言語モデルを利用した
類似文書検索

執筆者：紺野 憲一

指導教官：新納 浩幸

平成 15 年 3 月 3 日

目次

図目次

1章　はじめに

1．1　概要

コンピュータやインターネット等の発展と共に私たちは大量の情報を知ることができるようになった。しかし、人の手ではその膨大な量の情報を管理することが難しくなってきている。そこで、機械的にデータの中から必要とされる情報を抽出する手法が研究されている。その一つとして類似文書検索がある。類似文書検索とは入力した文書と内容の近い文書を見つけてくるものである。

現在、類似文書検索の手法として主にベクトル空間モデルを使った方法が研究されている。また、別の方法として言語モデルを用いた方法も考えられている。言語モデルを用いると、ベクトル空間モデルより検索精度を上げることができると考えられている[1]。

本研究では、この言語モデルでの検索を行う。また、評価としてベクトル空間モデルを用いた検索と比較する。

第２章 ベクトル空間モデルによる類似文書検索

２．１　ベクトル空間モデル

２．１．１　索引語

文書は単語から構成されている。よって文書中に含まれる単語の集合によって文書の内容を近似することが出来ると考えられる。しかし、全ての単語がその文書の内容と関係しているわけではない。したがって、文書を単語の集合で近似する場合、文書の内容を特徴付ける単語を抽出する必要がある。この、文書の内容を特徴付ける単語を索引語という。

２．１．２　ベクトル空間モデル

ベクトル空間モデルでは、文書を索引語の重みを要素とするベクトルで文書を表現する。

今、検索対象となる文書を$d_1,d_2,d_3,\cdots,d_n$とし、文書全体を通して全部でｍ個の索引語$\omega_1,\omega_2,\omega_3,\cdots,\omega_m$があるとする。 このとき、文書$d_j$の中の索引語$\omega_i$における重みを$d_{ij}$とし、文書$d_j$を以下のようなベクトルで表す。これを文書ベクトルという。

$$d_j = \left[d_{1j}, d_{2j}, d_{3j}, \cdots, d_{mj}\right]^t$$

よって、文書集合全体は次のようなｍ×ｎ行列Ｄによって表せる。

$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} & \cdots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \cdots & d_{2n} \\ d_{31} & d_{32} & d_{33} & \cdots & d_{3n} \\ \vdots & \vdots & \vdots & \cdot & \vdots \\ d_{m1} & d_{m2} & d_{m3} & \cdots & d_{mn} \end{bmatrix}$$

これを索引語文書行列といい、またベクトル空間モデルといわれるものである。
検索質問qも同様に、文書ベクトルで表すことができる。

$$q = [q_1, q_2, q_3, \cdots, q_m]^t$$

2．1．3　索引語の重み付け

ベクトル空間モデルを作成するにあたって、索引語の重みを決定することが必要になる。
索引語の重み付けとしては以下の３つが挙げられる。

・局所的重み：l_{ij}

文書中に現れる単語の出現頻度である。出現回数が多ければ多いほど大きな値が
えられる。

・大域的重み：g_i

文書集合全体にわたる索引語の分布を考慮して決定される重み。特定の文書にのみ
出現する索引語に対して大きな値が与えられる。

・文書正規化係数：n_j

文書が長いほどまれる索引語の出現回数は増える、よって索引語も重みも大きくな
ってしまう。文書正規化係数はこの文書の長さによる影響をなくす為に導入される。

これによって重みd_{ij}が求められる。

$$d_{ij} = \frac{l_{ij} g_i}{n_i}$$

2．2コサイン尺度

文書検索において、与えられた文書との類似度を測る方法としてよく用いられる方法がコサイン尺度である[2]。これは内積を正規化したものである。

・ コサイン尺度

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

例えば、ベクトルA＝｛0，1，3，0，6，1，7｝
ベクトルB＝｛9，5，3，9，5，3，7｝
があるとするとコサイン尺度は以下のように計算される。

$$\sum_{i=1}^{m} A_i q_i = 0\times9+1\times5+3\times3+0\times9+6\times5+1\times3+7\times7$$

$$= 96$$

$$\sqrt{\sum_{i=1}^{m} A_i^2} = \sqrt{0^2+1^2+3^2+0^2+6^2+1^2+7^2}$$

$$= \sqrt{96}$$

$$\sqrt{\sum_{i=1}^{m} B_i^2} = \sqrt{9^2+5^2+3^2+9^2+5^2+3^2+7^2}$$

$$= \sqrt{279}$$

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

$$= \frac{96}{\sqrt{96}\times\sqrt{279}}$$

$$= 0.59$$

２．３　ベクトル空間モデルを用いた類似文書検索

実際に例を挙げてベクトル空間モデルでの類似文書検索を示す。

・検索質問文書

ｑ：私は茨城県民です。

・検索対象文書

D_1：私は茨城大学の学生です。

D_2：私は茨城県に来たことがあります。

D_3：茨城県の大半は山である。

D_4：茨城在住の茨城県民です。

・索引語

ω_1：茨城

ω_2：県

ω_3：民

ω_4：大学

ω_5：学生

ω_6：山

・索引語・文書ベクトル

索引語の重みを単語の出現回数に限定（局所的重みのみを用いる、大域的重みと文書正規化係数は１とする）。

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

・検索質問文書ベクトル

$$q = [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]^t$$

・類似度の計算

$\cos(d_1, q) = 0.33$

$\cos(d_2, q) = 0.81$

$\cos(d_3, q) = 0.66$

$\cos(d_4, q) = 0.94$

・検索結果

検索順位	文書 No	類似度
1	d_4	0.94
2	d_2	0.81
3	d_3	0.66
4	d_1	0.33

3章　言語モデルによる類似文書検索

3．1　言語モデル

3．1．1 概要

言語モデルとは、ある言語において単語列αが発生する確率モデルである。単語列αが現れる確率をkとして以下の式で表現できる。

$$k = P(.)$$

このときのPが言語モデルと呼ばれるものである。

言語モデルの利点として以下のことが挙げられる。

- 自然言語には意味的曖昧性などの様々なレベルの曖昧性が存在する。確率的な情報を用いることによって、曖昧性のある複数の候補の中から最も確からしいものを選んだり、候補の間の順位付けを行うことが出来る。また、確率の小さいものを枝刈りすることによって、処理に効率化を行うことが出来る。
- 言語モデルのパラメータは確率理論や情報理論に基づき自動的に推定することが出来る。大量の言語データからパラメータを推定することにより高精度なモデルを構築できる。

ここでは、この言語モデルが文書毎に存在すると考える。そうすることにより、そのモデル同士の距離を測ることによって文書同士の類似度を測定できると考える。

3．1．2　Nグラムモデル

言語モデルは複数あり、その中で特に重要とされるものに、Nグラムモデル、隠れマルコフモデル、確率文脈自由文法の3つである。今回はこの中からNグラムモデルを利用する。N グラムモデルは非常に単純なモデルであり、多くの欠点がある半面非常に強力なモデルである。

Nグラムモデルは単語の生起がその直前の (N-1) 単語にのみ依存すると考えたモデルである。しかし、実際の文書においてどのような単語が用いられるかは、人それぞれの好みや文の構成、対話の流れなどによって様々な影響を受けており、直前の単語のみに影響されているとはいいがたい。一見、N グラムモ

デルの仮定は大きく間違っているように見られるが、この仮定が現実的に有効であることは多くの確率・統計的自然言語処理システムによって実証されている。

これによって単語の生起確率を表す。例えば、このような文書があるとする。

例）q：私 は 茨城 県 民 です。

単語列wの出現頻度を C（w）で表すことにすると、文書中の“茨城”という単語が“は”あるいは“私 は”の後に現れる確率は以下のようになる。

$$P(\text{茨城} \mid \text{は}) = \frac{C(\text{は 茨城})}{C(\text{は})}$$

$$P(\text{茨城} \mid \text{私 は}) = \frac{C(\text{私 は 茨城})}{C(\text{私 は})}$$

このように直前の単語のみを用いる（N=2）モデルをバイグラムと呼び、直前の2単語を用いる（N=3）モデルをトライグラムと呼ぶ。また、単語が独立して現れると考え（N=1）たときはユニグラムと呼ぶ。

一見、N の値が大きいほど、よりよいモデルになりそうに思えるが、N を大きくしすぎると問題が発生する。1つは、N を大きくするに従い急激にモデルのパラメータ（単語の条件付確率）数が増えるという点である。パラメータ数は指数オーダーで増えていくので、N が大きくなるとパラメータを正確に指定することが不可能になる。さらに、データが大きくなることによる計算時間の増大といった問題も現れる。また、N の値を大きくしても現実的に精度が向上しないことが証明されている。

これらの観点から通常はバイグラムまたはトライグラムを用いることが多い。

3．1．3　言語モデルの作成

今回の実験では言語モデルの要素として検索質問文書中に含まれる３単語列を用いる。

例）q：私 は 茨城 県 民 です。

検索質問文書qをこのように指定するとその文書中に含まれる３単語列は以下の４つである。

私	は	茨城
は	茨城	県
茨城	県	民
県	民	です

図３．１　３単語列

それぞれの要素の値は先程のＮグラムモデルを利用して計算できる。
３単語列をそれぞれ先頭から A,B,C としたときその３単語列が文中に現れる確率P(A,B,C)は以下のように表せる。

$$P(A,B,C) = P(A)P(B|A)P(C|A,B)$$

例文の３単語列に対しP(A,B,C)を導く。

A	B	C	P（A,B,C）
私	は	茨城	0.07×10^{-7}
は	茨城	県	0.05×10^{-7}
茨城	県	民	0.01×10^{-6}
県	民	です	0.09×10^{-8}

図３．２　３単語列の生起確率

これが検索質問qの言語モデルP_qとなる。

次に検索対象文書 d_1, d_2, d_3, d_4 においてのP(A,B,C)も求める。

	P_q	P_{d_1}	P_{d_2}	P_{d_3}	P_{d_4}
私　は　茨城	0.07×10^{-7}	0.09×10^{-6}	0.08×10^{-7}	0	0
は　茨城　県	0.05×10^{-7}	0	0.07×10^{-7}	0	0
茨城　県　民	0.01×10^{-6}	0	0	0	0.09×10^{-7}
県　民　です	0.09×10^{-8}	0	0	0	0.08×10^{-7}

図３．３ 言語モデル

3．1．4　スムージング

図3．3のように言語モデルでは文書中に含まれない単語の生起確率が全て0になってしまう為、空の要素が多い行列となってしまう。このままでは、文書同士の比較がうまく行えない。これはゼロ頻度問題といわれる重大な問題である。そのため、文書中に出ていない単語列に大しても何らかの方法で値を補正する必要がある。これらの補正方法にはディスカウンティングやスムージングと呼ばれるものがある。

