自然言語処理 124－12
（1998. 3. 13）

分野を特定した自動収集による WWW 情報検索

来住 伸子

津田塾大学情報数理科学科
187-0025ー 東京都小平市津田町 2-1-1
Tel:0423-42-5160 Fax:0423-42-5161 E-mail:kishi@tsuda.ac.jp

概要

分野を特定することにより、既存の WWW 情報検索ツール (検索エンジンやディレクトリサービス) より効率のよい情報検索が可能なツール、qBook を現在作成中である。qBook は WWW ロボットによる自動収集と、収集した文書の内容の特定の分野に関する関連性の評価を並行して行うことにより、小規模で効率のよい、自動収集を行う。この論文では、qBook の実現方法と、特定分野を料理の作り方 (レシピ) とした時の qBook の 情報収集効率の評価結果を紹介する。

Topic-oriented Information Retrieval on the Web

Nobuko Kishi

Department of Mathematics and Computer Science, Tsuda College

Abstract

To facilitate user's ability to find relevant information on the World Wide Web, there are various type of information retrieval methods such as search engines and directory services. This paper proposes qBook, a hybrid system of a search engine and directory services. qBook estimates and evaluates HTML document relevance while it collects the documents as a web robot. By doing so, qBook enables a small and efficient search engine and a user interface similar to directory services. This paper describes qBook's estimation and evaluation algorithm and some experimental results on its efficiency.

1 はじめに

World Wide Web 上の情報検索には、検索エンジンやディレクトリサービスが利用されることが多い。それぞれ長所があるが、欠点もまだ多い [2]。ディレクトリサービスの欠点としては、リンク先の情報が古い、ディレクトリの管理や更新に人手がかかる、などがあげられる。検索エンジンの欠点としては、関係のない情報が含まれる、重要な情報が含まれないなどがある。

そこで、これらの欠点を解決するため、特定の分野に関する WWW 上の情報を効率良く自動収集し、収集した情報の内容の検索を容易にすることを考えた。具体的には、以下の設計方針を持つ WWW 情報検索ツールを作成することにした。

1. 汎用の情報検索サービスではなく、分野を特定した情報検索サービスを目的とする。対象となる分野は、その分野を特徴づけるキーワードを十分な量、用意できる分野とする。
2. WWW ロボットによる自動収集と、収集した文書の内容の、特定した分野との関連性の評価を並行して行うことにより、小規模で効率のよい、自動収集を行う。
3. 自動収集した情報の検索において、1 で用意したキーワードを再利用することにより、検索に必要な専門知識が少なくてすむようにする。

このツールは、特定分野の WWW 上の情報 に関する質問をしやすくすることを目的としているので、qBook (Query Book) と名付けた。

qBook が扱う特定分野は、次の条件を満たせる分野であれば、どのような分野でも構わない。

1. その分野に関するリンク集が WWW 上にいくつか存在する。個々のリンク集は、単独では不完全なものでよい。
2. その分野に関するキーワードを十分な数、用意できる。
3. それらのキーワードを、使い方の面から、後述する 3 種のタイプに分類できる。

たとえば、「大学における数学教育」という分野を特定の分野とすることができる。条件 1 を満たすには、数学科リンク集や大学リンク集を利用する。条件 2 を満たすには、大学名リスト、科目名に関するキーワードリスト、(「解析」、「位相幾何」)、講義の内容説明によく使われる用語リスト (「講義」,「開講」,「演習」) などを利用して、キーワードリストを作成する。条件 3 を満たすには、条件 2 のために作成したキーワードリスト中の各語を、対象とする分野の文書で実際に使われるているかどうか、対象となる分野のリンク集でよく使われているかどうか、検索語としてよく使うかどうかを判断できる人がいればよい。

