平成９年度　卒業研究論文

日本語定型詩における連想空間の記述と解析

茨城大学工学部システム工学科

福士 里子

概要

「短歌」と聞いて思いつくものは何であろうか。「よく貴族たちが詠んでたやつ」、「5･7･5･7･7」、「難しくてよく意味が分からない」。よほど勉強している人でない限り、せいぜいこれくらいであろう。しかし、これも立派な「連想」である。

和歌では、5･7･5･7･7の31文字で自分の言いたいこと、感じたことなどを表現しなければならない。31文字という短さであるがゆえに想像力をたくましくして、とりかからなければならない。こういった状況から連想という習慣が、自然と出来上がっていったのではないだろうか。

はじめに書いてあるように、「短歌」から「意味が分からない」と考えてしまうのは、現代では和歌を詠むという風習が一般の人々にはないからである。ところが平安時代の人に同じく「短歌」から思いつくものを聞いたら、多分「ラブレター」とか「日記みたいなもの」などという答えが返って来るであろう。つまり、連想関係はその時代ごとの特徴を反映していると考えられるのである。

そこで、本研究では、計算機を用いた連想関係の調査を行なう。最終的には、自動検索のツールの開発が目的である。

本論文では、第１章の序論も含め、全７章からなる。
第２章では、研究の背景と目的について述べる。
第３章では、八代集について説明する。
第４章では、単語の抽出に関する手法、用いたシステム等について説明する。
第５章では、連想関係を表現するために導入した係数について説明する。
第６章では、集計した結果について考察を行ない、作成した検索プログラムについて述べる。
第７章で、本研究の成果を述べる。

目次

第１章

序論

むかしむかし、とはいってもそんなに遠いむかしではない時代に、スピニッチ君は、とある小さな村で生まれました。スピニッチ君は物心ついた頃から、毎日村の畑の中で、いろんなことについて考えることが習慣になっていました。例えば、どうして空は青いのだろうかとか、鳥はどうして飛べるのだろうかとか、トマトはどうしてはじめは青いのに赤くなるのだろうかというようなことを１日中考えていました。

スピニッチ君が畑の中で考え事をしていると、時々、その畑の持ち主であるおじいさんとおばあさんがやって来ることがあります。２人ともスピニッチ君に気づいても笑いかけるだけで、話しかけてくることはありませんでした。もちろんスピニッチ君も黙っています。２人は畑の草むしりや水やりなど、しばらく仕事をしたあとで帰っていきます。そんな２人の様子を眺めながらも、スピニッチ君は考え事を続けていました。

ある日曜日、おじいさんとおばあさんの孫が、犬と一緒に畑へ遊びに来ました。犬はスピニッチ君のほうへ来て、「ワン」と吠えました。スピニッチ君は驚いて、とても恐くなったけれども黙っていました。おじいさんとおばあさんの孫が「だめだよ」と言って、犬を連れて行きましたが、スピニッチ君のほうは見向きもしませんでした。スピニッチ君はちょっと悲しくなりました。でも、しばらくすると雲の形について考え始めてしまい、犬のこともおじいさんとおばあさんの孫のことも忘れてしまいました。

―――――――――― 月日は流れて・・・・ ――――――――――

スピニッチ君も大人になりかけたある日、相変わらず畑の中で考え事をしていると、おじいさんとおばあさんが、何か話しをしながらやってきました。おじいさんは今日に限って、スピニッチ君の方へ近づいてきます。スピニッチ君

が「何だろう」と思いながら黙っておじいさんを見ていると、おじいさんはスピニッチ君の前で立ち止まりました。すると、突然おじいさんは、草むしりをしているおばあさんの方を向いて、言ったのです。「れんそうくうかーーん」と。

「えっ、れんそうくうかん、って何?」と、スピニッチ君が考えていると、おばあさんがおじいさんに向かって「そうですねー」と返事をしました。つぎの瞬間、おじいさんはスピニッチ君の体をつかんで連れていってしまったのです。

一体、なにが起こったのでしょうか?もう一度、おじいさんとばおばあさんの会話を再現してみましょう。

おじいさん:「ほぅれんそう、くうかーー、んーー?」
おばあさん:「そうですねー」

もうお分かりいただけたでしょうか。そうです、スピニッチ君はほうれん草だったのです。

スピニッチ君、今度人間に生まれ変わってきたときは、「れんそうくうかん」についてよーく調べてみてね。

第２章

背景と目的

本研究で用いる連想とは、ある単語を聞いたとき、それに関連した別の単語を思いつくことである。たとえば、「はな」と聞いたときに「ゆき」を連想することである。なぜ、「はな」から「ゆき」を連想するのかというと、「はな」→「桜の花」→「桜の花が散る」→「白くて散るもの」→「ゆき」という構造からである。この連想は、和歌の世界ではよく用いられる。

ある時代の、ある階級で共通にもたれている連想関係を集計することによって、的確な裏付けを行うことができる。しかし、従来のこのような研究では、単語の集計等の作業をすべて手作業で行ってるため、大量に調べることは困難である。たとえば、２５年前の論文で、万葉集と古今集の一部についての連想関係を調べたものがある。これは、ある程度予測された名詞の組み合わせで、出現回数をカウントしただけであるが、すべて手作業で行ったため、３ヶ月もの時間がかかったそうである。このように、手作業では、調べる幅が広がらない。

そこで、本研究ではデータベースや古文の形態素解析システム、その他の手法を用いることによって、自動的かつ客観的に連想関係の調査を行う。特に、計算機を用いることによる自動検索のツールの開発を目的とする。

第３章

八代集

八代集とは、以下の八つの勅撰和歌集の総称である。

古今和歌集	９０５年	１１１１首	紀貫之
後撰和歌集	９５１年	１４２０首	中臣能宣、清原元輔
拾遺和歌集	１０１１年	１３５０首	藤原公任
後拾遺和歌集	１０８６年	１２１８首	藤原通俊
金葉和歌集	１１２７年	６５０首	源俊頼
詞花和歌集	１１５４年	４１１首	藤原顕輔
千載和歌集	１１８７年	１２００首	藤原俊成
新古今和歌集	１２０５年	１９８０首	藤原定家
	合計	９３４０首	

まだ歌道盛んな頃のもので、作品としても優れ後代に重んぜられた。

本研究で用いるデータとして、(有)文化工学舎のフロッピィーディスクのテキストファイルを利用した。内容は、歌番号、歌の部、歌、作者で構成されている。ただし、すべて歌は平仮名のべた書きになっており、濁点もついていない。

第４章

単語の抽出

4.1　dBXL を用いた単語集計

まずはじめに、単語の出現回数を調べるために、データベースマネージメントシステム dBXL を用いて、単語の集計を試みた。

4.1.1　データベースマネージメントシステム（ＤＢＭＳ）

データベースマネージメントシステム（ＤＢＭＳ）とは、その名の通りデータを管理するシステムのことである。

ファイル仕様の明らかでないアプリケーションソフトを利用してシステムを構築すると、データとの関係が図 4.1 のように１対１になり、後でそのデータを利用するにはかなりの技術を要する。

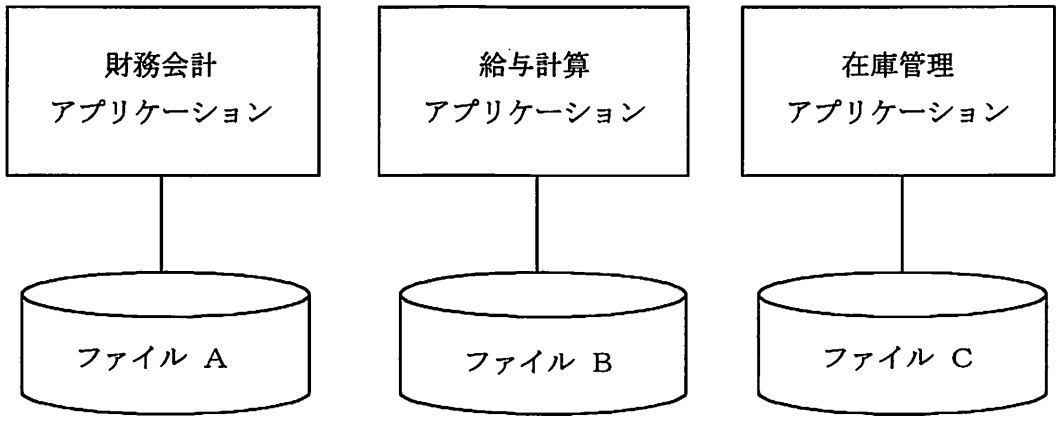


図 4.1 従来のアプリケーションソフト

しかし、データベースマネージメントの上にアプリケーションを作成すれば、データの取り扱いが非常に簡単になる。データベースマネージメントシステムでは、データベースとアプリケーションとは独立して存在している。各アプリケーションがデータベースを使用するときには、ＤＢＭＳを通して行う。

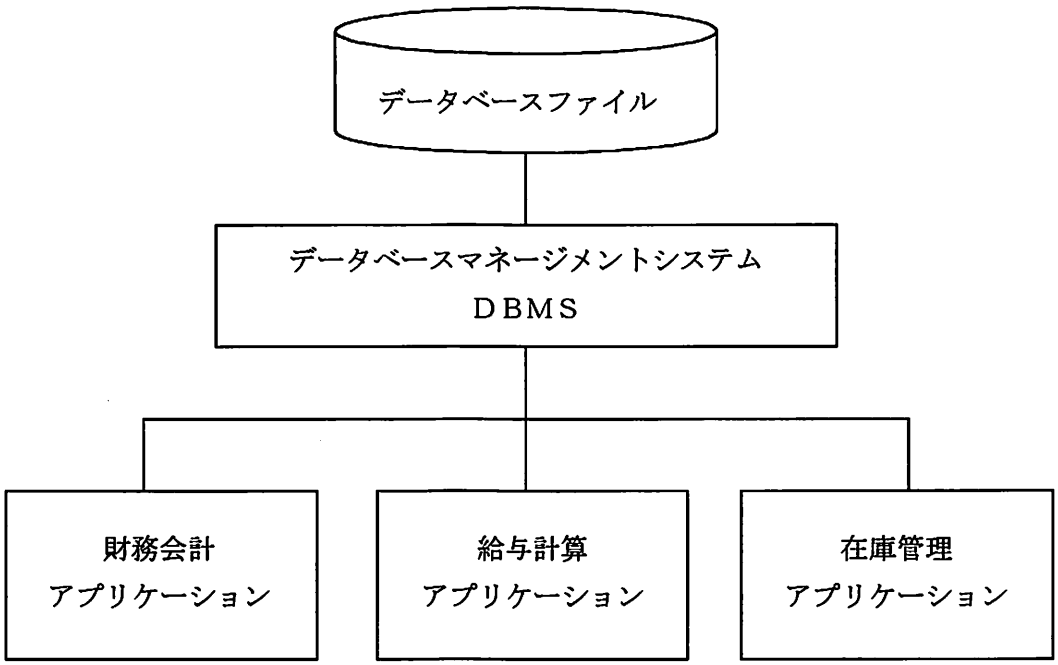


図 4.2 ＤＢＭＳ

4.1.2 dBXLを用いた単語集計

dBXLを用いて、すべての歌9420首について2～7文字の文字列をデータベース化した。その結果、755,444レコードを得た。この中から意味のある211単語を手作業で取り出した。

