

# 情報検索パッケージの実装

内山将夫 井佐原均  
通信総合研究所

情報検索のためのパッケージを実装した。このパッケージは、自然言語処理の分野において、質問応答などの、情報検索を前処理として使いたい場合、あるいは、情報検索の結果のみに興味があるときに使いたい場合に利用できる。このパッケージは、オブジェクト指向スクリプト言語 Ruby から容易に利用可能であり、検索精度は、ベースラインシステムとして十分なものである。また、このパッケージを利用することにより、関連語を得ることもできる。このパッケージは、

<http://www.crl.go.jp/jt/a132/members/mutiyama/software.html>

よりダウンロードできる。

## Implementation of an IR package

Masao Utiyama and Hitoshi Isahara  
Communications Research Laboratory

We have implemented an IR (Information Retrieval) package, which is intended to be used as a preprocessing tool for natural language processing tasks such as Q&A (Question and Answering) as well as an IR system by itself. The package is easily available from Ruby, an object-oriented scripting language, and its performance is high enough to be used as a baseline system for IR. The package is also used to get words that are relevant to a given word. The package is available at

<http://www.crl.go.jp/jt/a132/members/mutiyama/software.html>

## 1 はじめに

自然言語処理の研究分野において、情報検索を前処理として必要とするものがある。たとえば、質問応答において、データベース中の全文書を回答の探索対象とするのではなく、質問文から検索される関連度の高い文書のみを探索対象とすることが考えられる。また、情報抽出におけるイベント抽出パターンを自動獲得するために、あるイベントに該当する文書を検索したい場合がある (Sudo, Sekine, and Grishman 2001)。

情報検索をこのような目的のために使うとき、あるいは、情報検索の結果のみに興味があるときには、情報検索自体はツールとして扱いたい。そして、そのようなツールに必要な要件は、少なくとも以下のものである。

1. 高精度であること。
2. 柔軟であること。
3. それほど遅くないこと。

本稿で述べる情報検索のためのツールは、これらの要件を満たすことを意図して作成された。

まず、第1点を満たすためには、現時点で高精度であることが認められている手法を用いれば良い。そのような手法として、我々は、BM25(Robertson and Walker 2000)を用いた。また、第2点を満たすために、我々は、そのようなツールをスクリプト言語のライブラリとして提供することとした。更に、そのようなライブラリをC言語で実装することにより、ある程度の速度で検索ができることを意図した。そのため、我々は、オブジェクト指向スクリプト言語 Ruby の拡張ライブラリとして、情報検索パッケージを実装することにした。

このパッケージには、以下のような機能がある。

1. 質問文に適合するような文書を検索する機能
2. 単語についての統計量を求める機能

第1の機能は、通常の情報検索の機能である。第2の機能は、与えられた単語と関連の高い単語を得るなど、情報検索のための補助機能として必要な機能である。

以下では、2章において、情報検索のための機能を述べ、3章において、関連の高い単語などを求める機能を述べる。4章は、結論である。

## 2 情報検索

まず、質問が与えられたとき、文書に対するスコアを付与する関数について述べ、次に、文書中の単語を正規化する手法を述べ、更に、本パッケージの検索性能、および、本パッケージを使用したコードの例を述べる。

### 2.1 スコア関数

質問  $Q$  に対して、ある文書  $D$  のスコアは、(1) 式により示される、BM25 により計算される (Robertson and Walker 2000)。そして、このスコアにより文書を降順に並べたときの順位が、検索結果の文書の順位となる。

$$\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf}{K + tf} \frac{(k_3 + 1)qtf}{k_3 + qtf} \quad (1)$$

ただし,

$T$  は  $Q$  に含まれる単語である.

$w^{(1)}$  は, 以下の式で表される  $T$  の重みである.

$$\log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \quad (2)$$

$N$  は, 検索対象の文書集合における全文書数である.

$n$  は,  $T$  を含む文書の数である.

$R$  は,  $Q$  に適合する文書の数であり,  $r$  は,  $Q$  に適合する文書で, かつ,  $T$  を含む文書の数である. ただし, 適合する文書は, ユーザが指定したり, 情報検索システムが推測することにより定められる<sup>1</sup>.

$K$  は,  $k_1((1-b)+b\frac{dl}{avdl})$ .

