解 説

大規模データのための対話的テキストマイニング

吉田 一星
日本アイ・ビー・エム（株）東京基礎研究所

テキストマイニングの起こり

テキストマイニングとは，自然言語で書かれたテキストに対して，自然言語処理やデータマイニングの技術を応用して知見を得ることである．

本来この2つの技術は，目指すところの応用は異なっている．自然言語処理の目標は，コンピュータがテキストの意味を人間と同様に理解できることにある．その過程で，テキスト中から単語や係り受け（「何がどうした」といった，単語間の関係）などの意味のある単位を抽出する技術が，現在も盛んに研究されている．一方でデータマイニングは，その起源は「大量のデータから役に立ちそうなルールを抽出する」ことであり，したがって抽出技法が高速であることが求められる．データマイニングが扱うデータは，いわゆる関係データベースが扱う表の形式になっているもの(構造化データと呼ぶ)が対象である．例として，「商品の売上データベース」「人事情報データベース」などはおしなべて表形式である．

さて，昨今のWebページに代表されるように，蓄積されるテキストの量は増加の一途を辿っている．「統一されたフォーマットがない」「書く人間に依存する表記のゆれ」などの理由により，テキスト文書内の情報を集計したり検索したりするのは容易ではない．しかし，データ量はすでに人間が読んで解釈できる限度を超えている．Webページはもちろんのこと，より特定のデータでもそのような例はたくさんある．MEDLINEという医療文献の抄録を集めたデータベースは，1,600万件以上の文献情報を保持している[1)]（2007年8月現在）．企業が保管する顧客からの問合せログ（顧客の声はテキストとして保管される）も，数百万件という規模になることが珍しくなくなっている．問題なのは，これらがテキストとして保管されていることである．大規模データを得意とするデータマイニングの手法は，データが表として用意され，表の列ごとに値の範囲が決まっていることが前提のため，テキストデータに直接適用することができない．

このような状況を打破するため，テキストから意味のある単位を抽出する自然言語処理と，データマイニングとを組み合わせてみようというアプローチが起こったのは自然であろう．言い換えれば，データマイニングの手法が適用できるようにテキストを加工してあげればよく，自然言語処理の技術がその目的にうまく適用できるのである．この組合せによって，言語処理・データマイニングの個別の手法が適用できなかった，大規模なテキストデータの解析が可能になった．

本稿では，自然言語処理とデータマイニングの組合せのうち，ユーザからの問合せに対して結果を高速に（標語的には，「検索エンジンに近い応答速度で」）返す仕組みについて解説する．本稿ではこの仕組みを対話的テキストマイニングと呼ぶことにする．以下，タスクの定義，実現するためのデータ構造とアルゴリズム，その応用，の順で説明する．

なお，自然言語処理・データマイニング共に，統計・機械学習・計算幾何学など他分野のさまざまな道具が近年積極的に取り入れられるようになり，本稿で述べる要素技術およびその組合せは，その多様なアプローチの一例であることをお断りしておく．

対話的テキストマイニングとは

本稿が対象とする対話的テキストマイニングの概要を見てみよう．くだけた言い方でその目的を述べると，「適

当にテキストの集合を選んだとき，そこにたくさん出現する単語が何であるかを知りたい」ということである．この目的のために，以下の順で入力データを処理していく．

ステップ 1. 分析の対象となる可能性のあるすべてのテキスト文書(以下，文書と略す)から，単語や係り受け表現といった「キーワード」を抽出する

ステップ 2. ステップ 1 で抽出した情報を，対話的な分析のために必要な索引構造に保管する

ステップ 3. ステップ 2 の索引を用いて，ユーザからの問合せに対して，キーワードとその頻度からなるリストを計算し，ユーザに返す

それぞれのステップで行われる処理の概要を見ていこう．ステップ 1 では，与えられた文書を自然言語処理の技術を用いて，意味のある属性値の集合に変換する．たとえば以下の文書が入力データとして与えられたとしよう．

> パソコンの設定を変更したら，インターネットに接続できなくなった．

このとき，自然言語処理の手法によって，たとえば次のような情報を抽出することができる．

> [ハードウェア：パソコン]
> [名詞：設定]
> [名詞：インターネット]
> [動詞：変更する]
> [動詞：接続する]
> [係り受け：設定 - 変更する]
> [係り受け：インターネット - 接続できない]