今回Nグラムモデルの作成に利用したツールキットではバックオフスムージングが用いられていた。バックオフスムージングとは、N グラムの確率値を低次の (N−1) グラムの確率値と線形に補完する方法である。単語列$\omega_1\omega_2\cdots\omega_n$があった時のバックオフスムージングの式は以下のようになる。

$$P(\cdot_n \mid \cdot_{n-N+1}^{n-1}) = \cdot(\cdot_{n-N+1}^{n-1})P(\cdot_n \mid \cdot_{n-N+2}^{n-1})$$

このときのαはグッド･チューリング推定法などから求めることが出来る。このとき$P(\cdot_n \mid \cdot_{n-N+2}^{n-1})$はさらに低次の（N－2）グラムのモデルからバックオフスムージングを用いて値を求めることが出来る為、最終的に単語列$\omega_1\omega_2\cdots\omega_n$のNグラムモデルは各単語$\omega_1$、$\omega_2$、・・・・、$\omega_n$が存在していれば求めることが出来る。

	P_q	P_{d_1}	P_{d_2}	P_{d_3}	P_{d_4}
私　は　茨城	0.07×10^{-7}	0.09×10^{-6}	0.08×10^{-9}	0.08×10^{-16}	0.03×10^{-8}
は　茨城　県	0.05×10^{-7}	0.01×10^{-9}	0.07×10^{-10}	0.05×10^{-18}	0.01×10^{-9}
茨城　県　民	0.01×10^{-6}	0.06×10^{-10}	0.04×10^{-13}	0	0.09×10^{-7}
県　民　です	0.09×10^{-8}	0.08×10^{-12}	0.04×10^{-8}	0.02×10^{-17}	0.08×10^{-7}

図3．4　スムージング後の言語モデル

3．2　KL情報量

確率分布間の距離はKL情報量を用いて測ることができる。言語モデル P_a と言語モデル P_b のKL情報量 $d(P_a,P_b)$ は以下で定義される。

$$d(P_a,P_b)=\sum Pa\{.\}\log\frac{Pa\{.\}}{Pb\{.\}}$$

この値が小さければ類似度が高いと推測できる。

ここに2つの確率分布 $P_A=\{0．1、0．2、0．3、0．1\}$

$P_B=\{0．3、0．1、0．1、0．1\}$

があるとする。このときのAとBとのKL情報量を計算すると以下のようになる。

$$d(P_A,P_B)=0.1\log\frac{0.1}{0.3}+0.2\log\frac{0.2}{0.1}+0.3\log\frac{0.3}{0.1}+0.1\log\frac{0.1}{0.1}$$

$$=0.156$$

ここでαを検索質問文書に含まれる 3 単語列とすると、αがn個あるとして検索質問の文書に含まれない3単語列をβとすると以下の式で導き出せる。

$$.=1-\sum_{i=1}^{n}.$$

つまり確率分布の総和を 1 として考え、その足りない分を補完するものである。これを P_A と P_B に対して求めれば以下のようになる。

$$._A=1-(0.1+0.2+0.3+0.1)$$

$$=0.3$$

$$._A=1-(0.3+0.1+0.1+0.1)$$

$$=0.4$$

これは類似度が文書の長さに影響を受けないようにする為のものである。

これを加えたKL情報量の式は以下のようになる。

$$d(P_a,P_b)=\sum Pa\{.\}\log\frac{Pa\{.\}}{Pb\{.\}}+Pa\{.\}\log\frac{Pa\{.\}}{Pb\{.\}}$$

よってP_AとP_BとのKL情報量は以下のようになる。

$$d(P_a,P_b)=0.156+0.3\log\frac{0.3}{0.4}$$

$$=0.156-0.037$$
$$=0.119$$

この場合β_Aよりβ_Bの方が大きいのでマイナスの値となった、つまりαの総和が小さければ小さいほどベータの値が大きくなり、KL情報量が小さくなるので類似度が高くなる。これによって文書が短く単語が少ない文書に対しても適正な類似度が導き出されることが期待できる。

3．3　言語モデルを用いた類似文書検索

ベクトル空間モデルと同じ例を用いて実際の類似文書検索を示す。

・検索質問文書

q：私は茨城県民です。

・検索対象文書

D_1：私は茨城大学の学生です。

D_2：私は茨城県に来たことがあります。

D_3：茨城県の大半は山である。

D_4：茨城在住の茨城県民です。

・言語モデル

	P_q	P_{d_1}	P_{d_2}	P_{d_3}	P_{d_4}
私　は　茨城	0.0476212	0.030304	0.030304	0.00595251	0
は　茨城　県	0.0476212	0.030304	0.030304	0.00396826	0.300054
茨城　県　民	0.0476212	0	0	0	0.100023
県　民　です	0.00291541	0	0	0	0.000370339

・KL 情報量

d_1	0.0447737
d_2	0.00319704
d_3	0.0620342
d_4	0.376066

検索順位	文書 No	類似度
1	d_2	0.00319704
2	d_1	0.0447737
3	d_3	0.0620342
4	d_4	0.376066

4章　実験

4．1　実験手順

実験は以下の２つを行う。

・ベクトル空間モデルでの類似文書検索

・言語モデルでの類似文書検索

これらの結果の比較をする為、条件を同じものに定める。

・検索の質問となる文書は以下の文書とする。これは検索質問文書の中から取り出したものである。

阪神大震災（兵庫県南部地震）の被災地での都市ガス復旧が遅れているため、大阪ガスは一日から全国のガス事業者から五百人の追加応援を求めることになった。これで全国の応援は二千三百人になる。地震発生当初、「二月末には復旧」とみられていたが、交通渋滞などによる工事の遅れで復旧作業がはかどっておらず、被災者の問い合わせも多いため、同社は二、三日中に地域ごとの復旧見通しを発表する。　震災直後、大阪ガスは八十五万戸へのガス供給を停止。供給継続地域での約二万件に上るガス漏れ通報に対応するほか、応援部隊を供給停止地域での低圧ガス管の点検、修理に投入した。一月三十日現在、神戸、西宮、芦屋市など八十万戸でガスはストップしている。　復旧が予想以上に遅れているのは、交通渋滞に加え（１）ガス漏れ通報や供給停止地域でのガス管破損個所が想像を上回って多い（２）水道管の破損や雨でガス管内に水が入り、水を出す作業に手間取っているためだ。ガス管の修理は通常三千戸前後を一区画として、各家庭のバルブを閉めて、ガスを流す、圧力の変化でガス漏れの位置を確定するが、今回はガス管の破損場所が多い、三千戸では圧力が下がり過ぎてガス漏れ位置を特定できない。このため区画をさらに細分化するため時間がかかるた。　二十九日ごろから供給継続地域でのガス漏れ通報への対応がほぼ終了。こちらの人員を供給停止地域に振り替えている、一日の供給再開戸数は、三十日で九千四百戸まで増加。中心部ほど破損も激しくなるため、全面復旧にはまだ時間がかかるそうだ。

・ 検索の対象となる文書は毎日新聞95年度分の記事文書とし、その総数は約６７、０００記事である。

これは形態素解析済みの文書であり、文ごとにナンバーが付けられており、以下のような形式になっている。

```
950101001.0   未知語 950101001.0   名詞-サ変接続 17 * 0 * 0
                     記号-空白 80 * 0 * 0
「      「      「     記号-括弧開 82 * 0 * 0
私      ワタシ  私     名詞-代名詞-一般 14 * 0 * 0
の      ノ      の     助詞-連体化 71 * 0 * 0
大切    タイセツ       大切   名詞-形容動詞語幹 18 * 0 * 0
                        :
                        :
言葉    コトバ  言葉   名詞-一般 2 * 0 * 0
だっ    ダッ    だ     助動詞 74 特殊・ダ 55 連用タ接続 5
た      タ      た     助動詞 74 特殊・タ 54 基本形 1
。      。      。     記号-句点 78 * 0 * 0
EOS
950101001.1   未知語 950101001.1   名詞-サ変接続 17 * 0 * 0
「      「      「     記号-括弧開 82 * 0 * 0
そこ    ソコ    そこ   名詞-代名詞-一般 14 * 0 * 0
に      ニ      に     助詞-格助詞-一般 61 * 0 * 0
住む    スム    住む   動詞-自立 47 五段・マ行 16 基本形 1
                        :
                        :
```

１行ごとの構成は以下のようになっている。

```
単語　単語の読み　単語の原型　単語の活用形
```

半角の数字は {(記事毎の番号).(その記事中の文の番号)} である。
つまり上の 950101001.0 と 950101001.1 は 950101001 という記事の１文目と２文目である。
また、どちらの検索においても単語は原型を用いる。

４．２　ベクトル空間モデルを用いた類似文書検索

４．２．１　検索の条件設定

ベクトル空間モデルを用いた類似文書検索での固有の条件を以下のように指定した。

・索引語には検索対象の全文書中の含まれる、名詞、動詞、形容詞、副詞、とする。
その総数は、約１１０，０００単語である。
・索引語の重みには単語の出現回数を用いる。