ただし,  $k_1$ ,  $b$ ,  $k_3$  は経験的に定める定数である. これらの, 本パッケージでのデフォルト値は,  $k_1 = 1$ ,  $b = 1$ ,  $k_3 = 1000$  であり, 2.3節でも, この値が用いられた. また,  $dl$  は,  $D$  の長さであり,  $avdl$  は, 文書集合における文書の長さの平均値である. ただし, 文書の長さは, 本パッケージでは, その文書に含まれる単語の数のことである.

$tf$  は,  $D$  に含まれる  $T$  の数である.

$qtf$  は,  $Q$  に含まれる  $T$  の数である.

## 2.2 単語の正規化

単語の正規化は, 言語に依存した部分であり, 多言語検索の際には特に, 重要な研究の対象である (Oard, Levow, and Cabrezas 2000).

本パッケージでは, 日本語文書の検索に際しては, 茶筌 (松本, 北内, 山下, 平野, 松田, 高岡, 浅原 2001) を用い, 英語文書の検索に際しては, 文献 (Frakes and Baeza-Yates 1992) に記載されているプログラム<sup>2</sup>を用いることとし, それらへのインタフェースを作った.

まず, 日本語単語の正規化のためには, 茶筌の形態素解析の結果から, 主に名詞を抜き出し, 索引付けのための単語とした. 次に, 英語の単語の正規化のためには, 文献 (Frakes and Baeza-Yates 1992) のプログラムを利用して, ストップワードを除去し, Porter stemmer を適用した結果を索引付けのための単語とした.

<sup>1</sup>情報検索においては, (1) まず, ユーザが初期質問を情報検索システムに入力すると, (2) 次に, システムが, その質問に適合すると予測した文書を順位付けてユーザに提示する. (3a) このとき, 提示された文書の中から, ユーザが, 適合する文書を複数選んだとすると, その選んだ数が  $R$  となる. (3b) あるいは, 2.3節で使用している automatic feedback 手法のように, システムが, 単に, 上位  $R$  文書を適合文書とみなす場合もある. (4) このようにして, ユーザあるいはシステムにより適合文書が選ばれたとすると, 次に, この適合文書中の単語 (の一部) を初期質問に加えた質問を作り, その質問を利用して, システムは文書を検索する.

このような検索過程において, (1) の段階では,  $R$  や  $r$  の数は分からないので, 本パッケージでは, 便宜的に,  $R = r = 0$  としている. また, (3), (4) の段階では, システムまたはユーザにより選ばれた文書の数  $R$  とし, 選ばれた文書の中で,  $T$  を含む文書の数  $r$  としている.

<sup>2</sup><ftp://ftp.vt.edu/pub/reuse/IR.code/ir-code.tar.Z> からダウンロードできる.

## 2.3 検索性能

本パッケージを、IREX (IREX 実行委員会 1999), NTCIR2 (National Institute of Informatics 2001) の検索タスクに適用した結果の精度を表1に示す。ここで、IREXでは、毎日新聞94-95年の新聞記事に対しての30の日本語質問に対する検索結果についての R-Precision をとり、NTCIR2 JJ (NTCIR2 の日本語検索タスク) では、NTCIR2 の日本語データベースに対する49の日本語質問に対する検索結果に対しての Average-Precision をとり、NTCIR2 EE (NTCIR2 の英語検索タスク) では、NTCIR2 の英語データベースに対する49の英語質問に対する検索結果に対しての Average-Precision をとっている<sup>34</sup>。

表 1: 検索精度

|           | long   |        | short  |        | time    | 大きさ (Mb) |      |        |
|-----------|--------|--------|--------|--------|---------|----------|------|--------|
|           | this   | best   | this   | best   |         | テキストDB   | 索引   | 文書数    |
| IREX      | 0.4924 | 0.4929 |        |        | 1~5 秒   | 218      | 295  | 212555 |
| NTCIR2 JJ | 0.3968 | 0.4303 | 0.3353 | 0.3730 | 50~60 秒 | 815      | 1121 | 736166 |
| NTCIR2 EE | 0.3911 | 0.4043 | 0.3266 | 0.3193 | 3~6 秒   | 346      | 349  | 322058 |

表1において、「long」と「short」は、それぞれ、長めと短めの質問<sup>5</sup>ということであり、「this」とは、このパッケージの検索精度であり、「best」とは、それぞれの検索タスクにおける、そのタスクに参加したシステムの中での、最高精度である。空欄は、そのタスクにおける参加システムで、その長さの検索をしたものがないことを示す。