[ハードウェア：パソコン]という表記は，文書から抽出された単語が「パソコン」で，そのキーワードに対して「ハードウェア」というラベル(これをカテゴリと呼ぶ)が付与されたことを示す．[動詞：変更する]は，元のテキストに「変更したら」という形で出現している動詞が，やはり自然言語処理によって終止形に正規化されていることを示す．これは，集計の際に「変更する」「変更すれば」「変更させる」などの表現の違いを吸収する役目がある(この違い自体を分析対象にする研究ももちろんあるが，本稿では省略する)．本稿では，この「カテゴリと単語(または係り受け)」の組のことをキーワードと呼ぶ．いったんキーワードの形式になってしまえば，それが単語であろうと係り受けであろうとステップ 2 以下の手法を統一的に適用できる．また，日付や文書作成者など，文書に(テキストとは別途)付随している構造化データもキーワードとして処理することにより，構造化データと非構造化データとを統一的に扱うことができる．

ユーザが分析対象として指定する文書集合は動的に変わり得る（検索エンジンで例えれば，ユーザが事前にどんな検索語を使うかは予測できない)．このためステップ 1 は，ユーザが指定し得るすべての文書に対して前処理を行っておく．この「すべての文書」を全文書集合と呼ぶことにしよう．

ステップ 2 以下が本稿で述べる主題である．ステップ 2 では，ステップ 1 で抽出したキーワードを，それが抽出された文書の ID（文書を一意に識別するための番号）とともに外部記憶装置(主にハードディスク)に保管する．このとき，ステップ 3 での計算を高速に行うための索引構造を用いて保管する．ステップ 3 では，その索引を用いて，ユーザが指定した文書集合の中に頻出するキーワードの集合を求める．このタスクの明確な定義は次章で述べるとして，この一連の仕組みによって何が可能になるのかを例で示す．

例：企業のコールセンタは，電話やインターネット経由で顧客からの問合せ・苦情を受け付ける窓口である．いま，ある製品 X についてどのような不具合が報告されているか知りたいとしよう．このとき，「製品 X について記述されている文書」は，検索を用いて知ることができる．検索で見つかった文書集合の中で，係り受けカテゴリのキーワードで頻出するものが分かれば，その中に[インターネット - 接続できない]のような，問題に由来する表現が見つかる可能性が高まる．

例：同様に，医療文献データベースに対し，ある疾患 Y と関連する遺伝子を知りたいとする．この場合も，まず検索を用いて「疾患 Y について書かれた文献」に絞り込み，引き続きその絞り込まれた文書集合中に頻出する遺伝子名を調べればよい．

むろん，ある検索条件下で特定の表現が頻出するからといって，その条件と頻出表現との間に必ずしも意味的に深い関連があるとは限らない．しかし，文書数が膨大になってくれば，関連する表現が同一文書に出現する頻度は高くなるであろう．さらに，頻出するキーワードに対し，それとユーザが指定した条件との関連性を表す指標(相関値と呼ぶ，後述)を組み合わせて用いると，より精度の高い分析ができる．以下の章で，テキストデータに対してこれらの手法の有用性，およびその限界について述べる．