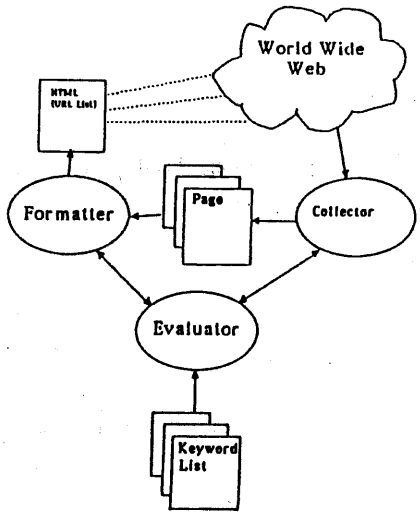

図 1: qBook の主な構成要素

この報告では、qBook の主な構成要素の紹介の後、各要素の動作の説明をし、最後に情報収集効率という観点からの評価結果を紹介する。また、この論文で紹介する qBook は、料理の作り方 (以下 レシピと表現する) という分野を特定分野の例としてとりあげる。WWW 上には レシピに関するリンク集がいくつか存在するので、条件 1 を満たすことができる。また、レシピは、レシピによく使われる形態素 (たとえば、「大匙」、「ジャガイモ」、「煮る」など) を、既存の料理に関する書籍から 約 1300 語用意することにより、上記の条件 2 を満たすことができる。

2 qBook の実現方法

2.1 qBook の主な構成要素

qBook は、図 1に示す 3 個のプロセスから構成される。なお、この図および、これ以降の説明では、URL とは URL 形式で記述された文字列、Page とは HTML

形式で記述された文字列（ある URL に対して HTTP プロトコルを利用して入手できる文書の内容）とする。

- Collector: Page を収集するロボット
- Evaluator: Page を評価するサーバー
- Formatter: 収集した Page をもとにディレクトリやリンク集のような HTML 文書を作成するツール

Evaluator は常時動作しているサーバープロセスで、文字ストリームが与えられると、そのストリームの中に、指定されたキーワードが何個含まれるかを評価するプロセスである。Collector は HTTP により Page を集めるプロセスで、タイマーによって定期的に起動される。Formatter は、Collector の出力として得られた Page から、関連度の高い Page を選び、それらに対応する URL をリンク集やディレクトリのような HTML 文書として出力する。図 2 は Formatter の出力例である。

2.2 Evaluator によるキーワードの抽出

Evaluator を起動する前に、キーワードリストを用意する。キーワードリストとは、対象とする特定分野でよく使われる語の集合で、以下の 3 種類のタイプにわける。

- タイプ 1
  特定分野の Page に現れることが多く、さらに、検索のキーワードとして利用されることの多い語の集合。たとえば、肉素材リスト (例「豚肉」、「ぶた」、「三枚肉」), 魚素材リスト (例「さんま」、「秋刀魚」、「サンマ」), 野菜素材リスト (例「玉ねぎ」、「玉葱」、「たまねぎ」、「タマネギ」) などがある。
- タイプ 2
  特定分野の Page に現れることが多いが、検索のキーワードとして利用されることがあまりない語の集まり。たとえば、道具リスト (例「大匙」、「大サジ」、「包丁」) 用語リスト (例「強火」、「三枚」、「沸騰」) などがある。
- タイプ 3
  特定分野の Page に現れることはあまりないが、特定分野の Page のリンク集に多い語の集まり。現在、レシピの分野では、リンク集用語リスト (例「レシピ」、「作り方」、「御総菜」、「メニュー」、「季節」, 「レシピ集」) だけがある。

このように 3 種類のタイプに分けた理由は、使用目的により、特定分野の「キーワード」が異なることが多いためである。そこで、各ツールが、以下のようにキーワードを使い分けることにより、Evaluator の共有ができるようにした。

- Collector
  タイプ 1,2,3 のすべてのキーワードリストを使用する。
- Formatter
  タイプ 1 と 2 のキーワードリストを使用する。
- ユーザ (Formatter の出力を利用する人)
  タイプ 1 のキーワードリストに含まれるキーワードを索引語として利用する。

Evaluator はサーバーとして起動され、クライアント (Collector と Formatter) からの要求をまつ。クライアントから入力ストリームが送られてくると、まず、形態素解析プログラム chasen [4] を利用して、入力ストリームを形態素列に分解する。次に、その形態素列に、予め指定したキーワードリストの中に含まれる語が何個あるかを調べ、その結果をクライアントに知らせる。

