品詞と平仮名のN-gramによる平仮名列の誤り検出

執筆者：河野 靖明

指導教官：新納 浩幸

平成１１年３月３日

目 次

Chapter 1

序論

1.1 はじめに

英文のスペルチェックの最も簡易な実装は、辞書にない単語をタイプミスであると指摘することである。このレベルの書き誤りに対応する日本語文のスペルチェックシステムは、有益であることは明らかだが、広く利用されているものではない。なぜなら、日本語文章の場合、単語切りを行うためには形態素解析が必要であるし、誤り箇所から誤った単語切りも生じ、辞書に単語が存在する、しなしで単純にスペルミスを検出することは困難だからである。一方、N-gram を利用して文章中の誤った単語の検出、修正が可能であることが知られている。これは N 個の単語列とその頻度などの統計的データを表の形であらかじめ用意しておき、存在しない単語列や出現回数が少ない単語列は誤りの可能性があると指摘する方法である。ただしスペルチェックに利用できる大規模な N-gram は N が 3 の場合でさえ、構築するのが困難である。また日本語の場合、単語 N-gram を利用するためには、単語 N-gram の種類数は膨大であり、その検索コストが高い点などから、スペルチェックとして手軽に利用できる方法とは考えられない。ただし、対象を日本語の平仮名列中に生じる書き誤りに限定すれば、平仮名 N-gram を用いる方法があるが、削除誤りに対しての正解率が低い。そこで、平仮名文字と前後の品詞で N-gram を構築することで、上記の問題を解決しつつ、誤り検出が可能である。N-gram を利用する場合、一般に N を大きくすれ

ば誤り検出の精度は増す。しかし現実的には大きなNに対しては、コーパスのスパース性から過度に誤りを検出してしまう結果となる。本論文では、$N = 4$として平仮名と品詞からなる$N - gram$を利用した場合の平仮名文字の挿入、削除、置換、転置による誤り検出の精度を調べた。

1.2 本論文の構成

本論文は最初に、形態素解析による平仮名列の誤り検出の説明(第2章)。平仮名N-gramによる平仮名列の誤り検出の説明(第3章)。その後品詞の利用について説明(第4章)、品詞と平仮名からなるN-gramによる平仮名列の誤り検出実験(第5章)及び考察(第6章)、結論(第7章)へと進む。

また、巻末にはプログラムのソースリストを添付した。

Chapter 2

形態素解析による誤り検出

2.1 概要

校正支援システムとは、ワープロソフトの機能として標準に備わっているスペルチェッカーのことをいう。

しかし、日本語の文書校正システムは英文スペルチェッカーと比較すると動作的には似てるように見えるが、細かく見ていくとかなりの違いがある。これは対象とする言語が全く違うからである。

英語に比べて日本語は、文字の意味の単語を表すものにも幾つもの表記が存在する。また、日本語は単語を分かち書きしない膠着語であり、単語を認識することが困難であるという特徴を持っている。

このような日本語の言語特徴をふまえて、校正支援システムが検出の対象とする現象を、その発生原因により分類してみると以下に示す4つに分類することができる。

1. 入力ミス

   タイプミス、変換誤りなど入力の操作ミスに起因する誤りで、書き手が意図していない誤り

2. 知識不足

「汚名挽回」(正しくは「汚名返上」)のように書き手の勘違いや関連する類似表現の混乱に起因する誤り

3. 表記の揺れ

「表す」と「表わす」のような同一語に対する2つ以上の異なる表記が1つの文章の中で揺れているもの。「受け付け」と「受付」のように慣用的に使い分けがはっきりしているものもあるが、誤りとは言いきれない場合も多くある。

4. 推敲

誤りではないが、曖昧な文章や複雑な文章を検出し、書き手に文章の推敲を促すもので、単純なものとしては一文の長さや句読点の間隔に制限をかけるものなどもこの範疇に入ると言える。

2.2 まとめ

2.2.1 文節の文法表現

「正しいパターンを定義し、それ以外を見つける」という方針に従えば、日本語文節において自立語の品詞に対する付属語の正しいパターンを定義し、正しい文節には正しいと、誤った文節に対しては誤りであると判定できるように表現できるようにしなければならない。

形態素解析や構文解析においては、誤りのある文節を誤りと判定することについて、やや寛容であるが、正しい文節に対して正しいと判定することは、このような文法表現では必然と言える。

ここで求められているのは、文節の中という狭い範囲での文法表現であるといえるので、形態素解析における文法表現を参考にして、どのような表現が、日本語スペルチェッカーに適しているかを考えてみると。

形態素解析の手法としては、最近では最小コスト法やn-gram、接続法などがある。最小コスト法はn個の品詞(または単語)の連接度合いを、人間が定義したコストという数値を用いる手法である。n-gramは連接度合いに1次言語資料から得た確率を用いた手法である。接続表は2つの品詞間の接続関係を示した表である。これらの手法において文法は、すべて2項またはn項の連接関係で表現されているといえる。

これらの連接関係では正しいものを記述するには十分だが、ありそうもないパターンも表現してしまうことがある。以下では2項関係で文法を表現した場合を例として考えてみる。

例えば、「彼女との会話では、彼女のと異なる意見を述べた」という文を考えてみる。このうち「彼女との」と「彼女のと」といった2つの文節を取り出して、文法的に解釈すると次のようになる。

彼女(名詞)/ と(助詞と)/ の(助詞の)

彼女(名詞)/ の(助詞の)/ と(助詞と)

この時、文法が2項間の接続関係で表現されているとすると、以下に示した品詞の組は、すべて接続できることになる。

(名詞) + (助詞と)

(名詞) + (助詞の)

(助詞と) + (助詞の)

(助詞の) + (助詞と)

このような連接が許される文法では、もし「彼女のとのと...」という日本語としては意味の通らない文節を解釈した場合、以下のような解釈を行い、正しい文節ということになってしまう。

彼女(名詞)/ の(助詞の)/ と(助詞と) / の(助詞の)/...

n項の接続を用いて文節を表現した場合でも、このように意図しない文法を許してしまう可能性は完全になくならない。

2.2.2 オートマトンを用いた文法表現

他の文法表現としては、図 2.1 に示すようなオートマトンを用いたものが考えられる。図 2.1 のように循環した遷移があると、接続の場合の問題と同様に有り得ない文節も表現してしまう。

循環しないようにオートマトンを作るのには有り得る文節のパターンの接頭辞を併合して樹状にオートマトンを作る方法がある。(図 2.1)

この方法ならば、定義した文節以外は認めないようにできる。

ここでは、遷移を矢印で、状態を丸で示している。また、文節の末端を示すために二重丸で終了状態を示している。このように終了状態を区別することで不完全な文節を許さないようにすることができる。

