(1,4)	(3,4)	(4,4)
値	1.0	1.7	1.1	1.3	1.1	2.1	2.0	1.3	1.5	1.2	1.0

次に上の表から行の部分だけを取り出す．

データ番号	1	2	3	4	5	6	7	8	9	10	11
位置	(1,1)	(5,1)	(7,1)	(3,2)	(5,2)	(6,2)	(2,3)	(6,3)	(1,4)	(3,4)	(4,4)
値	1.0	1.7	1.1	1.3	1.1	2.1	2.0	1.3	1.5	1.2	1.0
行位置	1	5	7	3	5	6	2	6	1	3	4

次に各列の最初の非ゼロの要素のデータ番号を列ポインタに記述する．上記の行列だと 1 列目の最初の非ゼロの要素は (1,1) の 1.0 であり,これに対するデータ番号は 1 である．次に 2 列目は，最初の要素は (3,2) の 1.3 でありこれに対するデータ番号は 4 である．このように各列の最初の非ゼロのデータ番号を記述したものが列ポインタである．よって，列ポインタの要素数は行列 D の列数となる．

データ番号	1	2	3	4	5	6	7	8	9	10	11
位置	(1,1)	(5,1)	(7,1)	(3,2)	(5,2)	(6,2)	(2,3)	(6,3)	(1,4)	(3,4)	(4,4)
値	1.0	1.7	1.1	1.3	1.1	2.1	2.0	1.3	1.5	1.2	1.0
行位置	1	5	7	3	5	6	2	6	1	3	4
列ポインタ	1	4	7	9							

ハーウェル・ボーイング形式とは，行列に対して，以下のような表を作り，列ポインタ，行位置，値を記述した形式である．こればスパース行列を圧縮した表現となる．

matrix では 4 行目以降に，列ポインタ，行位置，値が記述されている．列ポインタについては，最後の非ゼロ要素のデータ番号に 1 を足したものが付け加えられることに注意する．上記の例では以下のような形式になる．

```
1   4   7   9
1   5   7   3   5   6   2   6   1   3   4
1.0   1.7   1.1   1.3   1.1   2.1   2.0   1.3   1.5   1.2   1.0
```

matrix の最初の 4 行は，行列に関するその他の情報が記述されている．3 行目以外は意味はなく，1 行目はデータの名前であり，サンプルファイルの bellasit.Z を参考にして適当につければよい．2 行目，4 行目も意味はなく，belladit.Z のとおりに記述する．3 行目は以下のように 5 つのデータを空白で区切って記述する．

```
rra          7     4     11     0
```

この行の 1 列目と 5 列目は上記のとおり記述すればよい。2 列目は行列 D の行数，3 列目は列数，4 行目は非ゼロの要素の総数を記述する．以上のことから，matrix ファイルは以下のようになる．

```
Jikken Data                                    jikken
Transposed
rra          7     4     11     0
1   4   7   9
1   5   7   3   5   6   2   6   1   3   4
1.0   1.7   1.1   1.3   1.1   2.1   2.0   1.3   1.5   1.2   1.0
```

4．4　lap2 ファイルの記述方法

lap2 ファイルは las2 で使われるパラメータが 1 行で記述されている．配布されているキットの中には以下のような中身の lap2 ファイルが入っている．

'belladit' 44 10 -1.0e-30 TRUE 1.0e-6 0

1列目は matrix ファイルの1行目に記述したデータの名前だが，matrix と lap2 とのデータ名の一致は検査しないため，特に意味はない．2列目と3列目はどちらも行列の列数 nを設定する．4列目，5列目，6列目の数値は，特に変更する必要はない．7列目は特異値分解の結果の Uや Vの行列をファイルの保存するかどうかの指定で，TURE にしておけばファイルに保存される．8列目は特に意味はない．

出力結果

las2 による特異値分解の結果は，lav2 と lao2 というファイルに保存される．Lao2 はテキストファイルで，内容を見るのは容易にできる．重要な部分はファイルの下方に記述されている固有値の列である．また，lao2 では，個誘致は値の小さい順に出力されている．固有値は大きいほうが重要な意味を持つため，ファイルの下方にかかれている特異値ほど重要である．

lav2 はバイナリファイルであり，las2 のソースを見て出力形式を確認すれば，特異値分解の結果の Uや Vを得ることが可能となる．結局，lav2 をテキストファイル形式に変換するプログラムが必要になってしまう．

第5章

実験

実験の目的

実験では LSI による検索を行う．LSI を行うには，索引語文書行列を特異値分解する必要がある．特異値分解をするために，SVDPACKC というツールを使うのだが，索引語文書行列が大規模なためそのままでは特異値分解できない．そこで，ランダムサンプリングと索引語の選別という手法を用いて，索引語文書行列を次元削減し，特異値分解することが実験の目的である．

