QueReSeek *feat.*Wikipedia: 辞書を用いたキーワード繋がりによるウェブコンテンツの検索

QueReSeek *featuring* Wikipedia: Associated Web Search Using Keywords Found in Dictionaries

丹 英之[*1] 大向 一輝[*2] 武田 英明[*2]
Hideyuki TAN Ikki OHMUKAI Hideaki TAKEDA

[*1]株式会社アルファシステムズ Alpha Systems Inc.
[*2]国立情報学研究所 National Institute of Informatics

In this paper we propose QueReSeek, an associated web search engine that uses keywords defined in dictionaries. We use Wikipedia to take advantage of the collective intelligence of wide range of data contributers(i.e. Wikipedia authors). Furthermore, our system connect a wide range of web content based on the keywords found on the target web-page.

1. はじめに

ウェブを用いた情報検索は，日常生活においても無くてはならないものとなった．しかし，検索クエリとして入力した文字列の条件一致で結果を提示する検索エンジンでは，効果的な検索クエリの生成能力が重要であり，的確な結果を得るまでに要する時間や，得られた結果の質は検索者の情報検索スキルに依存してしまう．そこで我々は，検索履歴であるクエリログをコミュニティ内にて共有し,「いま閲覧中のウェブページは，他の人が入力したこの検索クエリで発見できます.」という情報をウェブ閲覧者に提示する，情報検索行為を支援するする仕組みQueReSeek(QRS) について検討してきた [丹 07] [丹 08]．しかし，この手法が効力を発揮するには，蓄積されたログの規模が重要になる．また，この手法の検討の中で，間違って利用されたクエリ文字列の除去が課題として残っていた．一般的に，この手の課題解決には統計学的手法が有効であると考えられが，入力ミスのクエリ文字列の除去にもある程度のログの量が必要となるのである．

そこで今回は，既に辞書で定義済みの言葉を QRS に適用することで，検索履歴を使用することなく閲覧中のウェブコンテンツに関連する，“既に他者に認知されているであろうキレイな文字列” である言葉の提示を試みた．そして，更にその認知可能な文字列で繋がっているという理由から結果を導く，関連ウェブコンテンツの連想検索について検討した．

2. QueReSeek の考え方

ここでは，QRS の考え方の基本となる，検索エンジンの入力と出力である検索クエリと検索結果の関係，そして，その検索クエリと検索結果の二部グラフの参照により得られる関連するウェブコンテンツの特定方法について述べる．

2.1 検索クエリと検索結果の関係

検索者がクエリ文字列を検索エンジンへ入力すると，検索エンジンはロボットによって収集したウェブコンテンツ集合に対して全文検索を行う．そして，採用しているアルゴリズムによって算出されたスコアに基づいてウェブコンテンツの順位を決定し，そのの URL 一覧を検索結果として検索者へ返す．言い換えると，検索クエリはウェブコンテンツを対象とした全文検索の結果であるコンテンツ集合に含まれる文字列であり，検索クエリはランキングアルゴリズムによってコンテンツの断片としてウェブコンテンツと結びついている．これを逆から見ると，“検索結果集合に含まれるウェブコンテンツを一言で表すと検索クエリになる” ということになり，ウェブコンテンツが一言に抽象化されたことに相当する．

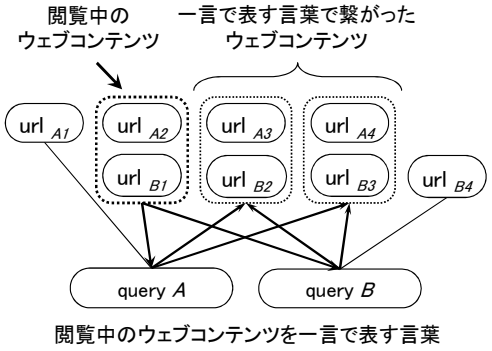

図 1: 検索クエリ-検索結果である URL の二部グラフ

検索クエリと検索結果として得られるウェブコンテンツを指す URL 集合の二部グラフを図 1 に示す．検索クエリ query$A$ の検索結果として得られるウェブコンテンツを指す URL を，結果のランキング順に url_{A1}，url_{A2}，... と表している．これを逆から見ると，ウェブコンテンツ url_{A1}，url_{A2}，... が query$A$ の一言になる．