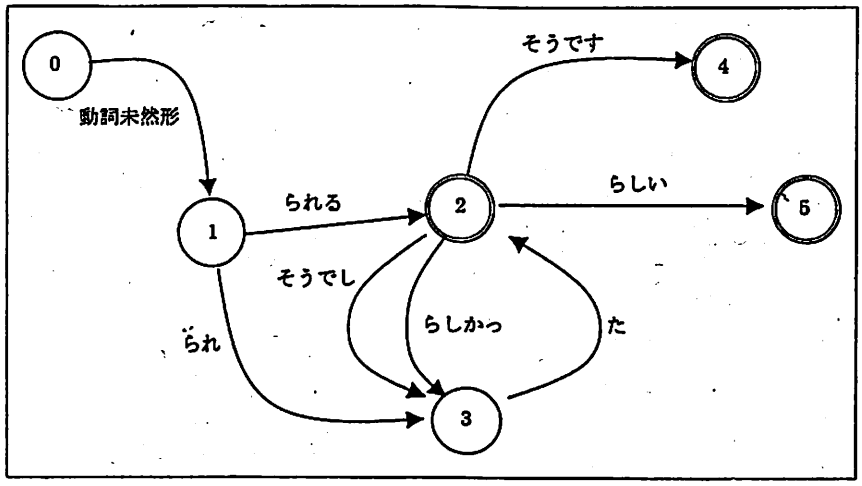

図２．１：オートマトン（1）

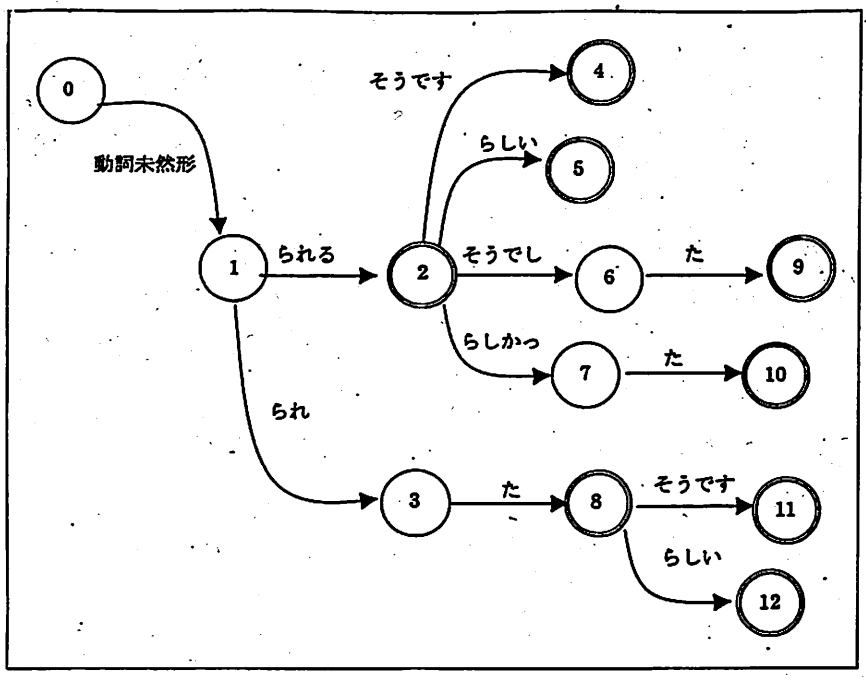

図２．２：オートマトン（2）

2.3 問題

このように構築した日本語スペルチェッカーの能力は、正しいと判断されたおよそ10万文字の文書データの平仮名列に対して機械的に脱字または文字ダブリを作って評価した結果、脱字に関しては5割程度、文字ダブリに関しては7割程度見つけることができた。

脱字については、機械的に作成したため「言ったのでしょう」が「言ったでしょう」となったものも含まれているので、少し低い値になっている。

問題点は、口語表現など文節のパターンを定義していない場合には、過剰な検出も多くなる欠点があることである。

Chapter 3

平仮名 N-gram による平仮名列の誤り検出

ここでは、平仮名 N-gram による平仮名列の誤り検出に用いた手法について説明を行う。

3.1 N-gram モデル

n 単語からなる単語列 $\omega_1...\omega_n$ が与えられたとき、単語列 $\omega_1...\omega_n$ の生成確率は次式により与えられる。

$$P(\omega_1...\omega_n) = P(\omega_1)P(\omega_2|\omega_1)...P(\omega_n|\omega_1...\omega_{n-1})$$

$$= \prod_{i=1}^{n} P(\omega_i|\omega_1...\omega_{i-1})$$

しかし、さまざまな単語の組み合わせに対し、条件付き確率 $P(\omega_i|\omega_1...\omega_{i-1})$ を求めるのは事実上不可能である。単語数 L の場合、$P(\omega_i|\omega_1...\omega_{i-1})$ を完全に指定するためには、L^i 個の値を求めなければならない。現実的なモデルとするためには、単語の履歴 $\omega_1...\omega_{i-1}$ を同値類に分割し、モデルのパラメータ数を削減する必要がある。

一般に、ある時点で生起する事象の確率が、その直前の N 個の時点で生起した事象だけの影響を受けるとき、これを N 重マルコフ過程というが、単語の生起を $N-1$ 重マル

コフ過程で近似したモデルを $N-gram$ モデルと呼んでいる。すなわち、$N-gram$ モデルでは、ある時点での単語の生起は直前の $N-1$ 単語のみに依存すると考えている。したがって、$N-gram$ モデルにおいては、

$$P(\omega_n|\omega_1...\omega_{n-1}) = P(\omega_n|\omega_{n-N+1}...\omega_{n-1})$$

となる。

いま、単語列 $\omega_1...\omega_n$ がコーパス中に出現する回数を $C(\omega_1...\omega_n)$ で表すことにする。$N-gram$ の確率は、コーパス中に出現する単語の N 個組と $(N-1)$ 個組の出現回数から、次のように推定することができる。

$$P(\omega_n|\omega_{n-N+1}...\omega_{n-1}) = \frac{C(\omega_{n-N+1}...\omega_n)}{C(\omega_{n-N+1}...\omega_{n-1})}$$

N の値が大きいほど、学習データから信頼性の高い $N-gram$ の値を推定するのが難しくなる。本研究では $N=4$ を用いる。

3.2 平仮名 N-gram

平仮名 $N-gram$ とは N文字長の平仮名列の頻度表のことである。

3.2.1 平仮名 N-gram の構築

平仮名 $N-gram$ の構築は容易である。コーパスを 1 本の長い文字列と考え、平仮名以外の文字を K という文字に変換しておく。i 番目の位置の文字から $i-N+1$ 番目の位置の文字までからなる長さ N の文字列を考え、その文字列が以下のいずれかのパターンになっている場合に、その文字列を取り出す。

1. $H\ H\ ...\ H$ (N 文字が全て平仮名)

   (例) あいさつ

2. $K\ H\ H\ ...\ H$ (先頭文字がひらがな以外で残りの $N-1$ 文字が平仮名)

   (例) 考えられる

3. $H\ H\ ...\ H\ K$ (先頭から $N-1$ 文字が平仮名で末尾文字が平仮名以外)

   (例) こういう問題

ここで、最初のケースのすべての文字が平仮名である平仮名列だけを取り出しても良いが、ここでは 1 文字の欠落による誤りからの修正を行うために、残りの 2 つのケースの文字列も抽出している。

上記の操作を $i=0$ から順にコーパスの最後の位置に至るまで繰り返し、取り出した文字列の頻度表を作成することで平仮名 $N-gram$ が構築できる。

上の 3 つのパターンの例から $N-gram(N=4)$ を取り出してみると

1. (例) これらのうち →

   これらの

   れらのう

   らのうち

2. (例) 増加させることができる →

   させるこ

   せること

   ることが

   ことがで

   とができ

   ができる

3. (例) 使われやすいという単語 →

   われやす

   れやすい

   やすいと

   すいとい

   いという

このようにして、文章から平仮名 $N-gram(N=4)$ を作成した例を表 3.1 に示す。

表 3.1 : 平仮名 $4-gram$

いしょう	1
えたかっ	3
かげりが	18
くらかず	1
げていけ	3
ごこちの	7
しょせん	9
たえるの	8
たれとま	1
っさにそ	2
とはきわ	3
ならぬも	5
のがない	9
まったん	3
もじった	3
やってく	33
りそうと	5
をかきた	1
んだのち	2

3.2.2 平仮名 N-gram による誤り検出

先ほど構築した平仮名 $N-gram$ を頻度の昇順に並べ、同時に総頻度を測る。頻度の少ないものから順に頻度の累計をとってゆく。ある平仮名列 h に書き誤りが存在するかどうかの判定は以下に従う。まず文字列 KhK から $N-gram$ を取り出し、それぞれの文字列の頻度を平仮名 $N-gram$ から調べる。それらの頻度の最小値 (この値を平仮名列 h に対

する $N-gram$ 最小頻度と呼ぶことにする) が先の閾値以下である場合に、平仮名列 h に書き誤りが存在すると判定する。

(例) 誤りを含む文 「自然なつながりがもつようにする。」 で誤り検出の手順と検出例を示すと

1. 文から平仮名列 (4 文字以上のもの) を取り出す

   自然なつながりがもつようにする。

   →なつながりがもつようにする

2. 4 文字列を取り出す

   なつながりがもつようにする

   → なつなが

   つながり

   ながりが

   がりがも

   りがもつ

   がもつよ

   もつよう

   つように

   ようにす

   うにする

3. 頻度表を参照し、頻度を求める

求めた結果を表3.2に示す

表3.2 平仮名4グラム検索結果

要素	頻度
なつなが	4
つながり	97
ながりが	15
がりがも	0
りがもつ	0
がもつよ	0
もつよう	2
つように	29
ようにす	182
うにする	169

このようにして頻度を求めたとき、閾値以下の場合誤りと判定する。ちなみに例の結果では、「がりがも、りがもつ、がもつよ」の頻度が0であるので明らかに誤りがあることがわかる。