４．２．２　文書ベクトルの作成と類似度の算出の手順

以上の条件でそれぞれの文書に対して文書ベクトルを作成する。その行程を以下のように分割して行った。

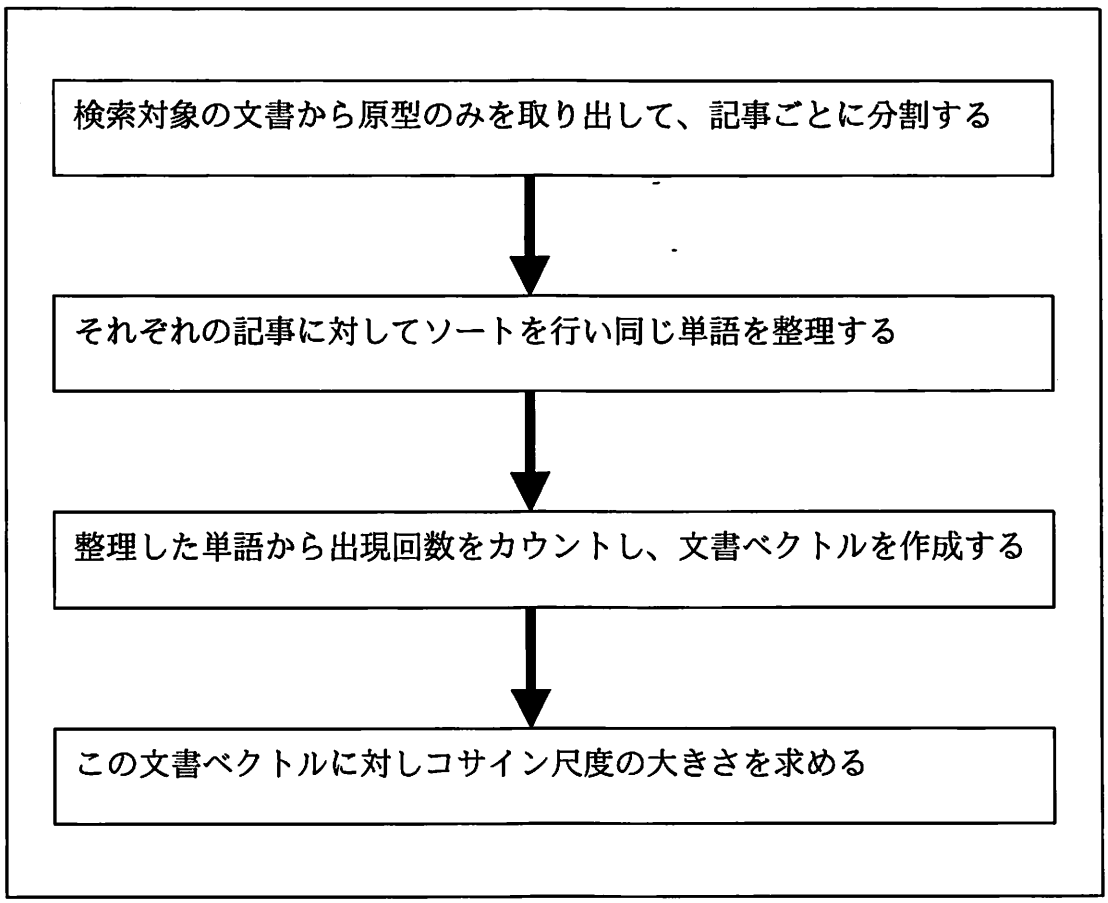

図４．２．１　実験の行程

４．２．３　記事の分割と原型の抽出

まず、記事ごとの比較をしやすいように文書を記事ごとに分割する。下で表したように記事番号が同じ文を一つのファイルとする、文の番号はこのとき取り除く。また出来る限り作業を高速化させるために必要な要素だけを取り出す。（名詞、動詞、形容詞、副詞の原型）これによって以下のような形式のファイルを作成する。(実行プログラム：添付 プログラムリストＡ　A_1)

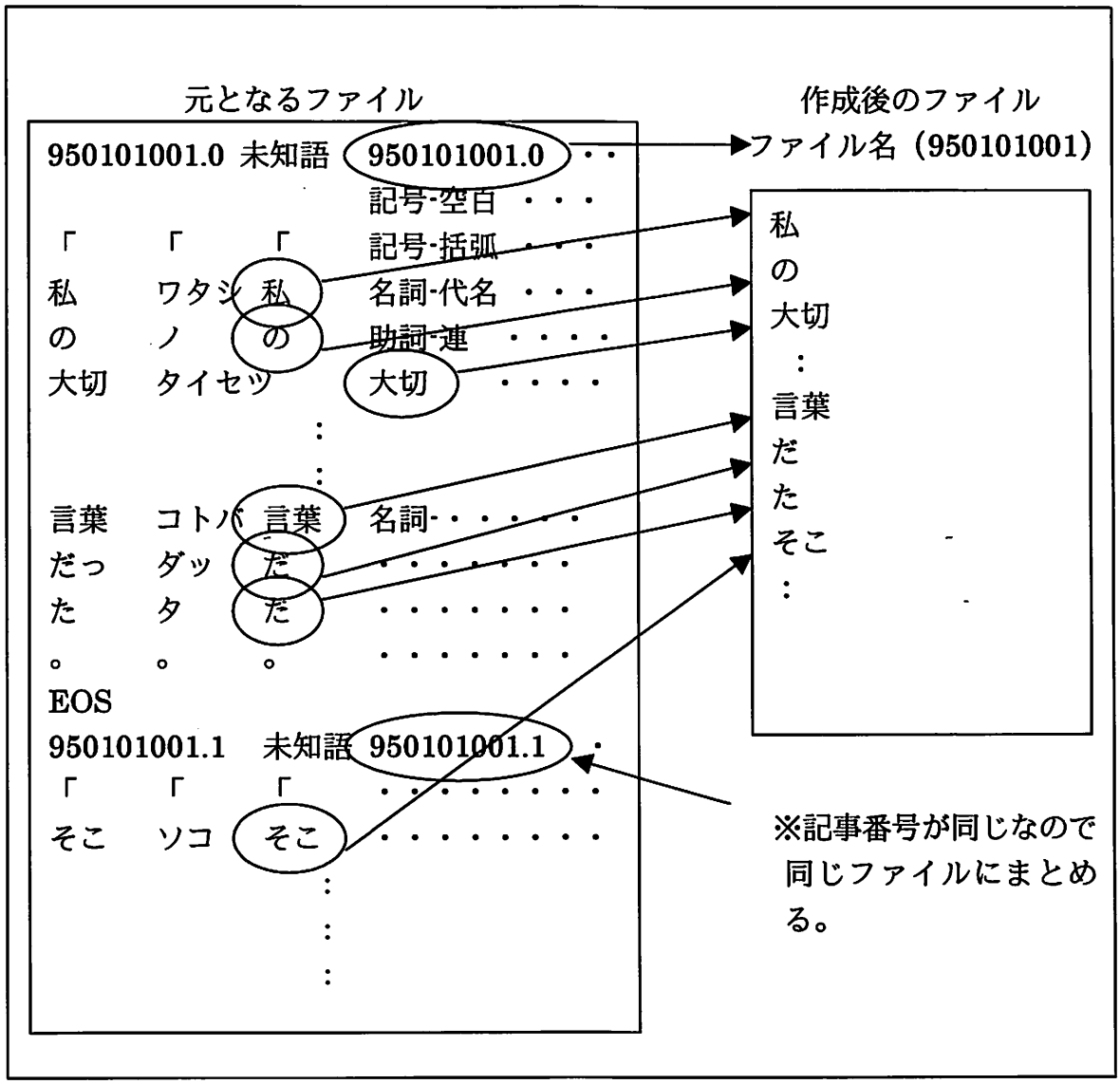

図４．２．２　原型の抽出と記事分割

４．２．４　各文書のソート

文書ベクトルの重みを算出する為、各単語の出現回数を数えなければならない。これは、生の文書から数えるよりも文書をソートしてから数えた方が早い。それは、同一単語が一つの場所に固められることによって、その場所の単語の数を数えるだけで出現回数を計算することが出来るからである。その例として以下にソートしたものとソートしないものとでの文字の出現回数を数えたときの例を挙げておく。(実行プログラム：添付 プログラムリストＡ　A_2)

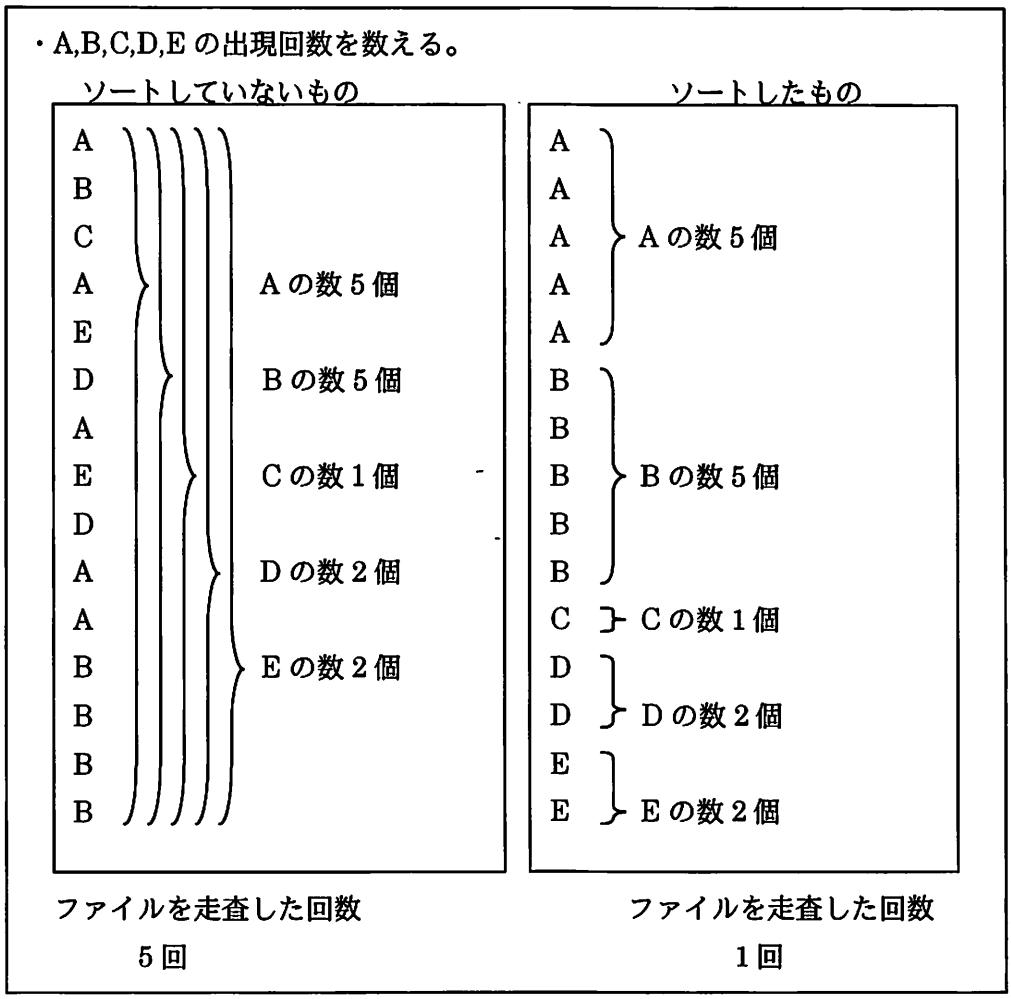

図４．２．３　単語カウント時のソートの有用性

４．２．５　単語の出現回数のカウント

以上のソートされた文書から単語の出現回数をカウントする。これによって文書ごとに以下のような書式のファイルを作成する。(実行プログラム：添付プログラムリストＡ　　A_３．A_4)