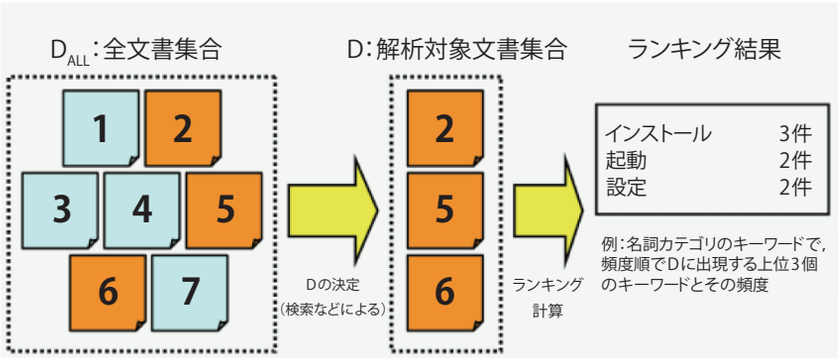

図 -1　ランキングの処理手順

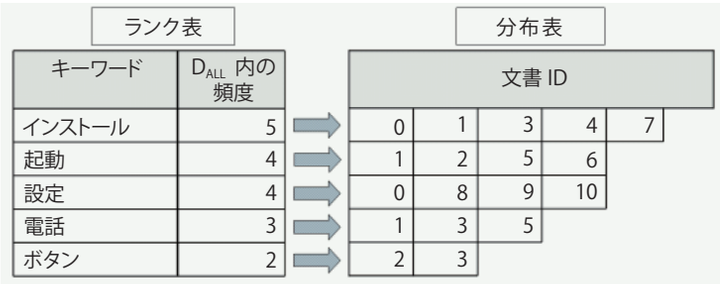

図 -2　ランキング索引の概念図

以上を要約すると，「検索などの手段でユーザが指定した条件下で」「テキスト内に最も多く出現する表現を列挙する」仕組み，が本稿で述べる対話的テキストマイニングの中核である．

高速化のためのデータ構造とアルゴリズム

ここでは，対話的テキストマイニングを高速に実現するための，言い換えるとユーザに分析結果を即座に返すための仕組みを述べる．

まず，問題をフォーマルに定義しよう．前章で述べた全文書集合を D_{ALL} と書くことにする．あらかじめ，D_{ALL} 内の各文書に対して，自然言語処理によってキーワードが抽出済みであるとする．D_{ALL} の部分集合 D を何らかの手段（大抵は，検索）によって指定したときに，「D に含まれるキーワードを，頻度の高い順に k 個求める」という処理を「ランキング」と呼ぶことにする．この D を解析対象文書集合と呼ぶ．解析対象文書集合は，ランキング実行のたびにユーザから動的に指定される．

具体例に対応させれば，D_{ALL} が「蓄積されたすべての顧客問合せログ」，D が「製品 X について書かれた文書全体」，頻度順で上位 100 個のキーワードを求めたい場合は k=100, となる．図 -1 にランキング処理手順を示す．このように，あるスコアを基準として上位固定件数の結果を返すタスクは，一般に Top-k Query として知られている．ここでは頻度は文書頻度を指すものとする．すなわち，1 文書に同じキーワードが何回出現しても，その文書における頻度は 1 とカウントする．

なお，各キーワードには自然言語処理によってカテゴリが付与されていることを前述したが，以下の定式化ではある 1 つのカテゴリ C のキーワードのみを計算対象とする．ハードウェア名と動詞とを混在させてランキングしても効果的でないことは明らかであろう．

さて，$D = \{d_1, d_2, ..., d_m\}$ をユーザが指定した解析対象文書集合，$T(C) = \{t_1, t_2, ..., t_n\}$ をカテゴリ C のキーワード全体の集合とする．また，$t \in T(C)$ に対し，Doc (t, D) を，「D の文書でかつキーワード t を含むもの全体の集合」と定義する．ランキングとは，T(C) の要素を ♯Doc (t, D) すなわち Doc (t, D) の要素数でソートしたときの，上位 k 個を取得する処理にほかならない．

図 -2 は，ランキング計算に用いるデータ構造（ランキング索引）の概念図である．この索引は，ランキング計算に必要なキーワードと文書の対応を，あらかじめ計算済みの頻度情報とともに保持している．この索引は各カテゴリ C に対し，ランク表と分布表の 2 つの表からなる．ランク表は，C のすべてのキーワードを D_{ALL} 内の頻度降順にソートし，キーワード t と頻度 f := ♯Doc (t, D_{ALL}) のペア (t, f) を並べた構造，および各ペアから分布表へのポインタを持つ．分布表は，ランク表の各ペア (t, f) に対し，Doc (t, D_{ALL}) に含まれる各文書の文書 ID（文書を一意に指定する整数で，あらかじめ付与しておく）のリストを持つ．図 -2 の例では，♯Doc ("インストール ", D_{ALL}) = 5 であり，キーワード "インストール " は文書 ID が 0, 1, 3, 4, 7 の 5 文書に含まれる．この索引を，前処理時に作成しておく．