ここで、dBXLを用いたこの集計方法では回避できない問題がある。dBXLの場合文字列の一致で集計を行うため、たとえば「はな」を含む歌を調べようとすると、

あきはきぬもみちはやとにふりしきぬ
みちふみわけてとふひとはなし　＜古今集･秋下＞

のように、明らかに意味の違うものもカウントしてしまうため、正しい集計結果が得られない。

ここで、形態素解析システムの導入が考えられる。しかし、古文の形態素解析システムについてはあまり研究がなされていないため、辞書の不足などが考えられる。また、大量の文字列の中から手作業で単語を選び出すのもかなり手間がかかる。そこで、形態素解析を行う前に連想空間で比較的よく用いられる名詞を、学習のアルゴリズムC4.5を用いて抽出することにした。

4.2 C4.5

C4.5 について説明する。

AI 手法である C4.5 は、記録された膨大な分類データを調べ、特定の例を一般化することにより、モデルを帰納的に作る方法の一つで、記録の中にあるパターンを見つけて解析することによって、2 番目の分類モデルを作るコンピュータプログラムである。

・属性ー値の記述

一つのオブジェクトあるいは事例(case)に関するすべての情報は、あらかじめ決められた性質あるいは属性(attributes)によって表現できなくてはならない。それぞれの属性は離散値をとる。しかし、ある事例を表現するために使われた属性のタイプは、別の事例で別なタイプとして扱われてはならない。この制約のため、オブジェクトが変化するような構造をもつ問題領域は扱うことができない。

・前もって定義されたクラス

事例が割り当てられるべきカテゴリはまえもって準備されていなくてはならない。これは、機械学習における教師付き(supervised)学習に対応し、解析によって事例の適当なグループを見つける教師なし学習とは対照的である。

・離散クラス

クラスは、事例がそのクラスに属するか属さないかを決めることができるように、はっきりと定義できる必要がある。さらに、事例の数はクラスの数より十分多い必要がある。

・データが十分であること

帰納的一般化は、データの中から同じようなパターンを見つけることによって行われる。もし、たまたま、はっきりと区別できるパターンを見つけることができなければ、この方法は失敗してしまう。この区別は、

一般に統計的テストによって行っているので、このテストが意味があるためには、十分な事例が必要である。必要となるデータの数は、属性やクラスの数と分類しようとするモデルの複雑さなどの要因から決まる。これらの要因が増えると、信頼性のあるモデルを作るために必要なデータの数は増える。簡単なモデルであれば、少数のデータから作ることができる。しかし、複雑な分類モデルを作るには、一般に数百から数千のトレーニング事例が必要となる。

・“論理的”分類モデル

C4.5 は、決定木あるいはプロダクションルールの集合を作ることができるだけである。一つのクラスは、属性の値に関する記述を集めた一つの論理的な表現で表すように制限されている。この制限を満足していない分類モデルの一般的な形の一つとして、線形判別 (liner discriminant) がある。線形判別では、属性からの寄与を重み付けしてから加え、閾値と比較する。クラスの記述は論理的というより算術的である。

次に、アルゴリズムを示す。

4.2.1 分割統治法

訓練事例の集合 T から、決定木を構成する。クラスを $\{C_1,C_2,...,C_k\}$ と表す。Tがどのような事例になるかにより、以下の三つの可能性が考えられる。

・Tは少なくとも一つ以上の事例を含み、しかも、そのすべての事例が一つのクラス C_j のに属する場合：
この場合は、Tに対する決定木は一つの葉だけからなり、それにクラス C_j のラベルを付与する。

・Tが全く事例を含まない場合：
この場合も決定木は一つの葉からなるが、それに付与すべきクラス名はT以外の情報に基づいて決めなければならない。例えば、問題領域の知識に基づいて、もっとも頻繁に現れるクラスを選ぶことができる。C4.5では、事例がまったく与えられなかった葉に付与すべきクラスとして、その親のノードの中で最も頻繁に現れたクラスを用いる。

・Tが種々のクラスに属する事例を含む場合：
この場合は、Tを部分集合に分割して、各部分集合ができるだけ単一のクラスに属するように改善する。例えば、T10というテストが選ばれたとして、それは一つの属性に基づいて、相違なる値 $\{O_1,O_2,...,O_n\}$ を出力するとする。このとき、テストの結果が O_i となるようなT内の事例の集合を T_i とすれば、Tは部分集合 $T_1,T_2,...T_n$ に分割される。Tに対する決定木は、テストT10を行うノードとかく出力値に対応する枝からなり、同様な手順がさらに再帰的に繰り返される。すなわち、そのi番目の枝の先には、訓練事例(training case)の部分集合 T_i に基づいて、同様な手順で構成された決定木がつながれる。

4.2.2 例

訓練事例の集合の分割は、すべての部分集合が単一のクラスに属するようになるまで繰り返される。このプロセスを説明するために一例を表 4.1 に示す。この小さな訓練集合では、4種の属性と二つのクラスがある。

これらの事例は単一クラスに属しているわけではないので、分割統治法により、それらの分割を試みる。ここで“天候”のテストを選んだとしよう。その出力値は、“晴れ”、“曇り”と“雨”の 3 通りある。“曇”のグループはすべて開催のクラスからなるが、“晴れ”と“雨”のグループでは複数のクラスが混在している。そこで、さらに“晴れ”のグループを強風か否かテストで分割すれば、その結果得られるグループは、すべて単一のクラスからなるようにできる。こうして最終的に得られた分割の様子と、それに対応する決定木を表 4.2 に示す。

表 4.1 小さな訓練集合

天候	温度(華氏)	湿度(%)	強風？	クラス
晴れ	75	70	真	開催
晴れ	80	90	真	中止
晴れ	85	85	偽	中止
晴れ	72	95	偽	中止
晴れ	69	70	偽	開催
曇り	72	90	真	開催
曇り	83	78	偽	開催
曇り	64	65	真	開催
曇り	81	75	偽	開催
雨	71	80	真	中止
雨	65	70	真	中止
雨	75	80	偽	開催
雨	68	80	偽	開催
雨	70	96	偽	開催

表 4.2 事例の最終的な分割と対応する決定木

事例の分割：

・天候＝晴れ

湿度≦75：

天候	温度(華氏)	湿度(%)	強風？	クラス
晴れ	75	70	真	開催
晴れ	69	70	偽	開催

湿度＞75：

天候	温度(華氏)	湿度(%)	強風？	クラス
晴れ	80	90	真	中止
晴れ	85	85	偽	中止
晴れ	72	95	偽	中止

・天候＝曇り：

天候	温度(華氏)	湿度(%)	強風？	クラス
曇り	72	90	真	開催
曇り	83	78	偽	開催
曇り	64	65	真	開催
曇り	81	75	偽	開催

・天候＝雨：

強風＝真：

天候	温度(華氏)	湿度(%)	強風？	クラス
雨	71	80	真	中止
雨	65	70	真	中止

強風＝偽：

天候	温度(華氏)	湿度(%)	強風？	クラス
雨	75	80	偽	開催
雨	68	80	偽	開催
雨	70	96	偽	開催

対応する決定木：

・天候＝晴れ：

湿度≦75　：　開催

湿度＞75　：　中止

・天候＝曇り：開催

・天候＝雨：

強風＝真　：　中止

強風＝偽　：　開催

4.2.3 テストの評価

各ノードで T を分割するテストをどのように選んでも、それが真の意味での分割であれば、最終的には単一のクラスからなる部分集合への分割が得られる。ここで、真の意味での分割とは、空にならない部分集合$\{T_i\}$が少なくとも二つ以上できるような分割のことである。もっとも、そのほとんどすべての部分集合がただ一つの訓練事例しか含まないことがあるかもしれない。しかし、決定木を構成する目的は、単にそのような分割を何でもよいから見つけることでなく、問題領域の構造を明らかにしたり、予測能力を獲得したりすることにある。そのためには、一つ一つの葉において、十分な数の事例が必要であり、したがって、あまり細かく分類しすぎないことが望ましい。

利得基準

事例の集合Sに対して、$freq(C_i,S)$は、S の中で、クラス C_i に属する事例の数を表す。また、集合Sに含まれる事例数を|S|と表す。

事例の集合 S からランダムに一つの事例を選びだし、それがクラス C_j に属していると知らせたとする。このメッセージの確率は、

$$\frac{freq(C_j,S)}{|S|}$$

であり、それが伝える情報量は、

$$-\log_2\left(\frac{freq(C_j,S)}{|S|}\right) \text{ビット}$$

となる。このようなクラスの所属関係に関するメッセージの平均情報量を求めるために、S 内での頻度で重み付けしてクラス全体に対する平均を求めると、

$$info(S) = -\sum_{j=1}^{k}\frac{freq(C_j,S)}{|S|}\times log_2\left(\frac{freq(C_j,S)}{|S|}\right) \text{ビット}$$

を得る。この量は、集合 S のエントロピーとも呼ばれる。これを訓練事例の集合 T に適用すれば、*info(T)*は、T 内のある一つの事例が属するクラスを同定させるのに必要な情報量の平均値となる。

テスト X のｎ通りの結果にあわせて、T が分割された後について同様な評価を考える。このとき、クラスを同定するのに必要な情報量の期待値は、部分集合上で荷重平均をとって、

$$info_x = \sum_{j=1}^{n} \frac{|T_i|}{|T|} \times info(T_i)$$

となる。これらの差

$$gain(X) = info(T) - info_x(T)$$

は、テスト X で T を分割することによって、獲得される情報量を表す。この情報量の利得は、テスト X とクラスとの相互情報量とも呼ばれる。これを最大にするようにテストを選ぶ基準を、利得基準と呼ぶ。

