IRツール利用説明書

1998年2月11日

土屋 雅稔
京都大学情報学研究科
言語メディア研究室
tsuchiya@pine.kuee.kyoto-u.ac.jp

関根 聡
Department of Computer Science
New York University
sekine@cs.nyu.edu

1 概要

この文書は、IREX配布パッケージに含まれている、IR関連のツールのインストールと利用方法について説明した文書です。

本文書の対象となるIR関連のツールは、2種類に大きく分類されます。

ファイル変換

IRで必要となる4種のファイル形式の変換を行ないます。全てPerl[1]で記述されていて、普通にシェルから呼び出して使用します。

- `irex-output-merge`
  IR結果提出ファイル(複数)→IR結果統合ファイル(複数)
- `irex-merge-merge`
  IR結果統合ファイル(複数)→IR結果統合ファイル(単一)
- `irex-kekka-trec`
  IR結果提出ファイル(複数)→TREC結果提出ファイル(複数)
- `trec-kekka-irex`
  TREC結果提出ファイル(複数)→IR結果提出ファイル(複数)
- `irex-seikai-trec`
  IR結果統合ファイル(複数)→TREC正解ファイル(単一)
- `trec-seikai-irex`
  TREC正解ファイル(複数)→IR結果統合ファイル(単一)

記事の関連性を判定するためのツール `irex-judge`

システムの検索結果が検索課題と関連しているか人手で判定を行なう必要がありますが、その作業を支援するためのツールです。Emacs-Lispで記述されていて、Mule[2]上で動作します。

[1] Perl5が必要です。Perl4では動作しません。jperlは特に必要ありません。
[2] Mule-2.2, Mule-2.3およびEmacs-20.3での動作を確認済みです。

IR 結果統合ファイルを閲覧するためのツール `irex-browse`

IR 結果統合ファイルに含まれている判定情報とシステム情報を閲覧すらためのツールです。Emacs-Lisp で記述されていて、Mule[3] 上で動作します。

IR 関連のツールが扱うファイル形式は 4 種類あります。

IR 結果提出ファイル (`*.irt`)

システム毎の検索結果を記述した SGML 形式のファイル

IR 結果統合ファイル (`*.irj`)

各システムが提出した検索結果とそれらの検索結果と検索課題の関連性の判定を SGML 形式で保存したファイル

TREC 結果提出ファイル (`*.trt`)

システム毎の検索結果を記述した行単位のデータ構造を持つファイル

TREC 正解ファイル (`*.trj`)

検索結果のそれぞれの記事に対して正解/不正解の 2 値を与えたデータを行単位に保存したファイル

前半の 2 つが IREX の独自形式であり、後半の 2 つが TREC で採用されたファイル形式です。NACSIS コレクション・ワークショップでも TREC 形式が採用されていますので、`irex-kekka-trec` や `trec-kekka-irex` などを用いて相互変換することが出来ます。

[3] Mule-2.2, Mule-2.3 および Emacs-20.3 での動作を確認済みです。

2 インストール

本節では、前節で説明したツール群をインストールする手順について説明します。

インストールしたいディレクトリに配布パッケージを展開すると、次のようなディレクトリツリーが得られます。IR 関連の全てのツールは、`IREX/`を基準ディレクトリとして設計されています[4]。

```
IREX/IR_TOOLS/
IREX/IR_TOOLS/bin/
IREX/IR_TOOLS/lisp/    Emacs–Lisp の source code
IREX/IR_TOOLS/perl/    Perl スクリプト
```

展開後、次のように`IREX/IR_TOOLS/`の下に移動して`make`を実行して下さい。

```
% cd IREX/IR_TOOLS/
% make
```