次に，対話的処理時にこの索引を用いるランキング計算のアルゴリズムを，具体例で見ていこう（図 -3）．

いま解析対象文書集合が D = {0, 1, 2, 3, 4, 5} である

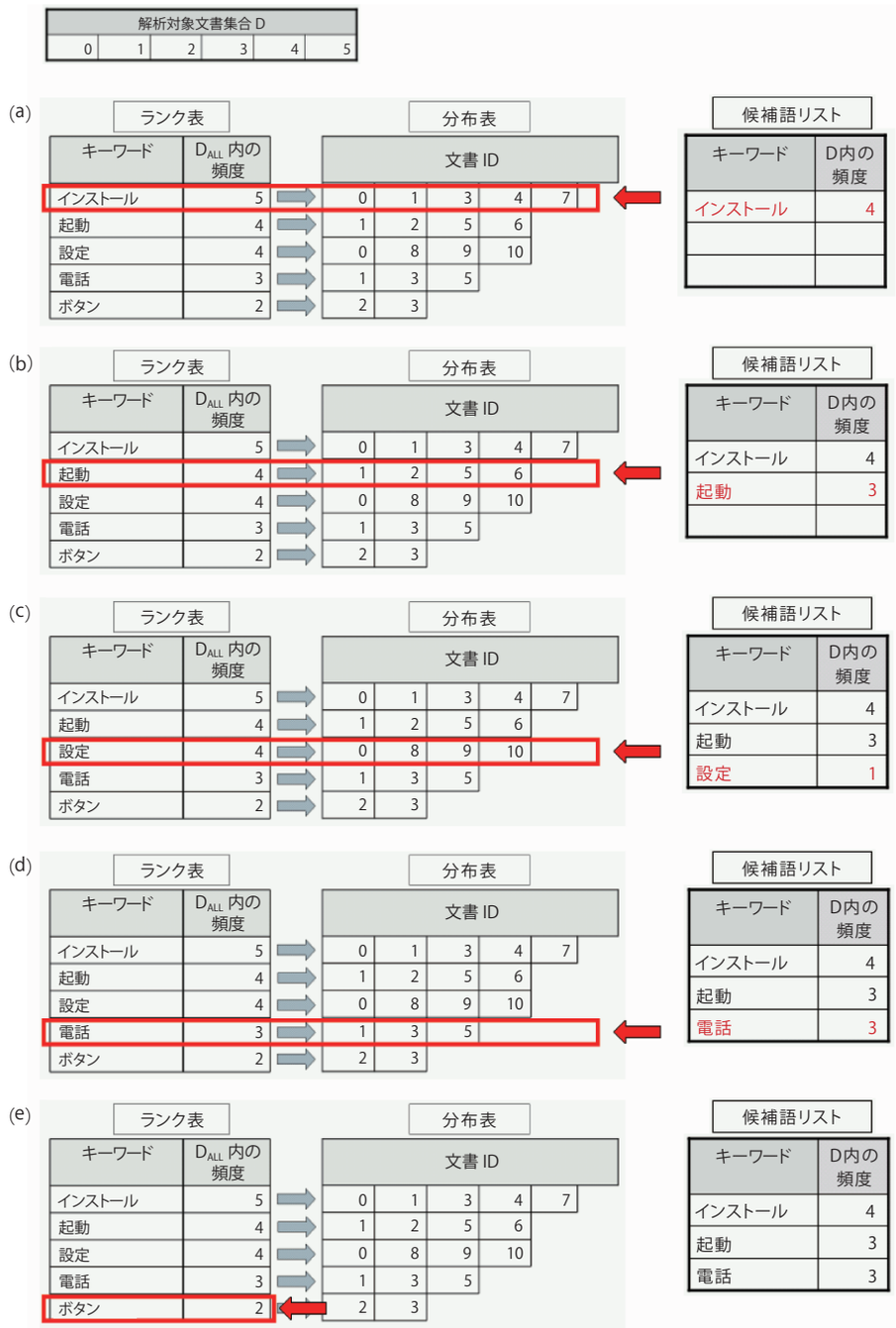

図 -3　ランキング計算アルゴリズム

とする．D に出現する頻度順上位 k=3 個のキーワードを求めよう．答えとなる 3 個が完全に求まるまで，ランク表および分布表を上から順に読んでいく．まず，ランク表の 1 番目のキーワード「インストール」は D 内に 4 文書 (= 0, 1, 3, 4) 出現することが分布表を読むと分かる (図 -3 (a))．この時点で「インストール」をランキング結果の候補として，頻度の値 4 とともに記憶しておく．同様に，「起動」「設定」を順に読むと，D 内の頻度がそれぞれ 3, 1 であり，これらも解の候補として記憶する (図 -3 (b), (c))．次に「電話」を読むと，D 内の頻度が 3 であるため，それまで候補として保管していた頻度 1 の「設定」を候補から落とし，代わりに「電話」を候補に入れる (図 -3 (d))．このように，ランク表と分布表を読み進めていくと，(ランク表を読み終えてしまった場合を除いて) 候補のリストには常に k 個のキーワードが蓄えられ，新しい候補が見つかると置き換えられていくことになる．