3.2.3 平仮名 N-gram による誤りのパターン

平仮名列の誤りは以下の 4 つのパターンのいずれかであると仮定する。

- 削除　平仮名列中のある位置の平仮名 1 文字が欠落した誤り

  (例) するかどうか → するどうか

  例文のように、「するかどうか」の「か」が欠落し「するどうか」となっているような平仮名列のことをいう。

- 挿入　平仮名列中のある位置に、ある平仮名 1 文字が挿入された誤り

  (例) するかどうか → するかがどうか

  例文のように、「するかどうか」に「が」という文字が挿入され「するかがどうか」となっているような平仮名列のことをいう。

- 置換　平仮名列中のある位置の平仮名 1 文字が、ある平仮名と入れ替わった誤り

  (例) するかどうか → するかとうか

  例文のように、「するかどうか」の「ど」が「と」に入れ替わり「するかとうか」となっているような平仮名列のことをいう。

- 転置　平仮名列中のあるとなりあう 2 つの平仮名文字が交換された誤り

  (例) するかどうか → するかうどか

例文のように、「するかどうか」の「ど」と「う」が交換されて「するかうどか」となっているような平仮名列のことをいう。

3.3 検索

品詞を利用するのに形態素解析を用いるため平仮名 $N-gram$ よりも検索時間が長くなるので検索プログラムを工夫する必要がある。

3.3.1 2分探索

2分探索は、データがソートされて小さい順 (または大きい順) に並んでいるときに有効な探索法である。

たとえば、次のようなデータからデータの 50 を 2 分探索する場合を考えてみる。

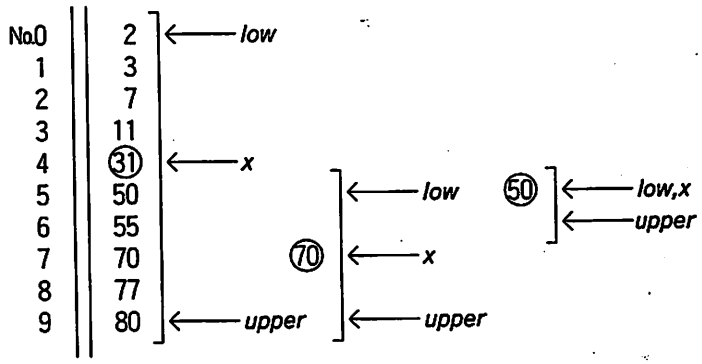

図３．１：２分探索

探索範囲の下限を low, 上限を $upper$ とし、$x=(low+upper)/2$ の位置のデータとキー (探すデータ) を比較する。もし、キーの方が大きければ、キーの位置は x より上にあるはずなので、lowを $x+1$ にする。逆に、キーの方が小さければ、キーの位置は x より下にあるはずなので $upper$を $x-1$ にする。これを、$low <= upper$ の間繰り返す。

実際に 2 分探索を行ってみると、

1. 最初 $low=0$ 、$upper=9$ なので、$x=(0+9)/2=4$ となり、4 番目のデータの 31

と探すデータ 50 を比較し、50 の方が大きいので $low = 4 + 1 = 5$ とする。

2. $low = 5, upper = 9$ のとき、 $x = (5 + 9)/2 = 7$ となり、7 番目のデータ 70 と 50 を比較し、50 の方が小さいので $upper = 7 - 1 = 6$ とする。

3. $low = 5, upper = 6$ のとき、$x = (5 + 6)/2 = 5$ となり、5 番目のデータ 50 と 50 を比較し、これでデータが探された。

つまり、2 分探索では、探索するデータ範囲を半分に分け、キーがどちらの半分にあるかを調べることを繰り返し、調べる範囲をキーに向かってだんだんに絞っていく。

もし、キーが見つからなければ、 lowと $high$ が逆転して $low > high$ となったとき、に終了する。

3.3.2 ハッシュ

たとえば、1 クラスの生徒のデータのように学籍番号で管理できるものは、この番号を添字 (レコード番号) にして配列 (ファイル) に格納しておけば、添字を元に即座にデータを参照することができる。

ところが、番号で管理できないデータは、一般に名前などの非数値データをキーにしてサーチしなければデータ参照できない。

しかし、ハッシュ (*hash*) という技法を使えば名前などをキーにして即座にデータ参照が行える。

ハッシング (*hashing*) は、キーの取り得る範囲の集合を、ある限られた数値範囲 (レコード番号や配列の添字番号などに対応する) に写像する方法である。この写像を行う変換関数をハッシュ関数という。

ハッシュ関数として次のようなものを考える。

キーは英字の大文字からなる名前とし、$A_1, A_2, A_3, ..., A_n$ の N 文字からなるものとする。キーの長さが n 文字とすると、取り得る名前の組み合わせは 26^{n+1} 個存在するにな

り、きわめて大きな数値範囲になってしまう。そこで、キーの先頭 A_1, 中間 $A_{n/2}$, 終りから 2 番目 A_{n-1} の 3 文字を用いて

$$hash(A_1A_2...A_n) = (A_1 + A_{n/2} * 26 + A_{n-1} * 26^2) mod\ 1000$$

という関数を設定する。

たとえば、キーが $'SUZUKI'$なら、

$$\begin{aligned} hash(SUZUKI) &= (ASC('S') - ASC('A')) \\ &\quad +(ASC('Z') - ASC('A')) * 26 \\ &\quad +(ASC('K') - ASC('A')) * 26^2) mod\ 1000 \\ &= (18 + 650 + 6700) mod\ 1000 \\ &= 428 \end{aligned}$$

となるから、428 という数値を配列なら添字、ファイルならレコード番号とみなして、キーの $SUZUKI$ を対応づければよい。

3.4 問題

平仮名 $N-gram$ だけで検索すると削除による誤りに対する正解率が悪い。

これは 1 文字削除によって、生成される平仮名列が妥当な平仮名列となる場合が多いからである。

(例) いずれかであると仮定する → いずれかである仮定する

これから 4 文字長の平仮名列を取り出すと

いずれか

ずれかで

れかであ

かである

となる。この取り出した4つの平仮名列はすべてコーパスに存在するため、平仮名列'いずれかである仮定する'は誤りと判定されない。

この種の誤りを対象の平仮名列だけから検出することは難しく、他の情報を利用する必要がある。有効なアプローチとして、品詞と平仮名からなる $N-gram$ を利用することが考えられる。こうすることで、対象となる平仮名列を含む前後の単語を含めた単語列の品詞のパターンからその平仮名列のと前後の品詞のパターンに誤りがある可能性が検出でき、そこから誤りの検出ができる。

(例) いずれかである仮定する

これを記号化すると

文いずれかである c する

となり、'である c'という文字列はコーパスに存在しないか頻度が少ないのはずなので誤りとして検出できる。

Chapter 4

品詞の利用

ここでは、平仮名 $N-gram$ での削除の正解率が低い点を改善するため品詞と平仮名からなる $N-gram$ の作り方を説明する。

4.1 品詞 → 記号

平仮名 $N-gram$ と品詞を組み合わせるために、文章を平仮名と品詞を記号化したものだけの文にする。まず形態素解析を用いて文章を品詞と平仮名の文する。それから品詞の部分を記号化する。

変換表を表 4.1 ～ 表 4.4 示す。

表 4.1:品詞 →記号の変換表 (その 1)

0	形容詞 且つ イ形容詞アウオ段
1	形容詞 且つ イ形容詞イ段
2	形容詞 且つ ナ形容詞
3	形容詞 且つ イ形容詞イ段特殊
4	形容詞 且つ ナノ形容詞
5	形容詞 且つ タル形容詞
6	形容詞 且つ ナ形容詞特殊
7	助動詞 且つ 無活用型
8	助動詞 且つ 助動詞く型
9	助動詞 且つ ナ形容詞
A	助動詞 且つ イ形容詞イ段
B	助動詞 且つ 判定詞
C	助動詞 且つ 助動詞だろう型
D	助動詞 且つ 助動詞そうだ型
E	助動詞 且つ ナノ形容詞
F	助動詞 且つ 助動詞ぬ型
G	動詞 且つ カ変動詞
H	動詞 且つ カ変動詞来
I	動詞 且つ サ変動詞
J	動詞 且つ ザ変動詞

表 4.2:品詞 →記号の変換表 (その 2)

K	動詞 且つ 動詞性接尾辞ます型
L	動詞 且つ ナ形容詞特殊
M	動詞 且つ 母音動詞
N	動詞 且つ 子音動詞カ行
O	動詞 且つ 子音動詞ガ行
P	動詞 且つ 子音動詞カ行促音便形
Q	動詞 且つ 子音動詞サ行
R	動詞 且つ 子音動詞タ行
S	動詞 且つ 子音動詞ナ行
T	動詞 且つ 子音動詞バ行
U	動詞 且つ 子音動詞マ行
V	動詞 且つ 子音動詞ラ行
W	動詞 且つ 子音動詞ラ行イ形
X	動詞 且つ 子音動詞ワ行
Y	動詞 且つ 子音動詞ワ行文語音便形
Z	指示詞 且つ 名詞形態指示詞
a	指示詞 且つ 連体詞形態指示詞
b	指示詞 且つ 副詞形態指示詞
c	名詞 且つ 普通名詞
d	名詞 且つ 副詞的名詞