また，次元削減した索引語文書行列による LSI がうまくいっているか，ベクトル空間モデルによる検索と比較してみる．

図 5．1 には LSI の通常の方法と，本研究で用いた方法を示す．

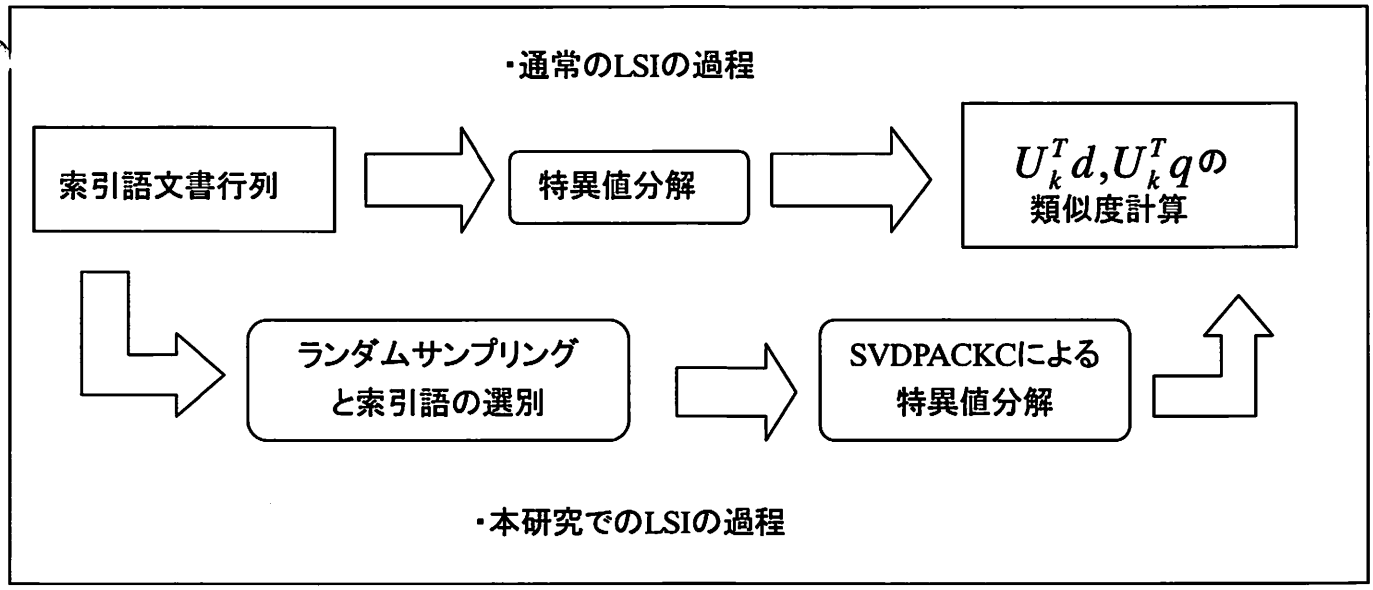

図 5．1　通常の LSI と本研究の LSI の流れ

５．２　実験の流れ

以下に実験の流れを示す．項目ごとに，どのようにして実験を行ったかも説明する．ベクトル空間モデルによる検索は，同研究室の紺野憲一氏の作成したプログラムから結果を出した．

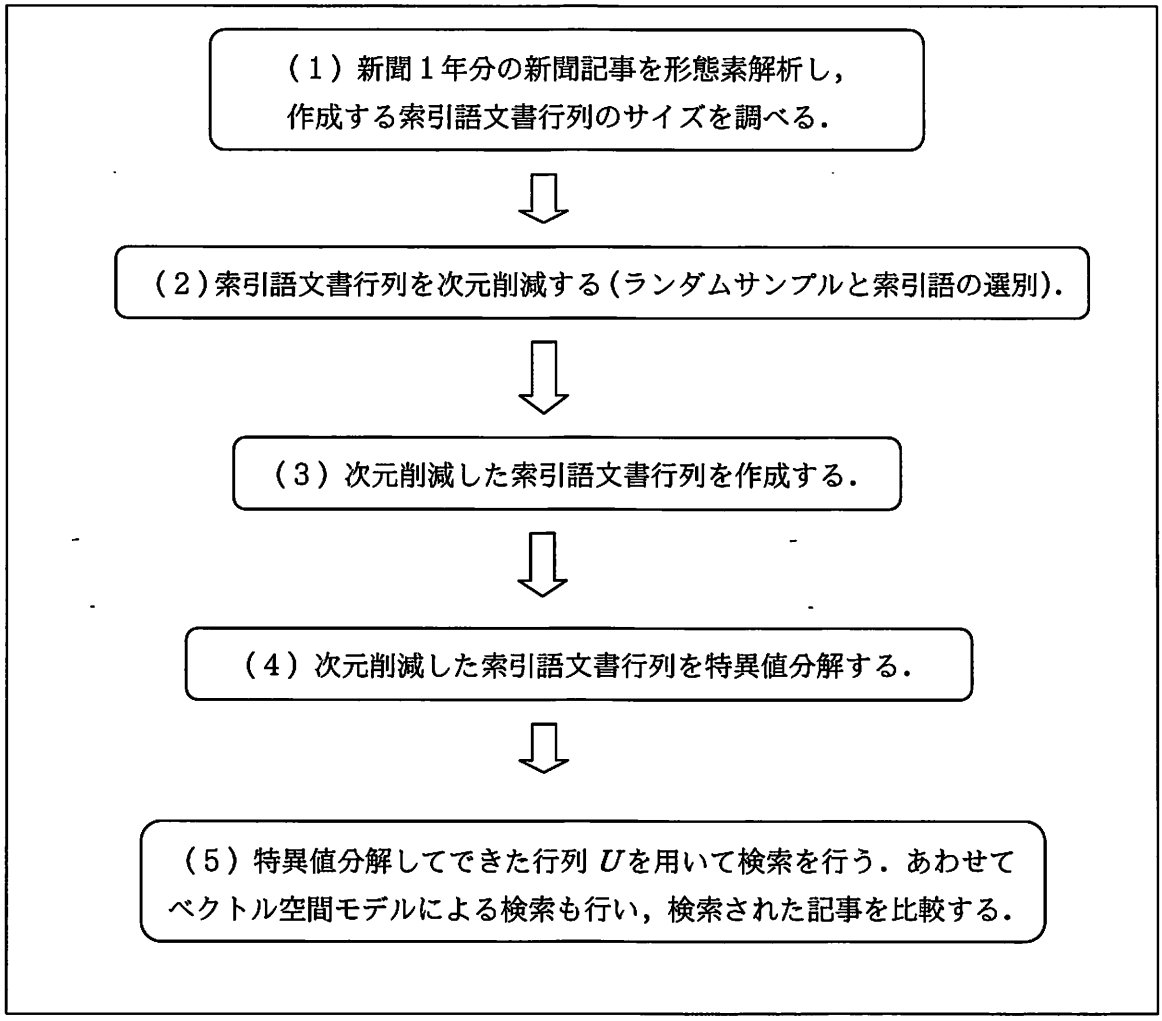

図５．２．１　実験の流れ

（１）新聞１年分の新聞記事を形態素解析し，索引語文書行列のサイズを調べる．

１年分の新聞記事を，「茶筌」というツールを用いて形態素解析する．図５．２．１に記事を形態素解析したときの中身を示す．

形態素解析前の記事

数年前、旅行先で何気なく英訳本を手にしたのが老子との出会いという。

形態素解析後

(見出し)(読み) (基本形) (品詞)