表1において、本パッケージの検索方法は、まず、与えられた質問文で検索し、その検索結果の上位5文献における全単語を質問文に追加して、再び検索するという (automatic feedback) 手法である。

表1における本パッケージの精度は、NTCIR2 EE の short を除いては、それぞれのタスクにおけるベストの検索精度に及ばないが、本パッケージの検索精度は、各タスクにおける精度が上位のシステムからなるグループに位置しているの、ベースラインシステムとしては十分であると考ええる。なお、各タスクにおいて、ベストのシステムは、NTCIR2 EE の short を除いては、BM25 をベースにしたスコア関数を使っているシステムであるので、本パッケージの精度が、それらのシステムにおよばないのは説明がつく。

なお、表1において、「time」は、一つの質問を処理するのに 750MHz/384Mb の PC で掛かった時間であり、「テキストDB」と「索引」の大きさは、それぞれの Mega Byte 単位の大きさであり、「文書数」はテキストに含まれる文書の数である。検索速度は OS のキャッシュが効けば速い。

<sup>3</sup>このように、IREX と NTCIR2 とで評価の指標が異なるのは、元々の検索タスクにおける評価の指標に合せたためである。なお、各質問における、R-Precision とは、質問文に適合すると判定された文書数と同数の文書を検索した時点における精度であり、Average-Precision とは、適合文書を検索することに精度を求め、その精度を平均した値である。また、質問文集合の R-Precision と Average-Precision とは、共に、各質問について求めたものを全質問で平均した値である (Salton and Buckley 1984)。表1における精度は、この全質問における精度である。

<sup>4</sup>表1において、IREX における文書 (記事) 数は 212555 であるが、これは IREX 本試験における記事数 211853 よりも多い。この理由は、IREX 本試験では 1995 年 8 月 23 日と 1995 年 8 月 24 日の記事が省かれているが、表1では、それらを含んでいるからである。厳密な比較のためには、ここでも、二日分の記事を省くべきだが、表1の目的は、厳密な比較ではなく、本パッケージの大まかな精度を見るためであり、その目的のためには、二日分の記事の差は問題ではないので、そのままにしてある。

<sup>5</sup>IREX と NTCIR2 のタスクでは、各質問文には、DESCRIPTION や NARRATIVE などのフィールドがある。厳密には、「long」とは、各質問文における (NTCIR2 では FIELD (分野) フィールドを除く) 全てのフィールドを利用して検索したということであり、「short」とは DESCRIPTION フィールド (検索要求を簡潔に述べたフィールド) のみを利用して検索したということである。

NTCIR2 JJ の処理速度が遅いのは、索引の大きさが主記憶の大きさを越えたためである。主記憶が索引程度に大きければ、もっと速くなる (3~10 秒程度) であろう。

## 2.4 Rochio タイプの automatic feedback による精度の向上

(1) 式は、(2) 式を利用することにより、フィードバック検索を実現しているが、フィードバック検索のためには、Rochio タイプの式 (Baeza-Yates and Ribeiro-Neto 1999) を使うこともできる。

そのためには、まず、(2) 式において、 $R = r = 0$  とする。次に、(1) 式における  $qtf$  を、

$$qtf = \alpha qtf_0 + (1 - \alpha) \frac{\sum_{i=1}^R qtf_i}{R} \quad (3)$$

とする。ただし、 $qtf_0$  とは、初期質問における当該単語の出現頻度であり、 $qtf_i$  とは、上位  $i$  位の文書における当該単語の出現頻度であり、 $R$  とは、automatic feedback に利用する上位文書の数である。また、 $\alpha$  は経験的に定める定数であり、本パッケージでのデフォルト値は 0.5 である。

このような変更を (1) 式に加えた場合の検索精度を表 2 に示す。この表における検索精度は、NTCIR2 JJ 以外は、それぞれのタスクにおける最高精度を上回っている。

表 2: 検索精度 (Rochio タイプの automatic feedback)

|           | long   | short  |
|-----------|--------|--------|
| IREX      | 0.5025 |        |
| NTCIR2 JJ | 0.4178 | 0.3407 |
| NTCIR2 EE | 0.4187 | 0.3371 |

## 2.5 使用例

本節では、本パッケージの使用例を述べる。まず、一定の書式のテキスト DB から索引を作るためには、

```
irbuilder テキスト DB へのパス
```