表 4.3:品詞 →記号の変換表 (その 3)

e	名詞 且つ 形式名詞
f	名詞 且つ 固有名詞
g	名詞 且つ 地名
h	名詞 且つ 人名
i	名詞 且つ 組織名
j	名詞 且つ サ変名詞
k	名詞 且つ 数詞
l	名詞 且つ 時相名詞
m	助詞 且つ 終助詞
n	助詞 且つ 接続助詞
o	助詞 且つ 副助詞
p	助詞 且つ 格助詞
q	接頭辞 且つ 名詞接頭辞
r	接頭辞 且つ 動詞接頭辞
s	接頭辞 且つ イ形容詞接頭辞
t	接頭辞 且つ ナ形容詞接頭辞
u	接尾辞 且つ 名詞性名詞助数辞
v	接尾辞 且つ 名詞性特殊接尾辞
w	接尾辞 且つ 名詞性名詞接尾辞
x	接尾辞 且つ 名詞性述語接尾辞

表 4.4:品詞 →記号の変換表 (その 4)

y	接尾辞 且つ 形容詞性名詞接尾辞
z	接尾辞 且つ 形容詞性述語接尾辞
α	接尾辞 且つ 動詞性接尾辞
β	特殊 且つ 句点
γ	特殊 且つ 読点
δ	特殊 且つ 括弧始
ε	特殊 且つ 括弧終
ζ	特殊 且つ 記号
η	特殊 且つ 空白
θ	連体詞
ι	副詞
κ	判定詞
λ	接続詞
μ	感動詞
ν	連語
ξ	未定義語 且つ カタカナ
o	未定義語 且つ アルファベット
π	未定義語 且つ その他

4.2 平仮名と品詞の組み合わせ

品詞と平仮名を組み合わせる条件と方法を示す。

品詞と平仮名を組み合わせるには以下の 2 つの条件を満たさなければならない。

- $K\ H\ ...\ H\ K$ (H:平仮名、K:平仮名以外の文字)
- 平仮名が 4 文字以上続くとき (例)　「自然なつながりをもつようにする。」

  ここでは「自然な」 がK で

  「つながりをもつようにする」 がH である。

このような形の場合に $N-gram$ を作成する。

例文「自然なつながりをもつようにする。」を用いて品詞と平仮名を組み合わせる方法を示す。

(1) 文章に形態素解析をかけ、平仮名以外の文字の品詞を求める。

(例) 自然なつながりをもつようにする。

この結果を示すと

自然な しぜんな 自然だ 形容詞 3 * 0 ナノ形容詞 22 ダ列基本連体形 4

つながり つながり つながり 名詞 6 普通名詞 1 * 0 * 0

を を を 助詞 9 格助詞 1 * 0 * 0

もつ もつ もつ 動詞 2 * 0 子音動詞タ行 6 基本形 2

ように ように ようだ 助動詞 5 * 0 ナ形容詞 21 ダ列基本連用形 7

する する する 接尾辞 14 動詞性接尾辞 7 サ変動詞 16 基本形 2

。 。 。 特殊 1 句点 1 * 0 * 0

EOS

となる。

(2) 文頭に「文」という文字を入れ、対応表を用いて平仮名と品詞を記号化したものだけの文にする。

(例) 自然なつながりをもつようにする。

→ 文4つながりをもつようにするβ

ここでは、形態素解析の結果を変換表に対応させると、「文頭」が「文」、「自然な」は形容詞のナノ形容詞なので「4」、「。」は特殊の句点なので「β」になる。

(3) K H ... H K の形のものを取り出す。(H が4つ以上続くとき)

(例) 文4つながりをもつようにするβ

→ 4つながりをもつようにするβ

(4) ここから $4-gram$ を取り出す。

(例) 4つながりをもつようにするβ

→ 4つなが
つながり
ながりを
がりをも
りをもつ
をもつよ
もつよう
つように
ようにす
うにする
にする β

このように取り出した文字列の頻度表を作成することで品詞と平仮名からなる $N-gram$ が構築できる。

例として品詞と平仮名からなる $N-gram(N=4)$ の一部を表4.5に示す。

表 4.5:品詞と平仮名の $N-gram$

Mのもか	2
cにいた	241
あるとき	97
しいがγ	174
なければ	10129
やるとγ	30
ょうがγ	75
らいをj	14
らわずか	135
るからま	6

Chapter 5

実験

5.1 実験の設定

日本経済新聞 $CD-ROM$ の '90 年度版から 1 年分の新聞記事から品詞と平仮名からなる $4-gram$ を作成し、それらを利用した場合の、誤り検出の効果を調べた。

まず、テストデータとして、先の新聞記事とは別の新聞記事を用意した。それらを形態素解析してから品詞を記号化し、品詞を記号化したものと平仮名だけの文に変換した。そこから $K\ H\ ...\ H\ K$ (H:平仮名、K:品詞を記号化した文字、H の長さ 4 以上) の文字列を 2651 種類取り出した。

テストデータの一部を示す

1. その日、狩りで一匹の獲物もなかったので、日記にただ一言「何もなし」▲一七八九年七月十四日の夜、早寝した国王は側近に揺り起こされた。

2. 「では、暴動じゃないか」「いいえ、陛下、これは革命でございます」。

3. ワクチンでエイズ感染を防ぐには、ＨＩＶに感染した細胞を見つけて異物として排除するＴリンパ球と、ＨＩＶそのものを攻撃する抗体を増やさなければならない。

4. 特捜部は職務権限などとの関連で収賄での立件が困難なケースについては「総選挙と

参院選に関し、国と請負等の契約をしている者は寄付をしてはならない」と規定した公職選挙法（一九九条）違反での摘発も検討している模様だ。

5. 来年度当初予算編成の越年をめぐる連立与党内のきしみの表面化、内閣支持率の低下など細川政権への逆風が続く中で注目された細川護熙首相の年頭会見だったが、訴えの中心は国民に「痛み」を求めたことだった。

6. 特別査察は、北朝鮮とＩＡＥＡが話し合う問題だとゲタをあずけてしまっており、核疑惑自体は何も解決していない。

このテストデータに対して品詞と平仮名からなる $4-gram$ を利用して、以下の実験を行った。

実験 1 各平仮名列に誤りがあるかどうかを判定する。

実験 2 各平仮名列の適当な位置の 1 文字を取り除くことで作成した平仮名列に誤りがあるかどうかを判定する。

また、テストデータを以下のように変えた文で実験を行った

(例) その日、狩りで一匹の獲物なかったので、日記にただ一言「何もなし」▲一七八九年七月十四日の夜、早寝した国王は側近に揺り起こされた。 なかったので もなかったので

実験 3 各平仮名列の適当な平仮名 1 文字を挿入することで作成した平仮名列に誤りがあるかどうかを判定する。

また、テストデータを以下のように変えた文で実験を行った

(例) その日、狩りで一匹の獲物ものなかったので、日記にただ一言「何もなし」▲一七八九年七月十四日の夜、早寝した国王は側近に揺り起こされた。 ものなかったので もなかったので

実験 4 各平仮名列の適当な位置の 1 文字を適当な平仮名 1 文字に変更することで作成した平仮名列に誤りがあるかどうかを判定する。

また、テストデータを以下のように変えた文で実験を行った

(例) その日、狩りで一匹の獲物のなかったので、日記にただ一言「何もなし」▲一七八九年七月十四日の夜、早寝した国王は側近に揺り起こされた。 のなかったので もなかったので

実験 5 各平仮名列の適当な隣り合う 2 文字を入れ換えることで作成した平仮名列に誤りがあるかどうかを判定する。

また、テストデータを以下のように変えた文で実験を行った

(例) その日、狩りで一匹の獲物なもかったので、日記にただ一言「何もなし」▲一七八九年七月十四日の夜、早寝した国王は側近に揺り起こされた。 なもかったので もなかったので

実験 6 閾値を 0、1、2、5、10 にとってみてそれぞれの正解率を求める。

また比較として平仮名 $N-gram(N=4)$ の実験も行う。

5.2 実験結果

実験 1 の結果を表 5.1、図 5.1 に示す。平仮名 $4-gram$ の検索結果も示す。

この結果を見ると平仮名 $N-gram(N=4)$ の方が品詞と平仮名からなる $N-gram(N=4)$ よりも良い結果となった。これは品詞と平仮名からなる $N-gram(N=4)$ のスパース性が原因である。ゆえに、品詞と平仮名からなる $N-gram(N=4)$ のコーパスの量を増やせば正解率が上がると考えられる。