具体例として、表 4.1 の訓練集合を再び使用する。ここは、９事例の“開催”と５事例の“中止”の２つのクラスがあり、

$$info(T) = -\frac{9}{14} \times log_2 \frac{9}{14} - \frac{5}{14} \times log_2 \frac{5}{14} \approx 0.940 \text{ ビット}$$

である。（これは、T 内の 1 事例のクラスを同定するために必要な情報量の期待値を表す。）また、“天候”を用いて、T を３つの部分集合に分割した後の結果は、

$$info_x(T) = \frac{5}{14} \times \left(-\frac{2}{5} \times log_2 \frac{2}{5} - \frac{3}{5} \times log_2 \frac{3}{5} \right) + \frac{4}{14} \left(-\frac{4}{4} \times log_2 \frac{4}{4} - \frac{0}{4} \times log_2 \frac{0}{4} \right) + \frac{5}{14} \left(-\frac{3}{5} \times log_2 \frac{3}{5} - \frac{2}{5} \times log_2 \frac{2}{5} \right)$$

$$\approx 0.694 \text{ ビット}$$

したがって、このテストによる情報量の利得は、0.940－0.694＝0.246 ビットとなる。ここで、天候の代わりに強風か否かの属性によって T を分割したとする。そうすれば、開催３事例と、中止３事例、開催６事例と、中止２事例からなる２つの部分集合が得られる。同様な計算により、

$$info_x(T) = \frac{6}{4} \times \left(-\frac{3}{6} \times log_2 \frac{3}{6} - \frac{3}{6} \times log_2 \frac{3}{6} \right) + \frac{8}{14} \times \left(-\frac{6}{8} \times log_2 \frac{6}{8} - \frac{2}{8} \times log_2 \frac{2}{8} \right)$$

$$= 0.892 \quad \text{ビット}$$

で、利得は 0.048 ビットとなり、これは先のテストで選られる利得より少ない。したがって、利得基準は、強風か否かの後者のテストよりも、天候についての前者のテストを選ぶことになる。

利得比基準

利得比基準は、多数の値を取るテストを偏重する欠点を持つ。このことは、患者の身分証明を属性として用いるような仮想的な医療診断タスクを考えてみれば分かる。身分証明は患者一人一人を識別するためのものであるから、これを属性として用いて訓練事例集合を分割すれば、１事例だけからなるたくさんの部分集合が得られることになる。当然のことながら、これらの１事例からなる部分集合は単一のクラスに含まれるので、$info_x(T)=0$ となる。したがって、この属性を用いて訓練事例の集合を分割すれば、情報量の利得は最大になる。しかしながら、クラスの予測能力を獲得しようという観点からは、このような分割はまったく無益である。

利得基準に伴うこのような偏重は、多数の値を取ることによって得られた利得部分を調整することによって、矯正することができる。ある事例に関して、それがどのクラスに属するかではなく、そのテスト結果自体を伝えるメッセージの情報量を考える。*info(S)*の定義からの類推により、分割情報量を

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times log_2 \left(\frac{|T_i|}{|T|} \right)$$

と定める。これは T をｎ個の部分集合へ分割することによって得られる全情

報量を表す。したがって、利得比

$$gain\ ratio(X) = \frac{gain(X)}{split\ \text{info(X)}}$$

は、分割によって選られる情報量のうち、有益な部分、すなわち、クラス分類に役立つ部分の割合を表す。分割が自明な分割に近いときは、分割情報量の値が小さいために利得比の値が不安定になる。そこで、利得比基準では、全テスト中で情報量利得が少なくとも平均以上であるという制約下でこの利得比を最大にするテストを選ぶ。

明らかに、この評価基準の下では、患者の身分証明という属性が高く評価されることはない。前述のように、クラスの数をｋとすると、上式の分子（情報量利得）は高々 $\log_2(k)$となる。一方、訓練事例をｎとすれば、テストの結果もｎ通りに分割されるので、分母は $\log_2(n)$となる。ここで、訓練事例数ｎはクラス数ｋよりずっと大きいので、利得比は小さな値となる。

前述の例に戻る。天候に関するテストは、訓練事例を 5 個、4 個、5 個の部分集合に分割する。この分割情報は、

$$-\frac{5}{14}\times log_2\frac{5}{14}-\frac{4}{14}log_2\frac{4}{14}-\frac{5}{14}\times log_2\frac{5}{14}$$

すなわち、1.577 ビットと計算される。また、前述したように、利得は 0.246 であるので、利得比は $\frac{0.246}{1.577}=0.156$ となる。

テストの候補

テストの候補が与えられれば、評価基準により各候補を評価して、その中で最良のテストを選び出すことができる。

通常、分類木を構成するシステムでは、テストの形式を定めて、その形式で表されるすべてのテストを調べる。１つのテストでは１つの属性だけを扱うのが普通である。なぜならば、それによって木が理解しやすくなるし、また、複数の属性を一度に扱ったときに起こるような組み合わせ爆発の問題が避けら

れるからである。

C4.5 は、次の３種類のテストを生成する仕組みを持っている。

・離散的な属性に関する“標準的”なテスト。各属性値をそのまま出力値として、一つの枝に対応させる。

・離散的な属性に基づく、より複雑なテスト。属性値をいくつかにグループ分けして、出力値を割り当てる。

・連続値をとる属性Ａに対して、閾値Ｚとの比較により、A≦Z または A>Z に分割するテスト。

これらのテストは、訓練事例を分割する際の利得比に基づいて統一的に評価される。また、自明に近い分割を避けるために、『分割の際、少なくとも 2 つの部分集合 T_i は、ある最小の数の事例を含まねばならない』という制約を加えることは有益である。このような制約は、特に訓練集合 T が小さいときに効果的である。

4.2.4 連続属性に関するテスト

まず、考えている属性Ａの値によって、訓練事例集合Ｔをソートする。その値は有限個しかないので、それらを大きさの順に並べて、$\{v_1,v_2,...,v_m\}$ と表す。v_i と v_{i+1} の間にある閾値はどれも同じ効果を持ち、属性Ａの値が $\{v_1,v_2,...,v_i\}$ に含まれるか、$\{v_{i+1},v_{i+2},...,v_m\}$ に含まれるかにより事例を分割する。したがって、Ａに基づく分割法は、ｍ－１通りしかなく、そのすべてを調べることは可能である。

通常、閾値として各九間の中点を選ぶ。したがって、ｉ番目の閾値は、

$$\frac{v_i + v_{i+1}}{2}$$

となる。

C4.5 は、閾値として中点そのものを選ばずに、訓練事例集合上でＡが取り

得る値の中で、その中点を超えない最大のものを選ぶ。こうすることによって、木や規則で用いられる閾値は、実際にデータ中に現れている“意味のある”値であると保証できる。

4.2.5 本研究で用いる属性

本研究でC4.5を用いるにあたり、以下の考え方をもとに属性を決定した。

長さｌである文字列ｎがあるする（図 4.3)。その文字列ｎが単語や慣用表現であったら、隣接する部分文字列であるｎ－１、ｎ－２の出現頻度は、ｎの文字列に含まれない語を含む部分文字列より、出現頻度に差が出る。図 4.4 に、ｎを慣用表現「することができる」とした場合について、ｎ－１、ｎ－２文字列の出現頻度を示す。

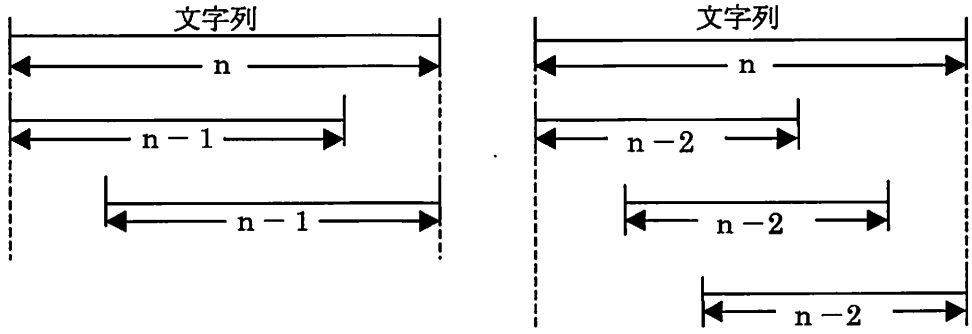


図 4.3 ｎ文字列とｎ－1 またはｎ－2 文字列

l	文字列	頻度
	することができる	188
8	現することができ	22
	することができる	○188
	ることができるが	12
7	現することがで	22
	することができ	○223
	ることができる	○452
	ことができるが	18

図 4.4 隣接する部分文字列の頻度

図 4.4 を見て分かるように、ｎ－１またはｎ－２がｎの部分文字列であるならば、頻度が極端に高くなる。つまり、頻度の高い部分文字列は名詞、または慣用句等の一部であると考えられるから、その前後にくる１文字、２文字もだいたい決まってくることになる。

この特性に注目し、本研究ではまず、すべての短歌から２～７文字の文字列を取り出しその頻度を集計した。そして、文字列、文字列の右側にくる１単語の種類数、一番頻度の高い１単語、１単語の頻度（全体の頻度に対する割合で表している)、左側にくる１単語の種類数、一番頻度の高い１単語、１単語の頻度を属性として C4.5 にトレーニングデータをあたえた。トレーニングデータとしては、1782 の文字列を与えた。トレーニングデータの一例を図４.に示す。

あか	20	つ	0.219512	32	S	0.158537
あかさ	6	り	0.363636	5	S	0.363636
あかし	9	の	0.388889	18	り	0.138889
あかしの	5	う	0.500000	10	は	0.285714
あかす	20	し	0.127660	15	を	0.127660
あかつ	1	き	1.000000	17	S	0.296296
あかつきか	2	た	0.666667	7	の	0.333333
あかつきの	11	ね	0.136364	8	S	0.590909
あかて	10	も	0.266667	8	S	0.266667
あかぬ	17	わ	0.282609	16	も	0.217391
あかぬわ	1	か	1.000000	7	の	0.461538
あかぬわか	1	れ	1.000000	7	の	0.461538
あかぬわかれ	6	の	0.461538	7	の	0.461538
あきか	7	せ	0.952941	23	S	0.376471
あきかせＥ	0	＊	0.000000	4	の	0.727273