を実行する。

### 2.5.1 単言語検索

次に、索引が作成されたテキスト DB に対して、検索を実行するためのプログラム断片の例は以下の通りである。

```
# 初期化
ire = Engine.new(テキスト DB へのパス)      # (1)
tokenizer = ChaSen.new                       # (2)
query = Query.new(ire)                       # (3)
# 検索
query.setTerms(tokenizer.parse(日本語質問文)) # (4)
docs = ire.retrieve(query)                   # (5)
# 表示
docs.each {|doc| puts doc}                   # (6)
```

ここで、(1)では、検索エンジンを設定し、(2)では、単語の正規化のためのライブラリを茶筌に設定し、(3)では、質問文を表現するインスタンスを設定している。次に、(4)で、質問文を分割し内容語を抽出した結果を質問として設定し、(5)で、その質問に基づいて、文書を検索している。最後に、(6)では、各文書を印刷している。なお、上記のプログラムでは、automatic feedback をしていないが、「# 表示」の直前に

```
query.setRelevantTerms(docs[0,5])
docs = ire.retrieve(query)
```

を追加すれば、(5)での結果の上位5文献を利用して automatic feedback が実行される(質問文中の単語に加えて、上位5文献中に含まれる全ての単語を利用して検索が実行される)。

### 2.5.2 言語横断検索

言語横断検索のために必要な、翻訳の機能は用意していないが、もし、たとえば、英語文を日本語単語集合に変換するようなモジュールとして、EJTrans というようなものがあるとすると、以下のようなコードにより、英語質問文から日本語テキストDBを検索できる<sup>6</sup>。

```
# 初期化
ire = Engine.new(日本語テキストDBへのパス)
tokenizer = EJTrans.new
query = Query.new(ire)
# 検索
query.setTerms(tokenizer.parse(英語質問文))
docs = ire.retrieve(query)
# 表示
docs.each {|doc| puts doc}
```

言語横断検索においても、automatic feedback が有効であるが、言語横断検索の場合には、英語質問文から日本語テキストDBを検索する場合には、まず、英語質問文で英語テキストDBを検索し、その検索結果の上位に含まれる英語文書の単語を検索質問に加えてから、その拡張された質問を日本語単語集合に翻訳し、その結果を利用して、日本語テキストDBを検索し、その検索結果における上位の文書の日本語単語を質問に追加して、更に、日本語テキストDBを検索するという、automatic feedback を2回使う方法がある(Ballesteros and Croft 1998; Murata, Utiyama, Ma, Ozaku, and Isahara 2001)。この方法は、概略、以下のようにして実装できる。

```
# 初期化
jre = Engine.new(日本語テキストDBへのパス)
ere = Engine.new(英語テキストDBへのパス)
engTokenizer = EngPreprocessor.new
translator = EJTrans.new
jqquery = Query.new(jre)
eqquery = Query.new(ere)

# 英語質問 => 英語DB => 英語拡張質問
eqquery.setTerms(engTokenizer.parse(英語質問文))
edocs = ere.retrieve(eqquery)
eqquery.setRelevantTerms(edocs[0,5])

# 英語拡張質問 => 日本語質問
jqquery.setTerms(translator.parse(eqquery.getTerms.collect{|t| t.to_s}.join(' ')))

# 日本語質問 => 日本語DB => 日本語拡張質問 => 日本語DB
```

<sup>6</sup>この節のコード断片が実働するかは確認していない。

```
jdocs = jre.retrieve(jquery)
jquery.setRelevantTerms(jdocs[0,5])
jdocs = jre.retrieve(jquery)

# 表示
jdocs.each {|doc| puts doc}
```

このように、本パッケージを用いれば、情報検索が非常に容易にできる。また、パッケージには、本稿で述べた実験に使用したスクリプトが付随しているので、それを参考にすれば、本パッケージの詳細な使い方が分かる。

### 3 単語の統計量

情報検索においては、質問文に出現した単語と関連する単語を提示したり、あるいは、文書中における重要そうな単語を提示したりすることも必要である。そのためには、単語の出現頻度や共起頻度を求める必要がある。

