全文検索ソフトを使った
QA システム

執筆者：星 秀明

指導教官：新納 浩幸

平成13年 3月 1日

目 次

図 目 次

表 目 次

Chapter 1

はじめに

1.1 概要

現存する情報検索システムは、「文書」の検索システムであり、ユーザは検索された文書を読んで必要な情報を取り出さなければならない。これに対してユーザが持っている質問 (情報要求) に対して、より直接的に答える技術を開発しようというのが QA システムである。例えば、「1994 年に交通事故で死んだ人は何名ですか?」と質問されたら「16,049 人」と答えるシステムである。このように自然言語の質問文に対し文書集合から解答を抽出する日本語質問応答システムについて報告する。

QA システムは次のような段階を踏めば良いと考えられている。

1. 質問文の解析
2. 文書データの検索
3. 検索結果の文書の解析

この三段階で処理を行う枠組みはすでに提案されているが [1]、どの程度うまくいくかは未知数である。

本研究の目的はQAシステムを試作することとQAシステムの問題点を整理することである。

今回TREC QA trackを参考に、人数表現を尋ねる質問を26問用意し、新聞記事より検索し、評価を行った。独自のシステムで検索した結果、新聞記事中に答えのあるものならば50％で解答を得ることができた。

1.2 本論文の構成

Chapter2ではQAシステムについて述べる。QAシステムはIRアプローチとIEアプローチの大きく2種類に分類される。ここでは2つのアプローチ方法を説明し、さらに本研究で用いたIEアプローチについて詳しく述べる。

Chapter3では全文検索システムのNamazuについて述べる。Namazuの用いている手法や基本構成について説明する。

Chapter4では本研究の目的の一つであるQAシステムの試作について述べる。概要、工夫点などを説明し、完成までの手順を説明する。

Chapter5では本QAシステムに複数の質問文を入力し、その精度をはかる実験について述べる。

Chapter6では実験の考察を述べ、さらに本研究の目的の一つであるQAシステムの問題点についても述べる。

Chapter 2

QA システム

米国では情報検索コンテスト TREC が 90 年代初頭から継続して開催されており、1999 年秋に TREC-8[2][3] が開催された。TREC-8 では、QA、WEB という 2 つのタスクが新たに導入された。TREC は情報検索の分野では最先端の会議であり、そこで取り上げられるタスクは今後重要な応用になると考えられている。そのため QA システムは今後の自然言語技術にとって重要な研究テーマである。

TREC-8 に参加した QA システムがとっている手法は、大きく以下の 2 種類に分類される。

(A) 情報検索技術を直接適用した手法 (IR アプローチ)

(B) 情報検索に NLP・情報抽出技術を融合した手法 (IE アプローチ)

2.1 IR アプローチ

文書を細かいパッセージに分割し、検索結果の上位 5 パッセージ (またはそれを若干修正したもの) を回答とする手法である。ただし、ベクトル空間法のような手法よりはむしろ、coordination match によるランキング (含まれる検索語の種類が多いほど上位にランクされる) や、検索語のパッセージ中での出現位置の近さを考慮した重み付け手法などがとられていることが多い。

2.2 IE アプローチ

質問文や検索結果の文書に対して形態素・構文解析や情報抽出処理を行い、これにより得られた情報を元に文やパッセージのフィルタリングやスコアリングを行うものである。

TREC-8 の参加システムの多くは IE アプローチの手法をとっているので、本研究でもこちらを使用する。以下のような手法が、IE アプローチの典型である。

(a) 質問文の解析

質問文を解析し、検索に使う語、および質問タイプ (答えとして何を求められているのか) を決定する。

(b) 文書データの検索

(a) で得られた検索語により、検索を実行して文書を得る。ここで検索された上位ランクの文書が、以降の処理の対象となる。

(c) 検索結果の文書の解析

検索結果文書を文単位に分割し、各文ごとに解析を実施する。ここで、構文解析、固有表現や数値表現の抽出処理が行われる (以下、固有表現・数値表現を会わせて NE(Named Entity) と呼ぶ)。

(d) 文または entity のスコアリング

上位 5 件を求めるためにスコアリングを行う。coordination match や検索語の出現位置の近さから重み付けするもの、質問タイプと NE との間の適合度算出、語の種類別の重み付け、ルールの適応/非適応による加点/減点など、さまざまな手法が試みられている。

(e) 結果の整形

スコアの高い文やentityについて、最終的な回答文字列が短くなるように結果を整形する。質問タイプに合致する語句を中心に、その周辺部分を切り出す手法が主流であるが、その他にも、不要語等を取り除いて単語の羅列にする、複数の文からNEや検索語周辺部分を抽出して結合する、といった手法も試みられている。

Chapter 3

全文検索システム

全文検索システムは、データベースの一種である。ただ、一般的なデータベースはあらかじめ検索用の項目を作成しておくが、全文検索システムではすべてのデータが検索の対象となる。ここが「全文」といわれるゆえんである。全文検索システムの構造を Figure3.1に示す。

もっとも、一般的なデータベースでも全文検索は可能である。ただ主たる目的ではないためにインデックスが作成されておらず、大変時間がかかってしまうのである。逆に全文検索システムでは検索対象を文法的に解析して単語に分割し、自動的にインデックスを作成するのでインデックス作成には時間がかかるが、検索自体は高速である。

最も知られた全文検索システムは、いわゆる Web エンジンである。あれほど扱う情報が多くなると大規模なシステムが必要となるが、自社のサーバだけならフリーのシステムでも十分対応できる。それが今回紹介する Namazu である。

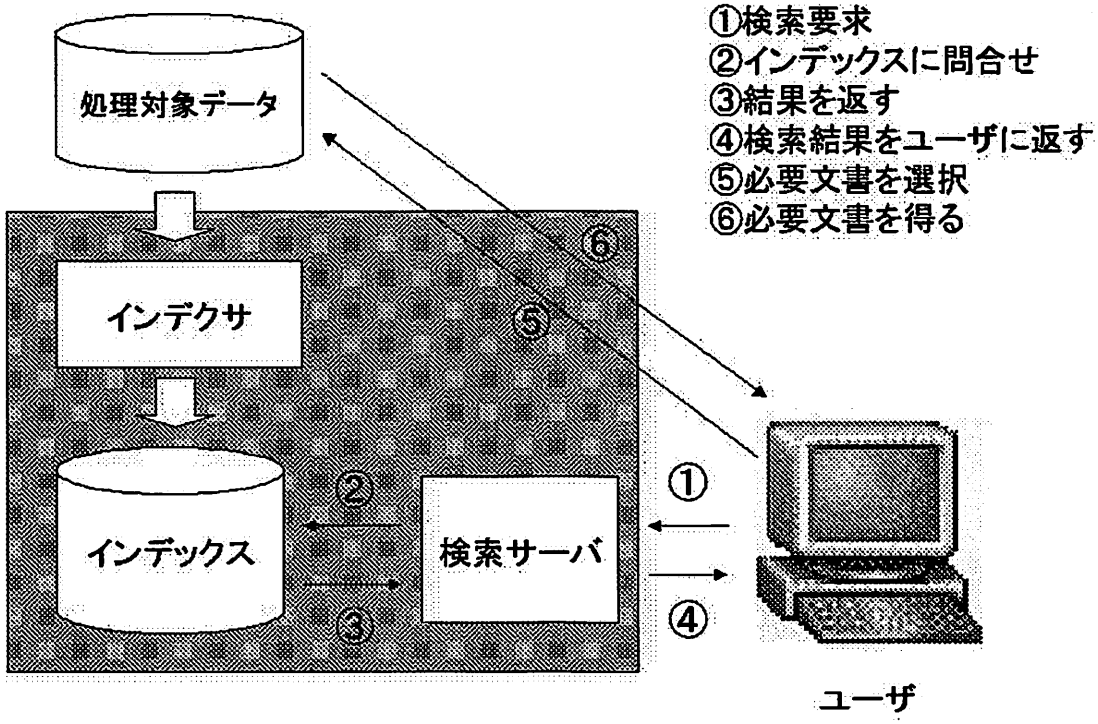

Figure 3.1: 全文検索システムの構造

3.1 Namazu

「Namazu」とは、手軽に使えることを第一に目指した日本語全文検索システムであり、作者は愛知大学の高林哲氏 (現奈良先端科学技術大学院大学生) である。Namazu には、フリーソフトウェアとしては画期的な多くの特徴がある [4][5]。まとめると次のようになる。

1. 各種プラットホームに対応
2. 高速な検索を実現
3. 多くの検索方法をサポート
4. 検索結果表示の柔軟性
5. 正確な HTML の取扱い
6. 複数のインデックスからの検索をサポート
7. 多彩な検索クライアント

次に Namazu で「できること」と「できないこと」をまとめておく。

Namazu でできること

- 一つの計算機の、一つまたは複数のディレクトリの下にある文書ファイルを指定して、それらの中の単語を調べ、どの文書ファイルにどの単語が含まれるかをインデックスに書き出しておく
- 利用者が入力した検索式と、上記の単語を比べて、その単語が含まれている文書ファイルを表示すること
- 上記単語の指定は、単語の一部分ではなく単語全体である。よって sys と指定しても sysytem は見つからない。そのような時は sys*や*sys*のように前後

