『言語情報学研究報告』No.15（2006）

日本語流行歌中の外来語に関する調査[1]

吉田　桃子
（東京外国語大学大学院博士後期課程）
望月　源
（東京外国語大学外国語学部　講師）

1. はじめに

本研究は流行歌の歌詞を手がかりにして、日本語における外来語の役割を、その量と使われ方の変化から明らかにすることを目的としている。

一般に外来語は、大きく分けて、

1. これまでになかった事物を言い表す
2. 洗練された雰囲気を作りだす

といった目的で用いられることが多い。外来語が多く使われる分野として、いわゆる「お役所文書」や「流行歌」が挙げられる。前者においては、情報伝達が第一の目的なので、外来語の多用が分かりにくさにつながるとしてしばしば問題になる。一方後者においては、その内容は基本的に歌詞の作者が自由に考えることができる。外来語の利用が意味のわかりにくさにつながるかどうかは問題にならず、語や表現の選択に特に制約はない。ただし、多くの人に受け入れられたものが流行歌となる。このことから日本語流行歌に使われる外来語の量と内容は日本人が外来語に対して持つイメージや理想を極端な形で反映していると考えられる。したがって流行歌歌詞の外来語使用の実際を調査することは、外来語一般のイメージを明らかにする一つの手段となりうる。

従来から流行歌を対象にした語彙調査は多いが、外来語に重点を置いたものは少ない。外来語を調査対象としたものでも、他の語種を考慮していなかったり、調査対象期間が限られているものがほとんどである（堀江、1966；原、1977；米田、1980）。本研究では、歌詞全体に占める外来語の実質的割合を知るために、語種に関しては、歌詞のすべての自立語を数え上げの対象にする。また、外来語使用の量的推移をよりはっきりした形で観察するために、調査対象期間に関しては、明治元年から平成 14 年上半期までの 135 年という比較的幅のある期間を調査対象にする。

本調査では、流行歌歌詞における外来語の量的変遷、外来語の品詞、外来語が多く使われる意味分野の変遷、具体的な高頻度外来語について報告する。

[1] 本研究報告は、言語処理学会第 9 回年次大会（NLP2003）において、大会優秀発表賞を受賞した研究発表に加筆修正したものである。

2. 単語の認定と語種の定義

2.1. 単語の認定基準

日本語における「単語」の定義は研究者によって様々である。そのため語彙調査では「単語」をどのような単位で設定するかが常に問題となる。本研究では、単語に区切る際の単位は国立国語研究所『テレビ放送の語彙調査』(国立国語研究所、1995）で用いられた「長い単位」に準ずる。

今までに国立国語研究所の行った語彙調査では、文節（単語）相当の長い単位系のもの（α 単位、長単位、W 単位、長い単位）と、形態素相当の短い単位系のもの（β 単位、短単位、M 単位）がある。一般に長い単位の系列は、素材テキストの特殊な側面（小説の文体的特徴など）を分析する場合に適しており、短い単位の系列は、素材テキストの一般的な側面（日本語のテキスト一般に共通する特徴）を分析する場合に適している。例えば日本語の基本語彙を選定したいという場合などである（伊藤、2002)。また、短い単位の系列は語構成や造語力などを調査する際に適していると言われる。このような事情から先行の語彙調査においても調査ごとに目的に適した単位が選ばれてきた。本研究で「長い単位」を採用したのは以下の理由からである。

- 「長い単位」は文節を認定して、そこから自立語を取り出すという手順で決定されるため、「短い単位」よりも語認定におけるばらつきが少ない。
- 複合語を単語として認めるため、外来混種語を扱うことが可能である。

また、「単語」の同語異語判定も単語の認定基準として問題になるが、調査全体に与える影響は「単語」自体の定義よりも小さい。本調査では文字テキストを基準にした処理が原則なので、語のまとめあげは表記を基準にして行う。表記（ひらがなと漢字など）が違っているだけで、同語だと判定したものは同一の見出しにまとめる。ただし、今回の調査対象の中で 2 種類以上の漢字表記があるものはまとめない。例えば「青い」と「蒼い」は別々の見出し語とする。さらに「あおい」があった場合はまた別の見出し語とする。「青い」と「あおい」だけがあり、他に「あおい」と読む同じ意味の漢字表記がない場合にだけ「青い」と「あおい」をまとめて同語とする。また、外来語の短縮形は元の形に戻さずそれぞれ別の見出しとする。例えば「ハンカチ」は「ハンカチーフ」とは別の見出しとして扱う。「イギリス」の省略形とみなせる「英」も別の見出しとする。ただし、「ハンカチ」と「ハンケチ」のように同一の外来語で表記が異なるものについては、音の揺れであるとみなして、同一見出しとする。

2.2. 語種の定義

語の原籍に基づいて語彙を分けていく分類法を語種分類という。一般的に日本語の語種分類は「和語」、「漢語」、「外来語」の 3 分法をとる（田中、1978）が、本研究では、「外来語」との対比において、「和語」と「漢語」を区別せずに「和語・漢語」とする。また、歌詞には日本語以外の言語種も使われているため、これらをまとめて「外国語」として扱う。さらに「外来語」と「和語・漢語」によって構成される複合語を「外来混種語」とする。

以上、本調査の語種・言語種分類は「外来語」「外国語」「外来混種語」「和語・漢語」の 4 種類である。なお、「外来語」、「外国語」の定義は文献（伊藤、2001）に準じ、以下のように定義する。

- 外来語：仮名かローマ字表記の外国出自の単語
- 外国語：アルファベット表記の外国出自の単語

その他、語種認定の細則は次のとおりとする。

- 漢字で書かれたものもでもルビがある場合は、ルビで語種や言語種を判断する。
- アラビア数字は表記でなく音声を頼りに決定する。日本語読み以外は外来語とする。
- 原語の異なるものは結合しない。違う言語どうしの複合語（英語＋日本語など）は認めないこととする。例えば「candle ライト」は「candle」（外国語＝英語）と「ライト」（外来語＝日本語）に分ける。ただし、アルファベットをアルファベット読みしたものと日本語の複合語は例外として認め、外来語に分類する（「T（ティー）シャツ」「Z（ゼット）旗」など）。

本研究での言語種、語種の区分をまとめると以下の表 1 のようになる

表 1　言語種と語種

言語種	日本語			4.外国語 （例：romantic）
語種	1.外来語 （例：アドレス、テールランプ）	2.外来混種語 （外来語と外来語以外の複合語） （例：ガラス窓）	3.和語・漢語 （例：家出、家出する）	

3. 調査方法

調査の具体的な手続きは次のとおりである。歌詞全体を単語に区切り、語種、原語、品詞、意味コードなどの情報をつけた語彙表を作成する。付加された情報に基づいて語の数え上げを行い、得られた数値を分析する。なお語の切り出しのために日本語形態素解析器「茶筅」と、日本語係り受け解析器「南瓜 windows 版[2]」を用いる（工藤・松本、2002）。

3.1. 歌詞の収集

調査対象期間は明治元（1868）年から平成 14（2002）年上半期までの 135 年とし、ほぼ 10 年ごとに 14 の年代に区切る。調査対象曲は、1868 年から 1967 年までは歌本『日本のう

[2]「南瓜」用の形態素解析器は「茶筅 2.1 for windows」が用いられている。

た』（椎葉、1998－2001）から 1 つの年代につき 50 曲を上限に無作為抽出、1968 年以降は『オリコン年鑑』（小池、1968－2001）の年間売上ランキングから各年 1 年あたりその年の上位 5 曲を選ぶ。