950101001.2 未知語 950101001.2 名詞-サ変接続 17 * 0 * 0
数 スウ 数 名詞-数 19 * 0 * 0
年 ネン 年 名詞-接尾-助数詞 35 * 0 * 0
前 マエ 前 名詞-副詞可能 16 * 0 * 0
、 、 、 記号-読点 79 * 0 * 0
旅行 リョコウ 旅行 名詞-サ変接続 17 * 0 * 0
先 サキ 先 名詞-接尾-一般 28 * 0 * 0
で デ で 助詞-格助詞-一般 61 * 0 * 0
何気なく ナニゲナク 何気ない 形容詞-自立 51 形容詞・アウオ段 50 連用テ接続 6
英訳 エイヤク 英訳 名詞-サ変接続 17 * 0 * 0
本 ホン 本 名詞-一般 2 * 0 * 0
を ヲ を 助詞-格助詞-一般 61 * 0 * 0
手 テ 手 名詞-一般 2 * 0 * 0
に ニ に 助詞-格助詞-一般 61 * 0 * 0
し シ する 動詞-自立 47 サ変・スル 3 連用形 7
た タ た 助動詞 74 特殊・タ 54 基本形 1
の ノ の 名詞-非自立-一般 21 * 0 * 0
が ガ が 助詞-格助詞-一般 61 * 0 * 0
老子 ロウシ 老子 名詞-固有名詞-人名-一般 6 * 0 * 0
と ト と 助詞-格助詞-一般 61 * 0 * 0
の ノ の 助詞-連体化 71 * 0 * 0
出会い デアイ 出会い 名詞-一般 2 * 0 * 0
と ト と 助詞-格助詞-引用 62 * 0 * 0
いう イウ いう 動詞-自立 47 五段・ワ行促音便 21 基本形 1
。 。 。 記号-句点 78 * 0 * 0
EOS

図 5．2．1 形態素解析による出力結果

索引語として名詞，動詞，形容詞，副詞を抽出したところ，計 110,000 種類になった．索引語には，名詞を用いることにした（約 90,000 種類）また，記事は約 67,000 であったため，90,000 行×67,000 列の索引語文書行列となることが分かった．

（2）索引語語文書行列を次元削減する．（ランダムサンプルと索引語の選別）

索引語文書行列を特異値分解するのに SVDPACKC を用いるのだが，特異値分解できる行列には制限がある．中間発表で SVDPACKC での特異値分解できる行列のサイズを調べたところ，特異値分解できる行列のサイズは 20,000 行×2,000 列と見積もった．このサイズに索引語文書行列を次元削減する．

・ランダムサンプリング

ランダムサンプリングによって行を削減する．67,000 の記事からランダムに 1,800 記事抽出した（各月 150 記事）．

抽出方法は bash スクリプトを用いた．各月のフォルダの中の記事数をカウントした．それと同時に 1 つのファイルに記事の名前と番号を 1 記事 1 行で入力する．そして記事数を 150 で割った値 x と記事に付けた番号 1，1+x，1+2x，…（x=1～180）と一致した記事を抽出した．例えば 600 記事から 150 記事抽出する場合，600 を 150 で割った値は 4 だから，記事を 3 つ飛ばしに 150 記事抽出することになる．（プログラム名 randomsumple）

・索引語の抽出

索引語の選別によって，列を削減する．索引語として名詞を用いた（約 90,000 種類）．そこから，使用頻度 25 以上のものを用いることにした（19,639 種類）．