ここで，検索クエリ query$B$ の検索結果である URL の集合も同様に示すと，検索クエリの近さによっては，同じウェブコンテンツを指す URL が得られる場合がある．図では url_{B1}，url_{B2}，url_{B3} がそれぞれ，url_{A2}，url_{A3}，url_{A4} と同じウェブコンテンツを指している．ここで，$\mathrm{url}_{A2}(=\mathrm{url}_{B1})$ が閲覧中のウェブコンテンツであるとするならば，$\mathrm{url}_{B2}(=\mathrm{url}_{A3})$，$\mathrm{url}_{B3}(=\mathrm{url}_{A4})$ は，query$A$，query$B$ で繋がったウェブコンテンツになる．

本提案では，query$A$，query$B$ が辞書で定義されている言葉であるので，$\mathrm{url}_{B2}(=\mathrm{url}_{A3})$ と $\mathrm{url}_{B3}(=\mathrm{url}_{A4})$ は，辞書で定義済みの一般に認知されているであろう言葉によって関連するウェブコンテンツということになる．これらをユーザに提示する．

連絡先: 丹英之，株式会社アルファシステムズ，川崎市中原区上小田中 6-6-1，044-733-4111，tanh@alpha.co.jp

2.2 関連する検索クエリととウェブコンテンツ

検索クエリと検索結果である URL 集合のセットを大量に用意しておくことで，閲覧中のウェブコンテンツが検索結果として得られる検索クエリを複数得ることができる．これを元に複数のウェブコンテンツをユーザへ提示するには，それらを提示する順番を決める必要がある．

まず，閲覧中のウェブコンテンツを示す url に関連する検索クエリのリストを求める必要がある．検索結果から得られるパラメータは，その検索クエリ query でのヒット数 $Hit(\mathrm{query})$，検索結果内における閲覧中 url の順位 $Rank_{\mathrm{url}}(\mathrm{query})$，そして，閲覧中 url の存在する検索結果を返した検索エンジンの個数 $Engines$ である．この検索エンジンの個数は，複数の検索エンジンからの結果を束ねた際に取得できる値である．これら三つの値を元に，閲覧中 url と検索クエリの関連性に関するスコア $Score_{\mathrm{url}}(\mathrm{query})$ を算出する式を式 (1) に定義する．

$$\begin{aligned} & Score_{\mathrm{url}}(\mathrm{query}) \\ = & \left(100 - \frac{\sum_{e}^{Engines} Rank_{\mathrm{url}}(\mathrm{query},e)}{Engines} + 1\right) \\ \times & \ Engines \\ \times & \ \log\left(\frac{\sum_{q}^{queries \in results(\mathrm{url})} \sum_{e}^{Engines} Hit(q,e)}{\sum_{e}^{Engines} Hit(\mathrm{query},e)}\right) \end{aligned} \tag{1}$$

第一項は，検索結果の上位に位置するウェブコンテンツほど，検索クエリとの関連は強いことを表している．検索結果の上位 100 件を対象としたので，その検索クエリを複数エンジンで検索した際，閲覧中 url の平均ランクが 1 位であれば 100 となる．第二項は，その検索クエリを複数エンジンで検索した際，閲覧中 url が存在する検索結果を返した検索エンジンの個数である．その検索クエリをどの検索エンジンで検索しても閲覧中 url が検索結果に含まれるということは，検索クエリと閲覧中 url の関連が強いことを表している．第三項は，ヒット数による検索クエリの語としての汎用性，専門性を表している．閲覧中 url が検索結果に含まれる複数の検索クエリのヒット数全ての積算値に対し，その検索クエリがどのくらいの割合を占めているかを求める．この値が大きいと，一般的に用いられる汎用的な言葉であり，小さいと特殊で専門的な言葉であると言える．ここではその逆数をとることで，値が大きい場合には複数個ある検索クエリの中でも専門性の高い言葉であり，その URL の指すウェブコンテンツを特徴づけている言葉であるということになる．ヒット数は値の範囲が広いので対数をとった．このスコアで検索クエリをソートすることで，閲覧中のウェブコンテンツに関連する検索クエリの順位が求まる．

次に，求まった検索クエリで繋がるウェブコンテンツを指す URL を求める．複数の検索クエリによる複数の検索エンジンからの検索結果を束ねるためのスコア $Score(\mathrm{URL})$ を算出する式を，式 (2) に定義する．

$$\begin{aligned} & Score(\mathrm{URL}) \\ & = \sum_{q}^{queries \in results(\mathrm{url})} \\ & \{(10 - Rank_{\mathrm{URL}}(q) + 1) \times Score_{\mathrm{url}}(q)\} \end{aligned} \tag{2}$$