このようにして得られる調査対象曲は合計 584 曲であり、年代区分と対象曲数は表 2 のとおりである。

表 2　年代区分と対象曲数

			年	曲数
1	1868 (明治元)	～ 1874 (明治 7)	7	8
2	1875 (明治 8)	～ 1884 (明治 17)	1	14
3	1885 (明治 18)	～ 1894 (明治 27)	1	28
4	1895 (明治 28)	～ 1904 (明治 37)	1	44
5	1905 (明治 38)	～ 1914 (大正 3)	1	50
6	1915 (大正 4)	～ 1924 (大正 13)	1	50
7	1925 (大正 14)	～ 1934 (昭和 9)	1	50
8	1935 (昭和 10)	～ 1944 (昭和 19)	1	50
9	1945 (昭和 20)	～ 1954 (昭和 29)	1	50
10	1955 (昭和 30)	～ 1964 (昭和 39)	1	50
11	1965 (昭和 40)	～ 1974 (昭和 49)	1	50
12	1975 (昭和 50)	～ 1984 (昭和 59)	1	50
13	1985 (昭和 60)	～ 1994 (平成 6)	1	50
14	1995 (平成 7)	～ 2002 (平成 14)	8	40

3.2. 前処理

流行歌の歌詞は書き言葉による文とは異なり、以下のような話し言葉に近い性格を持っている。

1. 意味のない繰り返しがある。
2. 倒置が頻繁に起こる。
3. 助詞･助動詞の脱落が多い。
4. 文が完結しない、あるいは他の語句の割り込みによって中断することがある。

これらの特徴を持つテキストは、そのままでは「南瓜」のような言語処理ツールによる解析がうまくいかない。そのため歌詞テキストに対し以下のような前処理を行う。

- 旧仮名遣いを現代仮名遣いに変換する。
  例：「言ふ」→「言う」
- 旧字体を新字体に変換する。
  例：「痴話が嵩じて」→「痴話が高じて」
- 標準的でない表記を適宜「南瓜」で認識される表記に変換する。
  例：「たぐい」→「類」、「ワルいヤツ」→「悪い奴」
- 助詞の脱落により文節区切りが明確でない箇所や語が並列されている箇所を区切

る（「紅い帯締め」→「紅い帯（を）、締め」）。

これらの前処理を歌詞テキストに対して行う際に、元のテキストの情報を完全に書き換えてしまうことは、将来、万が一元の歌詞が必要になった時などを考えると好ましくない。そのため、テキストの加工ではタグを定義し、いつでも変更箇所を元の状態に戻せるような形式で行うこととした。タグは、元の歌詞を<o></o>で囲み、修正した部分を<r></r>で囲むこととする。以下にタグ付けの例を示す。

例：都へ来てから <o>幾歳</o><r>幾年</r>ぞ

例：包丁一本 <o>さらし</o><r>サラシ</r>に巻いて

また、歌詞の特徴として、同じフレーズを繰り返すことが多い。その際繰り返しを直接記述せずに、記号として、「※」「△」「▲」「☆」や「★」を用いて繰り返す部分の指定と、何回繰り返すかを示していることがある。そこで、タグによる修正を反映させる部分と歌詞の繰り返し部分を展開する部分の両方を処理するプログラムを perl 言語を用いて自作した（付録 A に記す）。

3.3 文節の取り出し

既に述べたように本調査では単語の単位を「長い単位」とするが、これは文節における自立語にほぼ相当する。ここでは、「茶筅」と「南瓜」を使って次の手順で文節を作成する。

1. 形態素解析器「茶筅」により歌詞を形態素に区切り、各形態素に品詞を付与し、続いて係り受け解析器「南瓜」の文節作成機能を利用し、1 の形態素情報から、文節を作成する。（図に例を示す）

```
* 0 2D 1/2 2.09907294
南国  ナンゴク    南国  名詞-一般
土佐  トサ  土佐  名詞-固有名詞-地域-一般
を    ヲ    を    助詞-格助詞-一般
、    、    、    記号-読点
* 1 2O 0/1 0.00000000
後    ゴ    後    名詞-接尾-副詞可能
に    ニ    に    助詞-格助詞-一般
* 2 -1O 0/1 0.00000000
し    シ    する  動詞-自立   サ変・スル 連用形
て    テ    て    助詞-接続助詞
EOS
```

図 1 南瓜の処理結果の例

南瓜では「*」以下が文節のまとまりを示している。例えば、図 1 の最初の文節は「南国（名詞）」「土佐（名詞）」「を（助詞）」「、（記号－読点）」の 4 種類の形態素から 1 つの文節（文節 0 番）が出来ていることを示している。最後の「EOS」は End Of Sentence の略であり文の切れ目を示す。

2. 文節の中で、1 の処理ではうまくいかなかった部分を修正する。ここでは、1 で 1 つ

の文節として処理された中で、別々の文節に離して扱うべきものを別の文節に分割する。
例えば、次のような場合は、文節を分割する必要がある

- 接頭辞が連続する場合 1 つの文節と判断される。

```
* 3 4D 3/3 0.32699827
超　チョウ　超　接頭詞-名詞接続
超　チョウ　超　接頭詞-名詞接続
超　チョウ　超　接頭詞-名詞接続
超　チョウ　超　接頭詞-名詞接続
* 4 5D 0/0 0.00000000
いい　イイ　いい　形容詞-自立　不変化型　基本形
* 5 -1O 0/0 0.00000000
感じ　カンジ　感じ　名詞-一般
EOS
```

図 2　接尾辞が連続する例

例えば図 2 では、「超」が「超超超超」と連続するが、それぞれを 1 つに分ける必要がある。

- 名詞と未知語（外国語）が同じ文節になっている場合別にする。

```
* 0 1 1/1 0.00000000
乾杯　カンパイ　乾杯　名詞-サ変接続
baby　baby　baby　未知語
* 1 -1O 0/0 0.00000000
！　！　！　記号-一般
EOS
```

図 3　名詞と未知語の組み合わせの例

例えば図 3 では、名詞である「乾杯」と外国語である「baby」が同じ文節になっている。本調査では外国語は 1 つの単語として扱うため分割する必要がある。
そのほか以下のような場合にも、別々の文節に分割する必要がある。

- 「名詞」と「動詞」(「唄/ひびく」)。ただし「名詞-サ変接続」と「名詞-形動語幹」に接続する「スル」は切らない（「仕事する」「寝坊する」)。
- 「助詞、助動詞」と「名詞、未知語、動詞」(「大将たる/者」「好きに/なる」)。
- 「名詞-接尾（助数詞は除く）」と「名詞（接尾と非自立は除く）」または「動詞」(「見ずや/夕暮れ」「見合すネ/顔と」)（「十八・年・目・で」は切らない)。
- 「名詞」と「連体詞」、「連体詞」と「名詞」(「拙者/この/町に」)。
- 「形容詞」と「動詞」(「早く/来い」)。

以上のような問題に対処するため、自作プログラム（付録 B）を作成し、文節を分割

する。

3. 文節の中で、1 の処理ではうまくいかなかった部分を修正する。ここでは、1 で別々の文節として処理された中で、1 つの文節にまとめて扱うべきものを 1 つの文節にする。文節が必要以上に区切れてしまっている以下のような箇所は、直前の文節とつなげる。
   - 「助詞」または「助動詞」で始まる文節。
   - 「形式動詞（「…という」の「いう」など）で始まる文節。

   以上のような問題に対処するため、自作プログラム（付録 C）を作成し、文節を結合する。

最終的に以上の規則を適用しても、うまく区切れなかった部分をチェックし、手作業で修正する。なお、今回の形態素解析結果では、アルファベット語（本調査における外国語）は「未知語」として区分されるので、この段階で文脈上の品詞を付加する。

3.4. 「長い単位」の取り出し

一つの文節から一つの自立成分またはそれに準ずるものを取り出す。複合語は形態素に付加された情報を手がかりに認定し、複合した形で取り出す。複合語と規定するのは以下のような品詞の連続である。