以上のことから，90,000 行×67,000 列であった索引語文書行列を 19,639×1,800 に次元削減する．

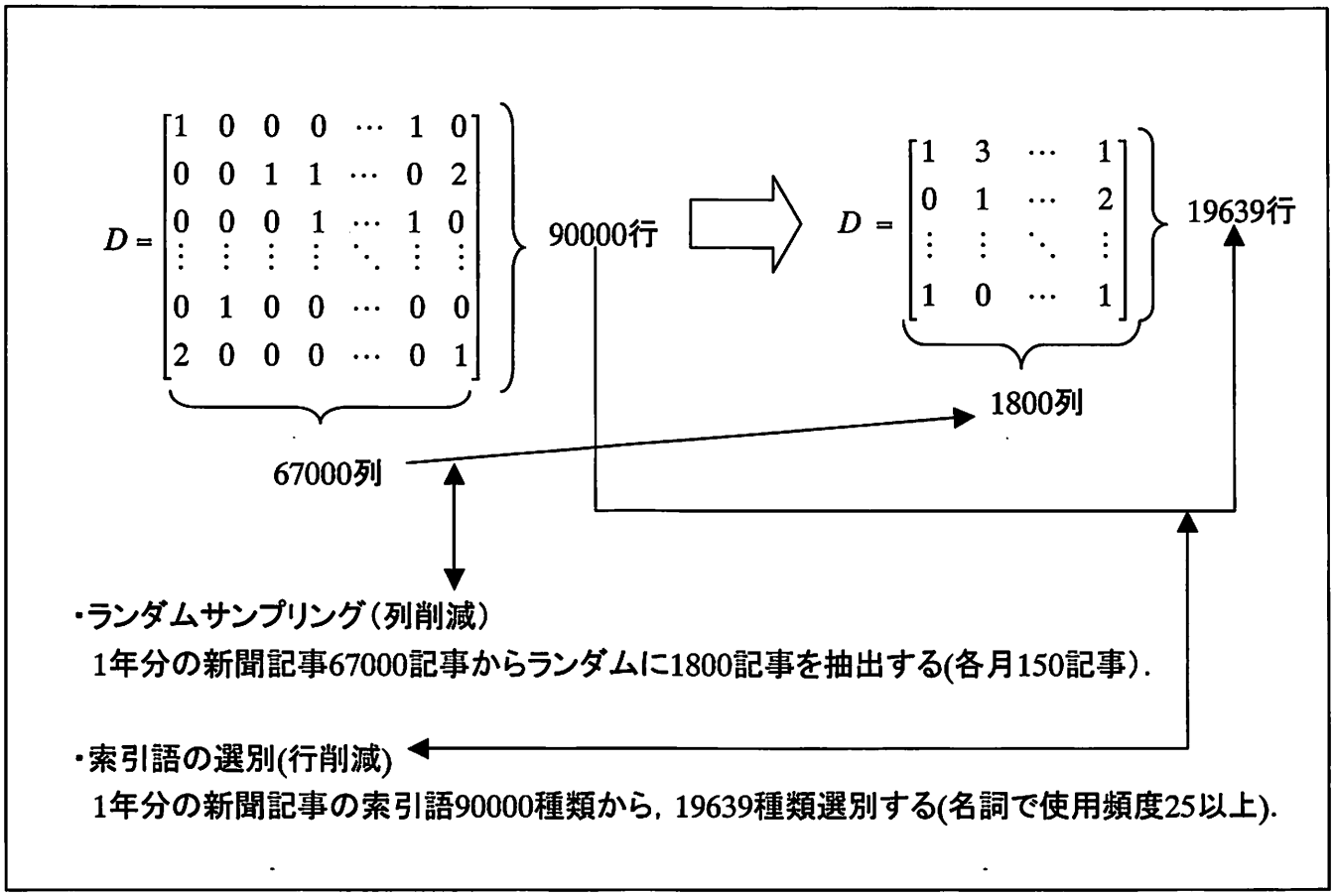

図５．２．２　特異値分解前の低次元化手法

（３）次元削減した索引語文書行列を作成する．

ここでは，索引語文書行列を作成するための行程を説明する．

・形態素解析した記事の使用頻度をカウントする．

Ｃプログラムを用いてカウントした．１行１単語で空白を置いて使用頻度も示しておく．

・使用する索引語に次元を付ける．(w_1～w_{19639})

bash シェルスクリプトを用いて，索引語の使用頻度が小さい順に次元をつけた．

上記の2つのファイルからGDBMを用いて，記事から索引語として使用されている単語と記事中の使用頻度を抽出する．GDBMとはハッシュデータベースを作成するライブラリで，key（今回は90,000種類の索引語）を鍵にして，記事からcon（索引語として使用されている単語）を抽出する．以下に索引語の抽出の図5．2．3を示す．