閲覧中 url に関連する検索クエリ全ての検索結果上位 10 件の URL に対し，式 (1) の第一項と同じ考え方に基づいた値と，式 (1) で求めたその閲覧中 url に関連する検索クエリ自体のスコア $Score_{\mathrm{url}}(\mathrm{query})$ を重みとして掛け，その値を総和したものを，閲覧中 url と関連するウェブコンテンツを指す URL の関連度を表すスコアとした．複数エンジンの結果に存在する場合も，そのまま積算する．このスコアの高い URL から順に並べることで，閲覧中ウェブコンテンツに対し，辞書で定義済みの言葉で繋がった関連度の高いウェブコンテンツを得ることができる．

図 2: QueReSeek の検索結果画面

3. 実装

ここでは，QueReSeek *feat.* Wikipedia にて実装したインタフェース，そして用意したデータセットについて述べる．

3.1 インタフェース

QRS の検索結果を得るためには，閲覧中のウェブコンテンツを指す URL をシステムに渡す必要がある．通常の検索クエリ文字列の入力とは異なり，URL 文字列は英数記号の羅列であるので，検索の都度 URL 文字列を入力することはユーザにとって負担が大きい．そこで，ウェブ閲覧中でも構築したシステムにアクセスしやすい手段を三つ用意した．

- ブックマークレット

  ブックマークレットとは，ブックマークに登録された JavaScript のことである．ウェブの閲覧中，そのウェブコンテンツに対する QRS の検索結果が必要になった際，ユーザはブックマークレットを実行することで QRS のシステムを呼び出すことができる．ユーザはブックマークレットの実行を能動的に行う必要があるが，ユーザの利用するウェブブラウザに最も依存しにくい汎用的な手段である．

- ユーザスクリプト

  ウェブブラウザの Firefox では，Add-ons の Greasemonkey [*1] を導入することで，ウェブコンテンツのロードの際，ウェブコンテンツの内容にかかわることなくユーザの用意した JavaScript を実行することができる．そこで，ウェブコンテンツのロード完了後，自動的に QRS の API を呼び出すユーザスクリプトを作成した．このスクリプトを用いることで，閲覧中のウェブコンテンツに関連するキーワードが存在する際，コンテンツ表示領域の左上にアイコンを提示しユーザへ通知する．

- Firefox の Add-ons

  上記ユーザスクリプトの機能を拡張機能のモジュールとして実装した．これにより，ウェブブラウザのインタフェー

∗1 https://addons.mozilla.org/ja/firefox/addon/748

スと QRS の結果表示を一体化することができる．この Add-ons を導入することで，閲覧中のウェブコンテンツに関するキーワードの個数が，ウェブブラウザの下部に位置するステータスバーに表示される．

QRS による検索結果の表示画面を図 2 に示す．画面上位には閲覧中のウェブコンテンツに関連したキーワードと，その語の定義が記述されている各辞書の記事へのリンクが一覧で表示される．そして，それらのキーワードによって繋がった他ウェブコンテンツが，繋がりの根拠となるキーワードと共に提示される．また，この他ウェブコンテンツから更に関連するキーワードによって繋がったウェブコンテンツを探し出すリンクも提供している．ユーザはこれらのリンクを辿り，繋がりの根拠となったキーワードである語の定義を確認したり，他のウェブコンテンツへと閲覧を進めることになる．

3.2 データセット

図 1 の二部グラフを構築するために用いた，キーワードと URL について述べる．

- キーワードのリスト

  キーワードの入手先となる辞書には，オンライン参加型百科事典である Wikipedia(日本語)[*2]，はてなキーワード[*3]，そして，小学館の「日本大百科全書 (ニッポニカ)」をベースに構築されている Yahoo!百科事典[*4]に掲載されている言葉を用いた．これにより，約 75 万語のキーワードを用意できた．

- URL のリスト

  用意したキーワードを検索クエリとし，検索エンジン Google[*5]，Yahoo![*6]，Live Search[*7] にて検索を行い，それぞれの結果に含まれる上位 100 件の URL について，明らかにコンテンツタイプが html ではないと想定される拡張子のものを除いて収集した．また，URL 文字列の末端が “/index.(htm|html|shtml)” である場合，“/” 以降を削除することで，そのディレクトリを表す URL 文字列にする簡単な正規化を行った．これにより，約 9,800 万件の URL を用意できた．