これによって以下の処理が行なわれます。

1. Emacs–Lisp の source を byte compile する。
2. Perl スクリプトの先頭の Perl のパスをシステムにあわせて変更する。

`irex-judge`および`irex-browse`を利用する場合は、毎日新聞の記事テキストを`mai2sgml`によって SGML 形式の 2 つのファイル`mai94.sgml`, `mai95.sgml`に変換し、それらのファイルが格納されているディレクトリを変数 *irex-mainichi-directory* に設定しておく必要があります。例えば、それらのファイルが`/usr/local/share/mainichi/`の下に置いてある場合には次のように指定して下さい。

```
(setq irex-mainichi-directory "/usr/local/share/mainichi")
```

特に指定しなかった場合は、`$IREX_ROOT/MAINICHI/`に 2 つのファイルがあることを仮定するので、その場所にシンボリックリンクを作っても良いでしょう。なお、環境変数 *IREX_ROOT* については 3.1節を参照して下さい。

```
% mkdir $IREX_ROOT/MAINICHI
% cd $IREX_ROOT/MAINICHI
% ln -s /usr/local/share/mainichi/mai94.sgml
% ln -s /usr/local/share/mainichi/mai95.sgml
```

[4] つまり、配布パッケージ自体を展開するディレクトリは自由に選べますが、`IR_TOOLS/`以下のディレクトリ構造については変更しないで下さい。

3 個人環境の設定

本節では、IR 関連のコマンドを利用するために必要な個人環境の設定について説明します。

3.1 環境変数 *IREX_ROOT* の設定

各コマンドは単体では動作せず、Perl や Emacs-Lisp の共有ライブラリを必要とします。環境変数 *IREX_ROOT* は、それらのライブラリの場所を指定するための環境変数で、全てのコマンドはこの環境変数によって指定されたディレクトリを基準として動作するように設計されています。`/usr/share/`以下に配布パッケージを展開した場合には次の設定を、`.cshrc` などに加えて下さい[5]。

```
setenv IREX_ROOT /usr/share/IREX
```

3.2 Mule の設定

通常はサンプルの設定をそのまま利用すれば良いので、次のコマンドを実行して`.emacs`を書き換えて下さい。

```
% cat $IREX_ROOT/IR_TOOLS/lisp/sample.emacs >>$HOME/.emacs
```

サンプル設定をそのまま利用した場合、以下のような設定が行なわれます。

1. 必要なライブラリを検索するためのパスを *load-path* に追加
2. `M-x irex-judge` を実行した時に自動的に必要なライブラリを load する設定 (autoload) を追加

場合によっては、以下の変数を設定する必要があるかもしれません。

irex-judge-default-pid
: 判定者の PID。この変数を設定しておくと、`irex-judge` 起動時に PID を質問しないようになります。従って、毎回入力したくない場合は、この変数を次のように設定しておいてください。

```
(setq irex-judge-default-pid "123")
```

irex-article-highlight-style

irex-article-highlight-color
: この 2 つの変数によって、記事本文中の強調表示する単語の見え方を制御しています。初期状態では青色に強調表示されるようになっていますが、カラー表示の出来ない状態で Mule を使用している場合はこの変数を例えば次のように変更してください。

[5] `tcsh` 以外のシェルを利用している場合は、適宜読みかえて下さい。

```
(setq irex-article-highlight-style 'underline)
```

なお、実行環境によっては、コマンドラインに一度に指定できる引数の長さに制限があるために、正常に irex-judge が実行できない場合があります[6]。その場合は、以下の設定を.emacs に追加してください。

```
(setq irex-mainichi-max-option-length 4096)
```

[6]AIX 4.2 では、この問題が発生することが確認されています。

4 マニュアル

4.1 irex-output-merge

irex-output-merge は、複数の結果提出ファイルをまとめて結果統合ファイルに変換する Perl スクリプトです。

使用例

最も典型的な使用例は次のようになります。

```
% irex-output-merge -f prefix file_1 file_2 ··· file_n
```

例えば、結果提出ファイル file_1, file_2,...,file_n に、TOPIC-ID が 0001 ～ 0003 のデータが含まれている場合、このコマンドの結果として 3 つの結果統合ファイルが生成されます。

```
prefix_0001.irj
prefix_0002.irj
prefix_0003.irj
```