図 4.5 トレーニングデータ

その後、すべてのデータを判定させた結果、321 の文字列が単語と認定された（図 4.6)。この中から、手作業で 105 個の正しい名詞を取り出した。これと、dBXL で取り出した文字列の中から手作業で選び出した 202 単語（動詞、形容詞を含む）をマージさせ、合計 237 単語を調査の対象、および形態素解析用辞書の補足に用いることにした。図 4.7 にはC4.5 によって新たに取り出された 35 の名詞のリストを示す。

あき	あきかせ	あきの	あきのよ	あけ
あさ	あふさか	あま	あめ	あら
あり	ありあ	ありあけ	いか	いく
いけ	いそ	いつれ	いの	いのち
いま	いろ	うき	うきよ	うくひす
うくひすの	うち	うめ	うめの	うめのは
えた	おい	おと	おもふこころ	かかり
かけて	かすか	かすみ	かせ	かた
かたみ	かみ	かも	からころも	かり
きく	きし	きに	きの	きみ
きみか	きみに	きも	きよ	くさ
くさの	くさは	くは	くま	くも
くもゐ	けか	けさ	けふり	こけ
こころ	こし	こすゑ	こそか	こと
ことし	このよ	こよひ	ころも	さかり
さき	さくら	さくらは	さくらはな	さこ
さため	さと	さめ	さら	さり
しくれ	しさ	した	しな	しに
しのふ	しひと	しま	しも	しもの
しらくも	しろ	すま	すゑ	そか
そた	そて	その	そめ	そら

図 4.6 名詞と判定された文字列

いま	えた	かた	けふり	こけ
こと	このよ	さくらはな	さと	した
しらくも	しろ	すま	たき	たけ
たひね	とり	なけき	なみ	ねさめ
のへ	はつき	はま	ふし	ふち
みね	みわ	もも	もり	やまた
やみ	ゆめ	よそ	よは	わかみ

図 4.7 C4.5 によって新たに取り出された名詞

4.3 形態素解析

4.3.1 形態素解析とは

自然言語は以下に示すような階層的構造を持っている。

音素(phoneme) 人間の意味(意志)伝達において音声をどのように使っているかをもとに考えた音の単位

形態素(morpheme) 意味をもつ最小の言語単位。一つ以上の音素からなる。

語(word) 一つの意味のまとまりをなし、文法上一つの機能を持つ最小の言語単位。一つ以上の形態素からなる。

文(sentence) あるまとまった内容を持ち、形の上で完結した(表記において句点が与えられる)言語単位。一つ以上の単語からなる。

文章・テキスト(text) あるまとまった内容を表現するための文の順序づけられた集まり。隣接する文相互間にはある種の関係性が存在する。

このうち、文字によって表記された自然言語においては形態素が最小の単位となるので、形態素解析(morphological analysis)を自然言語処理の第一段階と考えることが一般的である。しかし、形態素や語の適宜は言語の種類によって異なるため形態素解析が何を目標とするか、またどこにその難しさがあるかなどは言語依存の問題となる。

日本語の場合には語を区切る空白というものが存在しないため、語の厳密な定義を与えることは困難である。また、形態素という概念は欧米語の言語学からきたものであるため、日本語における形態素の定義もそれほど明確ではない。日本語の語は、文を構成する際の働きを基準とした場合、まず自立語(content word)と付属語(function word)に大別され、さらに細分化されて一般には図 4.3 に示す 10 の品詞に分類される。一方、語構成の立場から見ると、一つの要素からなる語と複数の要素からなる語がある。複数の要素からなる語について次の三つの結合形態がある。

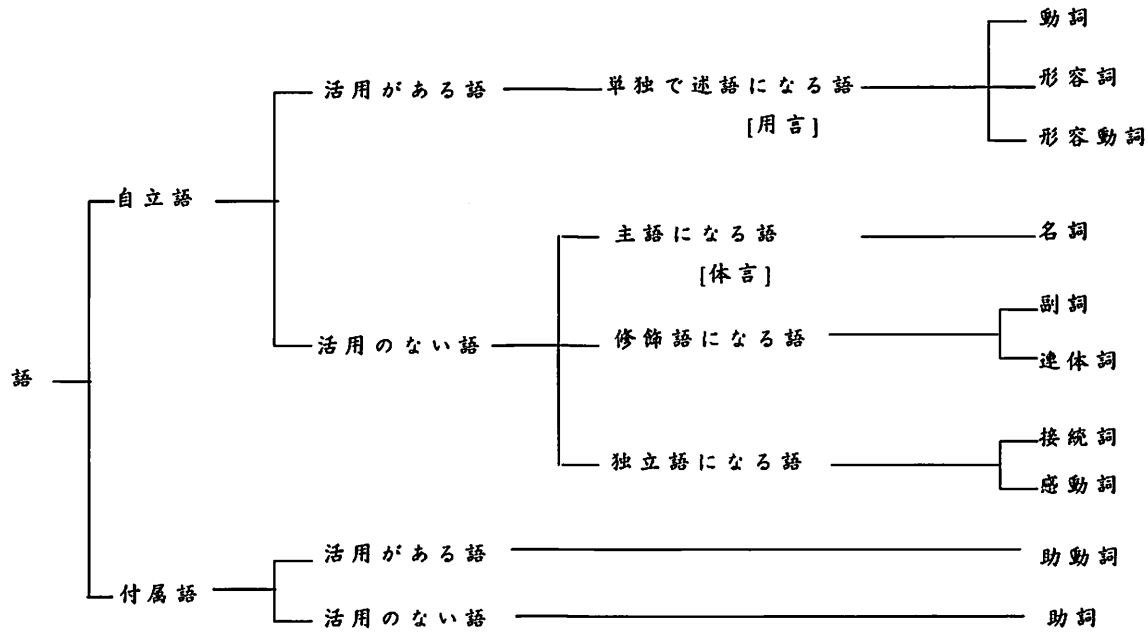


図 4.2　日本語の品詞分類

活用語　文中での働きの違いによって語形が変化する語。変化しない部分を活用語幹、変化する部分を活用語尾という。たとえば「食べる」では「食べ(活用語幹)+る(活用語尾)」となる。

派生語　ある語に付加的要素が付いてできる語。もとの語を派生語幹、付加的要素を接辞(接頭辞または接尾辞)という。たとえば「寒さ」では「寒(派生語幹)+さ(接尾辞)」、「ま冬」では「ま(接頭辞)+冬(派生語幹)」となる。

複合語　複数の語が結合して1語となったもの。「本+棚」、「うれし+涙」など。

日本語においては上記の結合形態の各構成要素がほぼ英語の形態素に相当するものである。工学的に見たときの形態素解析の目的は、入力文とにかく辞書中の行もkの組み合わせに分解することである。辞書には、図 4.2 の品詞に接辞を加えた 11 種類が登録されているとする。

以下で、2 語の連接可能性をチェックするもっとも基本的な形態素解析アルゴリズムについて説明する。

2 単語間の連接規則

語の並びとして日本語を見た場合、どのようなタイプ(品詞、活用形など)の2語が連続して文中にあらわれるかという条件は明確である。そこで、2語の連接可能性を規則(制約)として与え、それを参照しながら入力文を語の並びに変換するという方法がもっとも基本的な形態素解析の方法である。

この処理は、

1. 辞書を参照して入力文中の各位置から始まる語を取り出し、
2. 連接可能性をチェックしながら取り出された語をつないでいく

という二つの処理を繰り返し行なうことによって実現される。

このような解析を行なうために形態素解析では次の二つの辞書を必要とする。

単語辞書　語の品詞、読み、活用形などを指定する。

連接可能性辞書　連接可能な2語のタイプを指定する。語のタイプは具体的な語であっても、品詞であっても、活用形であってもよい。文頭にありえる語、文末にありえる語は、それぞれ文頭、文末という特別なタイプと連接可能であるとする。

これらの辞書は前処理によって形態素解析が高速に行なえる形式に変換される。まず、連接可能性辞書は連接可能性行列という形に変換される(図 4.3)。連接可能性辞書では行が左側にあらわれる語のタイプ、列が右側にあらわれる語のタイプを示し、この 2 タイプの語が連接可能であれば行列の値を1に、そうでなければ 0 にする。単語辞書も高速検索可能な形式に変換する。このとき、連接可能性が先に作成した行列の何番目の行と列に示されているかを調べて、その番号を各語とともに記憶しておく。こうすることによって、ある2語の連接可能性が左側の語の行番号と右側の列番号から行列の値を参照するだけですぐに調べられるようになる。

以上の操作で形態素解析は行なわれる。

4.3.2 形態素解析システムを用いた単語抽出

本研究では、短歌から単語を集計するために形態素解析システム JUMAN を用いて、短歌を単語切りと品詞付けを行った。

現代文における形態素解析システムは、ほぼ完成に近い域にあるといっても過言ではないが、古文における形態素解析システムについては、ほとんど研究がなされていないのが実状である。しかし、今回は運良く古文の形態素解析システム用辞書を作成された山本靖氏に許可をいただき、それを利用することができた。それが JUMAN 用古典文法・形態素解析辞書 kojuman（ただし、自立語の分は「古典対照語い表フロッピー版」＜笠間書院＞の koten.dat から作成）である。これに、C4.5 によって得られた名詞の一部を付加した。また、今回用いた八代集のデータはすべて、濁点なしのべた書きであるため、辞書から濁点は取り除いた。

また、データが濁点なしのべた書きということは、かなり解析が困難である。そこで、解析を容易にするために、八代集すべての歌 9420 首（短歌以外のものは除く）に 5・7・5・7・7 の区切りを入れた。字余りのない 7672 首については機械的に、字余りのある 1758 首については手作業で区切りを入れた。

JUMAN の解析によって得られた結果の一例を図に示す。

入力文：

「としのうちにはるはきにけりひととせをこそとはいはむことしとはいはむ」

10001	とし	とし	名詞	*
10001	の	の	助詞	*
10001	うち	うち	名詞	*
10001	に	に	助詞	*
10001	はる	はる	名詞	*
10001	はき	はし	形容詞	連体形
10001	に	に	名詞	*
10001	け	け	接尾辞	*
10001	り	り	名詞	*
10001	ひととせ	ひととせ	名詞	*
10001	を	を	助詞	*
10001	こそ	こそ	助詞	*
10001	と	と	助詞	*
10001	や	や	助詞	*
10001	いは	いふ	動詞	未然形
10001	む	む	助動詞	連体形
10001	こと	こと	名詞	*
10001	し	し	助詞	*
10001	と	と	助詞	*
10001	や	や	助詞	*
10001	いは	いふ	動詞	未然形
10001	む	む	助動詞	終止形
10001	EOS			