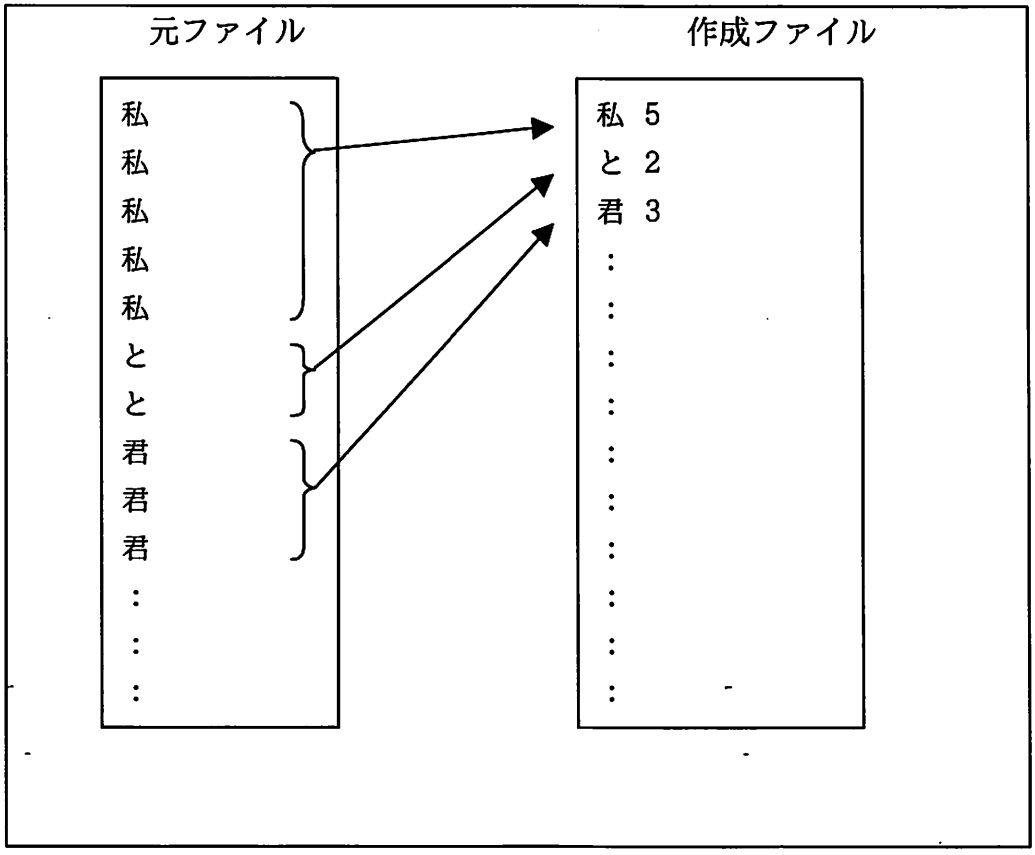

図４．２．４　文書ベクトルの作成

ここで作成されたファイルは出現回数０の単語を取り除いた文書ベクトルと考えることが出来る。

４．２．６　コサイン尺度を用いた類似度の計算

以上で作成した文書ベクトルファイルをコサイン尺度を用いて類似度を推測する。(実行プログラム：添付 プログラムリストＡ　A_５．A_６)

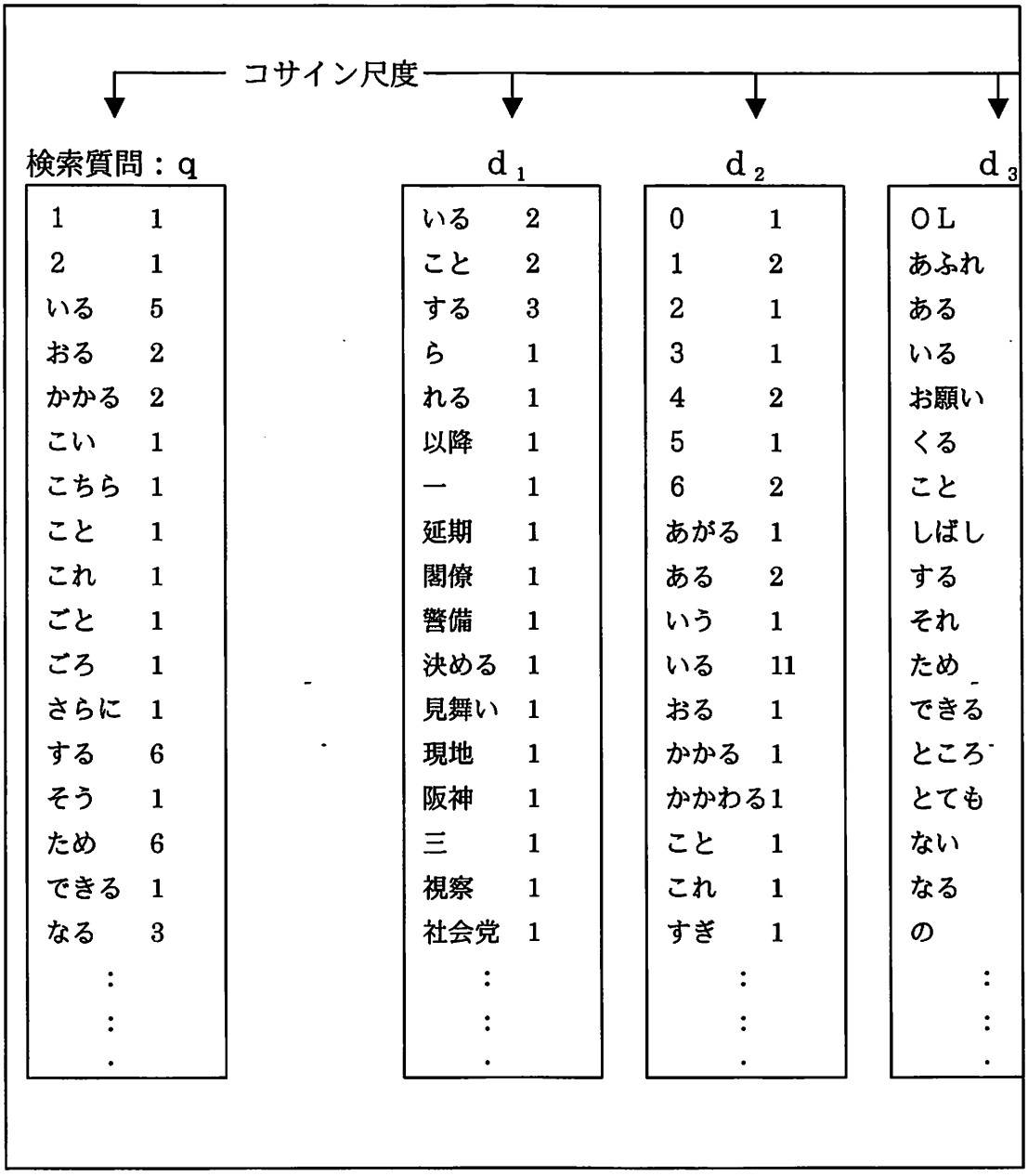

図４．２．５　コサイン尺度の計算

4．3　言語モデルを用いた類似文書検索

4．3．1 The CMU-Cambridge Statistical Language Modeling Toolkit

言語モデルの推定には、今回 The CMU-Cambridge Statistical Language Modeling Toolkit を用いる[3]。これは言語モデルの作成に必要な様々な計算を行う関数のセットである。

この中から以下の関数を利用する。

※ 関数の書式例
関数名
　　IN　：入力ファイル形式
　　OUT：出力ファイル内容
　関数の書式やオプションの指定など
　　・（不用と思われるオプションの説明は省く）

text2wfreq
　　IN　：テキストファイル
　　OUT：テキスト中に発生した言葉のリストと発生した回数
　関数の書式やオプションの指定など
　　・text2wfreq　＜　（入力ファイル名）.txt
　　　　＞　（出力ファイル名）.txt

wfreq2vocab
　　IN　：text2wfreq 関数で作成したファイル
　　OUT：語意ファイル
　関数の書式やオプションの指定など
　　・wfreq2vocab　＜　text2wfreq 関数で作成したファイル
　　　　＞　（出力ファイル名）.txt

text2idngram

IN　：テキストファイルと wfreq2vocab 関数で作成したファイル

OUT：単語の出現回数と id で表された N グラムリスト

関数の書式やオプションの指定など

・text2idngram　－vocab　（wfreq2vocab 関数で作成したファイル）

＜　（入力ファイル名）.txt

＞　（出力ファイル名）.txt

idngram2lm

IN　：wfreq2vocab 関数で作成したファイルと

text2idngram 関数で作成したファイル

OUT：ユニグラム＆バイグラム＆トライグラム

関数の書式やオプションの指定など

・text2idngram　－idngram　（text2idngram 関数で作成したファイル）

－vocab　（wfreq2vocab 関数で作成したファイル）

－arpa　（出力ファイル名）.txt

図４．３．１ 関数の説明

４．３．２　Ｎグラムモデルの計算

これらの関数を用いて言語モデルの作成を目指す。一つのファイルに対する関数の利用方法は以下のようになる。最終的に出力されるファイルはN=1~3のNグラムモデルである。(実行プログラム:添付 プログラムリストB　　B_1．B_2)

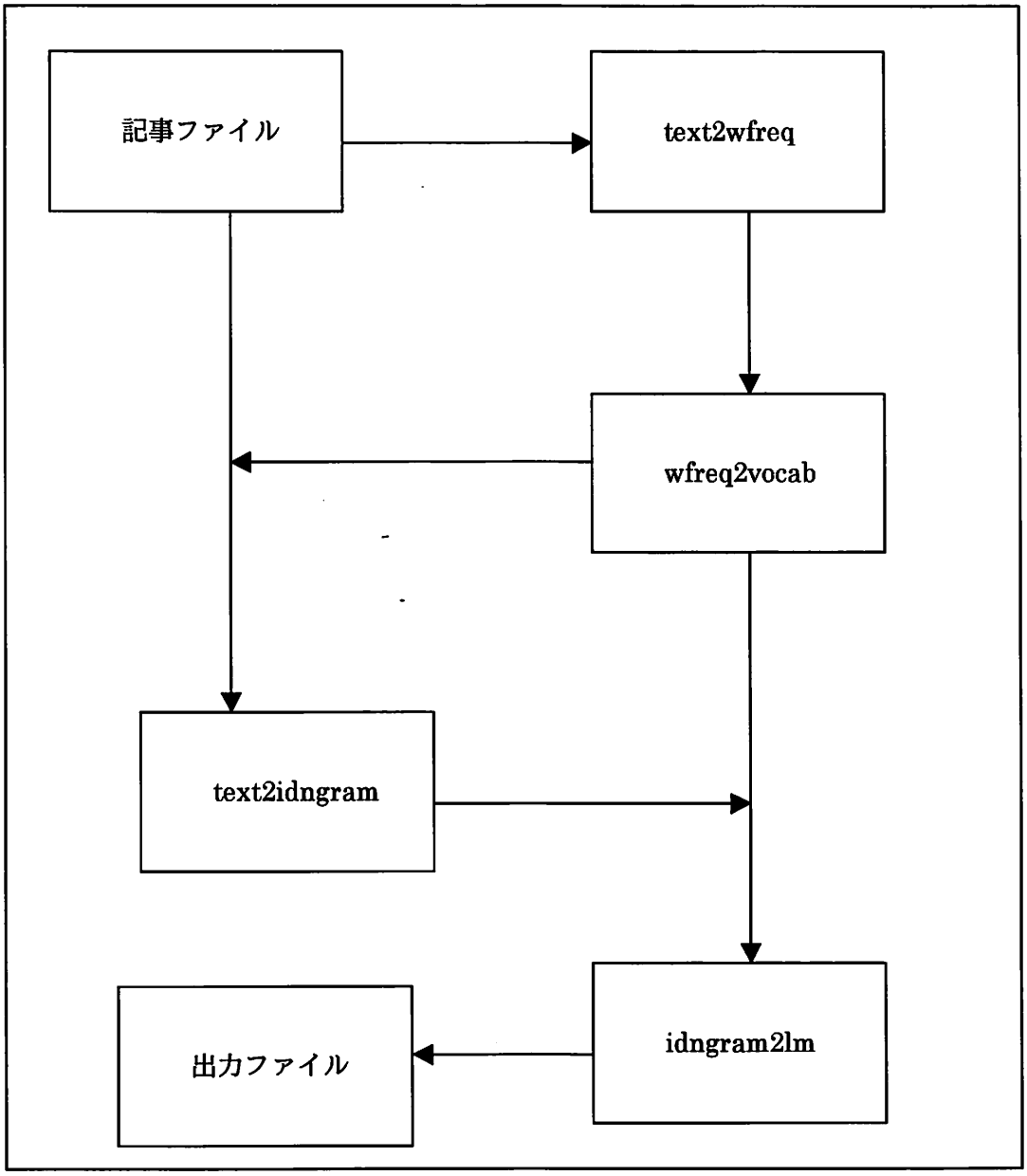

図４．３．２　Ｎグラムモデルの作成

これによって作成されたファイルは以下のような内容となる。

```
¥1-grams:
-2.6316 2          -0.0691
-1.9284 いる       -0.0691
-2.3263 おる       -0.0479
-2.3263 かかる     -0.0670
-2.1502 から       -0.0607
          :
          :
¥2-grams:
1.2923 百 戸 -0.0123
-0.6086 百 人 0.0463
-0.8151 部 ほど 0.0463
-0.8151 部隊 を -0.0004
-1.5933 復旧 」 0.0463
-0.9096 復旧 が -0.0146

          :
          :
¥3-grams:
-1.2385     阪神 大震災
-1.2385     震災 直後
-1.2385 、 「 二月
-1.2385 、 ガス を
-1.2385 、 芦屋 市
-1.2385 、 圧力 の
-1.2385 、 一 日
          :
          :
```