に*を付けることによって、sys で始まる単語 (sys*)、sys を含む単語 (*sys*)、sys で終了する単語 (*sys) と入力する

- 作られたインデックスの利用はコマンド行、または cgi-bin が動く HTTP server を Web ブラウザを通じて利用する形で行なう

Namazu でできないこと

- 他の計算機にまで文書ファイルを取りにいって検索すること
- 「一日に 100 万回の利用件数がある」というような大規模な構成のつかいかた

3.2 インデックス法

全文検索システムのインデックスは、書籍の索引とまさに同じ機能を持っている。全文検索システムの場合、インデックス作成で作られるインデックスは、どの単語がどの文書 (Web ページ) に含まれているかを示すものである。通常の本の場合には、重要と思われる単語についてのみ筆者がいくつか抜き出し、その単語についてのインデックスだけが作成される。一方、全文検索システムの場合は、文書に含まれるすべての単語についてのインデックスを作成することが通常行なわれる。検索時の処理はコンピュータが行なうのであるから、多少無駄な単語がインデックスに含まれていたとしても、それはインデックスの容量を若干増やすという方向になるが、そのことが検索スピードを遅くするという要因にはならない。ここでインデックスの概念図を Figure3.2に示す。

3.3 検索手法

ファイルごとに文字列一致を行なう grep などの原始的な検索方法とは違って、全文検索システムは圧倒的に速く検索結果を得ることができるのが最大の特徴で

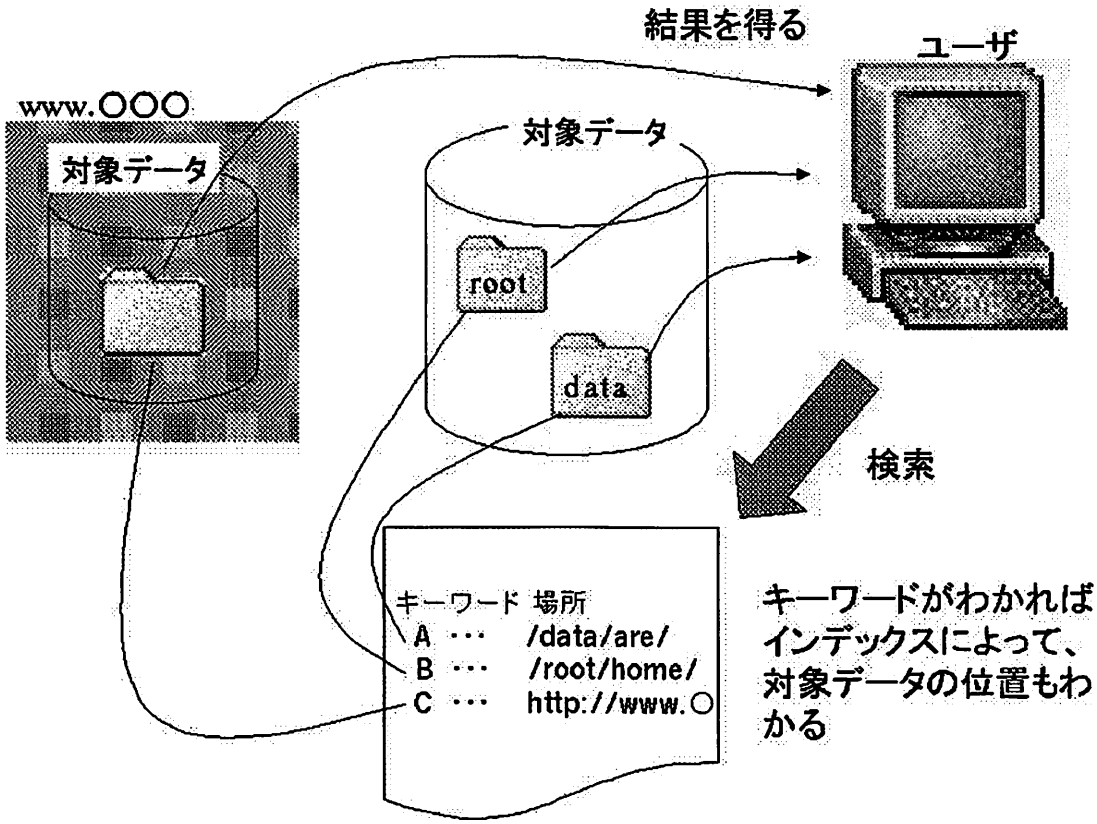

Figure 3.2: インデックスの概念図

あるといえる。

検索方法としては、AND 検索と OR 検索および NOT 検索など、データベース検索における基本的なものを Namazu はサポートしている。この 3 つを使いこなせるようになれば、ほとんどの局面に対応できる。

AND 検索は、一般的には、

word1 AND word2

のように入力し、word1 と word2 の両方が含まれるページを検索するために用いる。データを絞り込むためには必要不可欠な機能である。また、AND を省略すれば、デフォルトで AND 検索になる。

OR 検索は、複数の検索語のうち、一つ以上の検索語が含まれるページを検索するために用いる。OR 検索は、検索対象を広げることはできても絞り込むことはできない。

word1 OR word2

のように、検索語を羅列することによって、word1 あるいは word2 が含まれるページを検索できる。

NOT 検索は、

word1 NOT word2

のように用い、word2 が含まれないページを検索するために用いる。NOT 検索は、単独で用いられることはまずないが、AND 検索や OR 検索と組み合わせて用いられ、検索データの絞り込みを行なう際に、重要な役割を果たす。

以上のような 3 つの基本的な検索方法以外に、フレーズ検索および中間一致、後方一致、正規表現による検索もサポートしているなど、利用者のニーズに応じた自由度の高い検索を簡単に行なうことができるようになっている。

3.4 ランキング

一般に全文検索システムでは、キーワードを含む文書に対して、それぞれの文書がどの程度検索条件にマッチしているかを示す指標として「スコア」と呼ばれる点数付けを行ない、そのスコアの順に並び替えたものを検索結果として出力する。これをランキングと呼んでいる。ランキングの方法は、

1. 検索語の頻度
2. ファイルサイズ
3. 文書の構造

などを利用して総合的に評価してランキングを行なう。つまり、キーワードとなる文字列が複数出現している文書、あるいは、キーワードとなる文字列をタイトルや見出しの中に含んでいる文書などには高い点数が与えられる。

3.5 Namazu の基本構成

Namazu はインデックスの作成を行なう mknmz コマンドと、検索を行なう namazu コマンドから構成されている。

インデックスを作成するには mknmz を用いる。mknmz の引数にはインデックス作成の対象とするディレクトリ名を与える。たとえば、/home/foo/public_html を対象とするならば、

```
mknmz /home/foo/public_html
```

と実行する。すると、/home/foo/public_html 以下の*.html*.txt といったファイルに付いて索引付けを行ない、mknmz を実行したディレクトリに NMZ.* というファイルが作成される。この NMZ.*ファイルが Namazu のインデックスである。mknmz のコマンドオプションはいろいろあるが、今回使ったものを示す。

-a 全てのファイルを対象とする

-c 日本語の単語のわかち書きに Chasen を用いる

-O インデックスの出力先を指定する

作成されたインデックスに対して検索を行なうには namazu コマンドを用いる。

```
namazu bar /home/foo/Namazu/foobar
```

と実行すれば、/home/foo/Namazu/foobar にあるインデックスに対してキーワード bar を検索する。namazu のコマンドオプションはいろいろあるが、今回使ったものを示す。

-o 指定したファイルに検索結果を出力する

-n 一度に表示する件数

-s 短い書式で出力する

-h HTML で出力する

また、複数のキーワードを検索に使う時は、検索式を"ではさむ。

Chapter 4

QA システムの試作

4.1 概要

本研究では、質問文のタイプを人数を聞くものに限定する。検索の対象は毎日新聞の記事 95 年度 1 年分を使用する。以下のような手法が本研究で試作した QA システムの流れである。

(a) 質問文の解析

質問文を解析し、検索に使う検索式を作成する。本研究では「名詞」「未知語」を検索式候補とし、さらに、その検索式候補より「名詞－接尾」「名詞－副詞可能」「名詞－代名詞－一般」「名詞－数詞」に分類される語を除外し、「数」「人」「合計」を除いたものを検索式とした。

(b) 文書データの検索

(a) で得られた検索語により、検索を実行して文書を得る。ここで検索された上位ランクの文書が、以降の処理の対象となる。検索には Namazu を使う。

(c) 検索結果の文書の解析

検索された文書を Chasen にかけ「～人」という単語を解答候補として抜き出す。

(d) 文または entity のスコアリング

上位 5 件を求めるためにスコアリングを行う。本研究のスコアリング法としては、文書内に含まれるキーワードの数を 1 点とし、解答候補とキーワードが同一文内ならば 30 点、同段落内ならば 20 点、同文書内ならば 10 点としてその合計点を解答のスコアとした。

(e) 結果の整形

スコアの高い順に解答を並び替えて表示する。また、解答を含む文書を参照できるようにする。さらに見やすさを向上させるため JAVA で作成した表示系プログラム [6] を用いた。概観を Figure4.1に示す。

4.2 検索式の作成

質問文から検索式を作るには、単純に名詞だけを全て抜き取り、それを検索式とする方法が通常用いられる手法である。しかし、この方法では精度が悪い。検索を AND 検索で行っているためキーワードの数が多すぎて文書が検出されないためである。そこで本研究はキーワードを抽出した際に不要と思われる単語を除外することにした。