さて，「ボタン」をランク表から読み終えた時点で，「ボタン」の D_{ALL} における頻度が 2 であることが分かる (図 -3 (e))．一方，この時点で候補のリストに入っているキーワードの D における頻度最小値は 3(「起動」と「電話」) である．このことは，「ボタン」が候補に入り得ないだけでなく，ランク表で「ボタン」以下にあるどのキーワードも候補にならないことを示している．したがってこ

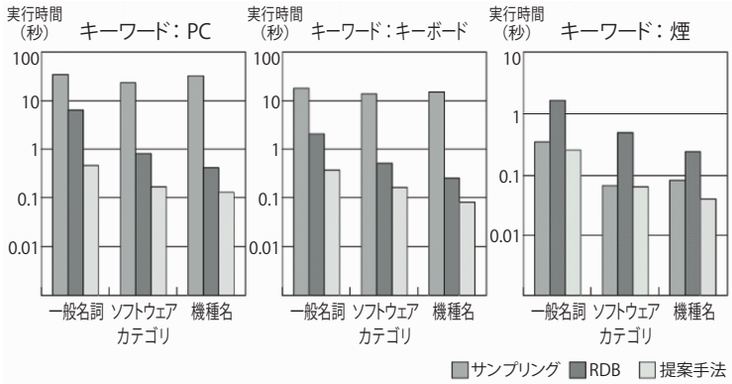

図 -4　ランキング計算時間の比較 [3]（実行時間軸は対数目盛り）

の時点で計算を停止し，候補リストに入っている語とその頻度の組を結果とすればよい．

この停止条件を用いた計算は，Threshold Algorithm と呼ばれる手法 [2] をランキング索引に自然に応用したものである．このアルゴリズムが高速に動作することの本質は，上のアルゴリズムにおける停止条件の適用と，索引が保管されるディスクへのアクセスがシーケンシャルアクセスのみで済むことの 2 点にある．後者は，ランク表・分布表共にディスクの連続した領域に保管しておくことにより，表を上から順に読むだけで計算が実行できることによる．

ランキングを計算する素朴な方法として，各文書 ID から，その文書に含まれるキーワードのリストを求める別の索引構造「文書索引」を用意して，D の中の各文書に出現するキーワードを足し合わせることが考えられる．この手法では，D の各文書にアクセスする際にディスクへのランダムアクセスが発生し，D の文書数に比例する計算コストがかかる．後の実験結果で示すように，この方法は D の文書数が少ない場合を除いてきわめて遅く，文書数を絞るためサンプリングを行うのが一般的である．

また，ランキング索引は，検索エンジンで一般的に用いられる転置索引とよく似ているが，転置索引は頻度順でソートされていない(大抵はアルファベット順である)．そのため，上のアルゴリズムを適用しようとすれば，やはりディスクへの低速なランダムアクセスを避けられない．情報検索のタスクでは，1 個のキーワードに対して 1 回のランダムアクセスで文書のリストを取得して終わりなのに対し，ランキングの場合ランダムアクセスが大量に発生することで対話性が落ちてしまうのである．

一方，ここで紹介したランキング計算手法は，D の文書数がきわめて小さく，かつキーワードの異なり数(ランク表に含まれるキーワードの個数)が極端に大きい場合にかぎり，終了条件に到達するまでに膨大な量のランク表および分布表を読み結果的に遅くなることがあり得る．本稿では詳細を割愛するが，これを解消するため，ランキング索引と文書索引をうまく使い分けることによってパフォーマンスを安定させることも可能である．

■パフォーマンス評価

次に，著者らが文献 3）で紹介している，このアルゴリズムの有効性を示す実験結果を述べる．比較対象として，文書索引によるサンプリング手法および関係データベース（RDB）による実装を挙げる．サンプリングは，解析対象文書集合 D からランダムに 10,000 文書(D に含まれる文書数がそれ以下の場合は，すべて)を選び，上で述べた文書索引を用いてキーワードの頻度分布を計算する．なお，RDB 実装の詳細は，文献 6）を参照されたい．