2.3 Collector による自動収集

Collector は、特定分野に関連する Page を効率よく収集するために、以下の 3 種の指標を利用する。

- Page Relevance (関連度)
  Page が, 特定分野にどの程度関連するかを示す指標。タイプ 1 と 2 のキーワードリストに含まれるキーワードの語数に、リスト別の重みを加重して合計することによって計算する。
- Link Weight (リンクの関連度)
  リンクの指している先に、関連度の高い Page, または、特定分野に関連の 高い リンク集がある可能性を示す指標。URL u に対応する Page に含まれるリンクの Link Weight を $l(u)$ と表し、以下の定義式で計算する。

$$l(u) \equiv C\frac{w}{s}\log_2\frac{s}{(t+1)}$$

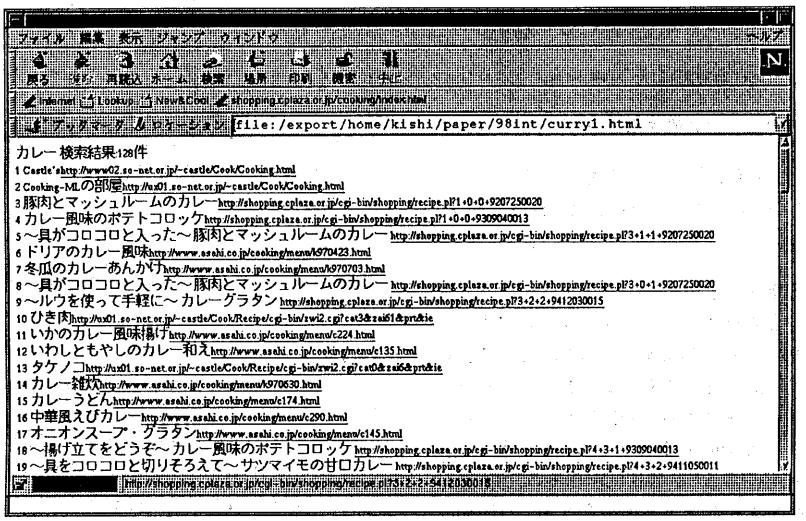

図 2: qBook のユーザ画面例 (Formatter の出力を WWW ブラウザで表示したもの)

w URL u に対応する Page に含まれる、タイプ 1,2,3 のキーワードリストのキーワードの語数

s Page の大きさ。URL u に対応する Page からタグを除いた部分のバイト数

t URL u に対応する Page に含まれるリンクの個数 (アンカータグ (<a href=) の個数).

C 適当な定数で、現在 100 を採用している。

- Estimated Relevance (Page Relevance の推定値)

通常の意味での関連度 (Page Relevance) は, Page の内容が入手できないと、計算できない。しかし、自動収集を効率よく行うには、URL u は分かっているが、それに対応する Page の内容が入手していない段階で、関連度を推定し、その推定値が高ければ、Page の内容を入手することが必要である。そこで、Page Relevance の推定値、Estimated Relevance $e_X(u)$ を次のように計算した。

$$e_X(u) \equiv \sum_{x \in TO(u) \cap X} l(x)$$

u Page の内容をまだ入手できていない URL u.

X すでに Page の内容を入手した URL の集合.

$TO(u)$ u をリンク先とするアンカータグ (<a href=u>) を含む Page の URL の集合.

これらの指標と、表 1 のアルゴリズムを利用して、Estimated Relevance が一定値以上の URL に対応する Page を集めた。

アルゴリズム 1 では $TO^{-1}(u)$ という集合を利用しているが、次のような意味で使用している (図 3)。

u URL 形式の文字列

U URL 形式の文字列の集合

$h(u)$ HTML 形式の文字列。$u \in U$ に対して、HTTP プロトコルを利用すると入手できる内容。Page と呼ぶ。

$x \to y$ $x, y \in U$ に対し、$h(x)$ に y へのアンカータグ (<a href=y>) が含まれるときに成り立つ関係。

$TO(u)$ $\{x \in U | x \to u\}$

$TO^{-1}(u)$ $\{x \in U | u \to x\}$

表 1 のアルゴリズムは 入力として要素 1 個の URL の集合 G 与えると、その集合を徐々に大きくしていくことを繰り返す (図 4)。その集合列を $G_0, G_1, , ..., G_n$ とすると、G_i から G_{i+1} からを生成する際に、$e_{G_i}(u)$ を評価関数として利用する。また、次のの性質を利用して、$e_{G_i}(u)$ から $e_{G_{i+1}}(u)$ を計算して利用している (付録)。