表 5.1:実験 1 の結果

N-gram	検出数	正解率
平仮名 4-gram	273(2651)	89.7%
品詞と平仮名の 4-gram	494(2651)	81.4%

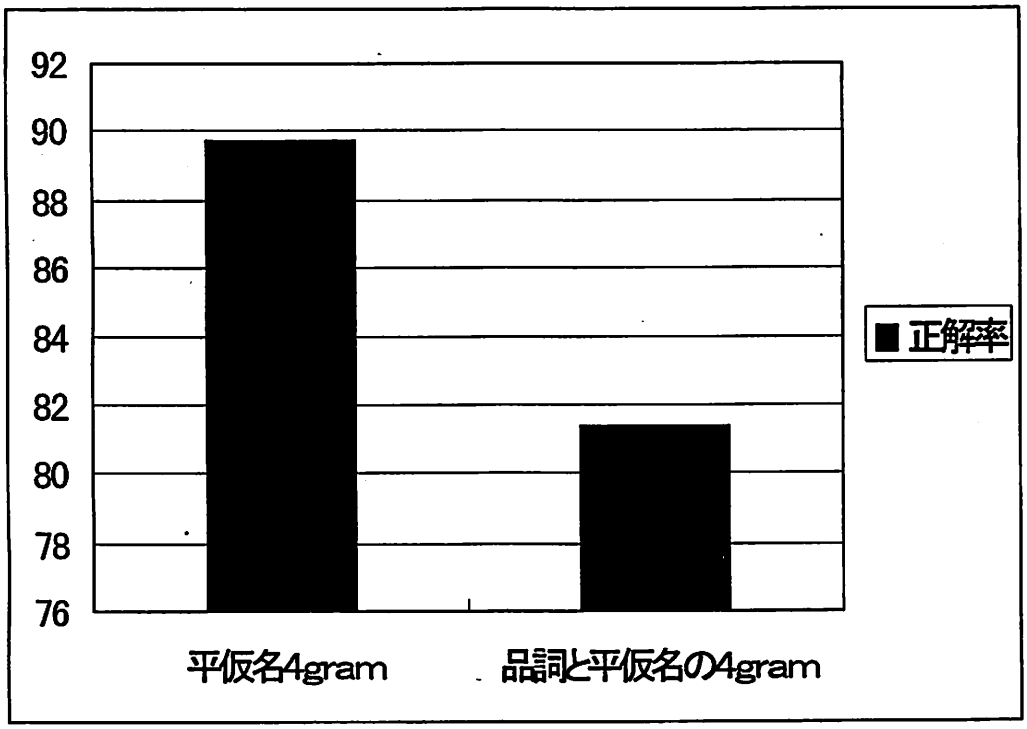

図５．１：実験１の結果

実験 2、3、4、5 の結果を表 5.2、図 5.2 に示す。この結果を比較すると全ての項目で平仮名 $N-gram(N=4)$ の結果を上回っていることがいえる。特に削除の正解率を比較するとかなり良くなっていることがいえる。

表 5.2:実験 2 ～ 5 の結果

N-gram	実験 2 (削除)	実験 3 (挿入)	実験 4 (置換)	実験 5 (転置)	平均
平仮名 4-gram	69.3%	94.9%	94.2%	96.5%	88.7%
品詞と平仮名の 4-gram	76.7% 2033(2651)	97.4% 2582(2651)	95.4% 2529(2651)	97.3% 2579(2651)	91.7% 9723(10604)

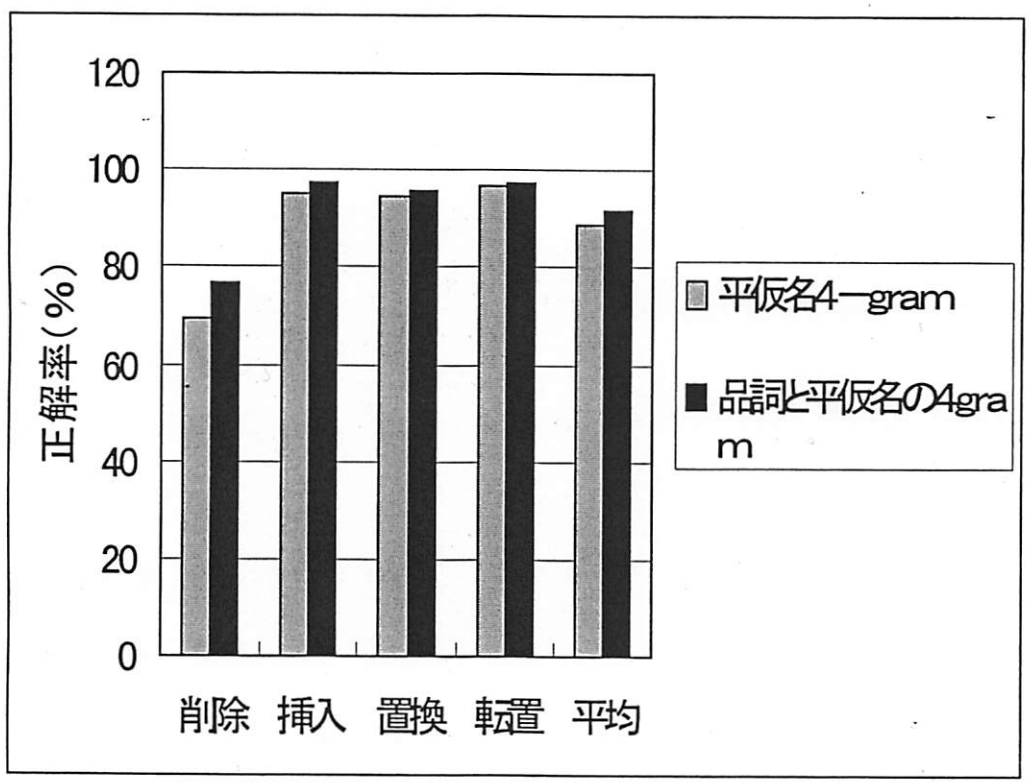

図５．２：実験２～５の結果

実験6の結果を表5.3、図5.3に示す。閾値の実験をした結果、閾値が2のときがもっとも良い結果を得ることができた。この結果より本研究の実験ではすべて閾値=2としている。

表5.3:実験6の結果

閾値	実験1	実験2 (削除)	実験3 (挿入)	実験4 (置換)	実験5 (転置)	平均
0	83.9% 2223(2651)	74.4% 1972(2651)	97.1% 2575(2651)	95.0% 2516(2651)	96.9% 2568(2651)	89.4% 11854(13255)
1	82.6% 2189(2651)	75.7% 2008(2651)	97.4% 2581(2651)	95.0% 2516(2651)	96.9% 2568(2651)	89.6% 11879(13255)
2	81.4% 2157(2651)	76.7% 2033(2651)	97.4% 2582(2651)	95.4% 2529(2651)	97.3% 2579(2651)	89.6% 11880(13255)
5	79.1% 2098(2651)	77.9% 2064(2651)	97.6% 2588(2651)	95.5% 2533(2651)	97.4% 2583(2651)	89.5% 11866(13255)
10	76.4% 2026(2651)	78.5% 2082(2651)	97.7% 2590(2651)	95.8% 2539(2651)	97.4% 2583(2651)	89.2% 11820(13255)