- 「名詞」と「名詞、未知語」。
- 「動詞」と「動詞、名詞、形容詞-非自立」（「押し+出す」「くわえ+煙草」「住み+良い」）。
- 「名詞、未知語、副詞」と「スル」。

以上の規定に準拠し、一律に単語の取り出しを行うための以下の出力を行うプログラムを自作した（付録 D）。

単語；品詞；ヘッド情報；ヘッドとなる形態素情報；全体的形態素情報；曲 ID

これにより各曲の各単語 1 つずつについて、後で語彙表のために集計したり見出し語を決定するための情報が得られる。なお、ここで「ヘッド」とは、長い単位の単語が複数の形態素からなる場合に、その中でヘッドとなる語を捉えるための処理を意味する。

3.5. 語彙表の作成

3.4 節で得られた「長い単位」の単語リストから、2.1 節で述べた基準により同じと認められるものは同一の見出し語の元にまとめて、見出し語のリストを作成する。重複する出典情報（どの歌に出現するか）をまとめて、出現回数をつける。外来語、外来混種語、外国語と判断したものに印をつける。外来語に分類したものには、分類語彙表（増補版）（中野、1996）の意味コードをつける。最終的に得られるデータの内容と形式は以下のとおり

である。

通し番号；語種；見出し語；品詞；出現形；出典情報；回数（；意味コード）

語彙表の例を図 4 に示す。

```
270;F;romantic;ADJ;1985-4:6
604;Ge;アドレス;名詞; 1984-3:1;1.1700
5657;Ged;テールランプ;名詞;1993-3:1;1.4600
2463;Hdj;ガラス窓;名詞;1947-5:1 1962-3:1 1985-5:1
779;J;家出;名詞;1888-3:1
780;J;家出する;動詞;1888-3:2 1970-1:1
```

図 4　語彙表の例

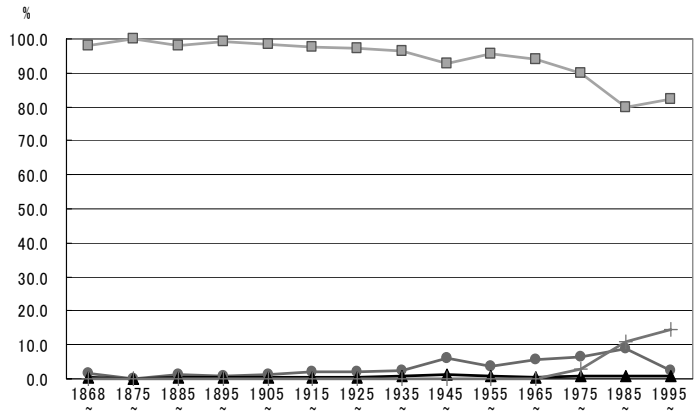
4. 結果と考察

調査対象全体から、異なりで 9,397 語、延べで 48,231 語が得られた。語種の内訳を表 3 に示す。

表 3　年代を通じての語種別語数

	異なり	延べ
和語･漢語	8,446	43,741
外来語	486	1,141
外来混種語	106	160
外国語	359	3,189

和語・漢語以外の語種が年代を通じて歌詞全体に占める割合は、異なり、延べともに 10%程度である。それぞれの語種が歌詞の異なり語数に占める割合を年代別に示すと図 5 のようになる。



	1868 ~	1875 ~	1885 ~	1895 ~	1905 ~	1915 ~	1925 ~	1935 ~	1945 ~	1955 ~	1965 ~	1975 ~	1985 ~	1995 ~
和語・漢語	97.9	100.0	98.0	99.1	98.3	97.7	97.3	96.5	92.8	95.6	93.8	89.8	79.8	82.2
外来語	1.7	0.0	1.4	0.6	1.3	1.9	2.2	2.6	6.0	3.7	5.5	6.3	8.7	2.6
外来混種語	0.4	0.0	0.6	0.2	0.4	0.3	0.5	0.9	1.2	0.7	0.5	1.0	0.7	0.6
外国語	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	2.9	10.8	14.5

図 5　異なり語数に占める語種の割合

全年代を通じて歌詞の大部分を占めているのは和語・漢語である。外来語の割合は歌詞全体から見れば多くはない。1945 期の増加と 1995 期の減少が多少目立つ程度である。外来語に代わって 1975 期から増加してくるのが外国語である。外来語が減って外国語が増えているのは、外来語が担っていた装飾的機能（1 節で述べた外来語使用の目的 2）が外国語に移行したからだと考えられる。つまり外来語の利用価値が下がって外国語の利用価値が相対的に上がったということである。1995 期に和語・漢語が増える兆候が見られるのは、和語・漢語の意味伝達機能が見直されたためだと考えられる。このように、それぞれの語種はその機能や利点によって使い分けられていると推測される。

外来語の品詞は名詞がほとんどである。一方、外国語の品詞を使用量の安定する 1975 期以降についてみると、名詞も多いが、その他の品詞も多いことが分かる。これは外来語が単語単位で取り入れられることが多いのに対し、外国語が文章単位で取り入れられることが多いことの表われである。

外来語がほとんど名詞であることは上に述べたが、実際の品詞の内訳を図 6 に示す。流行歌に使われる外来語名詞の意味を分類語彙表による分類に従って意味コード 2 桁目まで分類した結果を図 7 に示す。