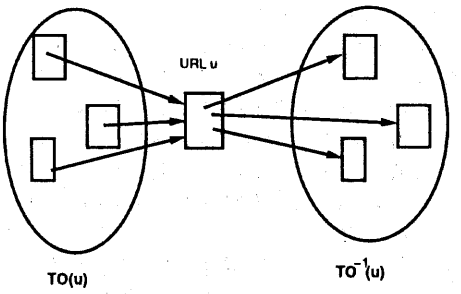

図 3: $TO(u)$ と $TO^{-1}(u)$ の関係

$G_{i+1} = G_i \cup \{y\}$, $y \notin G_{i+1}$ のとき、

$$e_{G_{i+1}}(u) = \begin{cases} e_{G_i}(u) & (u \notin TO^{-1}(y) \text{ のとき}) \\ e_{G_i}(u) + l(y) & (u \in TO^{-1}(y) \text{ のとき}) \end{cases}$$

2.4 Formatter による整形

Formatter は Collector が集めた Page の中から、Page Relevance の高い Page を選び、特定分野に関する Page の集合を作成する。ユーザの検索要求に対し、この Page の集合を全文検索をすれば通常の検索サービスと同じように利用できる。しかし、キーワードリストを利用して、キーワードをあまり知らない人が、ディレクトリやリンク集として、この Page の集合の URL 集を検索できるようにした。

まず、ユーザは、先述したタイプ 1 のキーワードリストのリストを選ぶ。さらに、選んだキーワードリストの中からさらにキーワードを選ぶ。選んだキーワードを含む Page の URL 数が一定数以下であれば、その URL のリストを、リンク集スタイルの HTML 文書として出力する (図 2) 一定数以上ある場合は、すでに選んだキーワードを含む Page に含まれるキーワードリストを出力する。ユーザは、キーワードをさらに選択することにより、対象となる URL 数を絞り込むことができる。

ユーザがこのような操作ができるようにするために、Formatter は以下の処理を行う。

1. キーワード表現の揺れの解消。
   タイプ 1 のキーワードに対し、各キーワードの正規形 (原則として読み) を定める。その正規形を使って、キーワードリストを正規形つきキーワードリストに書き換える。ある正規形 r に対し、

表 1: Estimated Relevance の高い Page を集めるアルゴリズム

入力

- URL a. a は対象となる分野のリンク集のどれか一つの URL. a の Estimated Relevance $e_\phi(a)$、は十分に高い値に初期化しておく。
- 定数 *lbound*. Estimated Relevance の 下限値。

出力

- URL の集合 K. 各 $k \in K$ に対応する Page の内容 h(k) は HTTP プロトコルによって入手ずみで、$e_K(k) > lbound$.

```
K = φ;
G = {a};
while( min_{x∈G}{e_K(x)} > lbound ) {
  g = G の要素で最も e_K(g) が高い URL
  h(g) を HTTP により入手する。
  G = G − {g};
  K = K + {g};
  for( each x ∈ TO⁻¹(g)) {
    if(x ∈ K) continue;
    if(x ∈ G)
      e_K(x) = e_K(x) + l(g);
    else {
      e_K(x) = l(g);
      G = G + {x};
    } // end if else
  } // end for
} // end while
```