また、入力ファイルを指定しなかった場合は、標準入力から読み込みます。従って、上記のコマンドと次のコマンドは同じ意味になります。

```
% cat file_1 file_2 ··· file_n | irex-output-merge -f prefix
```

処理結果を標準出力に出力したい場合は、-n オプションを指定してください。ただし、この場合は、TOPIC-ID が異なるデータも全てまとめて標準出力に出力されてしまいます。

```
% irex-output-merge -n file_1 file_2 ··· file_n
```

オプション一覧

-f *prefix* / --prefix *prefix*
: 生成する結果統合ファイルのファイル名の先頭部を指定するオプション

-n / --debug
: 処理した結果を標準出力に出力するオプション

-q / --quiet
: 処理の途中経過の報告を抑制するオプション

-h / -? / --help
: 簡易ヘルプを表示するオプション

4.2 irex-merge-merge

irex-merge-merge は、複数の結果統合ファイルをまとめて単一の結果統合ファイルに変換する Perl スクリプトです。

使用例

最も典型的な使用例は次のようになります。

```
% irex-merge-merge -f file file_1 file_2 ··· file_n
```

このコマンドは、結果統合ファイル file_1, file_2,...,file_n に含まれている全てのデータを統合した結果を file に出力します。irex-output-merge と同様、入力ファイルが指定されなかった場合は標準入力から読み込みます。従って、上記のコマンドと次のコマンドは同じ意味になります。

```
% cat file_1 file_2 ··· file_n | irex-merge-merge -f file
```

また、出力ファイルを指定しなかった場合は、標準出力に結果を書き出します。

```
% irex-merge-merge file_1 file_2 ··· file_n >file
```

オプション一覧

`-f` *filename* / `--file` *filename*
: 生成する結果統合ファイルのファイル名を指定するオプション

`-q` / `--quiet`
: 処理の途中経過の報告を抑制するオプション

`-h` / `-?` / `--help`
: 簡易ヘルプを表示するオプション

4.3 irex-kekka-trec

irex-kekka-trec は、IR 結果提出ファイルを TREC 結果提出ファイルに変換するための Perl スクリプトです。

使用例

最も典型的な使用例は次のようになります。

```
irex-kekka-trec -f prefix file_1 file_2 ··· file_n
```

複数の IR 結果提出ファイル file_1,file_2,...,file_n に複数のシステムの情報が含まれている場合、システム毎にファイルを切替えてデータを出力します。例えば、SYSTEM-ID が 101∼103 の値をとる場合、このコマンドの結果として 3 つの TREC 結果提出ファイルが生成されます。

```
prefix_101.trt
prefix_102.trt
prefix_103.trt
```

また、入力ファイルを指定しなかった場合は、標準入力から読み込みます。従って、上記のコマンドと次のコマンドは同じ意味になります。

```
% cat file_1 file_2 ··· file_n | irex-kekka-trec -f prefix
```

処理結果を標準出力に出力したい場合は、-n オプションを指定してください。ただし、この場合は、システムが異なるデータも全てまとめて標準出力に出力されてしまいます。

```
% irex-kekka-trec -n file_1 file_2 ··· file_n
```

オプション一覧

-f *prefix* / --prefix *prefix*
: 生成する TREC 結果提出ファイルのファイル名の先頭部を指定するオプション

-n / --debug
: 処理した結果を標準出力に出力するオプション

-q / --quiet
: 処理の途中経過の報告を抑制するオプション

-h / -? / --help
: 簡易ヘルプを表示するオプション

4.4 trec-kekka-irex

trec-kekka-irex は、TREC 結果提出ファイルを IR 結果提出ファイルに変換するための Perl スクリプトです。

使用例

最も典型的な使用例は次のようになります。

```
trec-kekka-irex -f prefix file_1 file_2 ··· file_n
```

複数の TREC 結果提出ファイル file_1,file_2,...,file_n に複数のシステムの情報が含まれている場合、システム毎にファイルを切替えてデータを出力します。例えば、SYSTEM-ID が 101〜103 の値をとる場合、このコマンドの結果として 3 つの IR 結果提出ファイルが生成されます。

```
prefix_101.irt
prefix_102.irt
prefix_103.irt
```