4. 考察

本章では，構築したシステムを利用してみた感懐を元に，QRS について議論する．

前回の試みでは，Web Proxy のログから抽出できた検索クエリを元に検索クエリ-URL の二部グラフを構築したが，閲覧中のウェブコンテンツに対して他者の利用した検索クエリを提示できる割合が少なく感じる場合が多々あった [丹 08]．しかし，今回の試行では関連キーワードとしての検索クエリを提示できる場合が多くみられ，データセットの量が質に転化した印象を持った．

QRS による検索結果は，微妙な URL 違いに敏感であることが判った．あるサイト example.com を提供しているサーバは，ウェブブラウザからのリクエストヘッダーやアクセス元の IP アドレスなどでクライアント側の利用言語を想定し，到達ページをリダイレクトで動的に切り替えていることがある．このとき，例えば `http://example.com/index.html` にアクセスすると，クライアントが日本語環境では `http://example.com/index-ja.html`，英語環境であれば `http://example.com/index-en.html` へのリダイレクトを返すとする．すると，QRS では閲覧中の URL を用いるので，日本語環境下での閲覧では，`http://example.com/index-ja.html` で検索クエリ-URL の二部グラフを参照することになる．この場合，同じウェブコンテンツを指す URL が複数存在することになるので，QRS による検索結果を得ることができる可能性が低下する．特に，検索エンジン側では，`http://example.com/` がその example のサイトを示す URL であるとしていたりもするので，組み合わせによっては，QRS の結果が異なることになる．現状の QRS は，URL 文字列の単純比較で処理しているので，これらの現象はやむを得ないと考えられる．しかし，URL 文字列に対し強力な正規化パターンを適用できれば，ある程度の揺らぎを緩和できると思われる．これに関し，検索エンジン側でも複数 URL を持つ同一ウェブコンテンツの問題を抱えおり，最近では唯一の正しい URL をサイト作成側にて指定できる Canonical Link 要素の導入を進めている [Google 09][Yahoo 09][Microsoft 09]．これが普及すると，閲覧中のウェブコンテンツについての正規化された URL を取得できるので，微妙な URL 違いにもロバスト性を持つ検索結果を返すことが可能であると考えられる．

QRS を呼び出すことで，閲覧中のウェブコンテンツから静的なリンクではなく，提示された定義が存在する言葉繋がりで他のウェブコンテンツへ閲覧を遷移することができる．ウェブコンテンツや繋がりとなる言葉の意味解析を行うことなく，単純な文字列マッチングだけで，Link-free Browsing[Iwazume 08] を実現している．

QRS では，全文検索エンジンによる検索結果をデータセットの準備に用いているので，ウェブコンテンツ内での共起度の高い近傍の言葉繋がりでウェブコンテンツの検索を行う．ところが，Yahoo! JAPAN[*8]について QRS の検索結果を求めると，“退場” が関連キーワードの上位に来る．この原因は，ウェブコンテンツ内に含まれる文字列ではなく，Yahoo! JAPAN へのリンクに付与されたアンカーテキストに含まれる文字列であることが想定される．これより，検索エンジンの基本はウェブコンテンツ内の全文検索であるが，アンカーテキストも十分考慮されていることが判る．このことから，QRS は単なる文書内での近傍にある言葉のみならず，ウェブコンテンツのリンク関係による言葉の繋がりをも提示できることになるので，検索結果には意外性が盛り込まれていると言える．つまり，意図した的確な情報を検索結果に求めない利用の仕方，例えば着想支援などアイデア想起にも有用で，現状のキーワードサーチだけでは探せないウェブコンテンツを見つけ出すことが可能であると思われる．

今回は，キーワードのデータソースとして，オンライン百科事典を用いた．オンラインでの自由編集型の百科事典では，記事内容についての信頼性について議論の対象になる場合が多い．しかし本手法では，事典の内容である語の定義を利用するのではなく，辞書のインデックスにその語が存在しているということのみしか利用していない．つまり，その文字列が意味をなしているという，辞書に掲載される程の認知度があるということだけを用いているに過ぎない．よって，辞書内における語の定義の信頼性に関する課題については，本手法とは別問題であり，信頼性の議論は，QRS の検索結果にて提供される，語