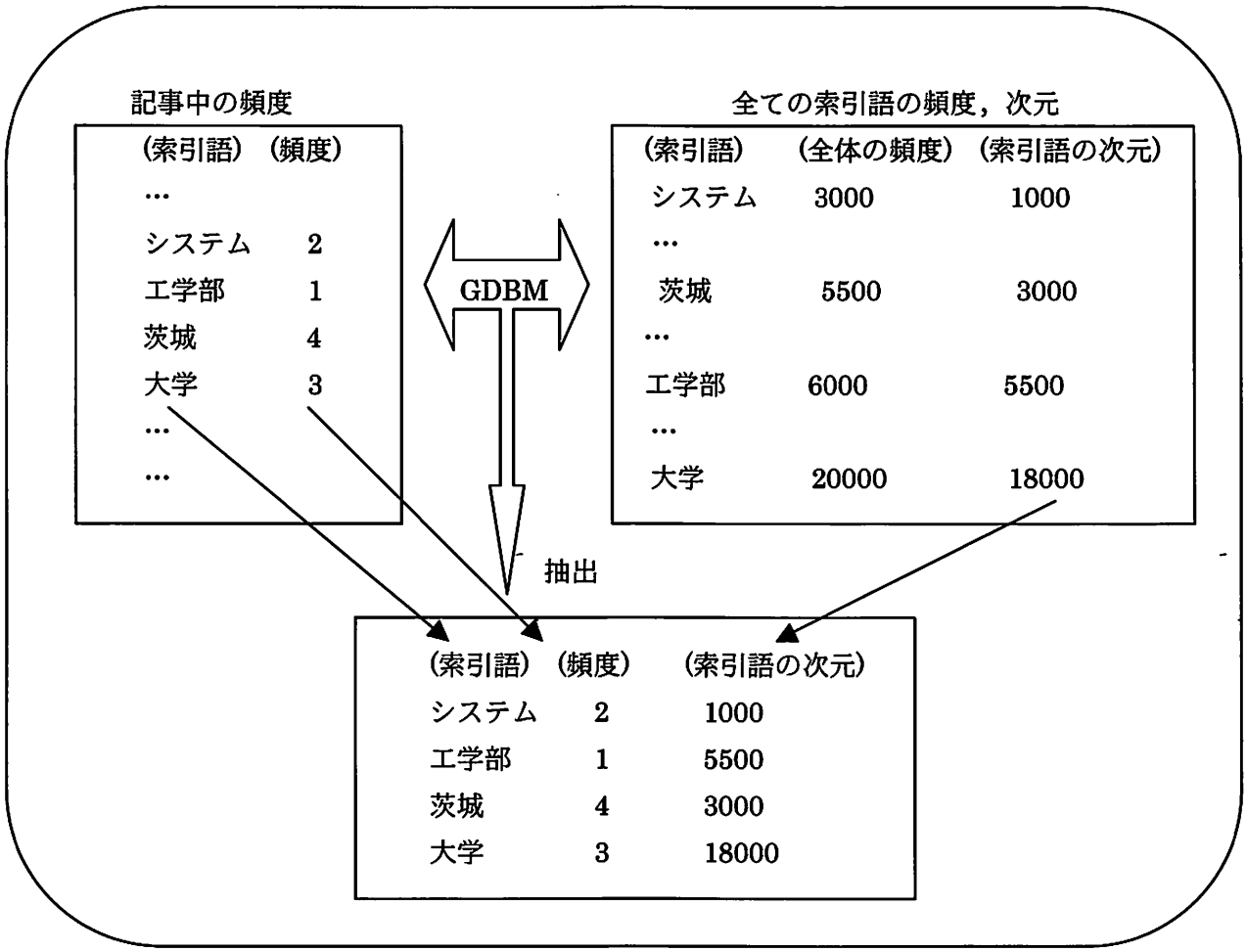

図5．2．3　GDBMによる索引語の次元，頻度抽出

実験では，mkdbmというデータベース作成プログラムから，使用頻度25以上の索引語の索引語名，索引語の次元が書かれているファイル（図5．2．3右の図）をデータベース化する．そして，kendbmという作成されたハッシュデータベースから検索するプログラムを用いて，索引語の頻度，次元を抽出する（図5．2．3下の図）．

・GDBMによって抽出した索引語の頻度,次元から索引語文書行列を作成する．
　1つの記事を1行として，以下ののように示す．「次元：頻度」と「次元：頻度」の間は空白で区切られている．

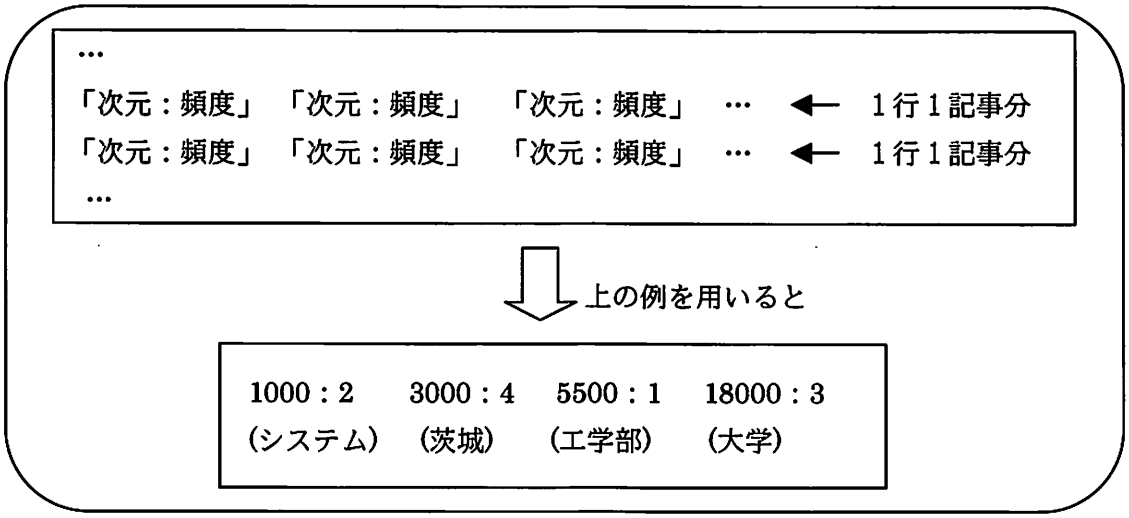

図 5．2．4　索引語文書行列の記述方法

以上のことから，1行1文書で索引語の次元，頻度を数値化した索引語文書行列を作成した．ここまでの行程を一括で処理するプログラムを，プログラムリストで紹介する．プログラム名は transpro である．

（4）次元削減にした索引語文書行列を特異値分解する．
　4章で紹介した，SVDPACKC を用いて特異値分解を行った．（1）～（3）の行程で作成した索引語文書行列から，matrix ファイルを作成し，特異値分解を行う．ここでは，低次元化した索引語文書行列から matrix ファイルを作成する手順を紹介する．

　bash シェルスクリプトを用いて，数値化した索引語文書行列を1行に1つの索引語の次元，頻度，行番号を出力する．例として今回は以下のような行列を用いて説明する．

11:5 17:2 37:5 49:2
7:3 47:4 58:1 60:2
1:4 12:5 19:3 25:5
29:2 35:5 45:2 50:4

表 5．2．1　数値化した索引語文書行列の例

・bash シェルスクリプトを用いて，上記のファイルから 1 行に 1 つの索引語，次元，頻度をを出力する．

1	11	5
1	17	2
1	37	5
1	49	2
1	7	3
2	47	4
2	58	1
2	60	2
3	1	4
3	12	5
3	19	3
3	25	5
4	35	5
4	45	2
4	50	4
(行番号)	(次元)	(頻度)

表 5．2．2　行番号，次元，頻度抽出

表５．２．２のファイルから bash シェルスクリプトを用いて，matrix ファイルに必要な値，行位置，非ゼロ数，を抽出する．また，行数，列数，列ポインタは C プログラムを用いて抽出する．これらの matrix ファイル作成に必要なデータを抽出するプログラムを一括で処理するプログラムも作成した。プログラム名は matrixdata である．Windows2000 上の cygwin の環境で
実行する．

プログラム matrixdata を使って matrix 作成に必要な情報を抽出する．matrix ファイル作成プログラム名は mkmatrix である．

（5）特異値分解してできた行列 U を用いて検索を行う．あわせてベクトル空間モデルによる検索も行い，検索された記事を比較する．U を用いたコサイン尺度による検索にツールを用いた．