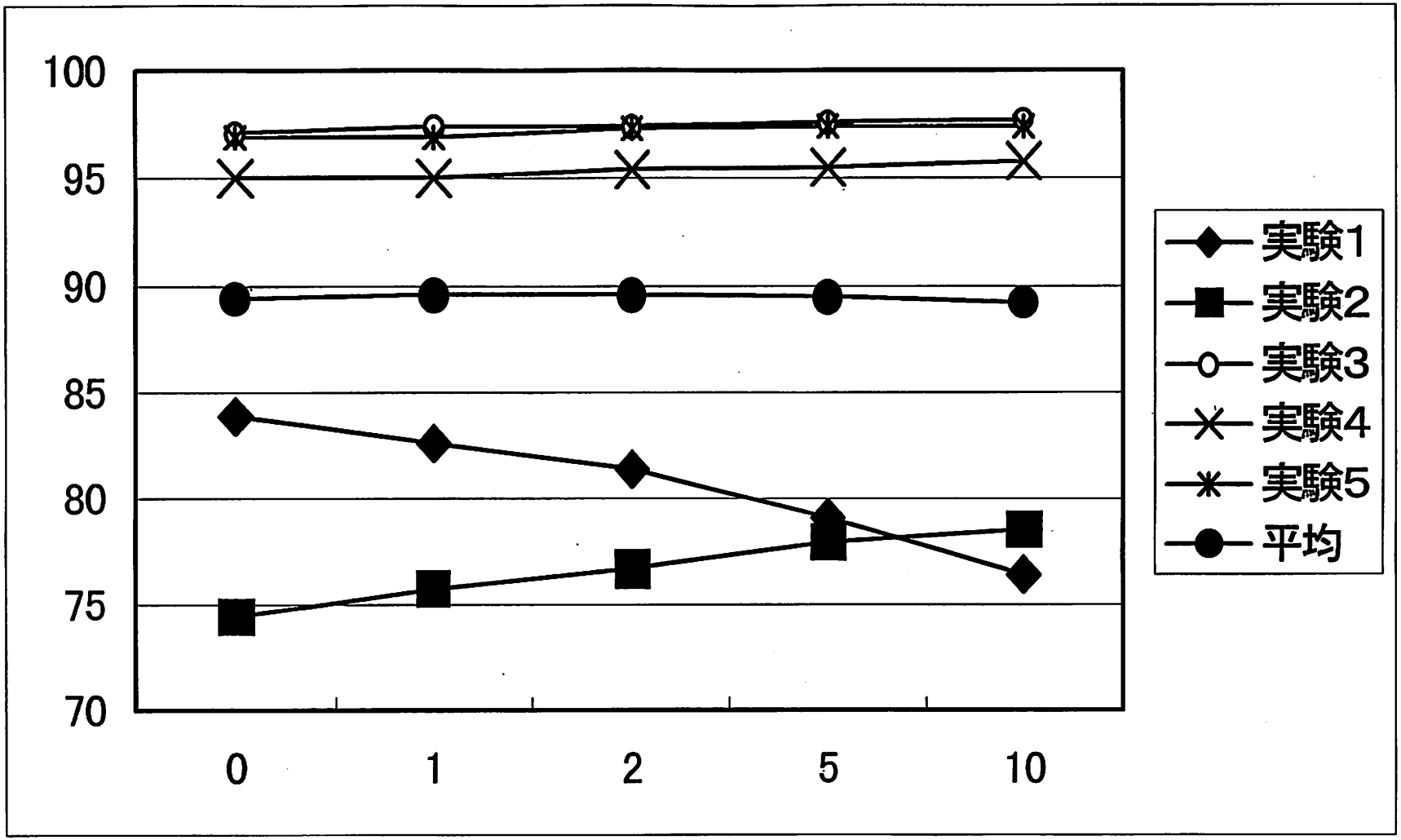

図５.３ 実験６の結果

Chapter 6

考察

実験の結果、本手法はデータを新聞記事1年分しか用いてないにも関わらず良い結果を得ることができた。ちなみに平仮名 $N-gram$ は新聞記事5年分用いている。

このことから、削除の優位性が顕著であることが言える。

問題点として次のようなことがあげられる。

- 形態素解析を用いているため平仮名 $N-gram$ よりも検出時間が長くなる
- 品詞を用いるとき、語尾の平仮名が消えてしまう

  (例) 動く (動詞) この場合、「動く」で1単語であるので品詞として考えると「く」が消えてしまう。

Chapter 7

結論

本研究では、平仮名と品詞からなる平仮名列の誤り検出を試みた。

平仮名 $N-gram$ の検索では削除の精度が悪いのを改善するため、品詞との組み合わせを用いた結果、削除の精度をあげることができた。本研究では平仮名と品詞からなる $N-gram$ を用いた平仮名列の誤り検出実験の結果、以下の成果が得られた。

- 従来手法との比較実験を行い、他手法よりも削除の精度が高いという点で良い結果が得られた
- 本研究では新聞記事 1 年分で全体の精度が上がった　(平仮名 $N-gram$ のデータは新聞記事 5 年分)

  これより本手法が平仮名列の誤り検出において有用と考えられる

Chapter 8

謝辞

本研究の遂行及び論文の作成において多大な御助言及び御指導を賜った新納 浩幸 教官 (茨城大学工学部システム工学科) に深い感謝の意を表します。

また、本研究で利用したコーパスは日本経済新聞 $CD-ROM'90$ 版と毎日新聞 $CD-ROM'94$ 版から得ています。利用を許可していただいた日本経済新聞社と毎日新聞社に深く感謝します。

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、同研究室の池谷昌紀君に感謝致します。

Bibliography

[1] 北研二、中村哲、永田昌明：“音声言語処理”, 森北出版株式会社 (1996).

[2] 白木伸征, 黒橋禎夫, 長尾真：“大量の平仮名登録による日本語スペルチェッカの作成”, 第 3 回言語処理学会年次大会 pp445–448 (1997).

[3] 新納浩幸：“平仮名 N-gram による平仮名列の誤り検出とその修正”, 第 4 回言語処理学会年次大会 pp512–515 (1998).

[4] 河西朝雄：“C 言語によるはじめてのアルゴリズム入門”, 株式会社技術評論社 (1996).

[5] 田中靖大：“日本語のスペルチェッカーを目指して”, bit,Vol.30,No.10,pp445–448 (1997).