*2 http://ja.wikipedia.org/
*3 http://k.hatena.ne.jp/
*4 http://100.yahoo.co.jp/
*5 http://www.google.co.jp/
*6 http://search.yahoo.co.jp/
*7 http://www.live.com/
*8 http://www.yahoo.co.jp/

の定義へのリンク辿ってからのこととなる．この場合でも，複数の辞書へのリンクを提供できれば，利用者自身が記事を比較することが可能となる．特に今回は，Yahoo!百科事典もキーワードのデータソースとして利用している．この事典は署名記事から成り立っており，この事典を参照できる場合には，語の定義に関してもそれなりの信頼性を担保できると思われる．キーワードと同様，提示する他のウェブコンテンツの方も，複数の検索エンジンの結果を採用することで，特定のウェブコンテンツが検索結果に表示されないなどの問題も解決できると思われる．

5. まとめ

本稿では，群衆の叡智である Wikipedia を辞書に用いることで，集合知の視点からウェブ空間を俯瞰し，閲覧中コンテンツの近傍にあるウェブコンテンツを探し出す，次世代型の情報検索エンジンである QueReSeek feat. Wikipedia について紹介した．QRS を用いると，一般的なウェブコンテンツに用語開設の footnote のようなリンク先を提示できるので，QRS のユーザはウェブコンテンツに関連する言葉を定義に戻って確認できるようになる．これは，辞書参照の新しい方法でもある．

今後は，知識ドメインを絞った専門領域への応用を検討したい．これには，特定分野を対象とした検索エンジンの利用や，専門分野に特化した辞書を用いることで可能となると思われる．また，予め二部グラフを構築しておくアーキテクチャの制約上，ニュースサイトやウェブログなど，即時性のあるウェブコンテンツにに対しては効果を発揮しにくいので，これを改善したい．また，URL- 検索クエリ-URL と二部グラフを参照したが，検索クエリ-URL-検索クエリで参照すると共起度の高い言葉のリストができる．共起している割合が多い言葉の集合は，概念体系が近いところにある言葉と言えるので，更なる応用も検討したい．

提示するキーワードに，集合知の要素を持った辞書を用いている．また提示する URL の入手先も，PageRank のようなリンクを張った人々の総意によってランキングされた結果を用いているので集合知の要素を含んでいる．そして，これらをデータソースに用いた QRS によって得られる結果は，集合知を掛け合わせたものであり，素性が幾分不透明なものであると言えよう．ところが QRS は，人間がなんとなく繋がる理由を理解できる検索結果を返すことが多い．検索者の意図を汲み取り，検索行為の文脈に合わせ，的確な検索結果を瞬時に返す検索エンジンは非常に有用であろう．しかし，自分の検索行動によって辿りついたウェブコンテンツから，一歩引いたところで周りを俯瞰し，それによって検索対象に到達できるような検索エンジンもまた役に立つのではないだろうか．

構築したシステムは，`http://rhodes.ex.nii.ac.jp` にて公開している．

参考文献

[丹 07] 丹, 大向, 武田:QueReSeek:検索履歴の逆引きによるコミュニティベースの Web ナビゲーション, 人工知能学会全国大会 (第 21 回) 論文集 (2007).

[丹 08] 丹, 大向, 武田:QueReSeek:検索履歴共有によるコミュニティ指向の連想検索, 人工知能学会全国大会 (第 22 回) 論文集 (2008).

[Google 09] Official Google Webmaster Central Blog: Specify your canonical, `http://googlewebmastercentral.blogspot.com/2009/02/specify-your-canonical.html`

[Yahoo 09] Yahoo! Search Blog, Blog Archive, Fighting Duplication: Adding more arrows to your quiver, `http://ysearchblog.com/2009/02/12/fighting-duplication-adding-more-arrows-to-your-quiver/`

[Microsoft 09] Live Search Webmaster Center Blog : Partnering to help solve duplicate content issues, `http://blogs.msdn.com/webmaster/archive/2009/02/12/partnering-to-help-solve-duplicate-content-issues.aspx`

[Iwazume 08] Iwazume M., Kaneiwa K., Zettsu K., Nakanishi T.,Kidawara Y. and Kiyoki Y.: KC3 Browser: Semantic Mashup and Link-free Browsing, Proc. of the 17th International World Wide Web Conference (WWW 2008).