5．3　実験結果

索引語文書行列を 19639×1800 行列に低次元化することで，SVDPACKC で特異値分解することが出来た．実験を行った環境と，実行結果を以下に示す．

CPU	Pentium-4 1.5GHz
メモリ	512 MB
使用メモリ	362MB
実行時間	1080秒

表5．3．1　実験環境と特異値分解で使用されたメモリと時間

次に，LSI とベクトル空間モデルによる検索結果の比較を示す．以下の図 5．3．1 は LSI とベクトル空間モデルで検索された記事の ID を上位 5 件示している．同様に，図 5．3．2 は LSI とベクトル空間モデルで検索された記事の ID を上位 10 件示している．また，検索された記事上位 10 件の類似度も表 5．3 で示す

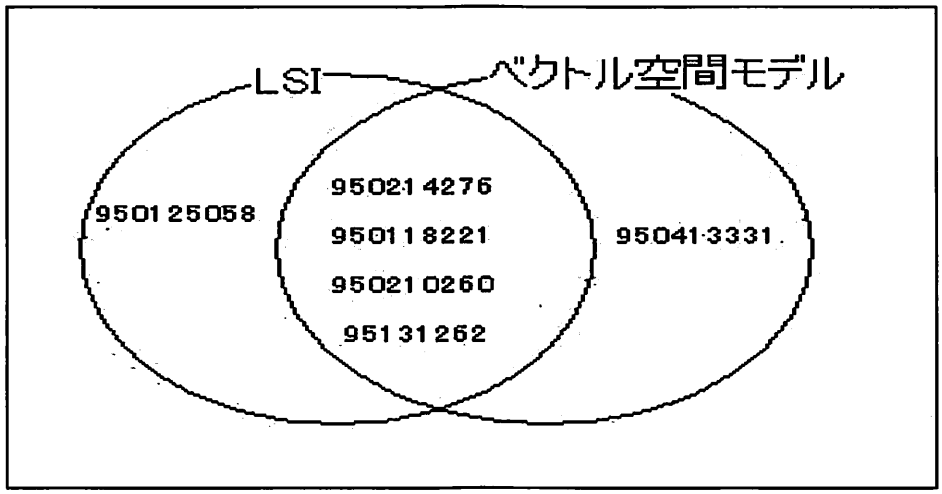

図 5．3．1　LSI とベクトル空間モデルによる検索結果（上位 5 件）

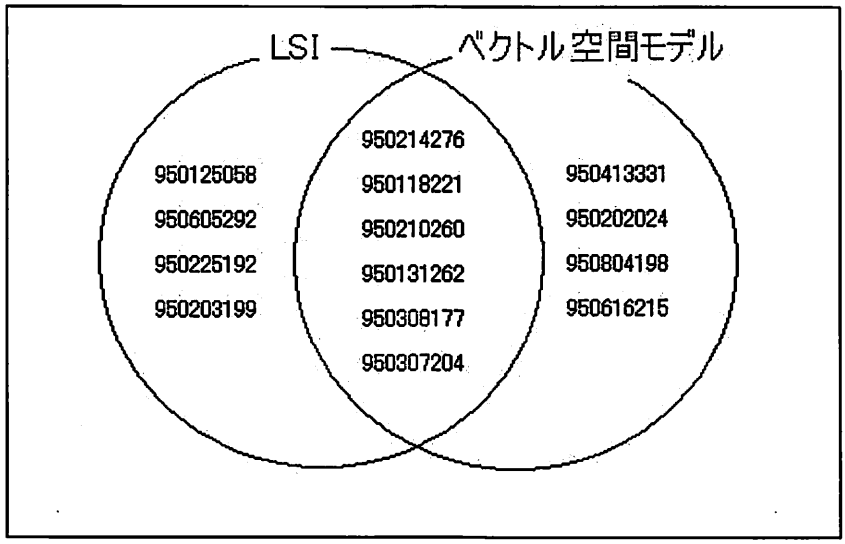

図 5．3．2　LSI とベクトル空間モデルによる検索結果（上位 10 件）

	LSI		ベクトル空間モデル	
検索順位	記事	類似度	記事	類似度
1	950214276	0.519758	950214276	0.412082
2	950118221	0.501067	950210260	0.381246
3	950210260	0.434821	950118221	0.344124
4	950131262	0.339593	950131262	0.3371
5	950125058	0.326030	950413331	0.320844
6	950605292	0.320630	950202024	0.306186
7	950225192	0.313648	950804198	0.293568
8	950308177	0.313055	950616215	0.288675
9	950307204	0.311856	950308177	0.288675
10	950203199	0.309108	950307204	0.288675

表 5．3．2　LSI とベクトル空間モデルで検索された記事（上位 10 件）

最後に，LSI とベクトル空間モデルで検索された記事の内容を示す．

・LSI

950214276

政府は十四日の閣議で、阪神大震災について政府が用いる一般呼称を「阪神・淡路大震災」とすることを決めた。一般に阪神大震災の呼び方が浸透していることを考慮した。

950118221
近畿通産局は阪神大震災による被害をすみやかに復旧するため、同局内に災害対策本部を設置した。

950210260
政府は十日の閣僚懇談会で、阪神大震災の呼称について「兵庫県南部地震」との正式呼称とは別に「阪神・淡路」を冒頭に付けた一般呼称を定めることを決めた。「阪神・淡路大震災」か「阪神・淡路震災」になる見通しだ。・・・

950131262
阪神大震災で断水していた被災地域の水道の復旧率が五〇％を超えたことが厚生省の三十一日までの調査で分かった。復旧が遅れていた西宮市、芦屋市も二月末までに仮復旧の見込みが立つなど、予想以上に回復が進んでいるという。

950125058
ドイツから阪神大震災支援として二十五日、粉ミルク千キロが関西国際空港に到着する。

950605292
長い間がれきの下に閉じ込められたために筋肉などが圧迫され、血液が全身に回らなくなることによってじん臓機能が急激に低下する症状。外傷がなくても重症になる恐れがあるとして阪神大震災でも問題となった。