・

・

図4. JUMANの解析結果の一例

第５章

連想関係の表現

5.1　相関係数

時代変遷にともなう言葉の使われ方を調べるために、時間軸に対する出現頻度の相関係数を求めることにした。相関係数とは、二つの変量、または現象の間に何らかの関係があると予想されるとき、その関係の程度を量的に表すものである。相関係数は以下の式で表せる。

$$r=\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2}\sqrt{\sum(y_i-\bar{y})^2}}$$

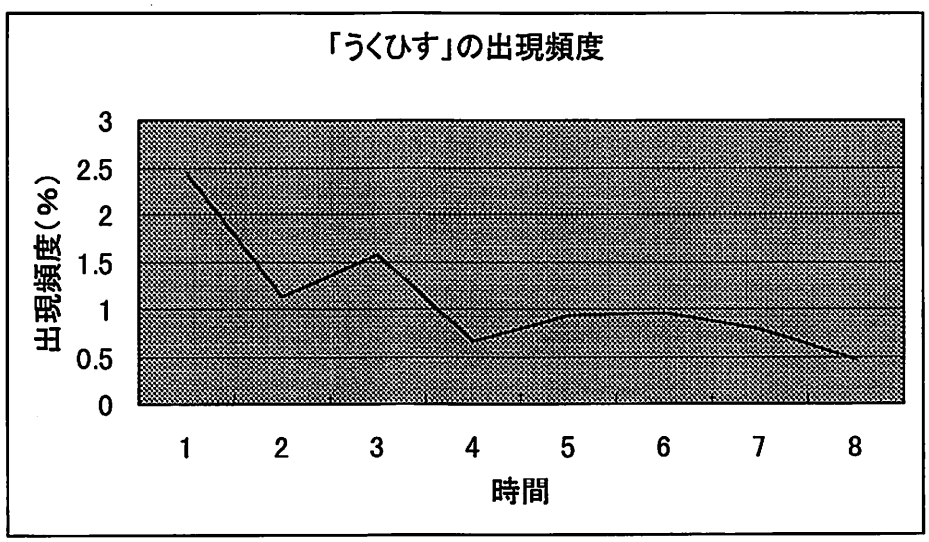
$$-1\le r\le 1$$

ここで、y_iは単語iのそれぞれの歌集に対する出現頻度(歌集全体に対する出現回数の占める割合《%》)、x_iは時間（八代集のそれぞれの歌集に対して、古いほうから１～８と番号付けしたもの)、$\bar{x}$，$\bar{y}$はx，yのそれぞれの平均である。

相関係数ｒが-1 に近い値を取るときは、古い方の時代（古今集側）の出現頻度が高く、ｒが+１に近い値を取るときは、新しいほうの時代（新古今集側）の出現頻度が高いということである。ｒが０に近い場合はほとんど、どの時代にも同じくらいの頻度で現れているということである。次に「つき」と「かり」の相関係数の一例を示す。

	古今	後撰	拾遺	後拾遺	金葉	詞花	千載	新古今	相関係数
うくひす	2.45	1.12	1.57	0.65	0.93	0.95	0.78	0.46	-0.792
つき	2.36	3.02	3.22	7.89	9.88	11.7	9.83	12.9	0.942

r =−0.792



r =0.92

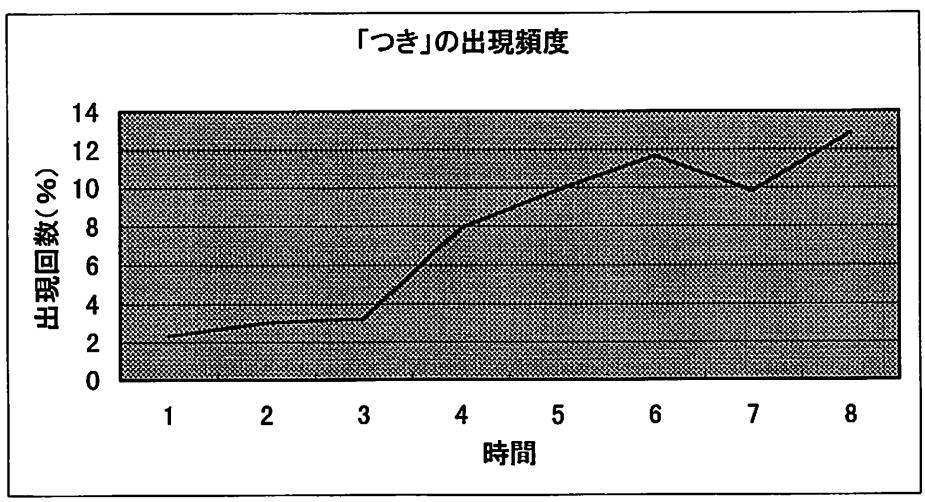


図 5.1 「つき」「かり」の相関係数

5.2 Dice 係数

Dice 係数は、単語間の共起性の強さを測る尺度の一つである。つまり、連想の強さを表すことができると考えられる。単語 x と単語 y の Dice 係数は以下の式で表せる。

$$Dice(x,y)=\frac{2\times f(x,y)}{f(x)+f(y)}$$

ここで、$f(x,y)$は単語xと単語yが同時に出現した歌数、$f(x)$, $f(y)$はそれぞれの単語が出現した歌数である。

Dice 係数が 1 に近いほど 2 単語間の連想が強いことになる。

第6章

結果

6.1 集計結果について

八代集すべての歌9420首について形態素解析を行った。その結果からプログラミング言語 Perl を用いて C4.5 で取り出した名詞を中心とする単語 237 について集計を行った。

ここで、Perl とは Practical Extract and Report Language の略であり、テキスト処理用のツールとして開発されたものである。

その結果、形態素解析システム JUMAN および JUMAN 用古典文法・形態素解析辞書 kojuman を用いた有用性は「かり」「はな」の集計結果から明らかである。この結果を表 6.1 に示す。

表 6.1 「かり」「はな」の集計結果の比較

	Dbxl	JUMAN	岩波書店八代集CD-ROM
かり	1186	109	87
はな	1,825	680	449

表から明らかなように、JUMAN を用いたほうが岩波の CD-ROM を用いて（手作業で）調べた歌数にほぼ近づいている。とくに「かり」の方は Dbxl の集計結果では、10 倍もの誤差が見られていたが、JUMAN の結果からはほぼノイズが取り除かれたことが分かる。

ここで、「岩波書店八代集 CD-ROM」とは、1 単語についてはそれを含む歌と歌数を表示できるものである。しかし、この CD-ROM では、ある単語

を含む歌を見ることを目的としているため、本研究で目的としている連想関係を調べるためのツールとしてはあまり適していないことを断っておく。

また、単語集計の結果から2単語について相関係数とDice係数を算出した。

6.2 考察

6.2.1 連想の表現について

連想関係を表現するために、時系列に注目した相関係数、連想の強さに注目したDice係数を導入した。ここで、どちらの方が2単語関の連想関係を表現するのにより適しているかについて考えてみる。

まず、相関係数の場合、2単語が同時に出現する回数を時系列に注目してとらえるため、一つの係数で、古い方の時代に出現する回数が多いか、新しいほうの時代に出現する回数が多いのかを表せる。しかし、下の図のような場合は、出現頻度が上がったり下がったりしているにもかかわらず、相関係数は0.076となっている。そのため、この数字だけを見た場合には、どの時代も平均して出現していると考えてしまいがちである。このように一つの係数で表現してしまうと、全体的な変化が見られなくなる。出現頻度の大体の目安としては適しているが、相関係数は連想関係を直接表現しているとは言い難い。図6.1に「ほととぎす・やま」の出現頻度のグラフを示す。

続いて、Dice係数の場合は、それぞれの単語の出現回数と、2単語が同時に出現している回数から算出される。そのため、2単語が同時に出現する回数のみに左右されず、2単語間の共起の強さを表すことができる。また、時系列に注目したい場合には歌集ごとのDice係数に注目すればよいことである。グラフ等に表せば、どこでの連想が強いかということはすぐ分かる。つまり、連想関係の強さを表すにはDice係数のほうが適していると考えられる。表6.2に「ほととぎす・やま」の出現回数表、図6.2に「ほととぎす・やま」のDice係数を示す。

r =0.076

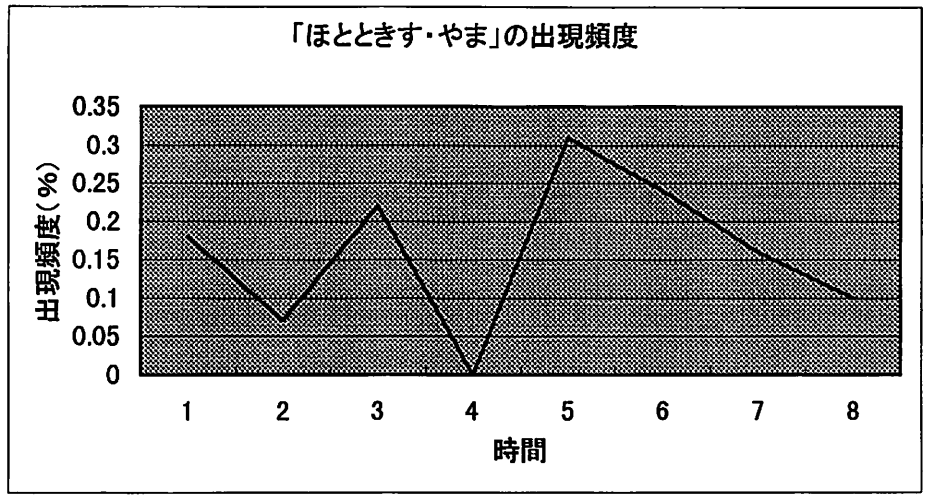


図 6.1 出現頻度

表 6.2 「ほとときす・やま」の出現回数

	古今	後撰	拾遺	後拾遺	金葉	詞花	千載	新古今
ほとときす　やま	2	1	3	0	2	1	2	2

図 6.2 Dice 係数

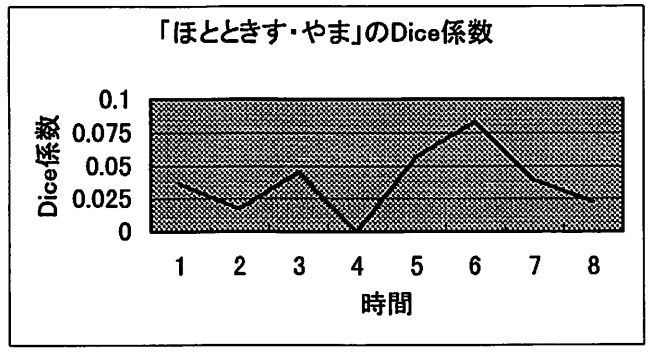


図 6.1 の出現頻度を見ると、時間が5、つまり、金葉集での出現頻度が高いが、図 6.2 の Dice 係数を見ると詞花集での連想が強い。このことから Dice 係数は、出現回数のみに左右されずに表せることが分かった。しかし、Dice 係数は出現回数が最低1であれば算出できてしまう。そのため、あまり重要でない組み合わせについても、場合によっては大きな値が出てしまうこともあるという欠点がある。しかし、これは出現回数を参照し、あらかじめ、ある程度の出現回数のものに絞っておくことで回避できる。