質問文を Chasen にかけると Table4.1 と Table4.2ような結果が出る。

これらよりキーワードとなりそうな単語は「名詞」と「未知語」に含まれていることがわかる。さらにキーワードとならなそうな単語は「名詞－代名詞－一般」「名詞－接尾」「名詞－副詞可能」「名詞－数詞」に分類される語と「数」「人」「合計」などだとわかる。

4.3 検索結果の解析

前述の方法で作成した検索式を用いて新聞記事より検索する。検索結果より解答候補 (本研究では「～人」) を抜きだし、スコアリング、ソートを行い出力する。

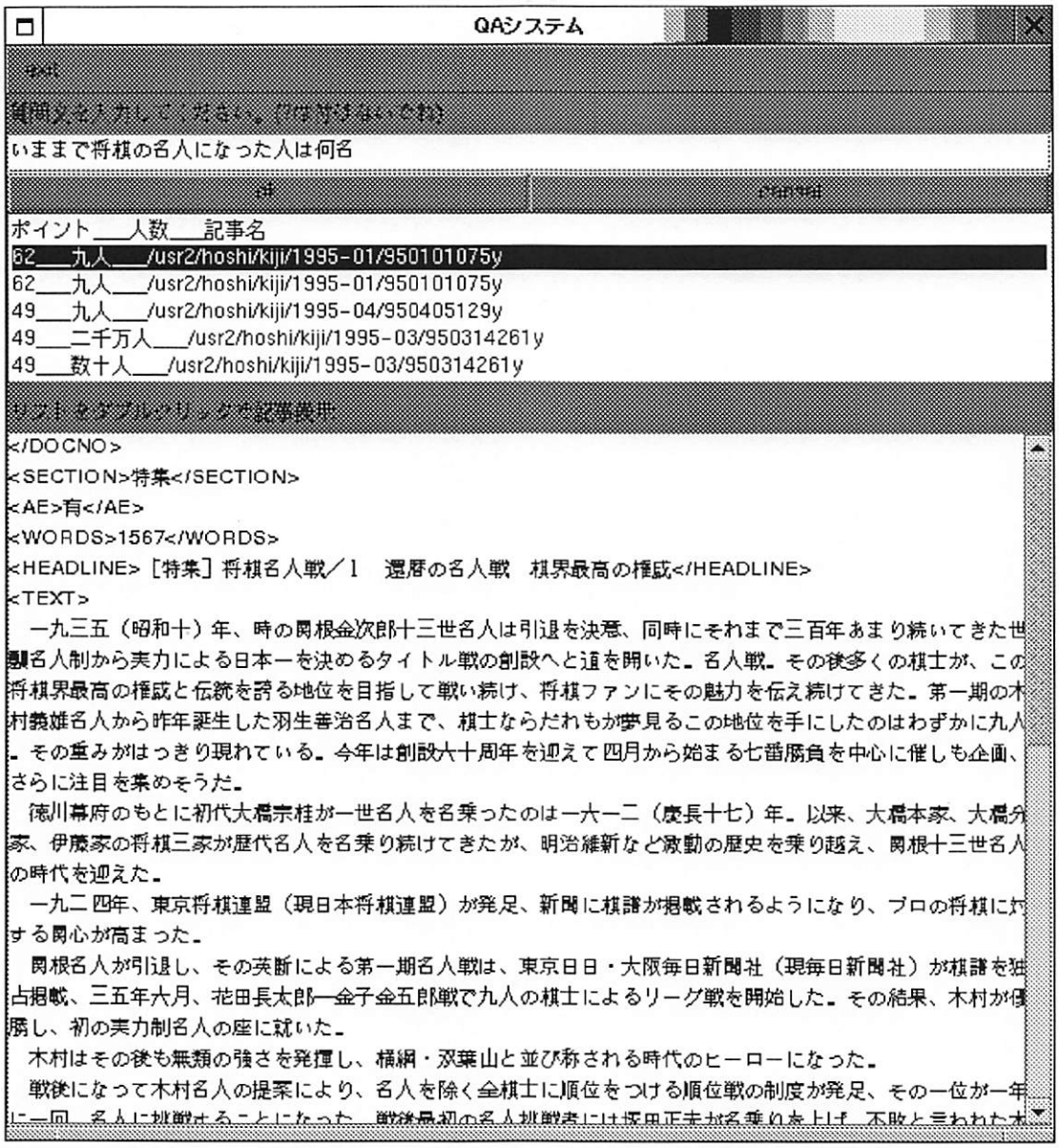

Figure 4.1: QA システム

Table 4.1: Chasen による質問文の解析結果 (その 1)

タイタニック	タイタニック	タイタニック	未知語
号	ゴウ	号	名詞-接尾-一般
に	ニ	に	助詞-格助詞-一般
乗っ	ノッ	乗る	動詞-自立
て	テ	て	助詞-接続助詞
い	イ	いる	動詞-非自立
た	タ	た	助動詞
人	ヒト	人	名詞-一般
は	ハ	は	助詞-係助詞
何	ナン	何	名詞-数
人	ニン	人	名詞-接尾-助数詞
EOS			

Table 4.2: Chasen による質問文の解析結果 (その 2)

これ	コレ	これ	名詞-代名詞-一般
まで	マデ	まで	助詞-副助詞
将棋	ショウギ	将棋	名詞-一般
の	ノ	の	助詞-連体化
名人	メイジン	名人	名詞-一般
に	ニ	に	助詞-格助詞-一般
なっ	ナッ	なる	動詞-自立
た	タ	た	助動詞
人	ヒト	人	名詞-一般
は	ハ	は	助詞-係助詞
合計	ゴウケイ	合計	名詞-サ変接続
何	ナン	何	名詞-数
名	メイ	名	名詞-接尾-助数詞
EOS			

TREC ではさまざまな手法でスコアリングが試みられているが、本研究のスコアリング法は、キーワードと解答候補との距離に注目した。キーワードと解答候補が同文内にあれば 30 点、同段落内にあれば 20 点、同記事内にあれば 10 点とし、さらに記事に含まれるキーワードの数を 1 点として解答候補にスコアを与えた。

4.4 試作

4.4.1 データベース作成

新聞記事は 1 年分が 1 つのファイルとなっていたために分割する必要があった。1 年分をすぐに 1 記事ごとに分割するとディレクトリ内に表示できるファイル数を越えてしまうので、まずは一ヵ月ごとに分割を行った。その後 1 記事ごとへ分割した。1 年文の記事は Figure4.2のようになっているので<DOCNO> のタグに注目して分割した。記事数は 111,396 記事であった。それらを Namazu でデータベース化する。

これらの作業を行なう際の実際のコマンドは次のようになる。まず、一ヵ月ごとに分割するには、

%bunkatu1

次に、分割されたファイルを手作業で Figure4.3のように別々のディレクトリに移動する。さらに bunkatu.c の fopen 部分をそれぞれのファイル名に書き換え、各ディレクトリにコピーする。次に、一記事ごとに分割するには、