IBM PC コールセンターのコールログ 324,677 文書に対し，[一般名詞] [ソフトウェア] [機種名]の各カテゴリについて，頻度順で上位 1,000 個のキーワードを求めるランキングの計算時間を測定した結果を図 -4 に示す．解析対象文書は，「PC」(高頻度語，全文書中 19,810 文書に出現)，「キーボード」(中頻度語，同 4,810 文書)，「煙」(低頻度語，同 4 文書)の各キーワードの検索結果を用いた．この結果から，紹介したランキング計算手法が，サンプリングや RDB 実装に比べて数倍から数十倍高速なことが分かる．

応　用

本章では，ランキング計算のためのデータ構造とアルゴリズムがテキストマイニング機能にどのように応用されるのかを，MEDLINE を分析対象にしたマイニングアプリケーションの操作例で説明する．本章で用いている例はすべて，MEDLINE データから論文誌名で絞り込んだ約 274 万文書を対象としている．また，個々の

キーワード：hypertension (Disease (List)) を含む　23794件 削除

キーワード	頻度	相関値
glucose	1146	2.7
norepinephrine	555	8.4
hydrochlorothiazide	370	51.8
propranolol	363	7.3
nitroprusside	311	10.6
nifedipine	284	9.1
atenolol	251	27.4
losartan	242	22.0
phenylephrine	207	8.3
epinephrine	189	3.6

図 -5　hypertension で検索した場合のランキング 500 語の上位 10 語

手法については文献 4）が詳しい．

■検索条件との相関分析

文書集合中から目的の文書を探す手段として検索は一般的ではあるが，たとえば「高血圧 (hypertension) に関係する薬物を知りたい」ときはどうすればよいだろうか．キーワードベースの検索では，hypertension というキーワードでヒットした文書が薬物について書かれているかどうか分からないため，各文書を 1 つ 1 つ目で追う必要がある．一方，自然言語処理で「高血圧に XX が効く」といった直接的な表現を抽出し，このパターンを検索する手法もある．この方法では見つかった文書が求めるものである可能性が高い反面，この表現が陽に述べられていないが高血圧と XX の関係を示す文書を逸してしまう．

前章で述べたランキング機能を用いるとこの問題にある程度解答を与えることができる．hypertension をキーワードとして検索し，ヒットした文書集合に対して，薬物カテゴリのキーワード上位 500 語を求めてみよう．図 -5 は，実際にランキングを実行して得られた結果 500 語のうち，上位の 10 語を示す．

約 274 万文書中，hypertension を含む文書が 23,794 文書あり，その中で glucose（ブドウ糖）が最も多く出現，つまり［薬物：glucose］というキーワードが自然言語処理によって抽出されたことを示す．このリストだけで高血圧に関連する情報が得られる場合もあるが，全文書集合に満遍なく出現するようなキーワードはこのリスト内でも上位にきてしまう．そこで検索条件との相関に着目するとより精密な分析が可能になる．ある検索条件 C とキーワード W との相対値 r(C,W) を，出現文書数についての C と D との相互情報量として以下のように定義する [4]．

$$r(C, W) = \frac{\dfrac{(C\text{ を満たしかつ }W\text{ を含む文書数})}{(\text{キーワード }W\text{ を含む文書数})}}{\dfrac{(\text{条件 }C\text{ を満たす文書数})}{(D_{ALL}\text{ に含まれる文書数})}}$$

直感的な言い方をすれば，glucose の相関値が 2.7 であることの意味は，「全文書集合 274 万件の中から適当に文書を選んできたときに比べて，hypertension について言及のある文書の中から適当に文書を選んだときの方が，glucose について書かれている文書を引き当てる確率が 2.7 倍高い」ということである．

図 -5 の結果を，相関値の高い順に結果をソートし直したものが図 -6 である．hydrochlorothiazide のように頻度順で上位にすでに入っていたキーワードもあるが，manidipine（血圧を低下させる薬物の一種）は頻度順では 109 位の単語である．したがって図 -5 のリスト内ではずっと下の方にある．

相関値が 1 より大きいことは，検索条件とそのキーワードとに何らかの関連があることを示す．つまり各々が確率的に独立に生起した場合の共起確率よりも実際に共起した割合の方が高い．さらにその値が高ければ高いほど相関が強いと考えられる．