連想関係の強さを距離に対応させ、主要な単語を配置させたn次元マップを作成する場合には、Dice 係数のほうが適している。

相関係数は連想の強さを表すには適していないが、1 単語についての時代変遷にともなう使われ方の変化をを見たいときには、よく特徴があらわれる。

そこで、1 単語の変遷については相関係数、2 単語については Dice 係数を見ることにする。以下に、集計結果の応用例として、簡単な考察を行う。

6.2.2 年表

まず、考察を行う前に八代集が編集された時代を確認する意味で、その年表を以下に示す。

表 6.3 八代集年表

世紀	八代集	他の文学作品
10 (901～1000)	905 古今和歌集 951 後撰和歌集 998 拾遺和歌集	伊勢物語、竹取物語
11 (1001～1100)	1186 後拾遺和歌集	枕草子、源氏物語 大鏡、今昔物語
12 (1101～1200)	1127 金葉和歌集 1155 詞花和歌集 1187 千載和歌集	梁塵秘抄 山家集(西行)
13 (1201～1300)	1025 新古今和歌集	方上記、金塊和歌集 小倉百人一首

6.2.3　1単語で見る言葉の変化

1 単語については、相関係数に注目した。結果、おもしろい関係になっている単語を発見した。「わひし」と「さひし」である。これを表 6.4 に出現頻度と相関係数を示す。

表 6.4 「わひし」「さひし」の比較

	古今	後撰	拾遺	後拾遺	金葉	詞花	千載	新古今	相関
わびし	1.09	2.03	0.82	0.33	0	0	0	0	-0.813
さびし	0.09	0.07	0.15	0.49	0.15	0	0.94	0.61	0.632

「わひし」の相関係数は金葉集以降の頻度が 0 であることから明らかなように、 - 1 に近い値を取っている。「さひし」の方は、後半になって急激に頻度が上がり、相関係数も 0.632 となっている。どちらも、同じような意味であるにもかかわらず、はやりすたりがあるのには理由があるはずである。そこで、まず 2 つの厳密な意味を下に示す。

わびし　気落ちして切ない、心細い、頼りない、身にこたえる、
　　やりきれない
さびし　もとの活気が失せて荒れ果てていると感じる、物足りなく感じる、孤独がひしひしと感じられる

「さひし」のほうが「わびし」より、荒廃した雰囲気を醸し出している意味を持つことがわかる。「わびし」は金葉集以降、まったく用いられなくなっているが、「さひし」はそれ以降で頻度が増えている。このことから、金葉集の前後に原因があると考えられる。

ここで、先ほどの年表をみてみると、金葉集が編集されたのは鎌倉時代に入ってからのことである。鎌倉時代といえば、武家社会である。しかし、和歌を詠むのはもっぱら貴族であった。その貴族は、都が平安から鎌倉へ移った時点で、権力を失っている。つまり、昔はよかったが今となっては活気が失せてしまい、まさに「さびし」という状態であったのである。このような、貴族達が「さびし」よりも、「わびし」を好むようになり、表 6.4 のような結果が得られたと考えられる。

政治の中心が変わり、時代も移ったことに影響され言葉の使われ方が変わっていったということを裏付けている一例である。

6.2.4 枕草子

また、ここで年表の中ほどをみると、「はるはあけぼの」で有名な清少納言の随筆「枕草子」が目を引く。「枕草子」は、清少納言が仕えていた中宮定子が亡くなり、宮廷を退いた後の、西暦1000年頃に完成したといわれている。この「枕草子」に着目し、「枕草子」の影響により八代集のの連想関係が変化していったのではないかという予測をたてた。枕草子の一節を示す。

枕草子

春は、曙。やうやう白くなりゆく、山ぎはすこし明かりて、紫だちたる雲のほそくたなびきたる。

・・・

秋は、夕暮。夕日のさして、山の端いと近うなりたるに、烏の、寝どころへ行くとて、三つ四つ二つなど、飛び急ぐさへ、あはれなり。

6.2.5 2単語の連想

特に、「枕草子」の傍線部分に着目し、「はる」と「あき」について、それぞれ連想関係になっている単語を調べた。その中から興味深いごく一部の組み合わせを取り出した。表6.5に示す。

表 6.5　「はる」「あき」を含む組み合わせの Dice 係数

		古今	後撰	拾遺	後拾遺	金葉	詞花	千載	新古今
はる	ゆき	0.143	0.047	0.14	0.1	0.036	0	0.018	0.094
あき	もみち	0.15	0.15	0.125	0.101	0.05	0.125	0.045	0.021
うくひす	はる	0.127	0.143	0.091	0.072	0.043	0	0.094	0.065
あけほの	はる	0	0	0	0.026	0	0	0.076	0.094
あき	ゆふくれ	0	0	0	0.194	0.143	0.061	0.048	0.181

まず、表の上から三つの組みあわせ、「はる・ゆき」「あき・もみち」「うくひす・はる」は、現代でも常識的に連想できる組み合わせである。この三つはどれも途中から連想が弱まっている。このことは、昔から多くの歌で用いられ常識的なこととなってしまい、連想の組み合わせとして飽きられてしまったためと考えられる。

一方、先ほどの「枕草子」に関連した組み合わせ、「はる・あけほの」「あき・ゆふくれ」の組み合わせは表から明らかなように、後拾遺以降で連想関係が発生している。ここで、再び年表をみてみると、枕草子が完成したといわれているのが西暦 1000 年、後拾遺集が編集されたのが 1186 年である。年代的にもほぼ一致している。つまり、「はる・あけほの」「あき・ゆふくれ」の組み合わせは、「枕草子」が出現したことにより、新たな連想としてはやり始めたと考えられる。このことから、八代集は枕草子の影響を強く受け、連想関係の常識が変わったといえる。以下に、枕草子に影響を受けたと思われる歌の一例を示しておく。

みよしののたかねのさくらちりにけり
あらしもしろきはるのあけほの
後鳥羽院 (新古今・春歌下)

みわたせははなももみちもなかりけり
うらのとまやのあきのゆふくれ
藤原定家 (新古今・秋歌上)

6.3 自動検索システムの開発

今までの集計結果等を利用して簡単な考察を行なってきたが、実際に古典について研究を行う場合は単語の出現回数等も大事であるが、その単語を含む歌も見る必要がある。岩波の CD-ROM では、1 単語を含む歌の検索は可能であるが、2 単語の場合は単語 x と単語 y を or で検索するのでノイズが多くなる。また、品詞は参照されないため、違った品詞のものが混じることがある。

そこで、dBXL を利用して、品詞を指定した 1 単語と 2 単語を含む歌を自動検索するプログラム“ＹＣＹ（YACHIYO）”を作成した。まず、形態素解析の結果を歌集ごとにデータベース化した。フィールドは、歌番号、もとの単語、原形、品詞、活用の五つである。はじめに、歌集を指定し、検索したい単語、品詞を入力すると、形態素解析の結果のデータベース（data1.dbf~data8.dbf）を参照して、歌数、または歌を検索する。目的に応じて、以下の 4 つにプログラムを分けた。

YTY11.PRG：品詞付き 1 単語を入力するとそれを含む歌の歌数を出力する。

YTY12.PRG：品詞付き 1 単語を入力するとそれを含む歌を UTA.DBF に書き込む。

YTY21.PRG：品詞付き 2 単語を入力するとそれを含む歌の歌数を出力する。

YTY22.PRG：品詞付き 2 単語を入力するとそれを含む歌を UTA2.DBF に書き込む。

これらのプログラムリストを付録に示す。

第７章

結論

本研究では、八代集全ての歌 9420 首において、品詞名付きの１単語、もしくは２単語から歌を自動検索するプログラム“ＹＣＹ（YACHIYO)”を新たに開発した。これを用いて八代集の意味空間の変遷について解析を行なった。

検索ツール作成のために行なった作業の要点は以下の通りである。

1. データベースマネージメントシステム dBXL を用いて２～７文字の文字列の抽出を行った。しかし、この方法では集計結果に誤差がありすぎることがわかった。
2. そこで、形態素解析システム JUMAN および JUMAN 用古典文法辞書 kojuman（山本靖氏作成）を利用して、形態素解析を行なった。JUMAN により形態素解析を行なった結果、誤差が大幅に減少し、品詞認定も可能となった。
3. 辞書無しで対象をある程度しぼるために、学習のアルゴリズム C4.5 を用いて、名詞の抽出を行なった。
4. 以上の作業から得られた結果をデータベース化した。そして、作成した検索プログラムによって、単語の出現頻度、その単語が出現する歌番号、歌、の検索を行なえるようにした。

１単語と、２単語の出現頻度の時系列に注目し、相関係数を導入した。また、２単語間の連想の強さを表すために Dice 係数を導入した。これらの算出結果の比較から、２単語間の連想の強さをより的確に表すには、Dice 係数の方が適していると考えられる。そこで、Dice 係数に注目した考察を行なった結果、八代集の連想関係には枕草子が大きく影響していることを確認した。

謝辞

本研究の指導教官である新納浩幸教官ならびに、多大なご助言を賜りました城道介教官に深く感謝いたします。

また、古典についてご指導下さった山戸竹男氏、kojuman の使用を許可して下さった山本靖氏に感謝の意をこめてお礼を申し上げます。

最後に、本研究を進めるにあたり、ご助言ご協力を頂きました、水野孝泰技官、近藤研究室の尾形達哉氏、寺島成樹氏、城研究室の大川直人氏、佐藤信一氏、延嶋聡氏、榎本和人氏、雨宮幸子氏、市川学氏、島村尚卓氏、鈴木規裕氏に感謝いたします。

参考文献

[1] 山戸 竹男；"連想表現論 –和歌と文学史–",昭和47年度東京大学修士文,(1972).