950308177
大阪ガスが６日現在でまとめた市区ごとの復旧率は次の通り。

950307204
大阪ガスが６日現在でまとめた市区ごとの復旧率は次の通り。

950203199
【テレビ】阪神大震災情報（放送範囲は近畿二府四県と徳島県）。

・ベクトル空間モデル

950214276
政府は十四日の閣議で、阪神大震災について政府が用いる一般呼称を「阪神・淡路大震災」とすることを決めた。一般に阪神大震災の呼び方が浸透していることを考慮した。

950210260
政府は十日の閣僚懇談会で、阪神大震災の呼称について「兵庫県南部地震」との正式呼称とは別に「阪神・淡路」を冒頭に付けた一般呼称を定めることを決めた。「阪神・淡路大震災」か「阪神・淡路震災」になる見通しだ。・・・

950118221
近畿通産局は阪神大震災による被害をすみやかに復旧するため、同局内に災害対策本部を設置した。

950131262
阪神大震災で断水していた被災地域の水道の復旧率が五〇％を超えたことが厚生省の三十一日までの調査で分かった。復旧が遅れていた西宮市、芦屋市も二月末までに仮復旧の見込みが立つなど、予想以上に回復が進んでいるという。

950413331
毎日新聞社、毎日放送は阪神大震災の被害の実態を検証し、その教訓を今後のまちづくや市民生活に生かすため、第一線の地震研究者による防災講演会「市民の暮らしと地震―阪神大震災を考える―」を開催します。・・・

950202024
大阪ガスの領木新一郎社長は一日記者会見、阪神大震災による同社の被害額は総額で約千九百億円との見通しを明らかにした。

950804198
・・・地震訓練どおりに復旧対象顧客をリストアップするのだが、段ボールは二百箱になる、運ぶのに困った――。大阪ガスはこのほど、阪神大震災後のガスの復旧活動を記録した「阪神大震災ガス復旧の軌跡」（変形Ａ４判、カラー）＝写真＝をまとめた。・・・復旧には、全国のガス事業者からの応援部隊を含める、九千七百人が従事。・・・

950616215
【ラジオ】「ネットワーク１・１７　阪神高速道路の復旧を考える」１７日１７時１０分―４４分。大震災で倒壊した阪神高速道路は、現在も復旧作業が進められている。しかし地域住民からは「阪神高速はもう要らない」との声もあがっている。・・・

950308177
大阪ガスが６日現在でまとめた市区ごとの復旧率は次の通り。

950308177
大阪ガスが６日現在でまとめた市区ごとの復旧率は次の通り。

図 5．3．1 を見ると分かるように，検索された上位５つの記事のうち４つが LSI とベクトル空間モデルで検索できたため，索引語文書行列の次元削減がうまくいき，LSI が行えたことが分かった．表 5．3 では，上位１位に検索された記事は，LSI もベクトル空間モデルも同じであったことが分かる．

また，LSI で検索されなかった記事 ID950413331，ベクトル空間モデルで検索されなかった記事 ID950125058 の中身を見ると，内容は少しずれているが阪神大震災について書かれていた．

第6章

考察

今回の実験では，索引語文書行列をランダムサンプリングと索引語の選別により，低次元にすることで特異値分解が実現できた．高次元な索引語文書行列でも，予め低次元化することで特異値分解が実現でき，LSIが可能となる．

LSIの結果もベクトル空間モデルの結果も，上位5の記事のうち4つが検索され，ほぼ納得のいく結果が得られたため，今回の実験では，ある程度妥当な次元削減が行えたと考えられる．

検索された文書を比較すると，内容に差はほとんどなかった．ただし，LSIでは検索された記事のサイズがやや小さく，ベクトル空間モデルの方が記事のサイズがやや大きいという傾向があったが，原因は分からなかった．

しかし，実験がうまくいったのには，今回キーワードにした索引語がすべて使用頻度25以上だったためとも考えられる．使用頻度の小さい索引語をキーワードに用いた実験の比較は今後の課題である．

おわりに

大規模な索引語文書行列を，ランダムサンプリングと索引語の選別によって低次元化し，SVDPACKC を用いて特異値分解することができた．

特異値分解によって得られた U も，低次元化する前の索引語文書ベクトルを用いたベクトル空間モデルとの比較で，妥当であることが確認できた．

今後は，低頻度の索引語を用いた実験，検索を行いたい．また，索引語文書行列を低次元化する他の手法を調べ，今回行った低次元化の手法と比較したい．

謝辞

本研究の遂行および論文作成において多大な御助言及び御指導を賜りました新納浩幸教官（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．

さらに，御指導を頂きましたシステム工学科計算機応用学講座の教官の方々，本研究を進めるにあたり御助言，ご協力を頂きました同研究室の阿部修也氏（茨城大学大学院理工学研究科システム学専攻），高橋篤史氏（茨城大学大学院理工学研究科システム学専攻），紺野憲一氏（茨城大学システム工学科４回生），山村一起氏（茨城大学システム工学科４回生）に深く感謝いたします．

参考文献

[1] 北研二，津田和彦，獅々堀正幹
「情報検索アルゴリズム」　共立出版　(2001)

[2] Michael Berry and Theresa Do and Gavin O'Brien Vijay Krishna and Sowmini Varadhan:SVDPACKC (Version 1.0) User's Guide
www.netlib.org/svdpack　(1993).

[3] 新納浩幸，佐々木稔：
SVDPACKC とその語義判別問題への利用，
自然言語処理,Vol.10, No.2, to appear (2003).