実際のアプリケーションでは，図 -6 の画面でさらに manidipine をクリックすることにより，hypertension と manidipline を共に含む文書（12 文書）を閲覧できる．また，manidipline をキーワードとして検索条件に追加して（たとえば "hypertension AND manidipline"），分析を続行することもできる．このように，検索条件と相関の高いキーワードに注目していくことによって，目的の文書に効率的に辿り着くことができる．

相関分析においては，試行錯誤によって検索条件をさまざまに変えながらランキングを見るため，アプリケー

キーワード：hypertension (Disease (List)) を含む　23794件 削除

キーワード	頻度	相関値
chlorthalidone	90	53.7
hydrochlorothiazide	370	51.8
doxazosin	54	35.6
amlodipine	106	30.8
manidipine	12	29.2
labetalol	58	28.2
deoxycorticosterone	127	28.1
atenolol	251	27.4
losartan	242	22.0
felodipine	60	21.1

図 -6　図 -5 の結果を相関値でソートした上位 10 語

キーワード：lung (Anatomy) を含む　52539件 削除

サブカテゴリ／キーワード	etoposide 519	cyclophosphamide 486	dexamethasone 447	doxorubicin 405	carboplatin 374	ovalbumin 333	methacholine 311	carbon monoxide 296	glucose 295	heparin 265	indomethacin 225	retinoic acid 207	hydrogen peroxide 184	hydroxyproline 177	oleic acid 174	cycloheximide 166
cancer 13051	456 3.2	260 1.9	41 0.2	269 2.3	354 3.4	6 0.0	2 0.0	26 0.2	47 0.4	15 0.1	20 0.2	63 0.9	20 0.2	3 0.0	4 0.0	16 0.2
adenocarcinoma 2309	18 0.4	16 0.4	15 0.4	16 0.5	12 0.4	2 0.0	0 0.0	1 0.0	3 0.0	6 0.1	11 0.5	11 0.6	4 0.1	0 0.0	2 0.0	6 0.2
syndrome 2125	9 0.2	25 0.8	16 0.5	6 0.1	3 0.0	1 0.0	4 0.1	20 0.9	15 0.7	8 0.3	4 0.1	3 0.0	8 0.4	3 0.1	21 1.7	3 0.1
asthma 1949	0 0.0	3 0.0	43 1.8	0 0.0	0 0.0	**175 11.9**	**181 13.2**	14 0.6	5 0.1	6 0.2	8 0.3	0 0.0	5 0.2	2 0.0	0 0.0	2 0.0
fibrosis 1303	5 0.1	14 0.6	11 0.5	2 0.0	0 0.0	12 0.7	7 0.3	30 2.6	4 0.1	5 0.2	6 0.3	0 0.0	3 0.1	**93 16.6**	1 0.0	2 0.0
cystic fibrosis 1076	0 0.0	0 0.0	6 0.2	2 0.0	0 0.0	0 0.0	2 0.0	4 0.1	4 0.1	3 0.1	1 0.0	0 0.0	2 0.0	0 0.0	0 0.0	2 0.0
respiratory distress 946	3 0.0	2 0.0	8 0.4	0 0.0	1 0.0	1 0.0	0 0.0	4 0.1	11 1.0	7 0.5	2 0.0	1 0.0	5 0.4	4 0.2	21 3.9	3 0.1

図 7　疾患と薬物カテゴリ間の相関分析結果

ションが対話的に動作することがきわめて重要である．相関値によるソート処理は主記憶上で実行可能で，計算コストはランキング計算に比べて非常に小さいため，ランキング計算の高速化が対話的処理のために本質的である．上で示した分析例において，ごく一般的な PC 上でも 1 回のランキング実行に 1 秒～3 秒しかかかっていない．

■2 つのカテゴリ間の相関分析

次に，「肺 (lung) について言及のある論文に絞ったときの，疾患と薬物との相関を見る」という操作を考えよう．疾患や薬物という指定には，自然言語処理の出力としてキーワードに付与されたカテゴリを使えばよい．

図 -7 は，lung をキーワードとして検索で絞り込んだ上で，疾患カテゴリと薬物カテゴリのキーワードの相関値を計算した結果である．表の縦軸には疾患カテゴリのキーワードが，横軸には薬物カテゴリのキーワードが，それぞれ頻度順で上位 100 個並んでいる（図ではその一部を示している）．これは各カテゴリに対する，lung を含む 52,539 文書におけるランキングの結果に等しい．また，表の各セルには，そのセルに対応する 2 つのキーワードの頻度および相関値が表示されている．頻度は，その 2 つのキーワードが共に出現する文書の数を示す．相関値は，検索条件とキーワードの相関値と同様