[2] Stuart J. Russell and Peter Norvig ；"Artificial Intelligence A Modern Approach ",Prentice Hall,pp.531-544(1995).

[3] Makoto Nagao and Shinsuke Mori ;"A New Method of Number of N-gram Statistics for Large Number of n and Automatic Extraction of Words and Phrases from Large Text Data of Japanese", Priceedubgs of COLLING-94, Vol.1, pp.611-615(1994).

[4] 新納 浩幸、井佐原 均；"片方向の共起性による述語型定型表現の自動抽出", 然言語処理学会,Vol.2,No.3,pp.73-84(1995).

[5] 梅田 真太郎；"コーパスからの『A の B』型慣用表現の自動抽出", 平成8年度茨城大学卒業論文(1997).

[6] 近藤 嘉雪;"実践Perl プログラミング" ,UNIX USER, vol.6, No.3, pp.54-62, No.4, pp.73-78, No5, pp.71-76, (1997).

[7] 長尾 真 偏；"自然言語処理", 岩波書店, pp.117-pp137, (1996).

[8] 鈴木 良美；"dBXL ハンディ・マニュアル", ナツメ社, (1990).

[9] 村山 集治;"dBASEⅢ基礎テキスト", アコムインターナショナル, (1986).

[10]村山 集治；"dBASEⅢビジネス活用法", アコムインターナショナル, (1986).

[12]石田 穣二 訳注；"新版 枕草子", 角川書店, (1979).

[11]橋本 治；"ハシモト式 古典入門", ごま書房, (1997).

[12]久保田 淳；"八代集総索引", 岩波書店, (1995).

[13]大野 晋, 佐竹 昭広, 前田 金五郎, : "岩波　古語辞典　補訂版", 岩波書店 (1974).

付録 プログラムリスト

YCY11

```
******YCY11.PRG*************************************************************
******JUMAN の解析結果から品詞付き１単語の出現歌数を検索するプログラム**********
**                                          1998.2.25    SATOKO FUKUSHI      **
*****************************************************************************
*** 変数一覧***
* FILE[]  :  ファイル名を入れた配列
* F1      :  ファイル名の配列の添字（キーボード入力）
* HIN[]   :  品詞を入れた配列
* I       :  F1 を数値型に変換した値
* J       :  I から小数部をとった値
* W1      :  検索したい文字列 1(キーボード入力)
* H1      :  品詞の配列の添字（キーボード入力）
* K       :  H1 を数値型に変換した値
* L       :  K から小数部をとった値
* UAKAZU[]:  それぞれの歌集の歌数を入れた配列
* KAISU   :  文字列１が一つの歌に含まれるなら１以上の値を持つ
* SHUTU   :  調べたい単語の出現回数
* ***************************************************************************
*
*入力
*
?"１: 古今集"
?"２: 後撰和歌集"
?"３: 拾遺和歌集"
?"４: 後拾遺和歌集"
?"５: 金葉和歌集"
?"６: 詞花和歌集"
?"７: 千載和歌集"
?"８: 新古今和歌集"
ACCEPT "ファイル番号を入力してください :" TO F1
ACCEPT "単語を入力してください >" TO W1
?"１: 名詞"
?"２: 助動詞"
?"３: 助詞"
?"４: 動詞(原型)"
?"５: 形容詞(原型) "
```

```
?"6：形容動詞"
?"7：副詞"
?"8：連体詞"
?"9：接続詞"
?"10：接尾辞"
?"11：枕詞"
ACCEPT "品詞番号を入力してください ＞" TO H1
clear
***
SET TALK OFF
DECLARE FILE[10]
FILE[1] = "DATA1"
FILE[2] = "DATA2"
FILE[3] = "DATA3"
FILE[4] = "DATA4"
FILE[5] = "DATA5"
FILE[6] = "DATA6"
FILE[7] = "DATA7"
FILE[8] = "DATA8"
**それぞれのデータファイルのレーコード数
DECLARE UTAKAZU[10]
UTAKAZU[1] = 19596
UTAKAZU[2] = 25420
UTAKAZU[3] = 23946
UTAKAZU[4] = 22183
UTAKAZU[5] = 11589
UTAKAZU[6] = 7565
UTAKAZU[7] = 23089
UTAKAZU[8] = 35573

DECLARE HIN[15]
HIN[1] = "名詞"
HIN[2] = "助動詞"
HIN[3] = "助詞"
HIN[4] = "動詞"
HIN[5] = "形容詞"
HIN[6] = "形容動詞"
HIN[7] = "副詞"
HIN[8] = "連体詞"
HIN[9] = "接続詞"
HIN[10] = "接尾辞"
HIN[11] = "枕詞"
***
*
*使用ファイルのオープン
*
I = VAL(F1)
J = INT(I)
```

```
FL=FILE[J]
USE &FL
***
*
*使用変数の初期化
*
KAISU=0
SHUTU = 0
K = VAL(H1)
L = INT(K)
HS=HIN[L]
UK =1
*
*検索開始
*
GOTO TOP
DO WHILE .NOT. EOF()
   IF GENKEI = W1 .AND. HINSI=HS .AND. LEN(TRIM(GENKEI)) = LEN(TRIM(W1))
      KAISU = KAISU + 1
   ENDIF
   IF   TANGO = "EOS"
     IF KAISU >0
        SHUTU = SHUTU+1
        KAISU = 0
     ENDIF
   ENDIF
   IF UK < UTAKAZU[J]
      UK = UK +1
      GOTO UK
   ELSE IF UK =UTAKAZU[J]
      BREAK
   ENDIF
ENDDO
CLOSE ALL
*
*結果出力
*
? W1
?? "("
?? HS
??")"
??   "の歌数は"
?? SHUTU
??"    です。"
RETURN
```

YCY12

```
******YCY12.PRG*************************************************************
***JUMANの解析結果から品詞付き単語の出現する歌をデータベース化するプログラム**
**                                          1998.2.25   SATOKO FUKUSHI    **
*****************************************************************************
*** ファイル一覧 ***
* DATA1   :  形態素解析の結果
* UTA     :  単語を含む歌を入れるファイル
* DUMMY   :  UTAファイルを初期化するためのファイル
* HATIDAI :  八代集のデータベース
*** 変数一覧***
* FILE[]  :  ファイル名を入れた配列
* F1      :  ファイル名の配列の添字（キーボード入力）
* HIN[]   :  品詞を入れた配列
* I       :  F1を数値型に変換した値
* J       :  Iから小数部をとった値
* FL      :  ファイル名
* W1      :  検索したい文字列1(キーボード入力)
* H1      :  品詞の配列の添字（キーボード入力）
* K       :  H1を数値型に変換した値
* L       :  Kから小数部をとった値
* HS      :  品詞名
* UTABAN  :  単語を含む歌の歌番号
* UAKAZU[]:  それぞれの歌集の歌数を入れた配列
* KAISU   :  文字列1が一つの歌に含まれるなら1以上の値を持つ
* SHUTU   :  調べたい単語の出現回数
*****************************************************************************
********入力********
?"1:古今集"
?"2:後撰和歌集"
?"3:拾遺和歌集"
?"4:後拾遺和歌集"
?"5:金葉和歌集"
?"6:詞花和歌集"
?"7:千載和歌集"
?"8:新古今和歌集"
ACCEPT"ファイル番号を入力してください:"TO F1
ACCEPT"単語を入力してください>"TO W1
?"1:名詞"
?"2:助動詞"
?"3:助詞"
?"4:動詞(原型)"
?"5:形容詞(原型)"
?"6:形容動詞"
```

```
?"7: 副詞"
?"8: 連体詞"
?"9: 接続詞"
?"10: 接尾辞"
?"11: 枕詞"
ACCEPT "品詞番号を入力してください >" TO H1
clear
***
*
*書き込み用ファイルの初期化
*
SET TALK OFF
USE DUMMY
COPY TO UTA
CLEAR

DECLARE FILE[10]
FILE[1] = "DATA1"
FILE[2] = "DATA2"
FILE[3] = "DATA3"
FILE[4] = "DATA4"
FILE[5] = "DATA5"
FILE[6] = "DATA6"
FILE[7] = "DATA7"
FILE[8] = "DATA8"

DECLARE UTAKAZU[10]
UTAKAZU[1] = 19596
UTAKAZU[2] = 25420
UTAKAZU[3] = 23946
UTAKAZU[4] = 22183
UTAKAZU[5] = 11589
UTAKAZU[6] = 7565
UTAKAZU[7] = 23089
UTAKAZU[8] = 35573

DECLARE HIN[15]
HIN[1] = "名詞"
HIN[2] = "助動詞"
HIN[3] = "助詞"
HIN[4] = "動詞"
HIN[5] = "形容詞"
HIN[6] = "形容動詞"
HIN[7] = "副詞"
HIN[8] = "連体詞"
HIN[9] = "接続詞"
HIN[10] = "接尾辞"
HIN[11] = "枕詞"
```

```
*
*使用ファイルのオープン&アリアス指定
*
SELECT B
USE UTA
SELECT A
I = VAL(F1)
J = INT(I)
FL= FILE[J]
USE &FL
***
*
*使用変数の初期化
*
KAISU=0
K = VAL(H1)
L = INT(K)
HS=HIN[L]
UK =1
UTABAN = 0
*
*検索開始
*
GOTO TOP
DO WHILE .NOT. EOF()
   IF GENKEI = W1 .AND. HINSI=HS .AND. LEN(TRIM(GENKEI)) = LEN(TRIM(W1))
      KAISU = KAISU + 1
      UTABAN = NO
   ENDIF
   IF   TANGO = "EOS"
     IF KAISU >0
        SELECT B
        *
        *UTA.DBF への書き込み
        *
        APPEND FROM HATIDAI FOR NO=UTABAN
        SELECT A
        KAISU = 0
     ENDIF
   ENDIF
   IF UK < UTAKAZU[J]
      UK = UK +1
      GOTO UK
   ELSE IF UK =UTAKAZU[J]
      BREAK
   ENDIF
ENDDO
CLOSE ALL
```