また、入力ファイルを指定しなかった場合は、標準入力から読み込みます。従って、上記のコマンドと次のコマンドは同じ意味になります。

```
% cat file_1 file_2 ··· file_n | trec-kekka-irex -f prefix
```

処理結果を標準出力に出力したい場合は、-n オプションを指定してください。ただし、この場合は、システムが異なるデータも全てまとめて標準出力に出力されてしまいます。

```
% trec-kekka-irex -n file_1 file_2 ··· file_n
```

オプション一覧

-f *prefix* / --prefix *prefix*
: 生成する TREC 結果提出ファイルのファイル名の先頭部を指定するオプション

-n / --debug
: 処理した結果を標準出力に出力するオプション

-q / --quiet
: 処理の途中経過の報告を抑制するオプション

-h / -? / --help
: 簡易ヘルプを表示するオプション

4.5 irex-seikai-trec

irex-seikai-trec は、IR 結果統合ファイルに含まれている各記事の関連性の判定結果を正解/不正解の 2 値に変換して TREC 正解ファイルを生成する Perl スクリプトです。

使用例

一般に 1 つの記事の判定に関わっている判定者は複数ですので、その複数の結果を 2 値に変換するための適当な判定関数が必要になります。そのような関数として、irex-seikai-trec に内蔵されている 2 種類の判定関数か、ユーザーが定義した関数が利用できます。どの判定関数を利用するかは、コマンドラインからオプションで指定します。

最も簡単な使用例は次のようになります。

```
% irex-seikai-trec -f file -a file_1 file_2 ··· file_n
```

このコマンドは、複数の IR 結果統合ファイル file_1, file_2,...,file_n に含まれている全てのデータを統合した結果を TREC 正解ファイル file に出力します。-a は、内蔵判定関数の 1 つを利用することを指定するオプションです。この判定関数の返す値は次のようにして決定されます。

1. 判定する記事の判定に参加している判定者から、最大の PID を持つ判定者を選ぶ。
2. その判定者の判定が A または A?の場合を正解とする。それ以外は不正解とする。

つまり、上記コマンドによって正解つき IR 結果統合ファイルを TREC 正解ファイルに変換できます。

-p オプションによって判定者の PID を指定すると、-a オプションによって指定された内蔵判定関数の動作が少し変化します。

```
% irex-seikai-trec -f file -a -p 1234 file_1 file_2 ··· file_n
```

この場合、判定関数の返す値を決定する手順は次のようになります。

1. 指定された PID の判定者の判定が A または A?の場合を正解とする。
2. それ以外は不正解とする。その記事が指定された判定者によって判定されていない場合も含まれる。

-b オプションは、-a オプションとよく似ていますが、正解と判定する条件のみが異なっている内蔵判定関数を指定するオプションです。すなわち、

- -a オプションの場合 → A または A?の場合を正解とする
- -b オプションの場合 → A, A?, B または B?の場合を正解とする

という差があります。-p オプションの有無による動作の変化は同じです。従って、次のコマンドにより、IR 結果統合ファイルを TREC 部分正解ファイルに変換できます。

```
% irex-seikai-trec -f file -b file_1 file_2 ··· file_n
```

内蔵関数では不十分な場合は、-l オプションによって外部の判定関数定義ファイルを読み込むことが出来ます。定義ファイルの例は$IREX_ROOT/IR_TOOLS/perl/sample-logic.plにあります。このファイルには、ファイルの形式と判定関数の引数及び返り値についての説明も含まれています。殆んどの場合、このファイルを少し書き換えれば必要な判定関数が得られるでしょう。

```
% cp -p $IREX_ROOT/IR_TOOLS/perl/sample-logic.pl logic.pl
# logic.pl を適当に編集
% irex-seikai-trec -f file -l logic.pl file_1 file_2···
```