図 -8　低頻度キーワード間の相関分析結果

に，2 つのカテゴリからそれぞれ 1 つずつ選んだキーワードの相互情報量として定義する．

この相関分析結果は，ランキング計算アルゴリズムで各カテゴリのキーワードと頻度を取得するときに，キーワードを含む文書 ID も保管しておくことにより，各セルの頻度をオンメモリで計算することで容易に求められる．

たとえば図 -8 において，cancer と etoposide の（lung を含む文書における）頻度は 456 で，その相関値は 3.2 である．アプリケーションは相関値の高いセルを赤くハイライトすることによって，ドリルダウンする価値のある（可能性のある）個所をユーザに伝えている．

検索条件との相関のときと同様，個々のキーワードの出現頻度自体が小さくても，相関値は高くなり得る．図 -8 は図 -7 と同じ相関分析結果の表の，より右下部分を見たものである．

頻度 9 のセルは，疾患カテゴリで頻度 23 位の pneumoniae（肺炎）と，薬物カテゴリで頻度 74 位の azithromycin（抗生物質の一種）の相関を示す．この程度の順位の周りではほとんどのセルの頻度が 0 であるが，このように意味のある相関が見つかることも多い．文書サンプリングによる手法では，たとえばサンプル数を 10,000 にすればこのセルの頻度もおよそ 5 分の 1 になり，サンプルの取り方にもよるがこの相関は見つからない可能性が高い．ランキング計算でサンプリングをせず解析対象文書集合をすべて分析対象にしていることは，このような低頻度かつ重要なキーワードをカバーする上で本質的であると言える．

今後の課題と展望

本稿で述べたランキング計算手法とその応用により，大規模なデータに対してもキーワードと文書の関係を対話的に分析することが可能となった．データの対話的な分析は，ユーザに対する利便性から今後もさまざまな応用で重要になっていくと考える．おわりに，この分野における今後の研究課題を述べる．

そもそも，キーワードと文書頻度ベースの分析には限界がある．テキストが長くなればなるほど，まったく関連のないキーワードがたまたま同じテキストに含まれる可能性が高まり，その結果相関値の信頼性が下がってしまう．これを解決するためには，文献 5）で紹介されているような文書間にまたがる概念の分析や，文書の構造に配慮した分析が必要であろう．本稿で述べた分析の枠組みやアルゴリズムの，この方向への拡張は興味深い課題である．

また，MEDLINE の例のように，ユーザ（研究者や医療従事者）がデータに対する十分な前提知識を持っている場合，そこにさらに差分となるような新規な知識をシステムが提示することは難しい．これは上のキーワードベースの限界とも関連するが，どのような知見が新規と言えるのかを，ユーザのフィードバックを必要ならば取り入れて学習するような仕組みが求められるであろう．

参考文献

1）MEDLINE Fact Sheet, http://www.nlm.nih.gov/pubs/factsheets/medline.html (2007/08/29).
2）Fagin, R., Lotem, A. and Naor, M. : Optimal Aggregation Algorithms for Middleware, Journal of Computer and System Sciences, Vol.66, No.4, pp.614-656 (2003).
3）吉田一星，宅間大介：対話的テキストマイニングのためのソフトウェアアーキテクチャー，Provision Winter 2007, No.52 (2007).
4）Nasukawa, T. and Nagano, T. : Text Analysis and Knowledge Mining System, IBM Systems Journal, Vol.40, No.4, pp.967-984 (2001).
5）小池麻子：テキストマイニングによる潜在的知識の発見支援，情報処理学会誌，Vol.48, No.8, pp.824-829 (Aug. 2007).
6）Inokuchi, A. and Takeda, K. : An Online Analytical Processing of Text Data, ACM Sixteenth Conference on Information and Knowledge Management (2007).

（平成 19 年 9 月 3 日受付）

吉田 一星（正会員）　issei@jp.ibm.com

2001 年東京大学大学院数理科学研究科修士課程修了．同年日本アイ・ビー・エム（株）入社．現在同社東京基礎研究所副主任研究員．テキストマイニングの基盤技術および応用の研究に従事．