図４．３．３　ツールキットの出力結果

このファイルの説明を例を用いて以下に示す。3単語列（A,B,C）があるとし、そのときのツールキットのを用いたデータは以下のようになる。

```
¥1-grams:
p_A      A     bo_wt_A
p_B      B     bo_wt_B
p_C      C     bo_wt_C
¥2-grams:
p_AB      A  B    bo_wt_AB
p_BC      B  C    bo_wt_BC
¥3-grams:
p_ABC  A    B    C
```

図４．３．４　Ｎグラムモデルの書式

これよりユニグラム、バイグラム、トライグラムの値を計算することが出来きる。それぞれの計算式は以下のようになる。

・ユニグラムＰ（A）
　Ｐ（A）　＝　p_A
・バイグラムＰ（B|A）※文書中にバイグラムが存在したとき
　Ｐ（B|A）＝　p_AB
・バイグラムＰ（B|A）※文書中にバイグラムが存在しないとき
　Ｐ（B|A）＝　bo_wt_A×p_B
・トライグラムＰ（C｜AB）※文書中にトライグラムが存在するとき
　Ｐ（C|AB）=p_ABC
・トライグラムＰ（C｜AB）※文書中にABのバイグラムが存在するとき
　Ｐ（C|AB）=bo_wt_AB×p(C|B)
・トライグラムＰ（C｜AB）※上記のどれでもないとき
　Ｐ（C|AB）=p(C|B)

これによって文中に含まれないトライグラムについても値を導き出すことが出来る。

４．３．３　言語モデルの要素の指定

以上のＮグラムモデルから今回利用する言語モデルを作成する。モデルの要素は検索質問文書中に現れる３単語列である。まず、検索質問文書の先頭から３単語筒を取り出し以下のようなファイルを作成する。この時のそれぞれ３単語列が言語モデルの要素となる。

```
    阪神　大震災
阪神　大震災　（
大震災　（　兵庫
（　兵庫　県
兵庫　県　南部
県　南部　地震
南部　地震　）
地震　）　の
）　の　被災
の　被災　地
被災　地　で
地　で　の
で　の　都市
の　都市　ガス
都市　ガス　復旧
　　：
　　．
```

図４．３．５ 言語モデルの要素の指定

４．３．４　言語モデルの作成

次に、この要素の値をＮ－グラムモデルから導き出す。(実行プログラム：添付 プログラムリストＢ　Ｂ_３．Ｂ_４)

	Pq	Pd_1	Pd_2
阪神　大震災	2.08449e-05	0	1.74703e-202
阪神　大震災　(	2.06443e-05	0	0
大震災　(　兵庫	2.06443e-05	0	0
(　兵庫　県	2.08449e-05	0	0
兵庫　県　南部	2.06443e-05	0	5.94429e-07
県　南部　地震	2.06443e-05	0	0
南部　地震　)	2.06443e-05	0	0
地震　)　の	2.08449e-05	9.9954e-07	0
)　の　被災	2.08449e-05	0	0
の　被災　地	2.08449e-05	0	0
被災　地　で	2.08449e-05	0	0
地　で　の	2.06443e-05	2.66379e-05	1.08518e-05
で　の　都市	0.000136176	0	0
の　都市　ガス	2.08449e-05	0	0
都市　ガス　復旧	2.06443e-05	0	0
:	:	:	:

図４．３．６　言語モデルの作成

４．３．５　KL 情報量を用いた類似度の算出

KL 情報量を用いて言語モデル同士の類似度を算出する。この計算を行う上で検索質問文書でのβ（検索質問の文書に含まれない３単語列）は、検索対象の文書と比べて使用回数が多く必要となるたびにα（検索質問の文書に含まれる３単語列）の総和から計算していては時間の無駄である。よってあらかじめ検索質問文書でのβの値を計算しておくことにする。）これを KL 情報量を計算するプログラムに定数として与えることにより処理の効率化を計る。

これらの条件で検索質問文書の言語モデルと検索文書群の言語モデルとの KL 情報量を計る。（実行プログラム：添付 プログラムリスト B　B_5、B_6）

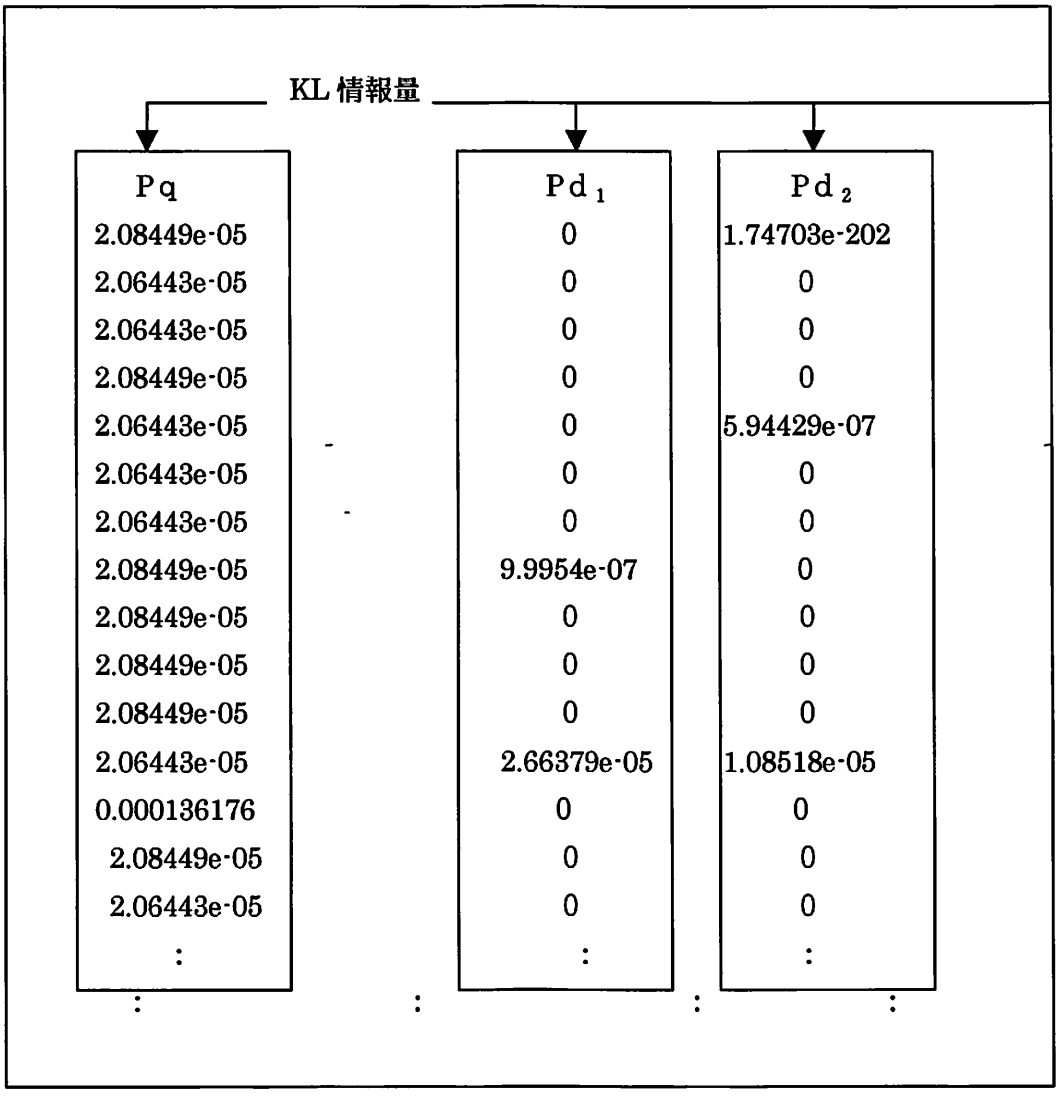

図４．３．７　類似度の計算

４．４　結果

４．４．１　類似度の高い文書とその内容

言語モデルとベクトル空間モデル出の類似文書検索の結果類似度が大きかったもの上位１０位までの記事をのファイル名（文書番号）と文書の書き出しの部分。１位の記事は共に検索質問文書自体となったので省略する。

・ベクトル空間モデル

2位　記事番号：950131337
　阪神大震災で被災するた神戸市などでの都市ガスの全面復旧時期が、当初見通しの二月末―三月初旬から大幅に遅れる見通しが強・・・・
3位　記事番号：950207305
　阪神大震災でまひすしたライフラインのうち、都市ガスの復旧が大幅に遅れているのは、地下に埋設されたガス管の・・・・・・・・・・
4位　記事番号：950227163
　阪神大震災後復旧が遅れていた都市ガスは、復旧率が二十五日、ようやく五割を突破した。大阪ガスのまとめでは、二十五日に・・・・
5位　記事番号：950224193
　大阪ガスが地震などによるガス漏れ防止のため家庭に設置していたマイコン制御装置が、地震時にガスを使用していない・・・・・
6位　記事番号：950119029
　通産省が十八日午後七時半現在でまとめた阪神大震災被害状況によると、大阪ガスが二次災害を防ぐため都市ガスの供給を遮断・・・・
7位　記事番号：950208205
　阪神大震災で現在も神戸市、芦屋市など七十万を超える世帯でガス供給が止まっているが、被害は旧式のガス導管接合部の破損が・・・・
8位　記事番号：950225189
　震度５（強震）クラスの地震の前後数分間にガスを使用していれば、家庭へのガス供給を自動的にストップするマイコン制御装置に・・・
9位　記事番号：950508178
　大阪ガスは七日、近く兵庫県や神戸市などが整備する防災拠点に、発電とともに冷暖房や給湯も行うコージェネレーション（電熱併・・・
10位　記事番号：950804198
　ガス臭の漂う家に「寝たきりのおじいさんと一緒に残る」というおばあさんのため、ガス管を仮修理。地震訓練どおりに復旧対象・・・・