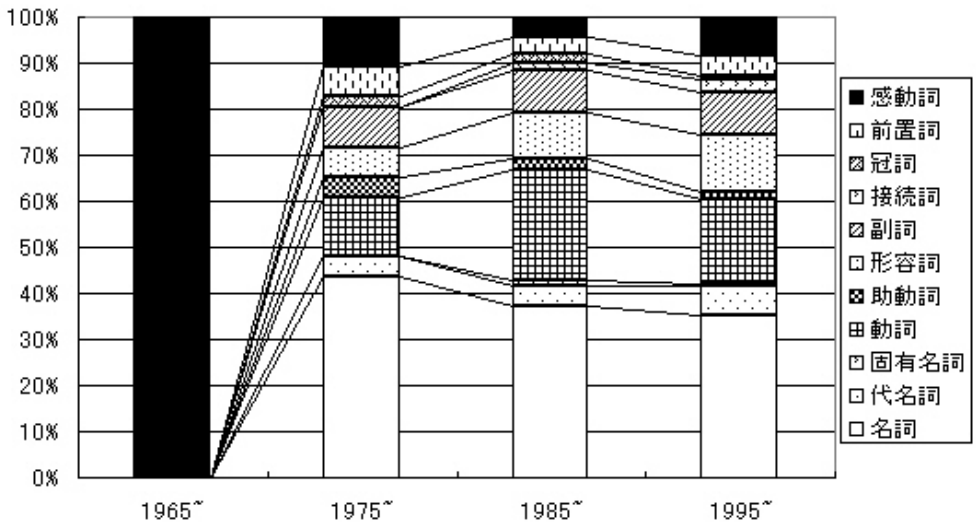


図 6　外国語の品詞（異なり）

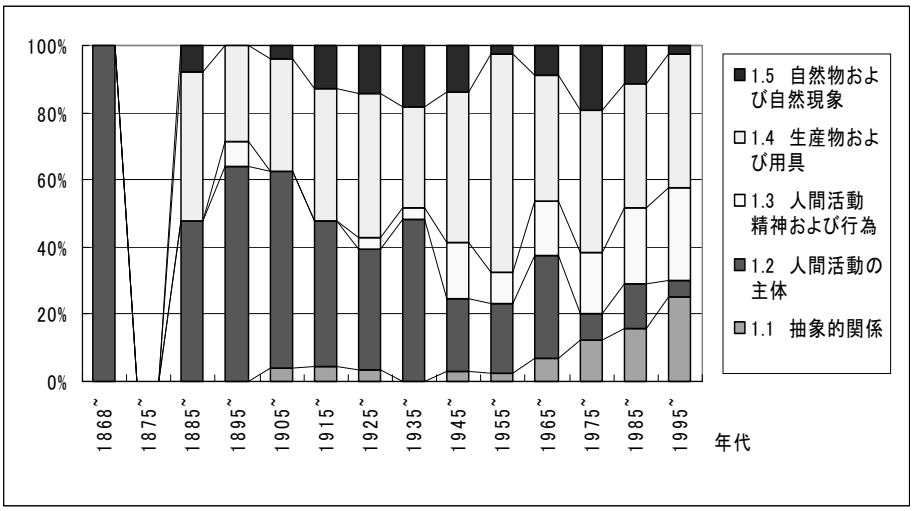


図 7　外来語（異なり）体の類下位区分の割合

意味コードの内容は表 4 のとおりである。

表 4　分類語彙表の意味コード

1.	体の類(名詞類)	
	1.1	抽象的関係
	1.2	人間活動の主体
	1.3	人間活動　精神および行為
	1.4	生産物および用具
	1.5	自然物および自然現象

どの年代の外来語にも多いのは、具体物を示す名詞（1.4 の分類）である。ただし 1945 期以降は抽象的な語（1.1、1.3 の分類）の割合が増えている。その理由は外来語一般に抽象語が多くなってきたことと、流行歌における外来語の使い方が変わってきたことが考えられる。雰囲気を演出する小道具として使われる外来語の他に、雰囲気や感情を直接表現する外来語が使われるようになってきたのである。

具体的な高頻度外来語を見ても上に述べたような傾向が分かる。年代別の高頻度外来語を表 5 に示す。高頻度語は通常、延べ語数から計算した使用率によって示されるが、歌詞特有の繰り返しの影響を避けるため、一つの年代について複数曲に登場する外来語を示している。なお 1868～1885 期の 3 つの年代は複数曲に出現する外来語が一つもない。

表 5　年代別、複数曲に出現する外来語

1895 期	ラッパ、トンネル
1905 期	バイロン
1915 期	シナ
1925 期	パリ、ポプラ、アカシア
1935 期	タバコ、シナ、ランタン
1945 期	グラス、ホテル、ハンカチ、パイプ、ブルース、ランプ、テープ
1955 期	ギター、ビル、ネオン
1965 期	ドア、タバコ、ブルー、レース（飾り）
1975 期	キス、タバコ、ボトル
1985 期	キス、ガラス、コイン、リズム、ベッド、ベル、ラジオ、ダイヤモンド、スキャンダル、エクスタシー、オレンジ、キャンドル・ライト、ターン、バラード
1995 期	ガラス、ドア、キス、ベル、メロディ、リスク、コロン、ストーリー

1935 期以前は地名・人名の固有名詞や具体名詞がほとんどである。1945 期以降は固有名詞が見当たらなくなり、一般名詞がほとんどになる。その中には具体名詞とともに抽象名詞が現われている。1935 期以前は外来語の数自体が少なかっただけなく、和語・漢語で置き換えにくい、つまり必要のため（1 節で述べた外来語使用の目的 1）に使われる外来語が多かったということである。

歌詞に使われる外国語はほぼすべてが英語であった。語の種類という点からは、外来語

より外国語に定型化の傾向がみられる。どの年代においても使用頻度が高い外国語は「I」と「you」である。これは近年の流行歌に描かれるのが基本的に「私」と「あなた」の世界であるということと、日本人の考える外国語（ほぼ＝英語）の構文的、語彙的バリエーションがまだ少ないということを意味していると考えられる。

5. おわりに

流行歌の歌詞を「単語」に区切り、語種、言語種別に分類し、それぞれの使用量を調べた。外来語の割合はそれほど多くなく、外国語の割合が近年になって急増していることが分かった。また、語種はその機能によって使い分けられているということがわかった。

今後、調査対象曲数を増やして、和語・漢語も含めた分析を行うことによって、語種の使い分けの原則、ひいては使用者の意識や語と語種に対するイメージを明らかにできると考えられる。