-l オプションによって指定されたファイルが相対パス名だった場合は、以下のディレクトリを順に検索します。

1. Current Directory
2. $IREX_ROOT/IR_TOOLS/perl
3. Perl5 の通常のライブラリ検索パス

-a, -b, -l のどれもが指定されていない場合、つまり全く判定関数が指定されていなかった場合は、-a オプションが指定されている時と同じ動作をします。

オプション一覧

-f *file* / --file *file*
: 出力ファイル名を指定するオプション

-l *func–file* / --logic *func–file*
: ユーザー個別の判定関数の定義ファイルを指定するオプション

-a / --aonly
: スコアが A または A?の場合を正解とする内蔵判定関数によって判定を行なうオプション

-b / --aorb / -a+b
: スコアが A, A?, B または B?の場合を正解とする内蔵判定関数によって判定を行なうオプション

-p *pid* / --pid *pid*
: 内蔵判定関数が参照する判定者を指定するオプション

-q / --quiet
: 処理の途中経過の報告を抑制するオプション

`-h` / `-?` / `--help`
: 簡易ヘルプを表示するオプション

4.6 trec-seikai-irex

trec-seikai-irex は、複数の TREC 正解ファイルをまとめて単一の IR 結果統合ファイルに変換する Perl スクリプトです。

使用例

最も典型的な使用例は次のようになります。

```
% trec-seikai-irex -f file -p 9000 file_1 file_2 ··· file_n
```

このコマンドは、TREC 正解ファイル file_1, file_2,...,file_n に含まれている全てのデータを統合した結果を IR 結果統合ファイル file に出力します。ただし、TREC 正解ファイルには判定者の PID の情報が含まれていないので、-p オプションによって指定する必要があります。-p オプションを省略した場合は、PID = 9999 を仮定しますので、次の 2 つのコマンドは同じ意味になります。

```
% trec-seikai-irex -f file -p 9999 file_1 file_2 ··· file_n% trec-seikai-irex
-f file file_1 file_2 ··· file_n
```

入力ファイルを指定しなかった場合は、標準入力から読み込みます。

```
% cat file_1 file_2 ··· file_n | trec-seikai-irex -f file
```

出力ファイルを指定しなかった場合は、標準出力に結果を書き出します。

```
% trec-seikai-irex file_1 file_2 ··· file_n >file
```

オプション一覧

-f *filename* / --file *filename*
: 出力ファイル名を指定するオプション

-p *pid* / --pid *pid*
: 入力の TREC 正解ファイルを作った判定者の PID を指定するオプション

-q / --quiet
: 処理の途中経過の報告を抑制するオプション

-h / -? / --help
: 簡易ヘルプを表示するオプション

4.7 irex-judge

irex-judge は、Mule 上で動作する IR の判定用のツールです。

使用例

判定者によって起動方法が若干異なります。

学生判定者の場合 (S-MODE)

Mule 上で判定しようとしている結果統合ファイルを開いてから、M-x irex-judge とします[7]。

上級判定者の場合 (E-MODE)

Mule 上で判定しようとしている結果統合ファイルを開いてから、C-u M-x irex-judge とします[8]。

irex-judge が起動すると、表 1のキーが利用可能となります。初めて起動する時は、毎日新聞全文記事データから必要な記事のみを取り出す必要があるので、結構時間 (数分) がかかります。取り出されたデータは、対象となっている IR 結果統合ファイルと同じディレクトリに保存されます。例えば、

```
$HOME/work/topic001.irj
```