%bunkatu

最後に、データベースを作成するには、

%mknmz -a -c -O /usr2/hoshi/index/ /usr2/hoshi/kiji/

と実行する。

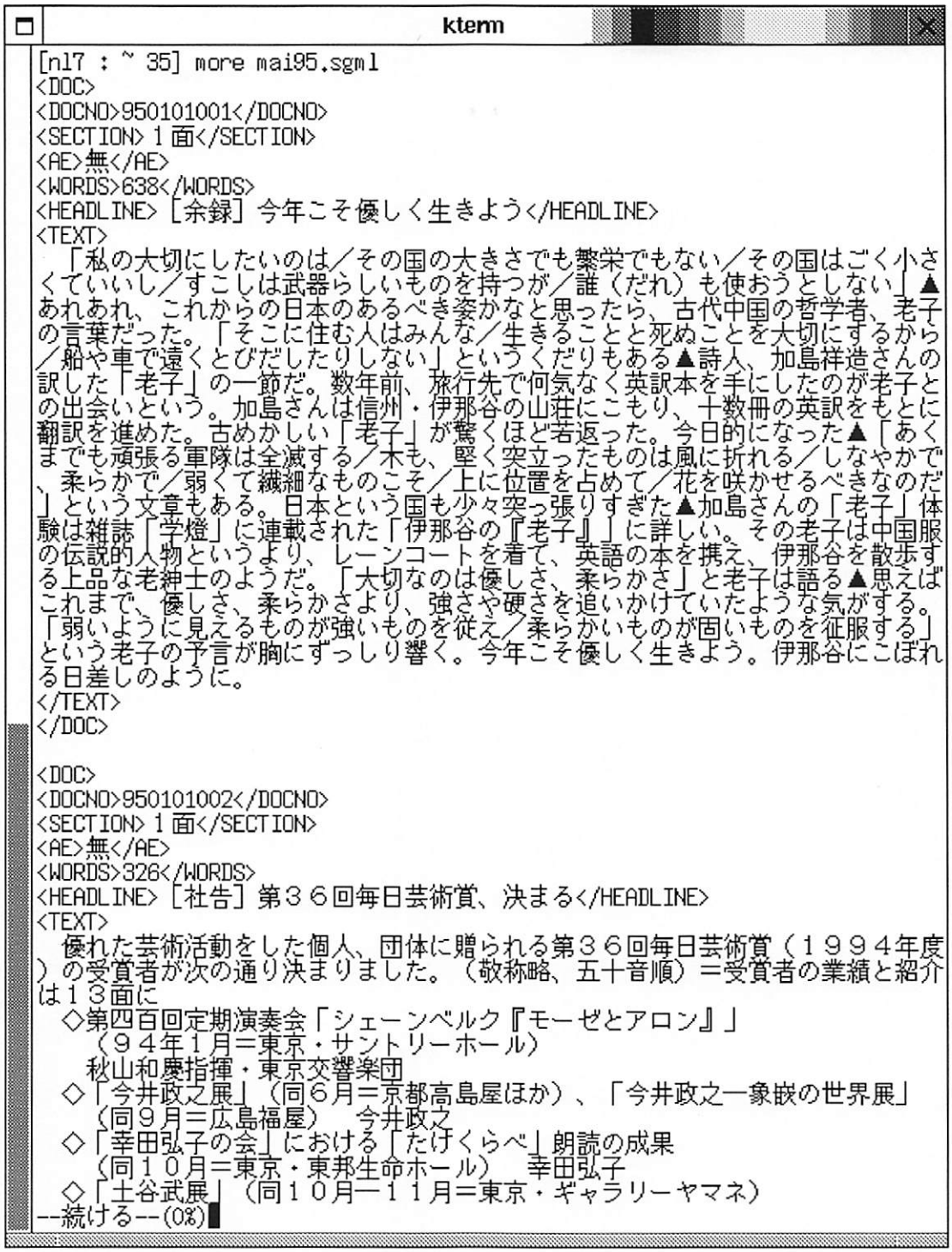

Figure 4.2: 新聞記事

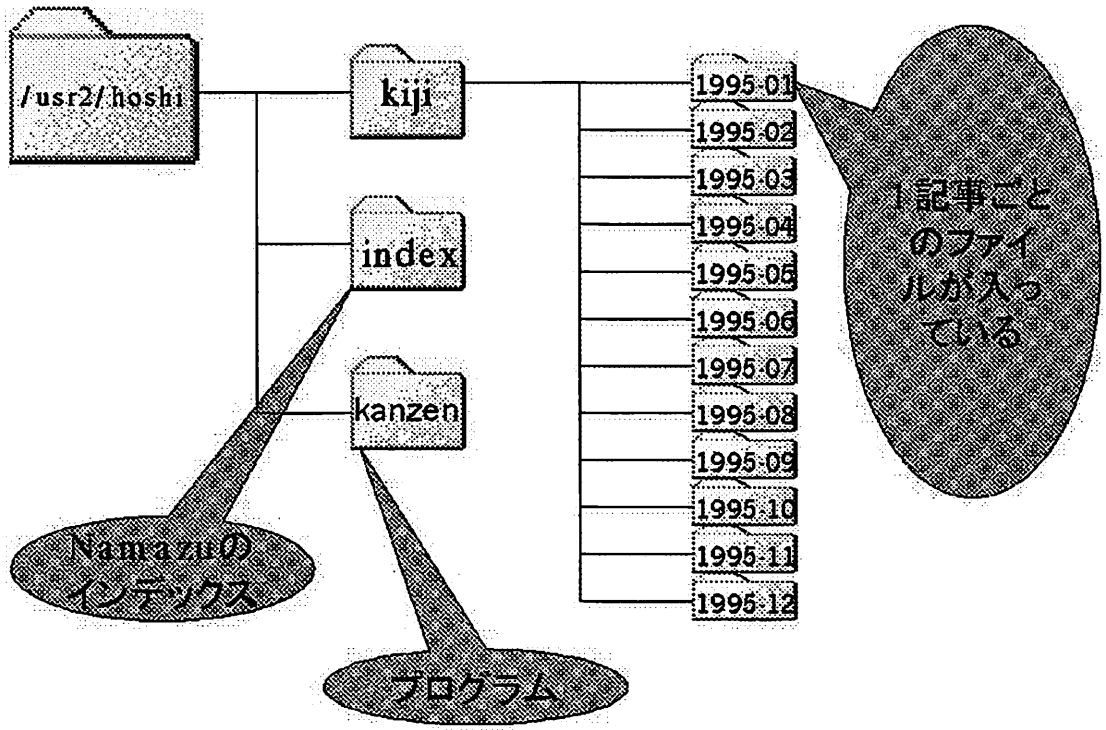

Figure 4.3: ディレクトリ構成

4.4.2 質問文の解析

Chasenの入力ソースはファイルなので質問文のみを記述したファイルを作成する。解析結果の出力はChasenのコマンドオプション-oを使ってファイルに書き出した。ここでは例として「これまで将棋の名人になった人は合計何名?」という質問文を用いる。

これらの作業を行なう際の実際のコマンドは次のようになる。まず、質問文のみを書いたaaaファイルを作成しておく。

%emacs aaa

次に、Chasenで解析する。

%chasen -o bbb aaa

4.4.3 検索式の作成

検索式を作る。改良の効果を調べるため、名詞のみを検索式とするプログラムと不要語を除外するプログラムを作る。Chasenの解析結果とこの二つのプログラムの結果はFigure4.4のようになる。

これらの作業を行なう際の実際のコマンドは次のようになる。名詞のみを検索式とする場合は、

%meisi bbb

を実行する。また、名詞のみを検索式とすると同時に不要語を除外する場合は、

%seisei bbb

を実行する。検索式の出力先は両プログラムともcccファイルとなっている。

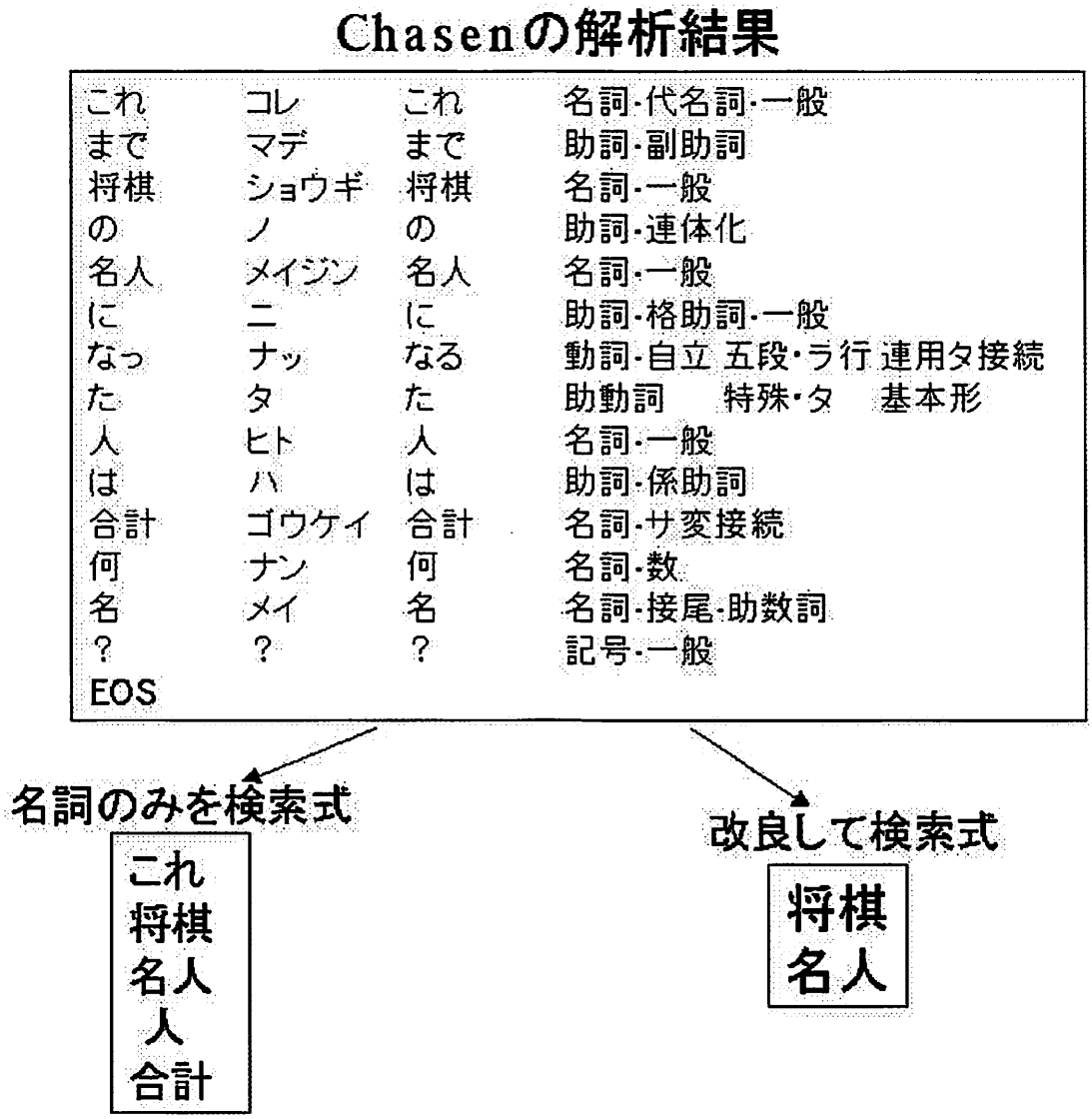

Figure 4.4: Chasen の結果とキーワード

4.4.4 記事の検索

検索式を使いNamazuで検索する。前述のようなコマンドオプションを用いてこの後の処理で使いやすいフォーマットで出力する。本研究では上位5件までの記事を扱う。

検索を行なうコマンドは、

```
%namazu -o zzz -n 5 -s -h "将棋 名人" /usr2/hoshi/index
```

と入力するがここではcccファイルに書かれている検索式を読み込むので、Cプログラムとshellを使い実行する。実際は、

```
%watasu1
```

でshellプログラムkanzentai.sh1を作り、

```
%chmod a+x kanzentai.sh1
%kanzentai.sh1
```

で実行する。また、kanzentai.sh1内で実行されているkensakuによって記事名のみをファイルに出力している(Figure4.5参照)。

4.4.5 解答の作成

スコアリングのために5つの記事を文ごとに分ける。それぞれの記事をChasenにかける。解析結果より人数表現を取り出すプログラムを作る。検索式と解答候補を読み込みスコアリングするプログラムを作る。