・言語モデル

2位　記事番号：950227163
　阪神大震災後復旧が遅れていた都市ガスは、復旧率が二十五日、ようやく五割を突破した。大阪ガスのまとめでは、二十五日に・・・・
3位　記事番号：9502033290
　阪神大震災で供給が止まった都市ガスの復旧が遅れるている。神戸市、芦屋市、西宮市などガス供給停止地区で復旧したの・・・・・
4位　記事番号：950508178
　大阪ガスは七日、近く兵庫県や神戸市などが整備する防災拠点に、発電とともに冷暖房や給湯も行うコージェネレーション（電熱併・・・
5位　記事番号：950217206
　――きょうで一カ月、復旧の見通しは。　被害の大きい地域を除いて一カ月半、三月上旬をめどにしている。　――いつ聞・・・・・
6位　記事番号：950119307
　阪神大震災の傷がなお広がる十八日、運転を再開するた阪急神戸線西宮北口駅に、人々の列が続いた。県外の親類、知人を頼る・・・
7位　記事番号：950119233
　ミナト神戸はよみがえるのか。阪神大震災から二日目を迎えた十八日、被災者の中には廃虚と化した阪神間の街を逃れる、近くの・・・
8位　記事番号：950127028
　阪神大震災から十日。生活や生産にかかわるライフラインの復旧も徐々に進んでいるが、ガス、水道の復旧の遅れが目立つている。・・・
9位　記事番号：950224193
　大阪ガスが地震などによるガス漏れ防止のため家庭に設置していたマイコン制御装置が、地震時にガスを使用していない・・・・・
10位　記事番号：950208205
　阪神大震災で現在も神戸市、芦屋市など七十万を超える世帯でガス供給が止まっているが、被害は旧式のガス導管接合部の破損が・・・・・

４．４．２　類似度の上位の比較表

各検索で類似度が大きかったもの上位１０位の文書に注目して、その対応表を以下に示す。

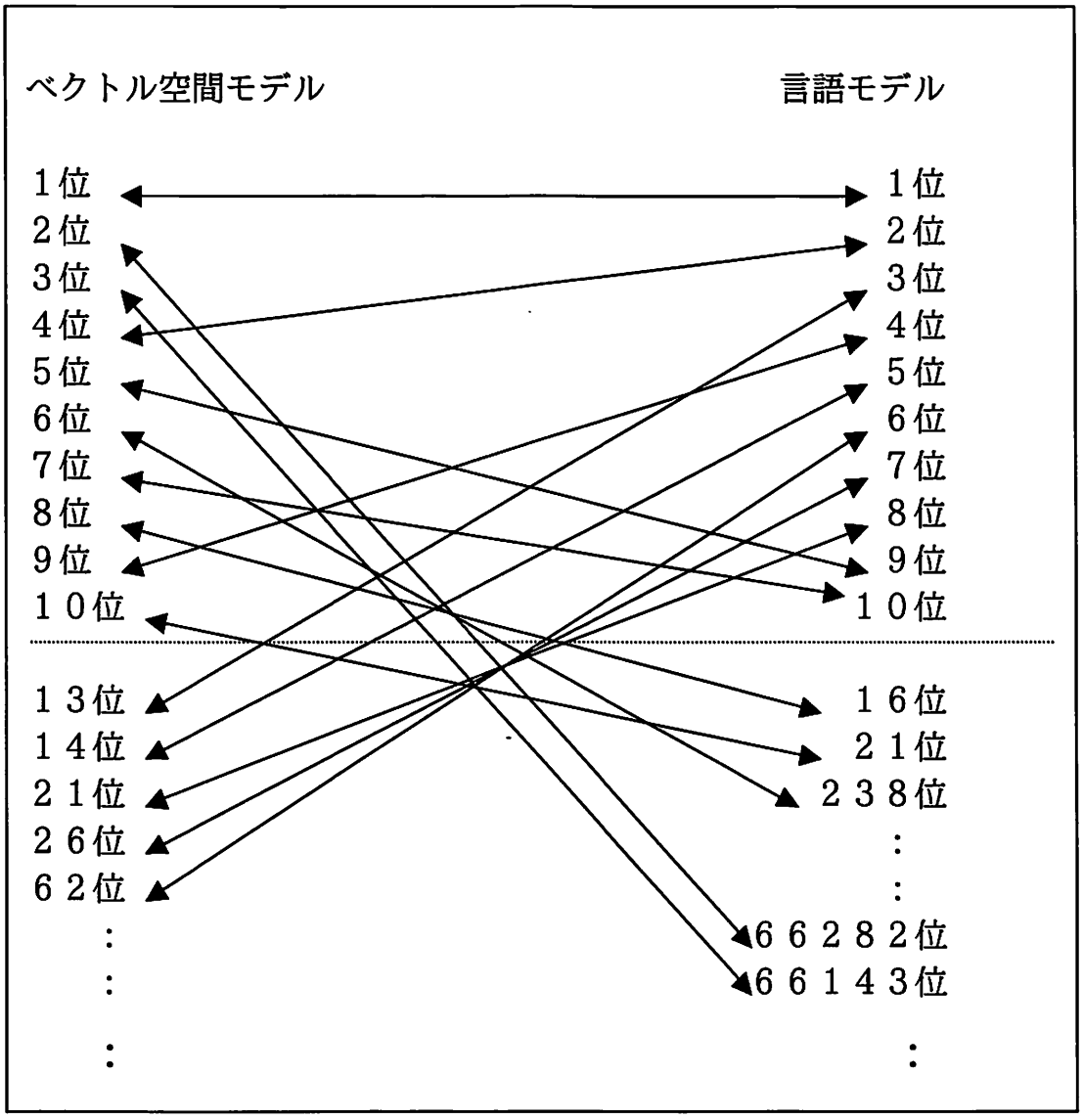

図４．４．１検索結果上位の比較

5章　考察

結果に挙げた言語ベクトルによる類似文書検索の結果の上位１０位までの内容を見た所、これらは十分類似した文書といえる内容であった。またベクトル空間モデルでの類似文飾の結果の上位 10 位までの内容も確認したところ、これも十分類似した文書であった。

両文書検索の高順位のものを比較した所、似たような文書が上位に挙げられており、これより言語モデルを用いた類似文書検索はベクトル空間モデルを用いた類似文書検索と同程度の精度が期待できそうであることがわかる。

しかし、この中に明らかに類似度の高さに差が現れたものが見られた。これは、ベクトル空間モデルで 2 位と 3 位に挙げられた文書である。これらの原因を推測する為、まずこの２文書の前文を挙げておく。

記事番号：950131337

阪神大震災でまひしたライフラインのうち、都市ガスの復旧が大幅に遅れているのは、地下に埋設されたガス管の中に、近くの破損するた水道管などから大量の水が流入するたことが大きな要因になっていることが、七日までに大阪ガスなどの調べで分かるた。　大阪ガスは現在、全国八十六社の応援を含める、八千三百人の態勢で復旧作業を続けているが、地震で供給が止まった八十五万七千四百戸のうち、今も約七十一万戸でガスが出ない状態。大半の地域でガスを供給できるのは三月上旬になる見通しという。　同社などによると、被災地のガス管は、「継ぎ手」と呼ぶれるつなぎ目付近でひびが入るたり、壊れるケースが多い。激震地付近ではこの継ぎ手や壊れた部分に、すぐ近くに埋設されていて破損した水道管から大量の水が流れ込む、復旧工事を始める前に、ポンプやバキュームカーを使うて水をくみ出す作業に追われている。兵庫県宝塚市では、ガス管から抜いた水が一日約一万三千リットル以上にもなる、水抜き作業だけで二日間もかかった現場もあったという。　同社広報部は「早期復旧を目指す連日最大限の作業を続けているが、管への水の流入量が多い、作業に手間取っている」と話している。

記事番号：950207305

阪神大震災で被災した神戸市などでの都市ガスの全面復旧時期が、当初見通しの二月末―三月初旬から大幅に遅れる見通しが強まり、大阪ガスは三十一日、復旧計画の全面的な見直し作業を始めた。予想以上にガス管が破損しているのに加え、交通渋滞やがれき除去のための工事が中断しがちだったことなどのため。二月一日中にも新たな全面復旧の時期の見通しをつける。　大阪ガスは、同社グループの六千人と、全国のガス事業者からの応援千八百人で復旧作業中。二月一日からは新たに五百人の応援を得て、作業のスピードアップを図る。しかし、作業は難航。ガス供給停止地域では二十四日に初めて供給を再開するたものの、その後は当初計画の一日二万戸を大幅に下回る一日平均約七千戸にしか供給を再開できておらず、三十日現在で約八十万八千八百戸のガスが止まっている。

見た限りどちらの文書も検索質問文書と類似した文書のようである。ではなぜこれらの文書の類似度に差が現れたのであろうか。これらの文書の特徴としては、数字が多用されていることが挙げられる。これが何かしらの影響を与えているのではないかと推測した。それを判断する為、それぞれの記事に対しての単語の出現回数を示した図を次に挙げる。