参考文献

堀江真喜雄　1966：「『ネオン』と『ギター』外来語使用から見た歌詞の一側面」『言語生活』178, pp.70-74　筑摩書房

原忠彦　1977：「歌謡曲における漢字音語と外来語」『アジア・アフリカ言語文化研究所通信』31, pp.5-9　東京外国語大学アジア・アフリカ言語文化研究所

米田武　1980：「流行歌の中の外国語」『文研月報』30-7, pp.32-38　日本放送協会

田中章夫　1978：『国語語彙論』　明治書院

伊藤雅光　2001：「ポップス系流行歌の語彙調査における外来語と外国語の判定基準」『計量国語学』23-2, pp.110-130　計量国語学会

工藤 拓・松本 裕治　2002：「チャンキングの段階適用による日本語係り受け解析」『情報処理学会論文誌』43-6, pp.1834-1842

野ばら社編集部・椎葉京一　1998-2001：『日本のうた』第 1 集～第 7 集　野ばら社

小池聰行　編　1968-2001：『オリコン年鑑』　オリコン

国立国語研究所・中野洋　1996：『「分類語彙表」形式による語彙分類表（増補版）』第 1・2 分冊　国立国語研究所

国立国語研究所　1995：『テレビ放送の語彙調査』

伊藤雅光　2002：『計量言語学入門』　大修館書店

付録

A　前処理用プログラム

主な機能：<o></o>タグで囲まれた部分を削り、<r></r>で囲まれた部分を残す。歌詞の繰り返し部分を示す「※」などの記号に従って、展開をする。

```
# kabochamae.pl
# Copyright (c) 2002 MOCHIZUKI Hajime <motizuki@tufs.ac.jp>
#   Fuculty of Foreign Studies,
#   Tokyo University of Foreign Studies
# usage: perl kabochamae.pl < lylics.txt
#
$zenkaku = "ａｂｃｄｅｆｇｈｉｊｋｌｍｎｏｐｑｒｓｔｕｖｗｘｙｚＡＢＣＤＥＦＧＨＩＪＫＬＭＮＯＰＱＲＳＴＵＶＷＸＹＺ０１２３４５６７８９";
$hankaku = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ０１２３４５６７８９";
@zen = split (//, $zenkaku);
@han = split (//, $hankaku);
$title=<>; # 先頭行はタイトル
chomp ($title);

while($line=<>) { # ファイルを１行読み込んで，$line に代入する.
     chomp ($line);
     if ($line!~/./ || $line =~ /^[ |　]+$/) { next;} # 空行ならこれ以上処理しない.
    # ORIGINAL の中身を無視して，REVISED の中身を残す部分.
    if ($line =~ /<o>/i) { # 修正個所があるので，以下を処理する.
          @a = split(/</, $line); # line の中身を '<' で区切って配列 a に代入する;
          $line = $a[0];
          for ($i=3; $i<=$#a; $i++) {
               @b = split(/>/, $a[$i]); # 「R>」を含む部分を>で区切る.
               $line = $line . $b[1];   # 例えば，$b[0]=R で $b[1]=雫
               $i++;
               @b = split(/>/, $a[$i]); # 「/R>」を含む部分を>で区切る.
               $line = $line . $b[1];   # 例えば，$b[0]=/R で，$b[1]=も
               $i=$i+2;
          }
     }
     $line =~ s/¥(/¥（/g; # 半角()を全角（）になおす
     $line =~ s/¥)/¥）/g;
     if ($line =~/¥（/) { # $line に（があれば以下の操作をする
          @a = split(/¥（/, $line); #　（を目印に$line を区切り@a に代入
          $line = $a[0]; #　（より前の部分を$line に代入
          $pre = $a[0]; # $pre にも代入
          for ($i=1; $i<=$#a; $i++) {
               @b = split(/¥）/, $a[$i]);
               if ($b[0]=~/^[ぁ-ん]+$/) { #　（）内が全部ひらがなで
                    if ($pre=~/[亜-腕]+$/ || $pre=~/[弐-熙]+$/) { # 直前が漢字の時は
                         $line = $line . $b[1]; # () 内を全部消す
                    } else { # それ以外の場合は
                         $line = $line . "（" . $b[0] . "）" . $b[1]; #
                    }
               } elsif ($b[0] eq "男" || $b[0] eq "女" || $b[0] eq "男女" || $b[0] eq "二人") {
                    $line = $line . $b[1]; # ()内を全部消す;
               } else {
                    $line = $line . "（" . $b[0] . "）" . $b[1];
```

```
            }
            $pre=$a[$i];
        }
}
$line =~ s/ [|] |¥[|¥]/ /g; # 全角/半角[]を削除し、空白を挿入.
$line =~ s/[ |  ]+/ /g; # 1個以上くりかえして半角/全角空白があるとき半角1個にする.
$line =~ s/^ //; # 行頭の空白を削除する
$line =~ s/ $//; # 行末の空白を削除する
$line =~ s/[･|・]{2,}/…/g; # 2個以上くりかえして半角/全角中点があるとき…1個にする.
$line =~ s/[.|. ]{2,}/…/g; # 2個以上くりかえして半角/全角ピリオドがあるとき…1個にする.
$line =~ s/ /、/g; # 空白を全角の「、」にする.
$line =~ s/[、]{2,}/、/g; # 2個以上繰り返して「、」があるときには1個にする
$line =~ s/¥?/？/g; # 記号類を全角にする
$line =~ s/¥!/！/g;
$line =~ s/¥./. /g; # 半角カンマ類を全角にする
$line =~ s/､/、/g;
$line =~ s/,/, /g;
$line =~ s/[,､]/、/g; #「, 」と「､」が連続したら「、」1個にする
$line =~ s/[、]{2,}/、/g; #「、」が連続したら「、」1個にする
# 全角英字を半角に, 半角数字を全角にする.
if($line =~ /[ａ-ｚＡ-Ｚ０-９]/) {
     for($i=0; $i<=$#zen; $i++){
          $line =~ s/$zen[$i]/$han[$i]/g;
     }
}
$line =~ tr/A-Z/a-z/; # 大文字英字を小文字にする
# 繰り返し対策
if ($mark{"※"}==1) {
     $repeat{"※"}= $repeat{"※"} . "¥n" . $line;
}
if ($mark{"△"}==1) {
     $repeat{"△"}= $repeat{"△"} . "¥n" . $line;
}
if ($mark{"★"}==1) {
     $repeat{"★"}= $repeat{"★"} . "¥n" . $line;
}
if ($line=~/^※/ && $mark{"※"}==0) {
     $repeat{"※"}=$line;
     $mark{"※"}=1;
}
if ($line=~/^△/ && $mark{"△"}==0) {
     $repeat{"△"}=$line;
     $mark{"△"}=1;
}
if ($line=~/^★/ && $mark{"★"}==0) {
     $repeat{"★"}=$line;
     $mark{"★"}=1;
}
if ($mark{"※"}==1 && $line=~/※+$/) {
     $mark{"※"}=2;
     $repeat{"※"}=~s/※//g;
     if ($repeat{"※"}=~/¥ (/) {
          @a = split(/¥ (/, $repeat{"※"});
          if ($a[$#a]=~/×/) {
               @b = split(/¥) /, $a[$#a]);
               @c = split(/×/, $b[0]);
               $kuri = $a[0];
```

```
                for ($i=1; $i<$#a; $i++) {
                    $kuri = $kuri."(".$a[$i];
                }
                $repeat{"※"}=$kuri;
                for ($i=1; $i<$c[1]; $i++) {
                    $repeat{"※"} = $repeat{"※"} . "¥n" . $kuri;
                }
            }
        }
    }
    if ($mark{"△"}==1 && $line=~/△+$/) {
        $mark{"△"}=2;
        $repeat{"△"}=~s/△//g;
        if ($repeat{"△"}=~/¥ (/) {
            @a = split(/¥ (/, $repeat{"△"});
            if ($a[$#a]=~/×/) {
                @b = split(/¥) /, $a[$#a]);
                @c = split(/×/, $b[0]);
                $kuri = $a[0];
                    for ($i=1; $i<$#a; $i++) {
                        $kuri = $kuri."(".$a[$i];
                    }
                $repeat{"△"} = $kuri;
                for ($i=1; $i<$c[1]; $i++) {
                    $repeat{"△"}=$repeat{"△"} . "¥n" . $kuri;
                }
            }
        }
    }
    if ($mark{"★"}==1 && $line=~/★+$/) {
        $mark{"★"}=2;
        $repeat{"★"}=~s/★//g;
        if ($repeat{"★"}=~/¥ (/) {
            @a = split(/¥ (/, $repeat{"★"});
            if ($a[$#a]=~/×/) {
                @b = split(/¥) /, $a[$#a]);
                @c = split(/×/, $b[0]);
                $kuri = $a[0];
                    for ($i=1; $i<$#a; $i++) {
                        $kuri = $kuri."(".$a[$i];
                    }
                $repeat{"★"} = $kuri;
                for ($i=1; $i<$c[1]; $i++) {
                    $repeat{"★"}=$repeat{"★"} . "¥n" . $kuri;
                }
            }
        }
    }
    if ($line=~/くりかえし/ || $line=~/くり返し/ || $line=~/繰り返し/) {
        if ($line=~/※/) {
            printf ("%s¥n", $repeat{"※"});
        } elsif ($line =~/△/) {
            printf ("%s¥n", $repeat{"△"});
        } elsif ($line =~/★/) {
            printf ("%s¥n", $repeat{"★"});
        } else {
            printf ("%s¥n", $line);
```

```
        }
    } else {
        if ($mark{"※"}==2) {
            printf ("%s¥n", $repeat{"※"});
            $mark{"※"}=3;
        } elsif ($mark{"△"}==2) {
            printf ("%s¥n", $repeat{"△"});
            $mark{"△"}=3;
        } elsif ($mark{"★"}==2) {
            printf ("%s¥n", $repeat{"★"});
            $mark{"★"}=3;
        } elsif ($mark{"※"}==1 || $mark{"△"}==1 || $mark{"★"}==1) {
            ;
        } else {
            printf ("%s¥n", $line);
        }
    }
}
exit;
```