記事を1文ごとに分けるのは、

```
%bun 記事ファイル
```

記事をChasenにかけるのは、

```
%chasen -o ddd 記事ファイル
```

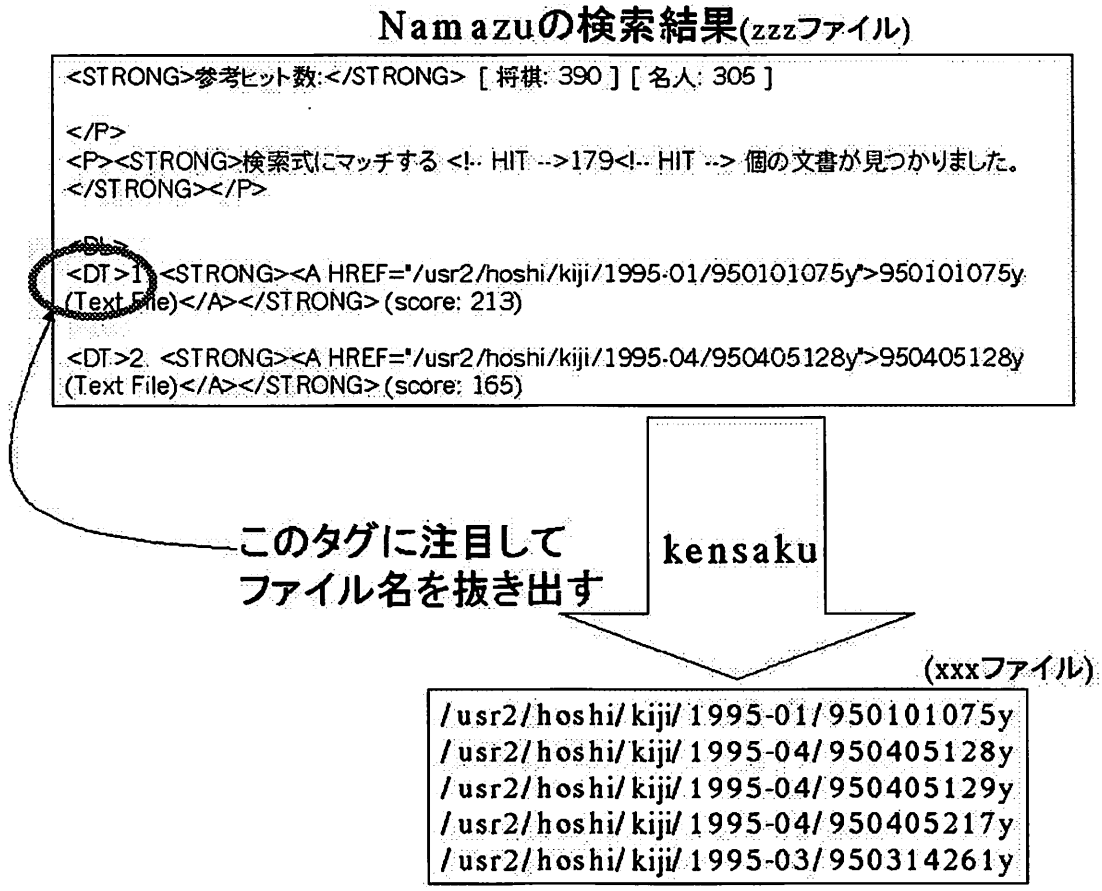

Figure 4.5: Namazu の検索結果

解答候補を抜き出すには、

%kouho

スコアリングするには、

%hyou

を実行する。これを5つの記事すべてに対して行なうため、shellプログラムkanzentai.sh3を用いる。よって、実際のコマンドは

%watasu2

でshellプログラムkanzentai.sh2を作成し、

%chmod a+x kanzentai.sh2
%kanzentai.sh2

を実行すると、kanzentai.sh2内でkanzentai.sh3が実行される。

4.4.6 結果の表示

スコアリング結果をソートして表示するプログラムを作る。表示はスコアの上位5位までとする。また、見やすさ・使いやすさを向上させるためJAVAを用いて表示系プログラムを作った。

解答候補を出力するまでのコマンドを1コマンドにしたshellはkanzentai.shである。これまでの結果をtextファイルに出力しておくために、

```
%kanzentai.sh > text
```

と実行し、

%tori

コマンドで結果をソートして表示することができる。また、JAVA のプログラムを実行させるには、

%java Tako

とする。

4.4.7 1コマンド化

全てのプログラムを 1 コマンドで実行するためにシェルスクリプトを用いた。

%hoshi.sh

で全てのプログラムを実行できる。JAVA プログラムではこのコマンドを呼び出している。さらにまとめとして、JAVA プログラムの起動も簡単に起動できるようにするために、

%QAsystem

コマンドを用意した。全てのプログラムを Figure4.3の kanzen 内に収納して、このコマンドを実行すればシステムが起動する。

Chapter 5

実験

5.1 設定

本研究では、人数を問う質問文に限定する。さらに質問文は記事中に解答が記述されていそうなものを予想して作ったので、必ずしも解答が記事中にあるわけではない。今回用いた 26 問の質問文を Table5.1 と Table5.2に示す。

5.2 結果

5.2.1 解答の有無

質問文を自由に作ったために新聞記事中に解答が載っていない場合があった。これらの質問文は本システムに入力しても解答を出すことができないのは当然である。全部で 5 問あり、問 11、問 15、問 17、問 18、問 19 であった。この 5 問は以降の実験には使わないようにした。

Table 5.1: 本研究で使用した質問文 (その 1)

問題 1 1994 年に交通事故で死んだ人は何名ですか？

問題 2 タイタニック号に乗っていた人は何名？

問題 3 これまで将棋の名人になった人は合計何名？

問題 4 私立園田小学校の児童の数は何名？

問題 5 日本人でこれまでにノーベル賞を受賞した人の数？

問題 6 第四十九回全日本学生音楽コンクールで予選を通過した人は何名？

問題 7 第 9 回ふくべらっきょう花マラソン大会に参加した人は何名？

問題 8 震災の影響で失業した人は何名？

問題 9 タイのチャオプラヤ川からの洪水による死亡者は何名？

問題 10 日本臨床スポーツ医学会の会員数は何名？

問題 11 その年のゴールデンウィークでの海外旅行者数？

問題 12 日本の国家公務員の数？

問題 13 第四十七回青少年航空教室に参加した人の数は何名？

Table 5.2: 本研究で使用した質問文 (その 2)

問題 14 ポケベルの所有者数？

問題 15 日本シリーズの観客動員数？

問題 16 センター試験の受験者数？

問題 17 サッカー FIFA ワールドカップアメリカ大会をテレビで見た人の数？

問題 18 お盆に海外へ行った人の数？

問題 19 ホノルルマラソンで完走した人の数？

問題 20 今年成人式を迎えた人の数？

問題 21 特殊法人へ大蔵省から天下りした人の数は？

問題 22 94 年に消費者問題に関する世論調査に回答した人の数は？

問題 23 ク製剤の薬害訴訟を起こしている患者の数は？

問題 24 沖縄慰霊の日に参加した米軍の退役軍人の数は？

問題 25 カバディという競技を行うには何名必要か？

問題 26 インドで起こった爆弾テロ事件で逮捕された人の数は？

5.2.2 名詞のみで検索

質問文を Chasen にかけ「名詞」「未知語」を抽出し、「何名」「何人」を除くことで検索式を作成して検索した。その後、検索された文書の上位 5 位以内に正解を含む文書が入っているか判定する。結果は 21 問中 4 問検出できた。詳しくは Table 5.3に示す。

5.2.3 本 QA システムで検索

名詞のみのシステムからさらに「名詞－代名詞－一般」「名詞－接尾」「名詞－副詞可能」「名詞－数詞」に分類される語と「数」「人」「合計」を除外することで検索式を作成して検索した。その後、検索された文書の上位 5 位以内に正解を含む文書が入っているか判定する。結果は 21 問中 9 問検出できた。詳しくは Table 5.4に示す。以降の実験はこの 9 問を対象とする。

5.2.4 スコアリング

検出された文書を Chasen により解析し、「～人」という単語を解答候補として抜き出す。また、スコアリングのために文書を 1 文ごとに分割しておく。その後、本研究のスコアリング法に従って解答にスコアを与え、スコアの高い順にソートし出力した。