プログラムリスト

ランダムサンプリングによる記事抽出プログラム　randomsumple

```
function randomsumple {
mkdir file
for directory in *
do
  if [ -d $directory ]
  then
      cd $directory
      for file in *.txt
      do
        if [ -f $file ]
        then
          echo "$file" >> $directory.kizi
          awk '{print NR" "$1}' $directory.kizi > $directory.ban
          sort -n -r < $directory.ban > $directory.sort
          awk '{print $1-1+NR" "$2" "NR}' $directory.sort > $directory.hin
          awk -f aaa.awk $directory.hin >> $directory.out
        fi
      done
      cd ..
  fi
done
}
```

・データベース作成プログラム　mkdbm.c

```
#include <stdio.h>
#include <gdbm.h>
#include <fcntl.h>

/*
#define NDBM_KEY_MAX        256
#define NDBM_CON_MAX        1024
*/

main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  int   r,count;
  char line[1000],*p,*q;
  GDBM_FILE *mydb;
  datum   key,content;

  if(argc != 2) quit("引数の数が違う"); /* prog datafile */
  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイル 1 が開けない");
  mydb = gdbm_open("MYDBM",512,GDBM_NEWDB,(O_RDWR|O_CREAT), 0640);
  count = 1;
  while(fgets(line,1000,f1) != NULL) {
      if ((count % 100) == 0) printf("%d 行終り¥n",count);
      key.dptr = strtok(line," ");
      key.dsize = strlen(key.dptr);

      p = strtok(NULL,"¥n");
      content.dptr = p;
      content.dsize = strlen(p);
      if (gdbm_store(mydb,key,content,GDBM_REPLACE) < 0) quit("登録できない");
      count++;
  }
  if((r = fclose(f1)) == -1) quit("ファイル 1 が閉じれない");
  dbm_close(mydb);
}
```

```
void quit(char *s)
{
  printf(s); putchar('¥n');
  exit(1);
}
```

・データベース検索プログラム　kendbm.c

```
#include <stdio.h>
#include <gdbm.h>
#include <fcntl.h>

/*
#define NDBM_KEY_MAX        256
#define NDBM_CON_MAX        1024
*/

main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  int r,c,len,size,i;
  char line[256],noun[200],*p;
  GDBM_FILE *mydb;
  datum   key, val;

  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイルが開けない");
  mydb = gdbm_open("MYDBM",512,GDBM_READER,O_RDONLY, 0640);
  while(fgets(line,256,f1) != NULL) {
       chop(line);

      key.dptr = strtok(line," ");
       p= strtok(NULL,"¥n");
      key.dsize = char_size(key.dptr);
      printf("%s %s ",line,p);
```

```
        val = gdbm_fetch(mydb,key);
        size = val.dsize;
        for(i=0; i < size ; i++) putchar((val.dptr)[i]);
        putchar('¥n');
    }
    if((r = fclose(f1)) == -1) quit("ファイル 1 が閉じれない");
    gdbm_close(mydb);
}


void quit(char *s)
{
    printf(s); putchar('¥n');
    exit(1);
}


int char_size(char *p)
{
    int i;
    for(i=0;p[i] != NULL;i++);
    return i;
}


void chop(char *p)
{
    int i;
    for(i=0;p[i] != '¥r';i++);
    if (i > 0) p[i] = '¥n';
}
```

列ポイント抽出プログラム　retupoint.c

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

main()
{
        FILE *fp;

        int first,second,third;
        int i=1,j=0;

        fp=fopen("out3","r");

        while(fscanf(fp,"%d %d %d",&first,&second,&third)!=EOF){
                if(j!=first){
                        printf("%d¥n",i);
                        j=first;
                        i++;
                }
                else{
                        i++;
                }
        }
        fclose(fp);
        printf("%d¥n",i);
}
```

索引語文書行列の行数，列数抽出プログラム　gyouretu.c

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

main()
{
        FILE *fp;

        int first,second,third;
        int i,j;

        fp=fopen("out3","r");
        while(fscanf(fp,"%d %d %d",&first,&second,&third)!=EOF){
        i=first;
        }
        fclose(fp);

        fp=fopen("out4","r");
        while(fscanf(fp,"%d %d %d",&first,&second,&third)!=EOF){
        j=second;
        }
        fclose(fp);
        printf("%d      %d",i,j);
}
```

索引語文書行列作成プログラム　transpro

```
function transpro {
  mkdbm2 meishi-jigen
  for file in *.txt
  do
    sort $file | uniq -c >| gomi
    awk '{print $2,$1,"¥r"}' gomi >| $file.gomi
    kendbm2 $file.gomi >| $file.data
    echo $(awk -f jigen-hindo.awk $file.data) >> saku-hindo
  done
}
```

* meishi-jigen は名詞で使用頻度 25 以上のものに次元をつけたファイル

* jigen-hindo.awk の中身

```
{
if($3!=NULL){
  printf($4":"$2" ")
}
}
```

matrix データ作成プログラム　matrixdata

```
function matrixdata {
 awk -f gyouiti1.awk saku-hindo > out1
 awk -f gyouiti2.awk out1 > out2
 sort -k1,1n -k2,2n out2 > out3
 sort -k2,2n -k1,1n out3 > out4
 retupoint.exe > retupoint
}
```

* gyouiti1.awk の中身

```
BEGIN { FS = ":" }
{
for(i=1;i<=NF;i++){
    printf ($i" ")
    if(i==NF){
       printf ("¥n")
    }
}
}
```

* gyouiti2.awk の中身

```
{
for(i=1;i<=NF;i++){
  if(i%2!=0){
     printf (NR" "$i" ")
  }
  else{
     printf ($i"¥n")
  }
}
}
```

matrix ファイル作成プログラム　mkmatrix

```
function matrix {
aaa=$(wc -l out3)
bbb=${aaa%*out3}
echo "Jikken Data                    jikken"
echo "Transposed"
echo "rra      $(gyouretu.exe) $bbb      0"
echo "   (10i8)       (10i8)       (8f10.3)       (8f10.3)"
echo "$(awk -f matrix1.awk retupoint)"
echo "$(awk -f matrix1.awk out4)"
echo "$(awk -f matrix2.awk out4)"

}
```

* matrix1.awk の中身

```
{
if(NR%10==0){
printf ($1"¥n")
}
else{
printf ($1" ")
}
}
```

* matrix2.awk の中身

```
{
if(NR%10==0){
printf ($2"¥n")
}
else{
printf ($2" ")
}
                                    }
```