RETURN

YCY21

```
****YCY21.P**************************************************************************
****JUMAN の解析結果から品詞付きの２単語を含む歌数を検索するプログラム**********
**                                                  1998.2.28    SATOKO FUKUSHI       **
*****************************************************************************
*** 変数一覧***
* FILE[]  :  ファイル名を入れた配列
* F1      :  ファイル名の配列の添字（キーボード入力）
* HIN1[]  :  品詞１を入れた配列
* HIN2[]  :  品詞２を入れた配列
* I       :  F1 を数値型に変換した値
* J       :  Ｉから小数部をとった値
* FL      :  ファイル名
* W1      :  検索したい文字列 1(キーボード入力)
* H1      :  品詞の配列の添字（キーボード入力）
* K       :  H1 を数値型に変換した値
* L       :  Ｋから小数部をとった値
* M       :  H2 を数値型に変換した値
* N       :  Ｍから小数部をとった値
* HS1,HS2 :  品詞名
* UAKAZU[]:  それぞれの歌集の歌数を入れた配列
* KAISU   :  文字列１が一つの歌に含まれるなら１以上の値を持つ
* SHUTU   :  調べたい単語の出現回数
* UK      :  調べている位置を表す
* SU      :  単語１含む歌番号
* CT      :  前の歌の EOS のレコード番号
*****************************************************************************
*
*入力
*
?"１: 古今集"
?"２: 後撰和歌集"
?"３: 拾遺和歌集"
?"４: 後拾遺和歌集"
?"５: 金葉和歌集"
?"６: 詞花和歌集"
?"７: 千載和歌集"
?"８: 新古今和歌集"
ACCEPT "ファイル番号を入力してください :" TO F1
ACCEPT "単語１を入力してください >" TO W1
ACCEPT "単語２を入力してください >" TO W2
?"１: 名詞"
?"２: 助動詞"
?"３: 助詞"
```

```
?"4: 動詞(原型)"
?"5: 形容詞(原型)"
?"6: 形容動詞"
?"7: 副詞"
?"8: 連体詞"
?"9: 接続詞"
?"10: 接尾辞"
?"11: 枕詞"

ACCEPT "単語1の品詞番号を入力してください >" TO H1
ACCEPT "単語2の品詞番号を入力してください >" TO H2
clear
***
SET TALK OFF
DECLARE FILE[10]
FILE[1] = "DATA1"
FILE[2] = "DATA2"
FILE[3] = "DATA3"
FILE[4] = "DATA4"
FILE[5] = "DATA5"
FILE[6] = "DATA6"
FILE[7] = "DATA7"
FILE[8] = "DATA8"

DECLARE UTAKAZU[10]
UTAKAZU[1] = 19596
UTAKAZU[2] = 25420
UTAKAZU[3] = 23946
UTAKAZU[4] = 22183
UTAKAZU[5] = 11589
UTAKAZU[6] = 7565
UTAKAZU[7] = 23089
UTAKAZU[8] = 35573

DECLARE HIN1[15]
HIN1[1] = "名詞"
HIN1[2] = "助動詞"
HIN1[3] = "助詞"
HIN1[4] = "動詞"
HIN1[5] = "形容詞"
HIN1[6] = "形容動詞"
HIN1[7] = "副詞"
HIN1[8] = "連体詞"
HIN1[9] = "接続詞"
HIN1[10] = "接尾辞"
HIN1[11] = "枕詞"

DECLARE HIN2[15]
```

```
HIN2[1] = "名詞"
HIN2[2] = "助動詞"
HIN2[3] = "助詞"
HIN2[4] = "動詞"
HIN2[5] = "形容詞"
HIN2[6] = "形容動詞"
HIN2[7] = "副詞"
HIN2[8] = "連体詞"
HIN2[9] = "接続詞"
HIN2[10] = "接尾辞"
HIN2[11] = "枕詞"
***
SET TALK OFF
***
*
*使用ファイルのオープン
*
I = VAL(F1)
J = INT(I)
FL=FILE[J]
USE &FL
***
K = VAL(H1)
L = INT(K)
HS1 = HIN1[L]
M = VAL(H2)
N = INT(M)
HS2 = HIN2[N]
*
*使用変数の初期化
*
KAISU=0
SHUTU = 0
UK = 1
CT = 0
*
*検索開始
*
GOTO TOP
DO WHILE .NOT. EOF()
   IF GENKEI = W1 .AND. HINSI = HS1 .AND. LEN(TRIM(GENKEI)) = LEN(TRIM(W1))
         KAISU = KAISU + 1
         SU = NO
   ENDIF
   IF TANGO = "EOS"
      IF KAISU >0
        SKIP   -1*(UK-CT-1)
        DO WHILE SU=NO
```

```
            IF GENKEI = W2 .AND. HINSI = HS2 .AND. LEN(TRIM(GENKEI)) =
LEN(TRIM(W2))
                SHUTU = SHUTU + 1
                KAISU = 0
                BREAK
            ENDIF
            SKIP 1
          ENDDO
          GOTO UK
        ENDIF
        CT = UK
    ENDIF
    IF UK < UTAKAZU[J]
        UK = UK+1
        GOTO UK
    ELSE IF UK =UTAKAZU[J]
        BREAK
    ENDIF
ENDDO
CLOSE ALL
*
*結果出力
*

? W1
?? "("
?? HS1
??")"
??"と"
?? W2
?? "("
?? HS2
??")"
??  "の歌数は"
?? SHUTU
??"    です。"
RETURN
```

YCY21

```
*YCY22.PRG*************************************************************************
*JUMANの解析結果から品詞付きの２単語の出現する歌をデータベース化するプログラム*
**                                                    1998.2.28    SATOKO FUKUSHI      **
************************************************************************************
*** ファイル一覧 ***
* DATA1    :  形態素解析の結果
* UTA2     :  単語を含む歌を入れるファイル
* DUMMY    :  UTAファイルを初期化するためのファイル
* HATIDAI :   八代集のデータベース
*** 変数一覧***
* FILE[]  :  ファイル名を入れた配列
* F1       :  ファイル名の配列の添字（キーボード入力）
* HIN1[]  :  品詞１を入れた配列
* HIN2[]  :  品詞２を入れた配列
* I        :  F1を数値型に変換した値
* J        :  Iから小数部をとった値
* FL       :  ファイル名
* W1       :  検索したい文字列1(キーボード入力)
* H1       :  品詞の配列の添字（キーボード入力）
* K        :  H1を数値型に変換した値
* L        :  Kから小数部をとった値
* M        :  H2を数値型に変換した値
* N        :  Mから小数部をとった値
* HS1,HS2:   品詞名
* UAKAZU[]:  それぞれの歌集の歌数を入れた配列
* KAISU    :  文字列１が一つの歌に含まれるなら１以上の値を持つ
* UK       :  調べている位置を表す
* SU       :  単語１含む歌番号
* CT       :  前の歌のEOSのレコード番号
************************************************************************************
*
*入力
*

?"1: 古今集"
?"2: 後撰和歌集"
?"3: 拾遺和歌集"
?"4: 後拾遺和歌集"
?"5: 金葉和歌集"
?"6: 詞花和歌集"
?"7: 千載和歌集"
?"8: 新古今和歌集"
ACCEPT "ファイル番号を入力してください :" TO F1
```

```
ACCEPT "単語１を入力してください ＞" TO W1
ACCEPT "単語２を入力してください ＞" TO W2
?"１：名詞"
?"２：助動詞"
?"３：助詞"
?"４：動詞(原型)"
?"５：形容詞(原型)"
?"６：形容動詞"
?"７：副詞"
?"８：連体詞"
?"９：接続詞"
?"10：接尾辞"
?"11：枕詞"

ACCEPT "単語１の品詞番号を入力してください ＞" TO H1
ACCEPT "単語２の品詞番号を入力してください ＞" TO H2
clear
***
SET TALK OFF
*
*書き込み用ファイルの初期化
*

USE DUMMY
COPY TO UTA2
CLEAR

DECLARE FILE[10]
FILE[1] = "DATA1"
FILE[2] = "DATA2"
FILE[3] = "DATA3"
FILE[4] = "DATA4"
FILE[5] = "DATA5"
FILE[6] = "DATA6"
FILE[7] = "DATA7"
FILE[8] = "DATA8"

DECLARE UTAKAZU[10]
UTAKAZU[1] = 19596
UTAKAZU[2] = 25420
UTAKAZU[3] = 23946
UTAKAZU[4] = 22183
UTAKAZU[5] = 11589
UTAKAZU[6] = 7565
UTAKAZU[7] = 23089
UTAKAZU[8] = 35573

DECLARE HIN1[15]
```

```
HIN1[1] = "名詞"
HIN1[2] = "助動詞"
HIN1[3] = "助詞"
HIN1[4] = "動詞"
HIN1[5] = "形容詞"
HIN1[6] = "形容動詞"
HIN1[7] = "副詞"
HIN1[8] = "連体詞"
HIN1[9] = "接続詞"
HIN1[10] = "接尾辞"
HIN1[11] = "枕詞"

DECLARE HIN2[15]
HIN2[1] = "名詞"
HIN2[2] = "助動詞"
HIN2[3] = "助詞"
HIN2[4] = "動詞"
HIN2[5] = "形容詞"
HIN2[6] = "形容動詞"
HIN2[7] = "副詞"
HIN2[8] = "連体詞"
HIN2[9] = "接続詞"
HIN2[10] = "接尾辞"
HIN2[11] = "枕詞"
***
SET TALK OFF
***
*
*使用ファイルのオープン&アリアス指定
*

SELECT B
USE UTA2
SELECT A

I = VAL(F1)
J = INT(I)
FL=FILE[J]
USE &FL
***
*
*使用変数の初期化
*
K = VAL(H1)
L = INT(K)
HS1 = HIN1[L]
M = VAL(H2)
N = INT(M)
```

```
HS2 = HIN2[N]

KAISU=0
UK = 1
CT = 0
*
*検索開始
*
GOTO TOP
DO WHILE .NOT. EOF()
   IF GENKEI = W1 .AND. HINSI = HS1 .AND. LEN(TRIM(GENKEI)) = LEN(TRIM(W1))
         KAISU = KAISU + 1
         SU = NO
   ENDIF
   IF TANGO = "EOS"
      IF KAISU >0
        SKIP   -1*(UK-CT-1)
        DO WHILE SU=NO
           IF GENKEI = W2 .AND. HINSI = HS2 .AND. LEN(TRIM(GENKEI)) =
LEN(TRIM(W2))
              SELECT B
              *
              *結果書き込み
              *
              APPEND FROM HATIDAI FOR NO=SU
              SELECT A
              KAISU = 0
              BREAK
           ENDIF
           SKIP 1
        ENDDO
        GOTO UK
      ENDIF
      CT = UK
   ENDIF
   IF UK < UTAKAZU[J]
       UK = UK+1
       GOTO UK
   ELSE IF UK =UTAKAZU[J]
       BREAK
   ENDIF
ENDDO
CLOSE ALL
RETURN
```