本研究のスコアリング法を使うと、「改良して検索」で正しい文書が検出されている質問文に対して全て正解が得られている。この結果から本スコアリング法の妥当性が示されたと考える。

Table 5.3: 名詞のみでの検索結果

問	キーワード	判定
1	１９９４, 年, 交通, 事故, 人	×
2	タイタニック, 号, 人	○
3	これ, 将棋, 名人, 人, 合計	×
4	市立, 園田, 小学校, 児童, 数	×
5	日本人, これ, ノーベル, 賞, 受賞, 人	×
6	四十九, 回, 全日本, 学生, 音楽, コンクール, 予選, 通過, 人	×
7	９, 回, ら, きょう, 花, マラソン, 大会, 参加, 人	×
8	震災, 影響, 失業, 人	×
9	タイ, チャオプラヤ, 川, 洪水, 死亡, 者	○
10	日本, 臨床, スポーツ, 医学, 会, 会員, 数	○
12	日本, 国家, 公務員, 数	×
13	四十七, 回, 青少年, 航空, 教室, 参加, 人, 数	×
14	ポケベル, 所有, 者, 数	×
16	センター, 試験, 受験, 者, 数	×
20	今年, 成人, 式, 人, 数	×
21	特殊, 法人, 大蔵省, 天下り, 人, 数	×
22	９４, 年, 消費, 者, 問題, 世論, 調査, 回答, 人, 数	×
23	ク, 製剤, 薬害, 訴訟, 患者, 数	×
24	沖縄, 慰霊, 日, 参加, 米, 軍, 退役, 軍人, 数	×
25	カバディ, 競技, 必要	×
26	インド, 爆弾, 事件, 逮捕, 人, 数	○

Table 5.4: 改良後のシステムでの検索結果

問	キーワード	判定
1	交通, 事故	×
2	タイタニック	○
3	将棋, 名人	○
4	市立, 園田, 小学校, 児童	×
5	日本人, ノーベル, 受賞	×
6	全日本, 学生, 音楽, コンクール, 予選, 通過	×
7	花, マラソン, 大会, 参加	○
8	震災, 影響, 失業	×
9	タイ, チャオプラヤ, 洪水, 死亡	○
10	日本, 臨床, スポーツ, 医学, 会員	○
12	日本, 国家, 公務員	×
13	青少年, 航空, 教室, 参加	○
14	ポケベル, 所有	×
16	センター, 試験, 受験	×
20	成人	×
21	特殊, 法人, 大蔵省, 天下り	○
22	消費, 問題, 世論, 調査, 回答	×
23	ク, 製剤, 薬害, 訴訟, 患者	○
24	沖縄, 慰霊, 日, 参加, 米, 軍, 退役, 軍人	×
25	カバディ, 競技, 必要	×
26	インド, 爆弾, 事件, 逮捕	○

Chapter 6

考察

今回の改良ではキーワードを抽出した際に不要と思われる単語を除外することにした。実験の結果からも分かるように21問中4問の正解から9問の正解まで精度が上がった。このことより、キーワードの数を多く用いることが良いことではないとわかり、改良の効果が確認できた。

また、スコアリング法に関しても、本研究の方法では9問中9問すべてが正しい解答を得ることができたことにより、配点の妥当性も確認できた。

次にQAシステムの問題点について考察する。本研究で用いた質問文は次の4タイプに分類される。

1. 記事中に正解が無いもの [5問]

2. 正解があるが間接的に書かれているもの [3問]

3. 正解があるが質問文より検索式の生成が困難なもの [9問]

4. 本研究のシステムで問題無く正解が得られるもの [9問]

1は正解が出せないのは当然である。2が正解を出せないのは解答候補を抜き出す際「～人」を機械的に抽出したためだと考えられる。これは検出された記事の意味を考慮にいれなくてはならない。例えば、問5の正解が含まれる記事には、

「京大と東大が各4人」というように書かれて、正解は「8人」である (答えの間接表現)。3が正解を出さないのは記事が検出されないためで、その原因はキーワードの生成である。例えば、問1の「1994」を「昨年」と換える場合である。これは本研究で作ったキーワード作成プログラムの改良が必要である (キーワードの変換法)。

適切なキーワードの作成のみでどの程度うまく検索できるかを調べたが、それだけではQAシステムの実現は難しい。QAシステムには少なくとも質問文の曖昧性と答えの粒度の問題がある。例えば、問1では事故現場の範囲が明示されていないし、問3ではいつの時点での「これまで」なのかがわからない。ここでの実験では全てシステムの都合の良いように解釈して判定している。また答えの粒度の問題は、本実験の設定では人数を問うものに限定しているため顕在化していないが、例えば場所を質問された場合、極端に言えば「地球」という答えで済んでしまう。このようなQAシステムの持つ問題を整理することも重要であり、今後適切なキーワードの作成法とともに考察していきたい。

Chapter 7

おわりに

本研究では独自のQAシステムを開発し実装させた。このQAシステムの特徴は、検索部分に全文検索ソフトを用い、結果の表示にJAVAの表示系プログラムを用いたところである。精度は、記事中に答えが一単語であるものならば50％で正解を含む文書が検出でき、その文書より解答のスコアリングを行うとすべてにおいて正しい解答を得ることができた。

また、本研究を通じてQAシステムの問題点として、

- 答えの間接表現
- キーワードの変換法
- 質問文の曖昧性
- 答えの粒度

があることが整理できた。

Chapter 8

謝辞

本研究の遂行及び論文の作成に多大な御助言及び指導を賜った 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、本研究で利用した文書データは、毎日新聞 CD-ROM'95 版です。利用を許可していただいた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助言、協力を頂きました、同研究室の 池谷 昌紀 氏 (茨城大学大学院修士課程)、徳永崇 氏 (茨城大学工学部システム工学科 4 回生)、高橋 篤史 氏 (茨城大学工学部システム工学科 4 回生) にも深く感謝します。

Bibliography

[1] 賀沢秀人 加藤桓昭 : "意味制約を用いた日本語質問応答システム", 情報処理学会自然言語処理研究会,NL-140-24 (2000).

[2] 田中穂積 (編) : "自然言語処理システムに関する調査報告書", (社) 日本電子工業振興協会, 自然言語処理技術委員会 (2000).

[3] 永田昌明 : "自然言語処理は機械学習の実験場になる？", 情報論的学習理論ワークショップ, pp.207–214(2000).

[4] 高林哲 : "Namazu:全文検索で文書の山に立ち向かう", 情報処理 vol.41 No.11, pp.1227–1232(2000).

[5] 馬場肇 : "日本語全文検索システムの構築と活用", ソフトバンク株式会社 出版事業部 (1998).

[6] Sun Microsystems,Inc : "JAVA プログラミング講座", 株式会社アスキー, (1996).

Appendix A

プログラムソースリスト

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

1年分の新聞記事を1ヵ月ごとに分割する bunkatu1.c

  =======================================================*/
#include<stdio.h>
#include<stdlib.h>

int main()
{
  FILE *fin,*fout;
  char ss[256];
  char kugiri1[]="<DOCNO>950201001</DOCNO>\n\0";
  char kugiri2[]="<DOCNO>950301001</DOCNO>\n\0";
  char kugiri3[]="<DOCNO>950401001</DOCNO>\n\0";
  char kugiri4[]="<DOCNO>950501001</DOCNO>\n\0";
  char kugiri5[]="<DOCNO>950601001</DOCNO>\n\0";
```

```
char kugiri6[]="<DOCNO>950701001</DOCNO>\n\0";
char kugiri7[]="<DOCNO>950801001</DOCNO>\n\0";
char kugiri8[]="<DOCNO>950901001</DOCNO>\n\0";
char kugiri9[]="<DOCNO>951001001</DOCNO>\n\0";
char kugiri10[]="<DOCNO>951101001</DOCNO>\n\0";
char kugiri11[]="<DOCNO>951201001</DOCNO>\n\0";


if((fin = fopen("mai95.sgml","r")) == NULL){
  printf("input error\n");
  exit(1);
}


  if((fout = fopen("d95-01","w")) == NULL){
    printf("output error\n");
    exit(1);
  }
    while(fgets(ss,256,fin)!= NULL){
    fputs(ss,fout);
    if(strcmp(ss,kugiri1) == 0)break;
  }
  fclose(fout);


  if((fout = fopen("d95-02","w")) == NULL){
    printf("output error\n");
```

```
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri2) == 0)break;
}
fclose(fout);


if((fout = fopen("d95-03","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri3) == 0)break;
}
fclose(fout);


if((fout = fopen("d95-04","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri4) == 0)break;
```

```
}
fclose(fout);


if((fout = fopen("d95-05","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri5) == 0)break;
}
fclose(fout);


if((fout = fopen("d95-06","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri6) == 0)break;
}
fclose(fout);


if((fout = fopen("d95-07","w")) == NULL){
```

```
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri7) == 0)break;
}
fclose(fout);


if((fout = fopen("d95-08","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri8) == 0)break;
}
fclose(fout);


if((fout = fopen("d95-09","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
```