Appendix A

プログラムソースリスト

```
/*============================================================

        品詞と平仮名の $N-gram$ による誤り検出 Program

              Written by : Yasuaki Kawano (Ver 1.00)

  ============================================================*/
\newpage

/*記号*/
#include<stdio.h>
```

```
#include<stdlib.h>
void  bbuf(int);
main(int argc,char *argv[])
{
  FILE *f1;
  char buf[800];
  char hinsi[100];
  char sai6[100];
  char sai8[100];
  char cc;
  int r;
  int HHHHH=0;
  int k,i,j,l,ll=0;
  if(argc !=2) { printf("引数の数が違います\n"); exit(1); }
  if((f1 = fopen(argv[1],"r"))==NULL) exit(1);
printf("文");
  while(fgets(buf,800,f1)!=NULL){
      if((buf[0]=='E') && (buf[1]=='O') && (buf[2]=='S')){
        printf("  ");
        bbuf(HHHHH);
        printf("EOS  \n");
        HHHHH++;
        printf("文");     }
     i=0,k=0;
  while(buf[i]!=' ')  {
      if(buf[i]==-92) { i+=2; /*printf("i=%d buf[i]= %d\n",i,buf[i]);*/}
      else {k=1;
```

```
    if((buf[0]=='E') && (buf[1]=='O') && (buf[2]=='S'));
/*************************************************************************/
    else{
        l=0;
        for(j=0;j<3;j++){
            while(buf[l]!=' ') l++;
            while(buf[l]==' ') l++;
         }
         j=0; ll=l;
         while(buf[l]!=' '){
            hinsi[l-ll]=buf[l];  l++;
         }
         hinsi[l-ll]='\0';

         l=0;
         for(j=0;j<5;j++){
            while(buf[l]!=' ') l++;
            while(buf[l]==' ') l++;
         }
         j=0; ll=l;
         while(buf[l]!=' '){
            sai6[l-ll]=buf[l]; l++;
         }
         sai6[l-ll]='\0';
         l=0;
         for(j=0;j<7;j++){
            while(buf[l]!=' ') l++;
```

```
            while(buf[l]==' ') l++;
        }
        j=0;  ll=l;
        while(buf[l]!=' '){
            sai8[l-ll]=buf[l]; l++;
        }
        sai8[l-ll]='\0';
        if((strcmp(hinsi,"形容詞\0")==0)&&(strcmp(sai8,"イ形容詞アウオ段\0")
            printf("0");
        }
        else if((strcmp(hinsi,"形容詞\0")==0)&&(strcmp(sai8,"イ形容詞イ段
\0")==0)){
            printf("1");
        }
        else if((strcmp(hinsi,"形容詞\0")==0)&&(strcmp(sai8,"ナ形容詞\0")==
            printf("2");
        }
        else if((strcmp(hinsi,"形容詞\0")==0)&&(strcmp(sai8,"ナ形容詞イ段
特殊\0")==0)){                printf("3");
        }
        else if((strcmp(hinsi,"形容詞\0")==0)&&(strcmp(sai8,"ナノ形容詞\0")
            printf("4");
        }
        else if((strcmp(hinsi,"形容詞\0")==0)&&(strcmp(sai8,"タル形容詞\0")
            printf("5");
        }
        else if((strcmp(hinsi,"形容詞\0")==0)&&(strcmp(sai8,"ナ形容詞特殊
```

```
\0")==0)){
                printf("6");
            }
            else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"無活用型\0")==
                printf("7");
            }
            else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"助動詞く型\0")
                printf("8");
            }
            else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"ナ形容詞\0")==
                printf("9");
            }
            else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"イ形容詞イ段
\0")==0)){
                printf("A");
            }
            else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"判定詞\0")==0)
                printf("B");
            }
            else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"助動詞だろう
型\0")==0)){
                printf("C");
            }
            else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"助動詞そうだ
型\0")==0)){
                printf("D");
            }
```

```
        else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"ナノ形容詞\0")
            printf("E");
        }
        else if((strcmp(hinsi,"助動詞\0")==0)&&(strcmp(sai8,"助動詞ぬ型\0")
            printf("F");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"カ変動詞\0")==0)
            printf("G");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"カ変動詞来\0")==
            printf("H");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"サ変動詞\0")==0)
            printf("I");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"ザ変動詞\0")==0)
            printf("J");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"動詞性接尾辞ま
す型\0")==0)){                printf("K");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"ナ形容詞イ段特
殊\0")==0)){
            printf("L");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"母音動詞\0")==0)
            printf("M");
```

```
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞カ行\0")
            printf("N");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞ガ行\0")
            printf("O");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞カ行促
音便型\0")==0)){                printf("P");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞サ行\0")
            printf("Q");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞タ行\0")
            printf("R");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞ナ行\0")
            printf("S");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞バ行\0")
            printf("T");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞マ行\0")
            printf("U");
        }
        else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞ラ行\0")
            printf("V");
```

```
            }
            else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞ラ行イ
形\0")==0)){
                printf("W");
            }
            else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞ワ行\0")
                printf("X");
            }
            else if((strcmp(hinsi,"動詞\0")==0)&&(strcmp(sai8,"子音動詞ワ行文
語音便形\0")==0)              ){
                printf("Y");
            }
            else if((strcmp(hinsi,"指示詞\0")==0)&&(strcmp(sai6,"名詞形態指示
詞\0")==0)){
                printf("Z");
            }
            else if((strcmp(hinsi,"指示詞\0")==0)&&(strcmp(sai6,"連体詞形態指
示詞\0")==0)){                  printf("a");
            }
            else if((strcmp(hinsi,"指示詞\0")==0)&&(strcmp(sai6,"副詞形態指示
詞\0")==0)){
                printf("b");
            }
            else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"普通名詞\0")==0)
                printf("c");
            }
            else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"副詞的名詞\0")==
```

```
    printf("d");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"形式名詞\0")==0)
    printf("e");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"固有名詞\0")==0)
    printf("f");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"地名\0")==0)){
    printf("g");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"人名\0")==0)){
    printf("h");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"組織名\0")==0))
    printf("i");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"サ変名詞\0")==0)
    printf("j");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"数詞\0")==0)){
    printf("k");
}
else if((strcmp(hinsi,"名詞\0")==0)&&(strcmp(sai6,"時相名詞\0")==0)
    printf("l");
}
else if((strcmp(hinsi,"助詞\0")==0)&&(strcmp(sai6,"終助詞\0")==0))
```

```
            printf("m");
        }
        else if((strcmp(hinsi,"助詞\0")==0)&&(strcmp(sai6,"接続助詞\0")==0)
            printf("n");
        }
        else if((strcmp(hinsi,"助詞\0")==0)&&(strcmp(sai6,"副助詞\0")==0))
            printf("o");
        }
        else if((strcmp(hinsi,"助詞\0")==0)&&(strcmp(sai6,"格助詞\0")==0))
            printf("p");
        }
        else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"名詞接頭辞\0")
            printf("q");
        }
        else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"動詞接頭辞\0")
            printf("r");
        }
        else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"イ形容詞接頭
辞\0")==0)){
            printf("s");
        }
        else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"ナ形容詞接頭
辞\0")==0)){
            printf("t");
        }
        else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"名詞性名詞助
数辞\0")==0)){                printf("u");
```

```
            }
            else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"名詞性特殊接
尾辞\0")==0)){                  printf("ｖ");
            }
            else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"名詞性名詞接
尾辞\0")==0)){                  printf("ｗ");
            }
            else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"名詞性述語接
尾辞\0")==0)){                  printf("ｘ");
            }
             else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"形容詞性名詞
接尾辞\0")==0))          {
                printf("ｙ");
            }
            else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"形容詞性述語
接尾辞\0")==0)){                printf("ｚ");
            }
            else if((strcmp(hinsi,"接頭辞\0")==0)&&(strcmp(sai6,"動詞性接尾辞
\0")==0)){
                printf("α");
            }
            else if((strcmp(hinsi,"特殊\0")==0)&&(strcmp(sai6,"句点\0")==0)){
                printf("β");
            }
            else if((strcmp(hinsi,"特殊\0")==0)&&(strcmp(sai6,"読点\0")==0)){
                printf("γ");
            }