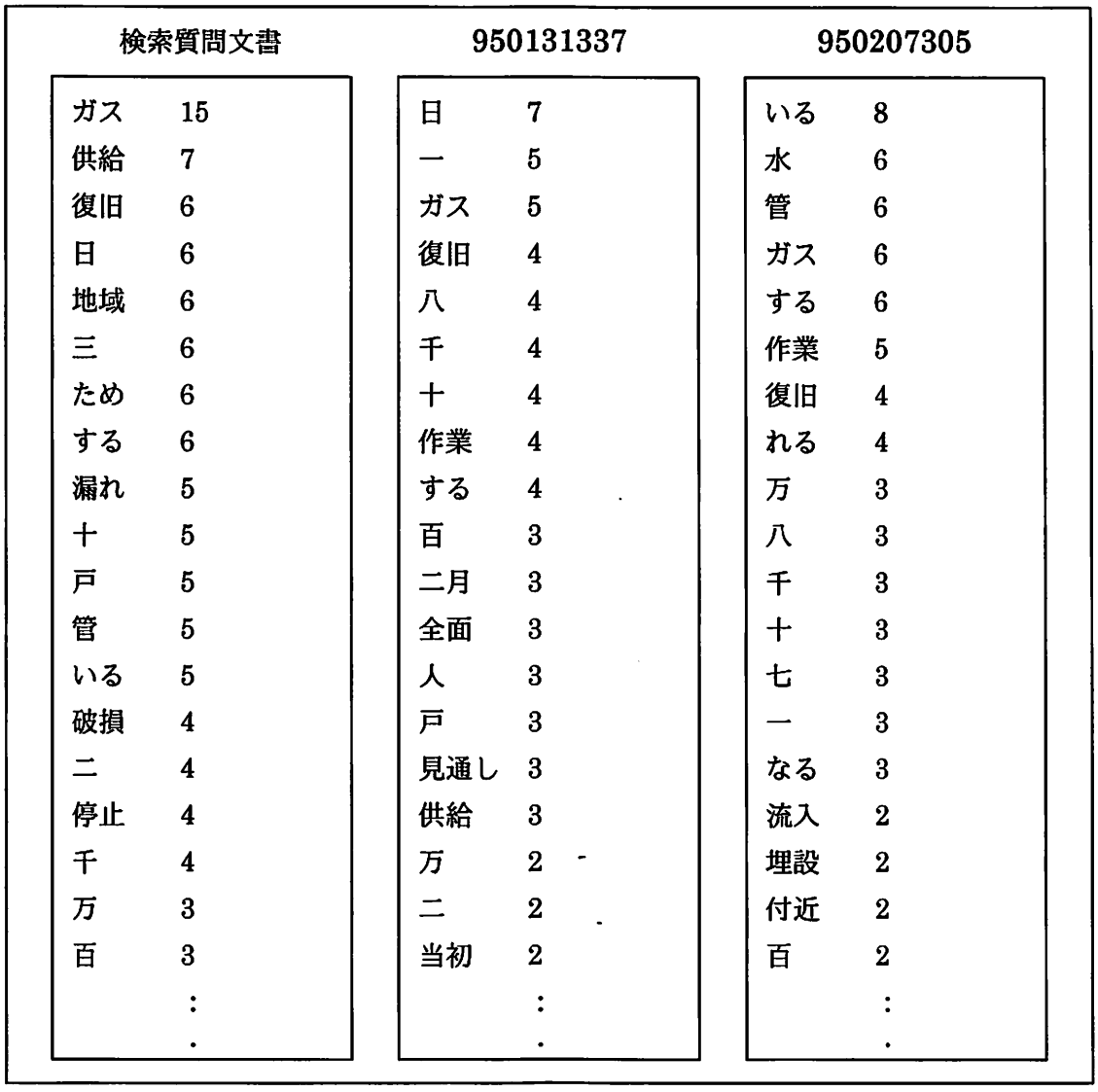

図５．１ 単語の出現回数

たしかにこれらの記事で数値が頻繁に使用割れているのがわかる。特に単位を表す“十”“百”“千”“万”等でそれが顕著に表れている。これより、ベクトル空間モデルでの検索において類似度が高くなったことが伺える。

ではなぜ言語モデルでの検索においてはあれほど類似度が低くなってしまったのであろうか。それもまた数字が多く含まれていたことが原因であると推測される。つまり数字の連なりに関連性がないため、単語の連なりから類似度を測定する言語モデルでは数値が多く含まれるほど類似度が小さくなってしまうのである。今回は数字を１桁ごとに分割して１単語としたためこのような問題が発生したと思われる。つまり、数字列を１単語として用いれば検索の制度を向上できるのではないかと予測される。

6章　おわりに

本研究では言語モデルを用いた類似文書研究についての可能性について検証した。

研究の内容としては、言語モデルを用いた類似文書検索の実装と、比較対照としてベクトル空間モデルでの類似文書検索の実装。さらに両類似文書検索の結果の比較を行った。

言語モデルを用いた類似文書検索はベクトル空間モデルを用いた類似文書検索と同程度の高い精度を上げることが出来ることがわかった。

しかし、数字が多く含まれる文書に対しての類似度の算出がうまくいかないなどの問題なども提示され、更なる改善が必要である。

また、今回のベクトル空間モデルを用いた類似文書検索では索引語の重みに単語の出現頻度を用いるなど簡単な方法を用いたが、ベクトル空間モデルでの類似文書検索についても改良を加えて、更なる比較実験が必要であると思われる。

言語モデルを用いた場合、ベクトル空間モデルのようにあらかじめモデルを用意しておくことが出来ない為、検索に時間がかかるという問題がある。これらの問題解決も今後の課題である。

謝辞

本研究の遂行および論文の作成に多大なご助言およびご指導を賜った新納浩幸 教官（茨城大学システム工学科）に深い感謝の意を表します。

また、ご指導いただきましたシステム工学計算機応用学講座の教官の方々にも深く感謝いたします。

最後に、本研究を進めるにあたり助言、協力をいただきました、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工学専攻)、高橋篤史氏（茨城大学大学院理工学研究科システム工学専攻)、藤枝雅一氏（茨城大学システム工学４回生)、山村一起(茨城大学システム工学４回生)に深く感謝いたします。

参考文献

[1] Jay M.Ponte and w.Burce Croft ;
"A LanguageModeling Approach to Information Retrieval " ,
SIGIR'98, pp.275-281 (1998).

[2] 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)

[3]P.R. Clarkson and R. Rosenfeld ;
"Statistical Language Modeling Using the CMU-Cambridge Toolkit ",
Proceedings ESCA Eurospeech (1997).

付録A　プログラムリスト

A_1.記事文書を記事ごとに分割するプログラム（C++）

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
void main(void)
{
    char  mei[6]="名詞-",dou[6]="動詞-",kei[8]="形容詞-",fuku[6]="副詞
    -",o[19]="abcdefghijklmnopqr";
        char file_name[20],A[600],tit[100],chk[100],*B,C[3],gg[2],rr[2];
        char *AA,*BB,*CC;
        int f,q;
        for(f=0;f<=4;f++)
        {
                gg[0]=o[f];
                gg[1]='¥0';
            for(q=0;q<=17;q++)
                {
                        rr[0]=o[q];
                    rr[1]='¥0';
                    strcpy(file_name,"date¥¥mai95.");
                strcat(&file_name[0],&gg[0]);
                        strcat(&file_name[0],&rr[0]);
                        strcat(file_name,".chasen");
             ifstream in_file(file_name);

                        while(! in_file.eof())
                        {
                                ofstream Ofile(tit,ios::app);
                                in_file.getline(&A[0],sizeof(A));
```

```
                    if(isdigit(A[0])!=0  &&  isdigit(A[1])!=0  &&
                    isdigit(A[2])!= 0&& strstr(&A[0],&chk[0])==0)
                    {
                            B=strtok(&A[0],".");
                            strcpy(&tit[0],"kizi¥¥");
                           C[0]=A[2];
                           C[1]=A[3];
                            C[2]=NULL;
                            strcat(&tit[0],&C[0]);
                            strcat(&tit[0],"¥¥");
                            strcat(&tit[0],B);
                            strcat(&tit[0],".txt");
                            strcpy(&chk[0],B);
                    }
                    else      if(strstr(&A[0],&chk[0])==0      &&
                    strlen(&A[0])>=5)
                    {
                            if(strstr(&A[0],&dou[0])!=0          ||
                            strstr(&A[0],&kei[0])!=0             ||
                            strstr(&A[0],&fuku[0])!=0            ||
                            strstr(&A[0],&mei[0])!=0)
                           {
                               AA = strtok(&A[0],"¥t");
                               BB = strtok(NULL,"¥t");
                               CC = strtok(NULL,"¥t");
                               Ofile << CC ;
                               Ofile <<endl;
                           }
                    }

                }
                in_file.close();
            }

        }
    }
```

A_2.記事文書をソートするプログラム（bush）

```
function all_sort {
for list in */
do
    cd $list
    for dats in *.txt
    do
        if [ -e $dats ]
        then
            tek=${dats%*".txt"}
            sort $dats | cat > sort/$tek.txt
        fi
    done
    for check in *
    do
        if [ -d $check ]
        then
            all_sort
        break
        fi
    done

    cd ..
done
}
```

A_3.単語の出現回数を数えるプログラム（C++）

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
void main (int argc,char *argv[])
{
    char A[50],C[100],D[100];
    int I=0,t;
        int count=0;
        strcpy(A,argv[1]);
        strcat(A,".txt");
        ifstream Ifile(A);
        strcpy(A,"bm¥¥c");
        strcat(A,argv[1]);
        strcat(A,".bm");
        ofstream Ofile(A);
    while(!Ifile.eof())
    {
                Ifile.getline(&C[0],sizeof(C));
                if(strstr(&C[0],&D[0])!=0 && strstr(&D[0],&C[0])!=0)
                {
                        count++;
                }
                else
                {
                        if(count>0)
                        {
                                t=strlen(D);
                                D[t-1]='¥0';
                            Ofile<<D<<"        "<<count<<endl;
                        }
                        count=1;
                        strcpy(&D[0],&C[0]);
                }
}}
```

A_4.A_3 を全文書に適用させる為のプログラム（bush）

```
function t_count {
for list in */
do
    cd $list
    for dats in *.txt
    do
        if [ -e $dats ]
        then
            tek=${dats%*".txt"}
            count $tek
        fi
    done
    for check in *
    do
        if [ -d $check ]
        then
            t_count
        break
        fi
    done

    cd ..
done
}
```

A_5 .コサイン尺度を計算するプログラム（C++）

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>

void main(int argc,char *argv[])
{
        char A[100],D[100],name[50];
        int   a,b,d;
        long aa=0,dd=0,ad=0;
       double Raa,Rdd,aadd,ans;
        ifstream in_file(argv[1]);
        ofstream o_file("ksain.txt",ios::app);
        while(! in_file.eof())
        {
            in_file.getline(&A[0],100);
            b=0;
            while(1)
            {
                if(A[b]=='¥t')
                    {
                        A[b]='¥0';
                        a=atoi(&A[b+1]);
                        break;
                    }
                b++;
            }
            aa=aa+(a*a);
            strcpy(name,argv[2]);
            ifstream ch_file(name);
            while(! ch_file.eof())
            {
```

```
                    ch_file.getline(&D[0],100);
                    if(strstr(&D[0],&A[0])!=0)
                    {
                            b=0;
                            while(1)
                            {
                                    if(D[b]==' ')
                                    {
                                            d=atoi(&D[b+1]);
                                            break;
                                    }
                                    b++;
                            }
                            ad=ad+(a*d);
                            break;
                    }
            }
    }
    ifstream ch_file(name);
    while(! ch_file.eof())
    {
            ch_file.getline(&D[0],100);
            b=0;
            while(1)
            {
                    if(D[b]==' ')
                    {
                            d=atoi(&D[b+1]);
                            break;
                    }
                    b++;
             }
             dd=dd+(d*d);
    }
    Raa=sqrt(aa);
    Rdd=sqrt(dd);
```

```
        aadd=Raa*Rdd;
        ans=ad/aadd;
        b=0;
        while(1)
        {
                if(name[b]=='.')
                {
                        name[b]='¥0';
                        break;
                 }
                 b++;
        }
        o_file<<ans<<" "<<name<<endl;
}
```

A_6.A_5を全文書に適用させる為のプログラム（bush）