という名前の IR 結果統合ファイルを対象として irex-judge を実行している場合は、

```
$HOME/work/topic001.mai
```

という名前のファイルに保存されます。2 回目の起動からは、このデータを使います。

起動したら数字キーを使って、記事の関連性を判定していきます。学生判定者の場合は 1 ～ 6 のキーを使い、上級判定者の場合は 1 ～ 3, 7 ～ 9 のキーを使って下さい。判定せずに前後の記事に移る場合は n,p などのキーが利用できます。また、Shift キーを押しながら n,p を押す[9]と判定済みの記事を飛ばして直前または直後の未判定の記事に移動します。

判定対象となっている記事は何らかの検索トピックに対してシステムが返した検索結果ですから、何らかのキーワードが共通して含まれていることが期待されます。irex-judge では判定を容易にするために、それらのキーワードを強調表示させることが出来ます。この機能を利用する場合は k を押して下さい。するとキーワード入力用のバッファが開きますので、指示に従って 1 つの行に 1 つずつ単語を入力して下さい。例えば、「自衛隊」「国連」「軍」という 3 つのキーワードを強調表示したい場合は、図 1のようになります。

学生判定者の場合、A?などのように疑問つきの判定をする場合がありますが、その場合は疑問点を記録しておくことが推奨されています。そのような場合は c を押して、コメント入力を開始します。別にコメント入力用のバッファが開きますので、指示にしたがって入力して下さい。上級判定者の場合も同様に c によってコメント入力できます。

[7] M-x irex-judge とは、ESC キーを押してから x を押し、続いて irex–judge と入力する操作のことです。
[8] C-u M-x irex-judge とは、Control キーを押しながら u を押し、続いて ESC キーを押してから x を押し、続いて irex–judge と入力する操作のことです。
[9] つまり、大文字の N,P を入力する

```
# ---- この行よりも下に入力して下さい ----
自衛隊
国連
軍
```

図 1: キーワード入力の例

```
# ---- この行よりも下に入力して下さい ----
これはコメント
この記事は部分的には関連しているような気がするのですが。
```

図 2: コメント入力の例

`irex-judge` を終了したくなったら、q を押して下さい。もし、あなたの判定作業によってバッファが変更されていた場合は、終了時にファイルに保存するかどうか確認します。普通は保存しておけば問題ないでしょう。

自動判定

`E-MODE` の場合、記事リスト表示バッファには他の判定者の判定結果が同時に表示されています。例えば、記事リスト表示バッファのある 1 行が次のような形式の場合について考えます。

```
1    A   A?    -      94/02/21     ［用語］太陽光発電
```

この時、この記事について A 判定を行なった判定者が「判定者 J1」であり、A?判定を行なった判定者が「判定者 J2」です。

`E-MODE` の場合、判定者の労力を削減するために他の判定者 J1,J2,... の判定結果を利用して自動的に判定させることができます。a を押すと新しいバッファが開くので、指示にしたがって自動判定用 S 式を入力して下さい。

例えば、「判定者 J1」の判定結果が A の時に t となるような S 式を作るためには、次のような cons cell を入力して下さい。

```
("J1" . "A")
```

S 式中では and や or, if, cond などの全ての特殊表現と関数を使うことが出来ます。従って、「判定者 J1 の判定したスコアが A であり、かつ、判定者 J2 の判定したスコアが A である時、その記事のスコアを A とする」という意味の S 式は次のようになります。

```
(and ("J1" . "A") ("J2" . "A") "A")
```

次の S 式は、「判定者 J1 と判定者 J2 がどちらも疑問を表明せずに判定結果が一致している場合は、その判定結果を採用する」という意味を持ちます。

```
(cond
 ((and ("J1" . "A") ("J2" . "A")) "A")
 ((and ("J1" . "B") ("J2" . "B")) "B")
 ((and ("J1" . "C") ("J2" . "C")) "C"))
```

さらに S 式中では以下の変数を利用できます。

docno	判定の対象となっている記事の記事番号
line	判定の対象となっている記事の表示順位 (記事リストバッファの左端に表示されている整数)
x,y,z	自由に利用できる一時変数