B　文節を分割するプログラム

```
#
# Copyright (c) 2002 YOSHIDA Momoko, MOCHIZUKI Hajime <motizuki@tufs.ac.jp>
#   Fuculty of Foreign Studies,
#   Tokyo University of Foreign Studies
# usage: perl bunsetsukugiri.pl < lylics.cbc
# cabocha で切れなかったところを切る
while (<>) {
    chomp;
    if ($_ =~ /¥*/ || $_ =~/EOS/) { # *か EOS が出てきた時に
        if ($wc == 1) { # $wc カウンタが 1 だったら
            printf ("%s¥n", $line[0]); # $line[0]を表示
        } elsif ($wc == 0) { # カウンタが 0 だったら
            ; # 何もしない
        } else {
            & cut_bunsetu_ck ($wc); # サブルーチンへ行く
        }
        printf ("%s¥n", $_); # $_を表示
        $wc = 0; # カウンタを 0 に戻す
    } else { # それ以外の場合は
        $line[$wc] = $_; # $line[$wc]に$_を代入して
        $wc++; # カウンタを 1 進める
    }
}
exit;

sub cut_bunsetu_ck {
    my ($wc) = @_;
    my (@a, @b, $i, $j);
    for ($i=0; $i<$wc; $i++) {
```

```
@a = split (/¥t/, $line[$i]);
if ($a[3] =~ /^接頭詞/ && $i>0) { # 接頭詞がきたとき前を切る
     print "*" . "¥n";
     printf ("%s¥n", $line[$i]);
} elsif ($a[3] =~ /^名詞-接尾/ && $a[3] !~ /助数詞/) { # 助数詞を除く名詞-接尾がきたとき
     printf ("%s¥n", $line[$i]);
     $j = $i+1;
     @b = split(/¥t/, $line[$j]); # 次に名詞(-接尾/-非自立を除く)か動詞が続いたら間を切る
     if ($b[3] =~ /^名詞|^動詞/ && $b[3] !~ /^名詞-接尾|名詞-非自立/) {
          print "*¥n";
     }
} elsif ($a[3] =~ /^名詞/) { # 名詞で始まる連続
     printf ("%s¥n", $line[$i]);
     $j = $i +1;
     @b = split (/¥t/, $line[$j]);
     if ($b[3] =~ /^連体詞/) { # 名詞＋連体詞の間を切る
          print "*" . "¥n";
     } elsif ($b[3] =~ /^動詞/) { # 名詞＋動詞のとき
          if (($a[3]=~/^名詞-サ変接続|^名詞-形容動詞語幹/ &&
              ($b[4]=~/^サ変・スル/||$b[2] =~/^できる$|^出来る$/)) ||
              ($b[3] =~ /^動詞-非自立/)) {
          # 名詞がサ変接続か形容動詞語幹で、動詞が「する」か「できる/出来る」の場合,
          # または名詞＋動詞-非自立なら切らない
               ;
          } else { # その他は切る
               print "*" . "¥n";
          }
     } elsif ($b[3] =~ /^名詞-サ変接続/) { # 名詞＋名詞-サ変接続のとき
          $k = $j +1;
          @c = split (/¥t/, $line[$k]);
          if ($c[3] =~ /^動詞-自立/ && $c[4] =~ /^サ変・スル/ &&
             $a[0] !~ /[ァ-ン]/ && $b[0] !~ /[ァ-ン]/) {
               # その後に続くのが動詞-自立の「する」で名詞＋名詞-サ変接続が
               # どちらもカタカナでなければ 2 つの名詞の間を切る
               print "*" . "¥n";
          }
     } elsif ($b[3] =~ /^未知語/) { # 名詞＋未知語のとき
          if ($b[0] =~ /^[a-zA-Z]+$/) { # 未知語がアルファベット語なら間を切る
               print "*¥n";
          }
     } elsif ($b[3] =~ /^副詞/) { # 名詞＋副詞のとき間を切る
          print "*¥n";            # ↓名詞＋形容詞(ひらがな以外)のとき間を切る
     } elsif ($b[3] =~ /^形容詞/ && $b[0] !~ /^[ぁ-ん]+$/) {
          print "*¥n";
     }
} elsif ($a[3] =~ /^動詞/) { # 動詞で始まる連続
     printf ("%s¥n", $line[$i]);
     $j = $i +1;
     @b = split (/¥t/, $line[$j]);
     if ($b[3] =~ /^未知語/) { # 動詞＋未知語なら間を切る
```

```
            print "*¥n";
        } elsif ($a[3] =~ /^動詞-自立/ && $b[3] =~ /^名詞-非自立/ &&
                $b[2] !~ "の|ん|よう|様|よ") {
                # 動詞-自立＋名詞-非自立(「の」「ん」「よう(様)」以外)なら間を切る
            print "*¥n";
        }
    } elsif ($a[3] =~ /^形容詞/) { # 形容詞で始まる連続
        printf ("%s¥n", $line[$i]);
        $j = $i +1;
        @b = split (/¥t/, $line[$j]);
        if ($b[3] =~ /^動詞/ && $b[0] !~ /^[ぁ-ん]+$/) {
            # 形容詞＋動詞(ひらがな以外)の間を切る
            print "*¥n";
        } elsif ($b[3] =~ /^名詞/ && $b[0] !~ /^[ぁ-ん]+$/) {
            # 形容詞＋名詞(ひらがな以外)の間を切る
            print "*¥n";
        }
    } elsif ($a[3] =~ /^連体詞/) { # 連体詞で始まる連続
        printf ("%s¥n", $line[$i]);
        $j = $i +1;
        @b = split (/¥t/, $line[$j]);
        if ($b[3] =~ /^名詞/) { # 連体詞＋名詞なら間を切る
            print "*" . "¥n";
        }
    } elsif ($a[3]=~ /^助詞-格助詞-連語/) {
        if ($a[0]=~/に従い/) {
            printf ("に¥t ニ¥t に¥t 助詞-格助詞-一般¥n");
            print "*" . "¥n";
            printf ("従い¥t シタガイ¥t 従う¥t 動詞-自立¥t 五段・ワ行促音便¥t 連用形¥n");
        } elsif ($a[0]=~/に従う/) {
            printf ("に¥t ニ¥t に¥t 助詞-格助詞-一般¥n");
            print "*" . "¥n";
            printf ("従う¥t シタガウ¥t 従う¥t 動詞-自立¥t 五段・ワ行促音便¥t¥n");
        } elsif ($a[0]=~/に従って/) {
            printf ("に¥t ニ¥t に¥t 助詞-格助詞-一般¥n");
            print "*" . "¥n";
            printf ("従って¥t シタガッテ¥t 従う¥t 動詞-自立¥t 五段・ワ行促音便¥t¥n");
        } else {
            print $line[$i] . "¥n";
        }
    } elsif ($a[3] =~ /^助動詞/) { # 助動詞で始まる連続
        printf ("%s¥n", $line[$i]);
        $j = $i +1;
        @b = split (/¥t/, $line[$j]);
        if ($b[3] =~ /^名詞-一般|^未知語/) { # 助動詞＋名詞-一般/未知語なら間を切る
            print "*¥n";
        } elsif ($b[3] =~ /^動詞/) { # 助動詞＋動詞で
            if ($b[2] =~ /^[ぁ-ん]+$/) { # 動詞がひらがななら間を切らない, その他は切る
                ;
            } else {
```

```
                    print "*¥n";
                }
            }
        } elsif ($a[3] =~ /^助詞/) { # 助詞で始まる連続
            printf ("%s¥n", $line[$i]);
            $j = $i +1;
            @b = split (/¥t/, $line[$j]);
            if ($b[3] =~ /^名詞|^未知語/) { # 助詞＋名詞/未知語なら間を切る
                print "*" . "¥n";
            } elsif ($b[3] =~ /^動詞-自立/) { # 助詞＋動詞-自立で
                if ($b[2] eq "いる") { # 動詞-自立が「いる」なら間を切らない，その他は切る
                    ;
                } else {
                    print "*¥n";
                }
            }
        } elsif ($a[3] =~ /^未知語/) {
            printf ("%s¥n", $line[$i]);
            $j = $i +1;
            $k = $j +1;
            @b = split (/¥t/, $line[$j]);
            @c = split (/¥t/, $line[$k]);
            if ($b[3] =~ /^記号/ && $c[3] =~ /^未知語/) { ; }
        } elsif ($a[0] eq "、") {
            printf ("%s¥n", $line[$i]);
            if ($i +1 < $wc) {
                print "*¥n";
            }
        } else {
            printf ("%s¥n", $line[$i]);
        }
        $line[$i] = "";
    }
}
```