```
function kos {
for list in */
do
    cd $list
    for dats in *.bm
    do
        if [ -e $dats ]
        then
            kosains $1 $dats
        fi
    done
    for check in *
    do
        if [ -d $check ]
        then
            kos
        break
        fi
    done

    cd ..
done
```

付録 B　プログラムリスト

B_1.記事を文書ごとに分割するプログラム（C++）

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>

void main(void)
{
    char  mei[6]="名詞-",dou[6]="動詞-",kei[8]="形容詞-",fuku[6]="副詞
    -",o[19]="abcdefghijklmnopqr";
        char file_name[20],A[600],tit[100],chk[100],*B,C[3],gg[2],rr[2];
        char *AA,*BB,*CC;
        int f,q;
        for(f=0;f<=4;f++)
        {
                gg[0]=o[f];
                gg[1]='¥0';
                for(q=0;q<=17;q++)
                {
                    rr[0]=o[q];
                    rr[1]='¥0';
                    strcpy(file_name,"date¥¥mai95.");
                    strcat(&file_name[0],&gg[0]);
                    strcat(&file_name[0],&rr[0]);
                    strcat(file_name,".chasen");
                    ifstream in_file(file_name);

                    while(! in_file.eof())
                    {
                        ofstream Ofile(tit,ios::app);
```

```
                in_file.getline(&A[0],sizeof(A));
                if(isdigit(A[0])!=0&& isdigit(A[1])!=0
               &&isdigit(A[2])!=0&& strstr(&A[0],&chk[0])==0)
                {
                    B=strtok(&A[0],".");
                    strcpy(&tit[0],"kizi¥¥");
                    C[0]=A[2];
                    C[1]=A[3];
                    C[2]=NULL;
                    strcat(&tit[0],&C[0]);
                    strcat(&tit[0],"¥¥");
                    strcat(&tit[0],B);
                    strcat(&tit[0],".txt");
                    strcpy(&chk[0],B);
                }
               else if(strstr(&A[0],&chk[0])==0 && strlen(&A[0])>=5)
               {

                     AA = strtok(&A[0],"¥t");
                    BB = strtok(NULL,"¥t");
                    CC = strtok(NULL,"¥t");
                    Ofile << CC ;
                    Ofile <<endl;
                }

                }
                in_file.close();
            }

        }

}
```

B_2.ツールキットの結果より言語モデルを作成するプログラム（C++）

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

void main (int argc,char *argv[])
{
        char line[200],line2[200],*QA,*QB,*QC;
        char *t,*u,*v,*w;
        char name[20];
        char AA[10]="0",AB[50]="0",AC[10]="0";
        char DA[10]="0",DB[50]="0",DC[10]="0";
        char EA[10]="0",EB[50]="0",EC[10]="0";
        char BAA[10]="0",BAB[50]="0",BAC[50]="0",BAD[10]="0";
        char BBA[10]="0",BBB[50]="0",BBC[50]="0",BBD[10]="0";
        char CA[14]="0",CB[50]="0",CC[50]="0",CD[50]="0";
        int check=0,ca=0,cb=0,cc=0,cd=0,cg=0,ce=0;
        double ans=0,gg=0;
        ifstream B_file(argv[2]);
        strcpy(name,"tri¥¥tri");
        strcat(name,argv[1]);
        ofstream O_file(name);
        while(!B_file.eof())
        {
                ifstream A_file(argv[1]);
                B_file.getline(&line2[0],sizeof(line2));
                if( strlen(&line2[0])<=5)break;
                QA = strtok(&line2[0]," ");
                QB = strtok(NULL," ");
                QC = strtok(NULL,"¥n");
                for(int r=0;r<=50;r++)
                {
                        if(QC[r]=='¥0')QC[r-1]='¥0';
```

```
}

memmove(QC,QC+1,50);
check=0;
ca=0;
cb=0;
cc=0;
cd=0;
ce=0;
cg=0;
gg=0;
ans=0;
while(!A_file.eof())
{
        A_file.getline(&line[0],sizeof(line));

        if(check==4 && strlen(&line[0])>=11)
        {
                t = strtok(&line[0]," ");
                u = strtok(NULL,"¥t");
                v = strtok(NULL,"¥n");
                 if(cg==0      &&      strstr(u,QA)!=0      &&
                 strstr(QA,u)!=0 )
                 {
                        strcpy(&EA[0],t);
                        strcpy(&EB[0],u);
                        strcpy(&EC[0],v);
                        cg=1;
                 }
                  if(ce==0      &&      strstr(u,QB)!=0      &&
                  strstr(QB,u)!=0 )
                  {
                      strcpy(&DA[0],t);
                      strcpy(&DB[0],u);
                      strcpy(&DC[0],v);
                      ce=1;
```

```
                    }
                    if(ca==0    &&    strstr(u,QC)!=0    &&
                    strstr(QC,u)!=0 )
                    {
                        strcpy(&AA[0],t);
                        strcpy(&AB[0],u);
                        strcpy(&AC[0],v);
                        ca=1;
                    }
                }

        if(check==5 && strlen(&line[0])>=11)
        {
                    t =strtok(&line[0]," ");
                    u =strtok(NULL," ");
                    v =strtok(NULL," ");
                    w =strtok(NULL,"¥n");
                    if(cb==0 && strstr(v,QC)!=0 && strstr(QC,v)!=0
                    && strstr(u,QB)!=0 && strstr(QB,u)!=0 )
                    {
                        strcpy(&BAA[0],t);
                        strcpy(&BAB[0],u);
                        strcpy(&BAC[0],v);
                        strcpy(&BAD[0],w);
                        cb=1;
                    }
                    if(cc==0 && strstr(u,QA)!=0 && strstr(QA,u)!=0
                    && strstr(v,QB)!=0 && strstr(QB,v)!=0)
                    {
                        strcpy(&BBA[0],t);
                        strcpy(&BBB[0],u);
                        strcpy(&BBC[0],v);
                        strcpy(&BBD[0],w);
                        cc=1;
                     }
```

```
            }

            if(check==6 && cd==0 && strlen(&line[0])>=11)
                    {
                        t = strtok(&line[0]," ");
                        u = strtok(NULL," ");
                        v = strtok(NULL," ");
                        w = strtok(NULL," ");
                        if(strstr(u,QA)!=0    &&    strstr(QA,u)!=0    &&
                        strstr(v,QB)!=0      &&      strstr(QB,v)!=0      &&
                        strstr(w,QC)!=0 && strstr(QC,w)!=0   )
                        {
                              strcpy(&CA[0],t);
                              strcpy(&CB[0],u);
                              strcpy(&CC[0],v);
                              strcpy(&CD[0],w);
                              cd=1;
                        }
                    }

        if(strstr(&line[0],"-grams:")!=NULL && strlen(&line[0])==9)
        {
                check++;
        }
}
if(ca==0)
{
        strcpy(AA,"0");
        strcpy(AB,"0");
        strcpy(AC,"0");
}
if(cb==0)
{
        strcpy(BAA,"0");
        strcpy(BAB,"0");
```

```
            strcpy(BAC,"0");
            strcpy(BAD,"0");
        }
        if(cc==0)
        {
            strcpy(BBA,"0");
            strcpy(BBB,"0");
            strcpy(BBC,"0");
            strcpy(BBD,"0");
        }
        if(cd==0)
        {
            strcpy(CA,"0");
            strcpy(CB,"0");
            strcpy(CC,"0");
            strcpy(CD,"0");
        }
        if(ce==0)
        {
            strcpy(DA,"0");
            strcpy(DB,"0");
            strcpy(DC,"0");
        }
         if(atof(AA)!=0 && atof(DA)!=0)
        {
            if(atof(BAA)!=0)
            {
                ans=atof(BAA);
            }
            else
            {
                ans=atof(DC)+atof(AA);
            }
            if(atof(CA)!=0)
            {
                ans=atof(CA);
```

```
                }
                else if(atof(BBA)!=0)
                {
                    ans=ans+atof(BBD);
                 }
                 gg=atof(BBA);
                 if(atof(BBA)==0)
                 {
                          gg=atof(EC)+atof(DA);
                 }
                 ans=ans+atof(EA)+gg;
                 ans=pow(10,ans);
                 }
                 O_file<<QA<<' '<<QB<<' '<<QC<<' '<<ans<<endl;
                 A_file.close();
        }
}
```

B_3.B_2 を全文書に適用させるプログラム（bush）

```
function trigra {
for list in */
do
    cd $list
    for dats in *.lm
    do
       if [ -e $dats ]
       then
            tri_count $1 $dats
       fi
    done
    for check in *
    do
        if [ -d $check ]
        then
             trigra
        break
        fi
    done

    cd ..
done
}
```

B_4.検索質問文書のその他の確率を求めるプログラム

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

void main (void)
{
    char A[100],B[100];
    char *a,*b,*c,*d;
    char *e,*f,*g,*h;
    double aa,bb,cc=0,ans=0;

    ifstream Ifile1("trit.txt");
    ofstream Ofile("out.txt",ios::app);
    while(!Ifile1.eof())
    {
            Ifile1.getline(&A[0],sizeof(A));
            if(strlen(A)<=5)break;
            a = strtok(&A[0]," ");
            b = strtok(NULL," ");
            c = strtok(NULL," ");
            d = strtok(NULL,"¥n");
            aa = atof(d);
            cc = cc + aa;
            Ofile<<cc<<endl;
    }
            cc = 1 - cc;
            Ofile<<cc<<endl;
}
```

B_5.KL 情報量による近似値を求めるプログラム。

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

void main (int argc,char *argv[])
{
        char A[100],B[100];
        char *a,*b,*c,*d;
        char *e,*f,*g,*h;
        double aa,bb,cc=0,ans=0,dd=0,ee=0.909293,gg;

        ifstream Ifile1(argv[1]);
        ifstream Ifile2(argv[2]);
        ofstream Ofile("outs.txt",ios::app);
        while(!Ifile1.eof())
        {
                Ifile1.getline(&A[0],sizeof(A));
                if(strlen(A)<=5)break;
                Ifile2.getline(&B[0],sizeof(B));
                e = strtok(&B[0]," ");
                f = strtok(NULL," ");
                g = strtok(NULL," ");
                h = strtok(NULL,"¥n");
                a = strtok(&A[0]," ");
                b = strtok(NULL," ");
                c = strtok(NULL," ");
                d = strtok(NULL,"¥n");
                aa = atof(d);
                gg=atof(h);
                if(atof(h)==0)
                {
                        gg=0.00001e-10;
```

```
        }
        cc = cc + gg;
        bb = aa/gg;
        ans = ans+(aa*log(bb));
        //   Ofile<<e<<" "<<f<<" "<<g<<" "<<ans<<endl;
    }
    cc = 1 - cc;
    dd = ee/cc;
    ans=ans+(ee*log(dd));
    Ofile<<ans<<" "<<argv[2]<<endl;
}
```

B_6 .B_5 を全文書に適用させる為のプログラム（bush）

```
function tri_ans {
for list in */
do
    cd $list
    for dats in *.lm
    do
       if [ -e $dats ]
       then
            tri_anser $1 $dats
       fi
    done
    for check in *
    do
        if [ -d $check ]
        then
             tri_ans
        break
        fi
    done

    cd ..
done
}
```