次の例は、表示順位によって他の判定者の判定結果の信用の仕方を変更する例です。

```
(cond
 ((and (< line 50) ("J1" . "A") ("J2" . "A")) "A")
 ((and (> line 200) ("J1" . "C") ("J2" . "C")) "C"))
```

この例では、50 位以上の記事について二人の判定者が一致して A 判定をしている記事は A とする、また 200 位以下の記事について二人の判定者が一致して C 判定をしている記事は C とするという意味になります。また、文字列中の小文字は全て大文字に変換されて評価されます。

判定	
1	記事の関連性の程度を A として、次の未判定記事に移動する
2	B として、次の未判定記事に移動する
3	C として、次の未判定記事に移動する
4	A?として、次の未判定記事に移動する (S-MODE のみ有効)
5	B?として、次の未判定記事に移動する (S-MODE のみ有効)
6	C?として、次の未判定記事に移動する (S-MODE のみ有効)
7	A として、次の記事に移動する (E-MODE のみ有効)
8	B として、次の記事に移動する (E-MODE のみ有効)
9	C として、次の記事に移動する (E-MODE のみ有効)
移動	
n	次の記事に進む
p	前の記事に戻る
l	まだ判定していない最初の記事に移動
g	指定された行に移動
s	人手によって判定した最後の記事または最後に s を押した記事に移動
>	最後の記事に移動
<	最初の記事に移動
その他	
a	自動判定用 S 式を入力 (E-MODE のみ有効)
c	コメントを入力
k	強調表示したい単語を入力
SPC	記事表示画面をスクロールアップする
b	記事表示画面をスクロールダウンする
?	ヘルプを表示
z	判定作業を中断する
q	判定作業を終了する

表 1: irex-judge 実行中に利用可能なキーとコマンド

4.8 irex-browse

irex-browse は、Mule 上で動作する IR 結果統合ファイルを閲覧するためのツールです。

使用例

Mule 上で判定しようとしている結果統合ファイルを開いてから、M-x irex-browse[10] と入力して、irex-browse を起動して下さい。

また、検索システムに関する情報を同時に表示させたい場合は、C-u M-x irex-browse[11] と入力して、前置引数を指定して irex-browse を起動して下さい。

irex-judge が起動すると、表 2のキーが利用可能となります。初めて起動する時は、毎日新聞全文記事データから必要な記事のみを取り出す必要があるので、結構時間 (数分) がかかります。取り出されたデータは、対象となっている IR 結果統合ファイルと同じディレクトリに保存されます。例えば、

```
$HOME/work/topic001.irj
```

という名前の IR 結果統合ファイルを対象として irex-judge を実行している場合は、

```
$HOME/work/topic001.mai
```

という名前のファイルに保存されます。2 回目の起動からは、このデータを使います。

閲覧している記事は何らかの検索トピックに対してシステムが返した検索結果ですから、何らかのキーワードが共通して含まれていることが期待されます。irex-browse では、それらのキーワードを強調表示させることが出来ます。この機能を利用する場合は k を押して下さい。するとキーワード入力用のバッファが開きますので、指示に従って 1 つの行に 1 つずつ単語を入力して下さい。例えば、「自衛隊」「国連」「軍」という 3 つのキーワードを強調表示したい場合は、図 3のようになります。

```
# ---- この行よりも下に入力して下さい ----
自衛隊
国連
軍
```

図 3: キーワード入力の例

[10] M-x irex-browse とは、ESC キーを押してから x を押し、続いて irex–browse と入力する操作のことです。

[11] C-u M-x irex-browse とは、Control キーを押しながら u を押し、続いて ESC キーを押してから x を押し、続いて irex–browse と入力する操作のことです。

移動	
n	次の記事に進む
p	前の記事に戻る
g	指定された行に移動
>	最後の記事に移動
<	最初の記事に移動
その他	
k	強調表示したい単語を入力
SPC	記事表示画面をスクロールアップする
b	記事表示画面をスクロールダウンする
?	ヘルプを表示
q	終了する

表 2: `irex-browse` 実行中に利用可能なキーとコマンド