本パッケージでは、各単語について、その単語が出現した文書数や、その単語の文書集合全体における出現頻度などを求めたり、あるいは、与えられた単語に対して、その文書集合において関連の高い単語を求めたりすることができる。たとえば、「検索」という単語に対して、対数尤度比 (Dunning 1993) が大きいものから 10 単語を得るには、

```
ire.getCoTerms("検索",10,"log-of-likelihood-ratio")
=> [{"パソコン", 93, 545.2879841}, {"情報", 148, 444.6491541}, {"データベース", 50, 423.100068},
["コンピューター", 68, 343.9604411], ["利用", 97, 326.3583694], ["通信", 79, 312.2293554],
["CD-ROM", 33, 263.1316569], ["電子", 51, 260.213618], ["システム", 68, 236.7180314],
["データ", 55, 233.8312139]]
```

のようにする。ここで、["パソコン", 93, 545.2879841] における、「パソコン」は抽出された関連語であり、「93」は、「検索」と「パソコン」の共起頻度であり、「545.2879841」は、それらの対数尤度比である。ここで、これらの数値は、毎日新聞 94-95 年の記事集合に基づくものである。

なお、単語  $v$  と単語  $w$  の対数尤度比  $\lambda$  とは、 $v$  と  $w$  の 2 単語が従属とした場合と独立とした場合との最尤推定量による尤度比であり、2 単語が従属している度合いが強いほど大きい値を取る。そして、その定義式は、

$$\lambda = 2 \sum_{i,j} f_{ij} \left\{ \log \frac{f_{ij}}{F} - \log \frac{f_i \cdot f_j}{F^2} \right\} \quad (4)$$

である。ただし、 $f(v, w)$  を単語  $v$  と  $w$  が同時に出現した文書の数、 $f(x)$  を単語  $x$  が出現した文書の数、 $F$  を全文書の数とすると、

$$\begin{aligned} f_{11} &= f(v, w) \\ f_{12} &= f(v) - f(v, w) \\ f_{21} &= f(w) - f(v, w) \\ f_{22} &= F - f_{11} - f_{12} - f_{21} \end{aligned}$$

である。また、

$$\begin{aligned} f_i &= f_{i1} + f_{i2} \\ f_j &= f_{1j} + f_{2j} \end{aligned}$$

である。

## 4 おわりに

ここで紹介したパッケージは、スクリプト言語 Ruby から容易に利用可能であり、検索精度は、ベースラインシステムとして十分なものである。また、このパッケージを利用することにより、連関語を得ることもできる。このパッケージは、

<http://www.crl.go.jp/jt/a132/members/mutiyama/software.html>

よりダウンロードできる。

## 参考文献

- Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley.
- Ballesteros, L. and Croft, W. B. (1998). "Statistical Methods for Cross-Language Information Retrieval." In *CROSS-LANGUAGE INFORMATION RETRIEVAL*, pp. 23-40. Kluwer Academic Publishers.
- Dunning, T. E. (1993). "Accurate methods for the statistics of surprise and coincidence." *Computational Linguistics*, **19** (1), 61-74.
- Frakes, W. B. and Baeza-Yates, R. (Eds.) (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall.
- IREX 実行委員会 (1999). IREX ワークショップ予稿集.
- Murata, M., Utiyama, M., Ma, Q., Ozaku, H., and Isahara, H. (2001). "CRL at NTCIR2." In *Proc. of the Second NTCIR Workshop* (<http://research.nii.ac.jp/ntcir-ws2/>), pp. 5-21 - 5-31.
- National Institute of Informatics (2001). *Proceedings of the Second NTCIR Workshop Meeting on Evaluation of Chinese & Japanese Text Retrieval and Text Summarization*.
- Oard, D. W., Levow, G.-A., and Cabrezas, C. (2000). "CLEF Experiments at the University of Maryland: Statistical Stemming and Back-off Translation Strategies." In *Working Notes for CLEF Workshop*.
- Robertson, S. E. and Walker, S. (2000). "Okapi/Keenbow at TREC-8." In *Proc. of TREC 8*, pp. 151-162.
- Salton, G. and Buckley, C. (1984). "README in trec\_eval.v3beta.shar." [ftp://ftp.cs.cornell.edu/pub/smart/trec\\_eval.v3beta.shar](ftp://ftp.cs.cornell.edu/pub/smart/trec_eval.v3beta.shar).
- Sudo, K., Sekine, S., and Grishman, R. (2001). "Automatic Pattern Acquisition for Japanese Information Extraction." In *Proc. of HLT 2001*.
- 松本裕治, 北内啓, 山下達雄, 平野善隆, 松田寛, 高岡一馬, 浅原正幸 (2001). "形態素解析システム『茶筌』version 2.2.5 使用説明書." <http://chasen.aist-nara.ac.jp/>.