C　文節を結合するプログラム

```
#
# Copyright (c) 2002 YOSHIDA Momoko, MOCHIZUKI Hajime <motizuki@tufs.ac.jp>
#   Fuculty of Foreign Studies,
#   Tokyo University of Foreign Studies
# usage: perl bunsetsukuttuke.pl < lylics.cbc
# cabocha で切れてしまったところをくっつける
#
$pre = "";
$EOS = "";
while (<>) {
    chomp;
    if ($_ =~ /^¥*/ && $pre ne "") {
        $tsugi = <>;
```

```
          chomp ($tsugi);
          @tsuginonakami = split (/¥t/, $tsugi);
          if ($tsuginonakami[3] =~ /^助詞|^助動詞/) { ;
          } elsif ($tsuginonakami[2] eq "もの") {
               @prenonakami = split (/¥t/, $pre);
               if ($prenonakami[3] =~ /^助動詞/) { ;
               } else {
                    print $EOS;
                    print $_ . "¥n";
               }
          } elsif ($tsuginonakami[3] =~ /^名詞/) {
               @prenonakami = split (/¥t/, $pre);
               if ($prenonakami[2] eq "ヶ") { ;
               } else {
                    print $EOS;
                    print $_ . "¥n";
               }
          } elsif ($tsuginonakami[3] =~ /^動詞/) {
               @prenonakami = split (/¥t/, $pre);
               if (($prenonakami[0] eq "と" && $tsuginonakami[2] eq "いう") ||
                    ($prenonakami[3] =~ /^助動詞/ && $tsuginonakami[2] =~ /^[ぁ-ん]+$/)) {
                    ;
               } elsif ($tsuginonakami[2] eq "する|できる|出来る|なさる|いたす") {
                    ;
               } else {
                    print $EOS;
                    print $_ . "¥n";
               }
          } elsif ($tsuginonakami[0] eq "さ" &&
                    $tsuginonakami[3] =~ /^副詞|^助詞/ && $prenonakami[0] =~ /./) {
               ;
          } elsif ($tsuginonakami[0] =~ /^[あいうえおやゆよつわアイウエオヤユヨツワ]/) {
               ;
          } else {
               print $EOS;
               print $_ . "¥n";
          }
          print $tsugi . "¥n";
          $pre = $tsugi;
          $EOS = "";
     } elsif ($_ =~/^EOS/) {
          $EOS = $_ . "¥n";
     } else {
          print $_ . "¥n";
          $pre = $_;
          $EOS = "";
     }
}
print $EOS;
exit;
```