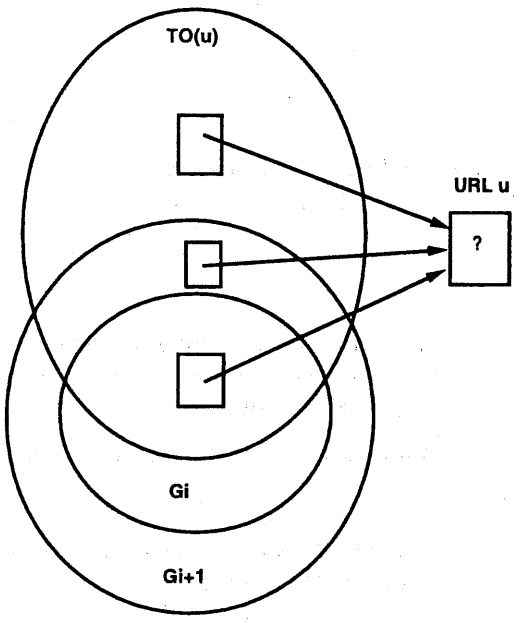

図 4: アルゴリズム 1 の途中の状態

n_r 個のキーワードが対応する場合、

$$k_0 : r, k_1 : r, ..., k_{n_r} : r$$

という正規形つきキーワードリストとして表現する。

たとえば、「玉ねぎ」、「玉葱」、「たまねぎ」、「タマネギ」、「オニオン」というキーワードリストは、

玉ねぎ: たまねぎ, 玉葱: たまねぎ,
たまねぎ: たまねぎ, タマネギ: たまねぎ,
オニオン: たまねぎ

という正規形つきキーワードリストに書き換える。

2. 関連度の低い Page の除去.
Evaluator を利用して、各 Page の Page Relevance (タイプ 1、タイプ 2 のキーワードリストに含まれる語の数) を計算し、低い Page は 除去する。同時に、各 Page で使われている、タイプ 1、タイプ 2 のキーワードリストに含まれる語の表を作成する。

3. キーワードによる分類.
各 Page を タイプ 1 キーワードリストにより、分類する。現在タイプ 1 のキーワードリストは、「肉素材リスト」「魚素材リスト」「野菜リスト」などの数種あり、それらのリストを利用して Page を 分類する。一つの Page に複数のキーワードリストの語が含まれる場合は複数のリストに分類する。たとえば、「豚肉」と「じゃがいも」の両方を利用した レシピは「肉素材リスト」の語のための リンク集と「野菜リスト」の語のための リンク集の両方に現れる。

なお、タイプ 2 のキーワードを含むが、タイプ 1 のキーワードを全く含まない Page は 分類できないが、未分類の Page として記録しておき、タイプ 1 にすべき語の登録もれの調査に利用している。

4. 共有キーワードの集計.
指定した キーワードを含む Page の数が、予め指定した一定数より少なければ次のステップ 5 に進む。多ければ、ステップ 3 で作成した表を調べ、各 Page で使用されている タイプ 1 のキーワードリストを出力し、ユーザがキーワードの追加指定を行えるようにする。

5. リンク集スタイルの HTML 文書の出力.
指定したキーワード群を含む Page に対応する URL をアンカータグとして使用した、リンク集の形式の HTML 文書を出力する。

3 評価

Collector の性能は情報収集効率という面から、Formatter の性能は情報検索作業効率という面から評価すべきである。しかし、後者の面での評価は今後の課題とし、今回は、前者の面での評価を以下の方法で行った。

- 評価対象:
  - qBook-1 (Page 5,000 個 を集めた状態)
  - qBook-2 (Page 31,674 個 を集めた状態)
  - goo[3]
    (Page 56,000,000 個を集めた, 1997 年 11 月 24 日の状態)
- 検索課題: 「カレー」の作り方

- 評価方法: 数通りの検索キーワード入力に対し、検索結果の上位の中に「カレー」の作り方が何個含まれるかを調べた。

使用した検索キーワードは次の5例である。

- 例1 カレー
- 例2 カレー 作り方
- 例3 カレー 玉ねぎ
- 例4 カレー 大さじ
- 例5 カレー 作り方 玉ねぎ 大さじ

これらのキーワードを利用して検索すると表2のようになった。

表2: qBook と goo の検索効率比較

検索キー		A	B	C	D
例1	qBook-1	128	91	30	54
	qBook-2	605	90	25	43
	goo	18318	99	5	0
例2	goo	1619	97	48	26
例3	goo	459	94	50	32
例4	goo	392	96	44	52
例5	goo	122	95	50	45

A 検索条件に適合した件数

B A の上位100位から重複を除いた件数

C B に含まれるカレーの作り方の件数

D B に含まれるカレー以外の料理の作り方の件数

Cの件数で比較すると、qBook は、不十分な検索キーワードで goo を利用したときより検索結果件数が多く、十分な検索キーワードで goo を利用したときより検索結果件数が少ないことが分かる。しかし、件数が少ないといっても 50 % 程度の少なさなので、収集した Page 数が 1,800 - 10,000 倍違うということを考えると、収集効率は十分に高いと考える。