```
  if(strcmp(ss,kugiri9) == 0)break;
}
fclose(fout);


if((fout = fopen("d95-10","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri10) == 0)break;
}
fclose(fout);

if((fout = fopen("d95-11","w")) == NULL){
  printf("output error\n");
  exit(1);
}
while(fgets(ss,256,fin)!= NULL){
  fputs(ss,fout);
  if(strcmp(ss,kugiri11) == 0)break;
}
fclose(fout);


if((fout = fopen("95-12","w")) == NULL){
```

```
      printf("output error\n");
      exit(1);
    }
    while(fgets(ss,256,fin)!= NULL){
      fputs(ss,fout);
    }
    fclose(fout);

  fclose(fin);

  return(0);

}

/*============================================================

1ヵ月ごとの記事を 1 記事ごとに分割する bunkasu.c

  ============================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main()
{
  FILE *fin,*fout;
```

```
char ss[256];
char kugiri[]="</DOC>\n\0";
char name[20];
int i=0;

if((fin = fopen("d95-10","r")) == NULL){
  printf("input error\n");
  exit(1);
}

while(!feof(fin)){

  fseek(fin,14L,SEEK_CUR);
  fscanf(fin,"%9c",name);
  if(i==0)strcat(name,"y");
  i++;

  if((fout = fopen(name,"w")) == NULL){
    printf("output error\n");
    exit(1);
  }

    while(fgets(ss,256,fin)!= NULL){
    fputs(ss,fout);
    if(strcmp(ss,kugiri) == 0)break;
  }
```

```
    fclose(fout);

  }
  fclose(fin);

  return(0);

}

/*=======================================================

名詞のみの検索式を作成する meisi.c

  =======================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main(int argc,char *argv[])
{
  FILE *f1,*f2;
  char buf[256];
  int r,i=0;

  char key0[]="名詞";
  char key1[]="未知語";
```

```
  char nkey0[]="何";
  char nkey1[]="名詞-接尾-助数詞";

  if(argc!=2){
    printf("argc=%d\n",argc);
    printf("引数が違う\n\a");
    exit(1);
  }

  if((f1=fopen(argv[1],"r"))==NULL){
    printf("File \"%s\" cannot open!! \n\a",argv[1]);
    exit(1);
  }

  if((f2=fopen("ccc","w"))==NULL){
    printf("File \"%s\" cannot open!! \n\a",argv[1]);
    exit(1);
  }

  while(fgets(buf,256,f1)!=NULL){
    if(strstr(buf,key0)!=NULL || strstr(buf,key1)!=NULL){

      while(buf[i]!='\t'){
if(strstr(buf,nkey0)!=NULL || strstr(buf,nkey1)!=NULL);
else fputc(buf[i],f2);
i++;
      }
```

```
      fprintf(f2," ");
      i=0;
    }
  }

  fprintf(f2,"\n");

  if((r=fclose(f1))==-1) exit(1);
  if((r=fclose(f2))==-1) exit(1);
  return(0);
}

/*============================================================

meisi.c を改良した検索式作成プログラム seisei.c

  ============================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main(int argc,char *argv[])
{
  FILE *f1,*f2;
  char buf[256];
  int r,i=0;
```

```
char key0[]="名詞";
char key1[]="未知語";

char nkey0[]="何";
char nkey1[]="名詞-接尾";
char nkey2[]="名詞-副詞可能";
char nkey3[]="名詞-代名詞-一般";
char nkey4[]="名詞-数";
char nkey5[]="ヒト";
char nkey6[]="カズ";
char nkey7[]="合計";

if(argc!=2){
  printf("argc=%d\n",argc);
  printf("引数が違う\n\a");
  exit(1);
}


if((f1=fopen(argv[1],"r"))==NULL){
  printf("File \"%s\" cannot open!! \n\a",argv[1]);
  exit(1);
}

if((f2=fopen("ccc","w"))==NULL){
  printf("File \"ccc\" cannot open!! \n\a");
  exit(1);
```

```
  }

  while(fgets(buf,256,f1)!=NULL){
    if(strstr(buf,key0)!=NULL || strstr(buf,key1)!=NULL){
      if(strstr(buf,nkey0)!=NULL || strstr(buf,nkey1)!=NULL ||
         strstr(buf,nkey2)!=NULL || strstr(buf,nkey3)!=NULL ||
         strstr(buf,nkey4)!=NULL || strstr(buf,nkey5)!=NULL ||
         strstr(buf,nkey6)!=NULL || strstr(buf,nkey7)!=NULL);
      else {
while(buf[i]!='\t'){
  fputc(buf[i],f2);
  i++;
}
fprintf(f2,"\n");
i=0;
      }
    }
  }

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

  if((r=fclose(f2))==-1) exit(1);

  return(0);
}
```

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

shell プログラム kanzentai.sh1 を作成する watasu1.c

  =======================================================*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int  main()
{
  FILE *f1,*f2;
  char buf[256];
  int r,i;

  if((f1 = fopen("ccc","r"))==NULL){
    printf("File \"ccc\" cannot open!! \n\a");
    exit(1);
  }

  if((f2 = fopen("kanzentai.sh1","w"))==NULL){
    printf("File \"kanzentai.sh1\" cannot open!! \n\a");
    exit(1);
  }

  fprintf(f2,"#!/bin/bash\n");
```

```
  fprintf(f2,"namazu -o zzz -n 5 -s -h \"");

  while(fgets(buf,256,f1)!=NULL){
    for(i=0;i<(strlen(buf)-1);i++)fputc(buf[i],f2);
    fprintf(f2," ");
  }

  fprintf(f2,"\" /usr2/hoshi/index\n");
  fprintf(f2,"kensaku\n");

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

  return(0);
}

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

検索された記事名を出力する kensaku.c

```
  =======================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main()
{
  FILE *f1,*f2;
```

```
  char buf[256];
  int r,i;

  char key0[]="<DT>";

  if((f1=fopen("zzz","r"))==NULL){
    printf("File \"zzz\" cannot open!! \n\a");
    exit(1);
  }

  if((f2=fopen("xxx","w"))==NULL){
    printf("File \"xxx\" cannot open!! \n\a");
    exit(1);
  }

  while(fgets(buf,256,f1)!=NULL){
    if(strstr(buf,key0)!=NULL){
      for(i=24;i<59;i++)
fputc(buf[i],f2);
      fputs("\n",f2);
    }
  }

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

  if((r=fclose(f2))==-1) exit(1);
```

```
  return(0);
}

/*=========================================================

記事を１文ごとに分ける bun.c

  =========================================================*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int main(int argc,char *argv[])
{
  FILE *fin,*fout;
  char buf[512];
  char a[10];
  int i;

  strcpy(a,"。");

  if(argc != 2){
    printf("argc=%d\n",argc);
    printf("引数が違う。\n\a");
    exit(1);
  }
```

```
  if((fin = fopen(argv[1],"r"))==NULL){
    printf("File cannot open!! \n");
    exit(1);
  }

  if((fout = fopen("fff","w"))==NULL){
    printf("File cannot open!! \n");
    exit(1);
  }

  while((fgets(buf,512,fin))!=NULL){
    for( i=0;i<strlen(buf)-2;i+=2){
      if((buf[i]==a[0])&&(buf[i+1]==a[1]))
fprintf(fout,"。\n");
      else
fprintf(fout,"%c%c",buf[i],buf[i+1]);
    }
  }

  fclose(fout);
  fclose(fin);
  return(0);
}
```

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

解答候補を抜き出す kouho.c

  ============================================================*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main()
{
  FILE *f1,*f2;
  char buf[256],mae[256];
  int r,i=0;

  char key0[]="名詞-数";
  char key1[]="名詞-接尾-助数詞";

  char hito[]="ニン";

  if((f1=fopen("ddd","r"))==NULL){
    printf("File cannot open!! \n\a");
    exit(1);
  }

  if((f2=fopen("eee","w"))==NULL){
    printf("File cannot open!! \n\a");
```

```
    exit(1);
  }


  while(fgets(buf,256,f1)!=NULL){
    if(strstr(buf,key0)!=NULL)strcpy(mae,buf);
    if(strstr(buf,key1)!=NULL && strstr(buf,hito)!=NULL){
      while(mae[i]!='\t'){
fputc(mae[i],f2);
i++;
      }
      i=0;
      while(buf[i]!='\t'){
fputc(buf[i],f2);
i++;
      }
      i=0;
      fprintf(f2,"\n");
    }
  }


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


  if((r=fclose(f2))==-1) exit(1);


  return(0);
}
```

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

スコアリングを行なう hyou.c

  ============================================================*/
#include<stdio.h>
#include<string.h>

int main(int argc,char *argv[])
{
  FILE *fin1,*fin2,*fin3,*fin4,*fin5;
  char a[64],key[64][64],b[64],ans[64][64],kiji[512];
  int i=0,j=0,k=0,r,kCOUNT=0,aCOUNT=0;
  int kgyou=0,agyou=0,gyou=1,point[64],kkazu[64],kazu=0;

  if(argc != 2){
    printf("argc=%d\n",argc);
    printf("引数が違う。\n\a");
    exit(1);
  }

  /*キーワードを 二次元配列 key に読み込む*/
  if((fin1=fopen("ccc","r"))==NULL){
    printf("File ccc cannot open!!\n\a");
    exit(1);
  }
```

```
while(fgets(a,64,fin1)!=NULL){
  strncpy(key[i],a,strlen(a)-1);
  key[i][strlen(a)-1]='\0';
  i++;
  kCOUNT++;
}
if((r=fclose(fin1))==-1) exit(1);
```