D　文節から単語を取り出すプログラム

```
#
# Copyright (c) 2002 YOSHIDA Momoko
# tangotoridasi.perl
# usage: jperl -s tangotoridasi.perl -fn="SHUTTEN_FILE_NAME" < filename
#
# 文節から単語を取り出す
#
if ($fn =~/./) {
     $fn=";" .$fn;
}
$wc = 0;
while (<>) {
     chomp;
     if ($_ =~ /¥*/ || $_ =~/EOS/) { # *かEOSが出てきたら
          if ($wc == 0) {
               ; # 何もしない
          } else {
               & pick_word ($wc); # サブルーチンへ行く
          }
          $wc = 0; # カウンタを0に戻す
     } else { # それ以外の場合は
          $line[$wc] = $_; # $line[$wc]に$_を代入して
          $wc++; # カウンタを1進める
     }
}
exit;

sub pick_word {
     my ($wc) = @_;
     my (@a, @b, $i, $j,$flag, $FLAG);
     my ($settou_fukugo, $settou_head, $settou_mrph);
     $FLAG=0;
     $i=0;
     $settou_fukugo = "";
     $settou_head=0;
     $settou_mrph="";
     while($i<$wc) {
       @a = split(/¥t/, $line[$i]);
       if ($a[3] =~ /^動詞-自立/) { # 動詞-自立で始まる複合基本形の計算を開始する
          $fukugo_kihon=""; $hyousou=$a[0]; $kihon = $a[2]; $allmrph=$line[$i];
          $j=$i+1;  $head=0;  $flag=0; $pos = "動詞";
          while($j<$wc) {
            @b = split(/¥t/, $line[$j]);
            if (($b[3] =~ /^動詞-自立/) ||     ($b[3] =~ /^動詞-非自立/ &&
                $b[2] !~/^てる$|^でる$|^ちゃう$|^じゃう$|^ちまう$|^じまう$|^とく$/)) {
                if ($pos=~/名詞/) {# 動詞＋名詞＋動詞の場合など;
                    $pos="動詞";
               }
```

```
        $fukugo_kihon =$fukugo_kihon . $hyousou;
        # $fukugo_kihon=$hyousou; これではだめ
        $hyousou = $b[0]; $kihon   = $b[2];
        $allmrph = $allmrph . " " . $line[$j];
      } elsif (($b[3] =~ /^名詞-一般/) ||
          ($b[3] =~ /^名詞-接尾/ && $b[2] !~ /^[ぁ-ん]$/ && $b[2] !~ /^そう$|^よう$|^もの$/)
                    ||  ($b[3] =~ /^名詞-サ変/)) {
        $pos = "名詞";  $head = $j-$i;
        $fukugo_kihon = $fukugo_kihon . $hyousou;
        # $fukugo_kihon = $hyousou; これではだめ
        $hyousou = $b[0]; $kihon   = $b[2];
        $allmrph = $allmrph . " " . $line[$j];
      } elsif ($b[3] =~ /^形容詞-非自立/) {
        # まだ続く;
        $pos = "形容詞"; $head = $j-$i;
        $fukugo_kihon = $fukugo_kihon . $hyousou;
        # $fukugo_kihon = $hyousou; これではだめ
        $hyousou = $b[0]; $kihon   = $b[2];
        $allmrph = $allmrph . " " . $line[$j];
      } else { # この１個前までで終わり;
        # ここで、プリントする
        #printf ("%s%s;%s;0;%s¥n", $a[2], $b[2], "動詞", $line[$i], $line[$j]);
        $fukugo_kihon = $fukugo_kihon . $kihon;
        printf ("%s%s;%s;%d;%s%s%s¥n",
        $settou_fukugo, $fukugo_kihon, $pos, $settou_head+$head, $settou_mrph, $allmrph, $fn);
        $j=$wc; $flag=1; $FLAG=1;
      }
      $j++;
    }
    if ($flag==0) {
      $fukugo_kihon = $fukugo_kihon . $kihon;
      printf ("%s%s;%s;%d;%s%s%s¥n", # 動詞＋動詞-自立/非自立→品詞は動詞、ヘッドは前項
      $settou_fukugo, $fukugo_kihon, $pos, $settou_head+$head, $settou_mrph, $allmrph, $fn);
      $FLAG=1;
    }
    $i=$wc;
  } elsif ($a[3] =~ /^名詞/) { # 名詞で始まる複合基本形の計算を開始する
    $fukugo_kihon=""; $hyousou=$a[0]; $kihon = $a[2]; $pos = $a[3];
    $allmrph=$line[$i];  $head=0;  $j=$i+1;  $flag=0;
    while($j<$wc) {
      @b = split(/¥t/, $line[$j]);
      if ($b[3] !~ /^名詞|^未知語/ && $b[2] ne "・")  { # 終わり
        if ($b[3] =~ /^動詞-自立/ && $b[2] eq "する") { # ここも含んで終わり
          $fukugo_kihon =$fukugo_kihon . $hyousou . $b[2];
          $allmrph=$allmrph . " " . $line[$j];
          printf ("%s%s;動詞;%d;%s%s%s¥n", # 名詞＋(名詞/未知語)*(←0回以上の繰り返し)
                                           # +動詞-自立→品詞は動詞、ヘッドは動詞の直前
          $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
        } else { # この１個前までで終わり;
          $fukugo_kihon =$fukugo_kihon . $hyousou;
```

```
            printf ("%s%s;名詞;%d;%s%s%s¥n", # 名詞＋(名詞/未知語)*→品詞は名詞、ヘッドは
        #そのまま(下の else にかかる要素はヘッドのカウンタが上がっている。それ以外は上がっていない)
            $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
          }
          $j=$wc; $flag=1; $FLAG=1;
        } elsif (($b[3] =~ /^名詞-非自立/ && $b[2] =~ /^みたい$|^きり$/) ||
                 ($b[3] =~ /^名詞-接尾/ && $b[2] =~ /^さ$|^ゆえ$|^故$|^そう$|^め$/)) {
                  # この１個前までで終わり;
          $fukugo_kihon =$fukugo_kihon . $hyousou;
          printf ("%s%s;名詞;%d;%s%s%s¥n", # 名詞＋(名詞/未知語)*→品詞は名詞、ヘッドは
#そのまま(下の else にかかる要素はヘッドのカウンタが上がっている。それ以外は上がっていない)
          $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
          $j=$wc; $flag=1; $FLAG=1;
        } else {
          if ($b[3] !~/^名詞-接尾/ && $b[3] !~ /^名詞-非自立/ && $b[0] ne "・") {
            $head = $j-$i;
          }
          $fukugo_kihon =$fukugo_kihon . $hyousou;
          $hyousou = $b[0]; $kihon   = $b[2];
          $allmrph = $allmrph . " " . $line[$j];
        }
        $j++;
      }
      if ($flag==0) {
        $fukugo_kihon =$fukugo_kihon . $hyousou;
        printf ("%s%s;名詞;%d;%s%s%s¥n",
        $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
        $FLAG=1;
      }
      $i=$wc;
    } elsif ($a[0] =~/^[a-zA-Z'¥-]+$/) { # アルファベット語に対応;
      printf ("%s;%s;0;%s%s¥n", $a[2], $a[3], $line[$i],$fn);
      $FLAG=1;
    } elsif ($a[3] =~ /^未知語/) {
      $fukugo_kihon=""; $hyousou=$a[0]; $kihon = $a[2]; $pos = $a[3];
      $allmrph=$line[$i];  $head=0;  $j=$i+1;  $flag=0;
      while($j<$wc) {
        @b = split(/¥t/, $line[$j]);
        if ($b[3] !~ /^名詞|^未知語/ && $b[2] ne "・" ) { # この未知語つながりは終わり
          if ($b[3] =~ /^動詞-自立/ && $b[2] eq "する") { #ここも含んで終わり
            $fukugo_kihon =$fukugo_kihon . $hyousou . $b[2];
            $allmrph=$allmrph . " " . $line[$j];
            printf ("%s%s;動詞;%d;%s%s%s¥n",
            $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
          } else { #この１個前までで終わり;
            $fukugo_kihon =$fukugo_kihon . $hyousou;
            printf ("%s%s;名詞;%d;%s%s%s¥n",
            $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
          }
          $j=$wc; $flag=1; $FLAG=1;
```

```
            } elsif (($b[3] =~ /^名詞-非自立/ && $b[2] =~ /^みたい$|^きり$/) ||
                    ($b[3] =~ /^名詞-接尾/ && $b[2] =~ /^さ$|^ゆえ$|^故$|^そう$|^め$/)) {
                     # この１個前までで終わり;
                $fukugo_kihon =$fukugo_kihon . $hyousou;
                printf ("%s%s;名詞;%d;%s%s%s¥n", # 名詞＋(名詞/未知語)*→品詞は名詞、ヘッドは
#そのまま(下の else にかかる要素はヘッドのカウンタが上がっている。それ以外は上がっていない)
                $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
                $j=$wc;    $flag=1;   $FLAG=1;
             } else {
                if ($b[3] !~/^名詞-接尾/ && $b[3] !~ /^名詞-非自立/) {
                  $head = $j-$i;
                }
                $fukugo_kihon =$fukugo_kihon . $hyousou;
                $hyousou = $b[0];    $kihon   = $b[2];
                $allmrph = $allmrph . " " . $line[$j];
            }
            $j++;
         }
        if ($flag==0) {
          $fukugo_kihon =$fukugo_kihon . $hyousou;
          printf ("%s%s;名詞;%d;%s%s%s¥n",
               $settou_fukugo, $fukugo_kihon, $settou_head+$head, $settou_mrph, $allmrph, $fn);
          $FLAG=1;
        }
        $i=$wc;
      } elsif ($a[3] =~ /^形容詞/) {
        printf ("%s%s;形容詞;%d;%s%s%s¥n", $settou_fukugo, $a[2], $settou_head, $settou_mrph,
                 $line[$i], $fn);
        $i=$wc; $FLAG=1;
      } elsif ($a[3] =~ /^副詞/) {
        if ($wc > $i+1) {
           @b = split (/¥t/, $line[($i+1)]);
           if ($b[3] =~ /^動詞-自立/ && $b[2] eq "する") {
             printf ("%s%s%s;動詞;%d;%s%s %s%s¥n",
             $settou_fukugo, $a[0], $b[2], $settou_head, $settou_mrph, $line[$i], $line[($i+1)], $fn);
           } else {
             printf ("%s%s;副詞;%d;%s%s%s¥n",
             $settou_fukugo, $a[0], $settou_head, $settou_mrph, $line[$i], $fn);
           }
        } else {
           printf ("%s%s;副詞;%d;%s%s%s¥n", $settou_fukugo, $a[0], $settou_head,
                    $settou_mrph, $line[$i], $fn);
        }
        $i=$wc; $FLAG=1;
      } elsif ($a[3] =~ /^接続詞/) {
         printf ("%s%s;接続詞;%d;%s%s%s¥n", $settou_fukugo, $a[2], $settou_head,
              $settou_mrph, $line[$i], $fn);
         $i=$wc;      $FLAG=1;
      } elsif ($a[3] =~ /^感動詞/) {
         printf ("%s%s;感動詞;%d;%s%s%s¥n", $settou_fukugo, $a[2], $settou_head,
```

```
            $settou_mrph, $line[$i], $fn);
        $i=$wc;      $FLAG=1;
      } elsif ($a[3] =~ /^連体詞/) {
        printf ("%s%s;連体詞;%d;%s%s%s¥n", $settou_fukugo, $a[2], $settou_head,
            $settou_mrph, $line[$i], $fn);
        $i=$wc;      $FLAG=1;
      } elsif ($a[3] =~ /^その他/) {
        printf ("%s%s;その他;%d;%s%s%s¥n", $settou_fukugo, $a[2], $settou_head,
            $settou_mrph, $line[$i], $fn);
        $i=$wc;      $FLAG=1;
      } elsif ($a[3] =~ /^接頭詞/) {
        $settou_fukugo = $a[2]; $settou_head=1; $settou_mrph=$line[$i] . " ";
      }
      $i++;
    }
    if ($FLAG==0) {
      if ($settou_fukugo =~ /./) {
        chop ($settou_mrph);
        printf ("%s;接頭詞;0;%s%s¥n", $settou_fukugo, $settou_mrph, $fn); # 接頭詞の連続は独立させる
      } else {
        $i=0;  $settou_fukugo = "";  $settou_head=0; $settou_mrph="";
        while ($i<$wc) {
          @a = split(/¥t/, $line[$i]);
         if ($a[3] =~ /^動詞-非自立/) { # 文節の中に動詞-非自立しか自立成分がない場合，それをとりだす
            printf ("%s%s;動詞;%d;%s%s%s¥n", $settou_fukugo, $a[2], $settou_head,
                $settou_mrph, $line[$i], $fn);
            $i = $wc; $FLAG = 1;
         }
         $i++;
       }
       if ($FLAG == 0) {
         printf ("-- NO_WORDS_PICKED_UP --%s¥n", $line[0]);
       }
      }
    }
}
```


目次ページに戻る。