また、qBook の検索結果 に含まれて goo の検索結果に含まれない Page と、その逆の Page が共に数割程度存在した。そのため、どちらの収集能力が優れているとは、正確には判定できない。しかし、日本国内のカレー料理のページは多くても 、例2の値から推定すると 1619×48/97 = 約 800 件、例3 の 値から推定すると 459 × 50/94 = 約244件、程度と考えられる。そこで、qBook-2 程度の収集件数、605×25/90 = 約170件、で、既存の検索エンジンと実質的に同程度の検索結果件数を得られていると考えている。

4 まとめと今後の方向

特定の分野に関する WWW 上の情報を効率良く自動収集し、収集した URL の容易な検索を可能にしたツール、qBook を提案し、その第一版の収集機能を既存の検索エンジンと比較した。

qBook では、収集と整形の各段階で、キーワードリストを利用することにより、特定の分野に限定した収集や整形を効率化している。「料理の作り方」(レシピ) という分野を特定するためには、約1300 語のキーワードリストを使用した。大きな検索エンジンと比較した結果、検索結果件数では qBook の方が少ないが、収集した文書数との相対比でみると、検索効率は高いことが確認できた。

その他の課題として、 qBook がレシピ以外の分野を対象とすることができるのか、qBook は利用者の作業効率をどの程度向上させるのか、というような面での評価があげられる。また、巨大な検索エンジンが少数存在した場合と、qBook のような小さな検索エンジンが多数存在した場合のネットワークの負荷といった面での評価も可能であれば取り組みたいと考えている。今後、より現実的な利用形態の中で、qBook の実装方法の改良と評価を重ねていきたい。

参考文献

[1] 松岡淳子, 来住伸子, 鈴木悦子, 小川貴英: "WWW 情報の重要度に基づく自動収集の試み," 情報処理学会第52回全国大会講演論文集第1分冊,1996, pp.1-165 - pp.1-166.

[2] 山名早人: "WWW 情報検索の現状," コンピュータソフトウェア, Vol.5, No.14, 1997.

[3] "goo," http://www.goo.ne.jp

[4] 松本裕治 他: "日本語形態素解析システム「茶筅」version 1.5 使用説明書," NAIST Tecnical Report

NAIST-IS-TR97007, 1997.
http://cactus.aist-nara.ac.jp/lab/nlt/chasen.html

[5] Golovchinsky, G. : Queries? Links? Is there a Difference?, *Proc. CHI'97 Conferece - Human Factors in Computing Systems,* 1997, pp. 407–414.

付録

$e_X(u)$ の性質
$X_{n+1} = X_n \cup \{y\}$, $y \notin X$ のとき、

$$e_{X_{n+1}}(u) = \begin{cases} e_{X_n}(u) & (u \notin TO^{-1}(y) \text{ のとき}) \\ e_{X_n}(u) + l(y) & (u \in TO^{-1}(y) \text{ のとき}) \end{cases}$$

$e_X(u)$ の性質 の理由
$u \notin TO^{-1}(y)$ のとき,

$$\begin{aligned} e_{X_{n+1}}(u) &= \sum_{x \in TO(u) \cap X_{n+1}} l(x) \\ &= \sum_{x \in TO(u) \cap X_n} l(x) + \sum_{x \in TO(u) \cap \{y\}} l(x) \\ &= \sum_{x \in TO(u) \cap \{y\}} l(x) \\ &= e_{X_n}(u) \end{aligned}$$

$u \in TO^{-1}(y)$ のとき,

$$\begin{aligned} e_{X_{n+1}}(u) &= \sum_{x \in TO(u) \cap X_{n+1}} l(x) \\ &= \sum_{x \in TO(u) \cap X_n} l(x) + \sum_{x \in TO(u) \cap \{y\}} l(x) \\ &= \sum_{x \in TO(u) \cap X_n} l(x) + l(y) \\ &= e_{X_n}(u) + l(y) \end{aligned}$$