```
/*答えの候補を二次元配列 ans に読み込む*/
if((fin2=fopen("eee","r"))==NULL){
  printf("File eee cannot open!!\n\a");
  exit(1);
}
while(fgets(b,64,fin2)!=NULL){
  strncpy(ans[j],b,strlen(b)-1);
  ans[j][strlen(b)-1]='\0';
  j++;
  aCOUNT++;
}
if((r=fclose(fin2))==-1) exit(1);
```

```
/*距離をはかる*/
```

```
  for(j=0;j<aCOUNT;j++){
    for(i=0;i<kCOUNT;i++){

      /*同じ文*/
      if((fin3=fopen("fff","r"))==NULL){
printf("BUN cannot open!!\n\a");
exit(1);
      }
      while(fgets(kiji,512,fin3)!=NULL){
if(strstr(kiji,ans[j])!=NULL){
  agyou=gyou;
}
if(strstr(kiji,key[i])!=NULL){
  kgyou=gyou;
  if(agyou==kgyou){
    //     printf("同じ文の中にあります。\n");
    point[k]=30;
  }
}
gyou++;
      }
      if((r=fclose(fin3))==-1) exit(1);


      /*同じ段落*/
      gyou=1;
      if((fin4=fopen(argv[1],"r"))==NULL){
```

```
printf("KIJI cannot open!!\n\a");
exit(1);
        }
        while(fgets(kiji,512,fin4)!=NULL){
if(strstr(kiji,ans[j])!=NULL){
  agyou=gyou;
}
if(strstr(kiji,key[i])!=NULL){
  kgyou=gyou;
  if(agyou==kgyou){
    //printf("同じ段落内にあります。\n");
    if(point[k]!=30)point[k]=20;
  }
}
gyou++;
        }
        if((r=fclose(fin4))==-1) exit(1);


        /*同じ文書*/
        if(agyou!=0 && kgyou!=0){
if(point[k]!=20 && point[k]!=30)point[k]=10;
        }

        /*ない*/
        if(point[k]!=10 && point[k]!=20 && point[k]!=30)point[k]=0;
        k++;
```

```
  }
}


/*キーワードの数を数える*/
for(i=0;i<kCOUNT;i++){
  kazu=0;
  if((fin5=fopen("ddd","r"))==NULL){
    printf("cha cannot open!!\n\a");
    exit(1);
  }
  while(fgets(kiji,512,fin5)!=NULL){
    if(strstr(kiji,key[i])!=NULL)kazu++;
  }
  kkazu[i]=kazu;
  if((r=fclose(fin5))==-1) exit(1);
}


/*結果*/
k=0;
for(j=0;j<aCOUNT;j++){
  for(i=0;i<kCOUNT;i++){
    printf("%d___%s___%s\n",point[k]+kkazu[i],ans[j],argv[1]);
```

```
      k++;
    }
  }

  return(0);
}

/*=========================================================

shell プログラム kanzentai.sh2 を作成する watasu2.c

  =========================================================*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int  main()
{
  FILE *f1,*f2;
  char buf[256];
  int r,i;

  if((f1 = fopen("xxx","r"))==NULL){
    printf("File \"ccc\" cannot open!! \n\a");
    exit(1);
  }
```

```
  if((f2 = fopen("kanzentai.sh2","w"))==NULL){
    printf("File \"kanzentai.sh2\" cannot open!! \n\a");
    exit(1);
  }

  fprintf(f2,"#!/bin/bash\n");
  fprintf(f2,"kanzentai.sh3 ");

  while(fgets(buf,256,f1)!=NULL){
    for(i=0;i<(strlen(buf)-1);i++)fputc(buf[i],f2);
    fprintf(f2," ");
  }

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

  return(0);
}

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

検索結果の解析系を 1 コマンド化した kanzentai.sh3

```
  =========================================================*/
#!/bin/bash

for kijifile
```

```
do
    chasen -o ddd $kijifile
    kouho
    bun $kijifile
    hyou $kijifile
done

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

結果のソート以前を 1 コマンド化した kanzentai.sh

```
  ==========================================================*/
#!/bin/bash

chasen -o bbb aaa
seisei bbb
watasu1
chmod a+x kanzentai.sh1
kanzentai.sh1
watasu2
chmod a+x kanzentai.sh2
kanzentai.sh2
rm bbb
rm ccc
rm ddd
rm eee
rm fff
```

```
rm xxx
rm zzz
rm kanzentai.sh1
rm kanzentai.sh2

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

解答候補のソートをする tori.c

```
  ============================================================*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int main()
{
  FILE *f1;
  char buf[256],gomi1[256][256],moto[256][256],yobi[256];
  int r,a=0,b=0,gomi2[256],max,s,t,i,j;

  if((f1 = fopen("text","r"))==NULL){
    printf("File \"text\" cannot open!! \n\a");
    exit(1);
  }

  while(fgets(buf,256,f1)!=NULL){
```

```
    strcpy(moto[a],buf);
    strncpy(gomi1[a],buf,2);
    gomi1[a][2]='\0';
    a++;
  }
  if((r=fclose(f1))==-1)exit(1);

  b=a;
  for(a=0;a<b;a++){
    gomi2[a]=atoi(gomi1[a]);
  }

  gomi2[b]='\0';

  for(i=0;i<b-1;i++){
    max=gomi2[i];
    s=i;
    for(j=i+1;j<b;j++){
      if(gomi2[j]>max){
max=gomi2[j];
s=j;
      }
    }
    t=gomi2[i];gomi2[i]=gomi2[s];gomi2[s]=t;
    strcpy(yobi,moto[i]);strcpy(moto[i],moto[s]);strcpy(moto[s],yobi);
  }
```

```
   if(b>5)b=5;
  for(i=0;i<b;i++)
    printf("%s",moto[i]);
  return(0);
}

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

全てを 1 コマンド化した hoshi.sh

```
  =========================================================*/
#!/bin/bash

kanzentai.sh >text
tori
rm text
rm aaa

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

JAVA の表示系プログラム

```
  =========================================================*/
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;
```

```
public class Tako extends Frame implements ActionListener{
    JTextArea jta;
    JTextField jtf;
    Button but1,but2;
    List lis;
    public static void main(String args[]){
Tako f=new Tako();
f.start();
    }

    public Tako(){
super("QA システム");
Menu ex=new Menu("exit");
ex.add(new MenuItem("exit"));
ex.addActionListener(this);
MenuBar mb=new MenuBar();
mb.add(ex);
setMenuBar(mb);
Panel ue=new Panel();
ue.setLayout(new BorderLayout());
Panel ueue=new Panel();
ueue.setLayout(new GridLayout(3,1));
Label lab=new Label("質問文を入力してください。(?は付けないでね)");
ueue.add(lab);
jtf=new JTextField();
ueue.add(jtf);
```

```
Panel botan=new Panel();
botan.setLayout(new GridLayout(1,2));
but1=new Button("ok");
but1.addActionListener(this);
botan.add(but1);
but2=new Button("cansel");
but2.addActionListener(this);
botan.add(but2);
ueue.add(botan);
ue.add("North",ueue);
lis=new List(6,false);
lis.addActionListener(this);
ue.add("Center",lis);
Label moji=new Label("リストをダブルクリックで記事表示");
ue.add("South",moji);
setLayout(new BorderLayout());
jta=new JTextArea();
jta.setLineWrap(true);
jta.setEditable(false);
JScrollPane jsp=new JScrollPane(jta);
add("North",ue);
add("Center",jsp);
    }

    public void start(){
setSize(500,500);
setBackground(Color.red);
```

```
show();
    }

    public void actionPerformed(ActionEvent ae){
if("exit".equals(ae.getActionCommand())){
    dispose();
    System.exit(0);
}
else if("ok".equals(ae.getActionCommand())){
    String s=jtf.getText();
    try{
FileOutputStream fos=new FileOutputStream("aaa");
PrintStream ps=new PrintStream(fos);
ps.println(s);
ps.close();
fos.close();
    }catch(Exception e){
System.err.println("だめ");
    }
    try{
Process process=Runtime.getRuntime().exec("hoshi.sh");
InputStream is=process.getInputStream();
BufferedReader br=new BufferedReader(new InputStreamReader(is));
String line;
lis.removeAll();
lis.add("ポイント___人数___記事名\n");
while((line=br.readLine())!=null){
```

```
    lis.add(line+"\n");
}
    }catch(Exception e){
System.err.println("error だよ");
    }
}
else if("cansel".equals(ae.getActionCommand())){
    jtf.setText("");
    jta.setText("");
    lis.removeAll();
}
else if(lis.getSelectedIndex()!=0){
    String s1=lis.getSelectedItem();
    String s2=s1.substring(s1.lastIndexOf('_')+1,s1.length()-1);
    try{
Process process=Runtime.getRuntime().exec("cat "+s2);
InputStream is=process.getInputStream();
BufferedReader br=new BufferedReader(new InputStreamReader(is));
String line;
jta.setText("");
while((line=br.readLine())!=null){
    jta.append(line+"\n");
}
    }catch(Exception e){
System.err.println("error だよ");
    }
```

```
}
    }
}

/*==========================================================

まとめ QAsystem

  ==========================================================*/

#!/bin/bash

java Tako
```