```

```
else if((strcmp(hinsi,"特殊\0")==0)&&(strcmp(sai6,"括弧始\0")==0))
    printf("δ");
}
else if((strcmp(hinsi,"特殊\0")==0)&&(strcmp(sai6,"括弧終\0")==0))
    printf("ε");
}
else if((strcmp(hinsi,"特殊\0")==0)&&(strcmp(sai6,"記号\0")==0)){
    printf("ζ");
}
else if((strcmp(hinsi,"特殊\0")==0)&&(strcmp(sai6,"空白\0")==0)){
    printf("η");
}
else if((strcmp(hinsi,"連体詞\0")==0)){
    printf("θ");
}
else if((strcmp(hinsi,"副詞\0")==0)){
    printf("ι");
}
else if((strcmp(hinsi,"判定詞\0")==0)){
    printf("κ");
}
else if((strcmp(hinsi,"接続詞\0")==0)){
    printf("λ");
}
else if((strcmp(hinsi,"感動詞\0")==0)){
    printf("μ");
}
```

```
            else if((strcmp(hinsi,"連語\0")==0)){
                printf("ν");
            }
            else if((strcmp(hinsi,"未定義語\0")==0)&&(strcmp(sai6,"カタカナ\0")
                printf("ξ");
            }
            else if((strcmp(hinsi,"未定義語\0")==0)&&(strcmp(sai6,"アルファベ
ット\0")==0)){                printf("ο");
            }
            else if((strcmp(hinsi,"未定義語\0")==0)&&(strcmp(sai6,"その他\0")==
                printf("π");
            }
        }
       break;
    }
    }
    if(k!=1){
       i=0;
       while(buf[i]!=' '){
         printf("%c",buf[i]);i++;
       }
    }
    k=0; i=0;
    }
printf("\b\b");
    if((r=fclose(f1))==-1) exit(1);
}
```

```
void bbuf(int ggg)
{
  FILE *f2,*fopen();
  char buf2[800];
  int r2;
  char f_name[25]="data";
  int KKKK=0;
  if((f2 = fopen("test.data","r"))==NULL) {printf("index がひらけません\n");
    exit(1);}
  while(fgets(buf2,800,f2)!=NULL){
   if(KKKK==ggg){
      printf(buf2);
      if((r2=fclose(f2))==-1) { printf("ファイルが閉じれないよ"); exit(1); }
      return;
   }
   KKKK++;
 }

}


\newpage


/*kh4k.c*/
#include<stdio.h>
#include<stdlib.h>
```

```
main(int argc,char *argv[])
{
FILE *f1;
char buf[2000];
int r;
int i=0,j=0,k=0,l,n=0;
char m;
if(argc !=2) { printf("引数の数が違います\n"); exit(1); }
if((f1 = fopen(argv[1],"r"))==NULL) exit(1);
while(fgets(buf,2000,f1)!=NULL){
i=0;
while((buf[i]!='\n')&&(buf[i]!='\0')&&(buf[i]!=' ')){
   k=0;
   while(buf[i]==-92){
          k++;
          i+=2;}
    if(k>=4){
        for(j=i-2*(k+1);j<i+2;j++)
            printf("%c",buf[j]);
 /*       printf("\n");*/
    while(buf[n]!=' ') n++;
    while(buf[n]!='\0'){ printf("%c",buf[n]);
    n++;}
    k=0;
    }
  i+=2;
  n=0;
```

```
  }
}
if((r=fclose(f1))==-1) exit(1);
}

\newpage

/*4グラム*/
#include<stdio.h>
#include<stdlib.h>
main(int argc,char *argv[])
{
FILE *f1;
char buf[800];
int r;
int i=0,j=0,k,l=0;
int  m=0;
if(argc !=2) { printf("引数の数が違います\n"); exit(1); }
if((f1 = fopen(argv[1],"r"))==NULL) exit(1);
while(fgets(buf,800,f1)!=NULL){
m=0;
l=0;
while(buf[l]!=' ')
l++;
l=l-1;
   for(i=0;i<=l-6;i+=2)
      {
```

```
        for(k=0;k<=7;k++){
          printf("%c",buf[i+k]);}
        printf("\n");
       }
    while(buf[m]!=' ')
      m++;
    while(buf[m]!='\0'){
      printf("%c",buf[m]); m++;}
}
if((r=fclose(f1))==-1) exit(1);
}

\newpage


/*4グラム検索*/
#include <stdio.h>
#include <math.h>
long index(char bstr[]);
int  kensaku(char astr[],long lll);
main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  char buf[1280];
  int e,r;
  int test=0;
  long ee;
```

```
  void quit(char *s);
  if(argc != 2) quit("引数の数が違う"); /* prog  KensakuFile */
  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイル１が開けない");
  while(fgets(buf,1280,f1) != NULL) {
    if(buf[0]!=' '){
       buf[8] = '\0';
       ee = index(buf);
       e  = kensaku(buf,ee);
       if(e<=10) {
          test=1;
          printf("%s  %d\n",buf,e);
       }
    }
    else{ if(test==1){printf(buf); test=0;}}
  }
 if((r = fclose(f1)) == -1) quit("ファイル１が閉じれない");
}
void quit(char *s)
{
  printf(s); putchar('\n');
  exit(1);
}
/*********************************************/
long index(char bstr[])
{
  FILE *f2,*fopen();
  char buf2[228];
```

```
int r2;
char index_name[25]="Index";
char a2[1500][228];
char b2[1500][228];
int i2=0,j2=0,l2=0,k2,ii2;
int low2,high2,mid2;
if((f2 = fopen("Index","r2"))==NULL) quit("index がひらけません");
while(fgets(buf2,228,f2)!=NULL){
   /***データの右側の値を配列 b にいれる*************/
   while(buf2[l2]!=' ') l2++;
   while(buf2[l2]==' ') l2++;
   while(buf2[l2]!=' ') l2++;
   while(buf2[l2]==' ') l2++;
   while((buf2[l2]!=' ')&&(buf2[l2]!='\n')&&(buf2[l2]!='\0')){
        b2[i2][j2]=buf2[l2]; j2++; l2++; }
   b2[i2][l2]='\0';
   /***データの左側の値を抜き出し整数に直し 配列 a にいれる。*******/
   j2=0;
   while(buf2[j2]!=' ') j2++;
   buf2[j2]='\0';
   j2=0,l2=0;
   strcpy(a2[i2],buf2);
   i2++;
}
/***<<<<<fgets 終わり>>>>>***/
low2=0;high2=i2;
/****二分探索 ****/
```

```
  while (low2<=high2){
      mid2=(low2+high2)/2;
      if (strcmp(a2[mid2],bstr)<=0)  low2=mid2+1;
      if (strcmp(a2[mid2],bstr)>=0)  high2=mid2-1;
  }
  /****結果の出力*****/
      if(low2==high2+2) {  /*printf("%s \n",b2[mid2]); */   /* データの中にあ
る場合 */
        if((r2=fclose(f2))==-1) quit("ファイルが閉じれないよ");
        return(atol(b2[mid2]));        }
      else /*printf("%s\n",b2[high2]);*/ {
        if((r2=fclose(f2))==-1) quit("ファイルが閉じれないよ");
           return(atol(b2[high2]));}
/*
  if((r2=fclose(f2))==-1) quit("ファイルが閉じれないよ");
  return(****);
*/
}

/*************************************************************/
int kensaku(char astr[],long lll);
{
  FILE *f3,*fopen();
  char buf3[228];
  int r3;
  char a3[1500][228];
  char b3[1500][228];
```

```
  char ret[100];
  int k3;
  int kenc3,z3,l3=0;
  char kk3[20];
  int ii3,i3=0,j3=0;
  int low3,high3,mid3;
  if((f3 = fopen("all.hind","r3")) == NULL) quit("h4g が開けない");
  fseek(f3,lll,SEEK_SET);
 ii3 = 0;
 while(fgets(buf3,228,f3) != NULL) {
   if(ii3==1000) break;
   else{
     /***データの右側の値を配列 b にいれる*************/
     while(buf3[l3]!=' ') l3++;
     while(buf3[l3]==' ') l3++;
     while((buf3[l3]!=' ')&&(buf3[l3]!='\n')&&(buf3[l3]!='\0'))
{ b3[i3][j3]=buf3[l3]; j3++; l3++; }b3[i3][l3]='\0';
     /***データの左側の値を抜き出し整数に直し 配列 a にいれる。*******/
     j3=0;
     while(buf3[j3]!=' ') j3++;
     buf3[j3]='\0';
     j3=0,l3=0;
    /* k=atoi(buf);*/
     strcpy(a3[i3],buf3);
     i3++;
   ii3++;}
}
```

```
  low3=0;high3=ii3;
  /****二分探索 ****/
  while (low3<=high3){
      mid3=(low3+high3)/2;
      if (strcmp(a3[mid3],astr)<=0)  low3=mid3+1;
      if (strcmp(a3[mid3],astr)>=0)  high3=mid3-1;
  }
  /****結果の出力*****/
      if(low3==high3+2) {  /*printf("%s \n",b3[mid3]); */
   if((r3 = fclose(f3)) == -1) quit("ファイル1が閉じれない");
         return(atoi(b3[mid3])) ; }
            /* データの中にある場合 */
       else { /*printf("%s\n",b3[high3]); */
           if((r3 = fclose(f3)) == -1) quit("ファイル1が閉じれない");
           /*return(atoi(b3[high3]));}*/
           return(0);}
/*
  if((r3 = fclose(f3)) == -1) quit("ファイル1が閉じれない");
*/
 /*   return(k);*/
}
\newpage

/**/
#include <stdio.h>
#include <math.h>
long index(char bstr[]);
```

```
int  kensaku(char astr[],long lll);
main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  char buf[1280];
  int e,r;
  int test=0;
  long ee;
  void quit(char *s);
  if(argc != 2) quit("引数の数が違う"); /* prog  KensakuFile */
  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイル１が開けない");
  while(fgets(buf,1280,f1) != NULL) {
    if(buf[0]!=' '){
     buf[8] = '\0';  /* この部分はこの前説明したから省略 */
     ee = index(buf);
     e  = kensaku(buf,ee);  /* ここが問題 */
     if(e<=10) {test=1;
/*     printf("%s  %d\n",buf,e);*/}}
    else{ if(test==1){printf(buf); test=0;}}
  }
 if((r = fclose(f1)) == -1) quit("ファイル１が閉じれない");
}
void quit(char *s)
{
  printf(s); putchar('\n');
  exit(1);
}
```

```
/*******************************************/
long index(char bstr[])
{
  FILE *f2,*fopen();
  char buf2[228];
  int r2;
  char index_name[25]="Index";
  char a2[1500][228];
  char b2[1500][228];
  int i2=0,j2=0,l2=0,k2,ii2;
  int low2,high2,mid2;
  if((f2 = fopen("Index","r2"))==NULL) quit("index がひらけません");
  while(fgets(buf2,228,f2)!=NULL){
     /***データの右側の値を配列 b にいれる*************/
     while(buf2[l2]!=' ') l2++;
     while(buf2[l2]==' ') l2++;
     while(buf2[l2]!=' ') l2++;
     while(buf2[l2]==' ') l2++;
     while((buf2[l2]!=' ')&&(buf2[l2]!='\n')&&(buf2[l2]!='\0')){
          b2[i2][j2]=buf2[l2]; j2++; l2++; }
     b2[i2][l2]='\0';
     /***データの左側の値を抜き出し整数に直し 配列 a にいれる。*******/
     j2=0;
     while(buf2[j2]!=' ') j2++;
     buf2[j2]='\0';
     j2=0,l2=0;
     strcpy(a2[i2],buf2);
```

```
    i2++;
  }
  /***<<<<<fgets 終わり>>>>>***/
  low2=0;high2=i2;
  /****二分探索 ****/
  while (low2<=high2){
      mid2=(low2+high2)/2;
      if (strcmp(a2[mid2],bstr)<=0)  low2=mid2+1;
      if (strcmp(a2[mid2],bstr)>=0)  high2=mid2-1;
  }
  /****結果の出力*****/
      if(low2==high2+2) {  /*printf("%s \n",b2[mid2]); */   /* データの中にあ
る場合 */
        if((r2=fclose(f2))==-1) quit("ファイルが閉じれないよ");
        return(atol(b2[mid2]));        }
      else /*printf("%s\n",b2[high2]);*/ {
        if((r2=fclose(f2))==-1) quit("ファイルが閉じれないよ");
           return(atol(b2[high2]));}
/*
  if((r2=fclose(f2))==-1) quit("ファイルが閉じれないよ");
  return(****);
*/
}
/*************************************************************/
```