「Namazuを用いた検索システムの構築について」

Namazu[1]を用いた検索システムの構築について(報告)

教育研究システム課
三浦　淳

1. はじめに

WWW を利用して自分が必要とする情報をみつけることはなかなか難しい。これは WWW 全体を対象とする場合のみならず、任意の WWW サーバーを対象とした場合においても同様である。そこで、その情報を見つけるための補助的な手段として全文検索型の検索サービスが多く利用されている。

現在、www.meiji.ac.jp でも、より適切な情報の提供が行えるよう全文検索型の検索サービスを運用している。しかし、利用者からは以下の点が指摘されている。

- www.meiji.ac.jp以外のMIND内のサーバーにある情報が取得できない。
- 遅い。(www.meiji.ac.jp の情報を検索する場合、ときにより MIND 外の検索システムを利用した方が早く結果が得られることが多い。)

MIND 外の検索システムを利用すれば、それらの問題は解消されることも無いわけではないが、それらは収集している情報量が多いが故に、その更新周期が長く、「リンク切れ」にでくわす可能性も大いにある。

そこで、これらの点を解消することを目的として Namazu と Kakasi[2]、wget[3]を中心とした検索システムの構築を試みたので報告する。

2. 基本となるシステム

2.1. ロボットの利用

前述の問題のうち、www.meiji.ac.jp 以外の MIND 内 WWW サーバーにある情報の検索、ならびに「リンク切れ」に対しては、単純に各 WWW サーバーへ頻繁にアクセスし、その時々に存在する情報のみを取得するように情報収集の仕組みを構築することで解決できる。

これには「ロボット」と呼ばれるソフトウェアの利用が大変有効である。ロボットとは、あるページの情報をひとまず取得した後、そのページ内に記載されているリンク先の情報を読み取り、順次ページの収集を繰返すようなソフトウェアの総称である。

そこで、今回の検索システムで情報の収集にはロボットを使用することとして、

- wget
- httpdown[4]
- WWWcp[5]

の三つのフリーソフトウェアを試用した。

各ソフトウェアとも機能的には遜色なく、今回のシステムで用いるに十分であると思われたが、httpdown と WWWcp はここ数年、バージョンが変化していないようであった。また、速度という点では若干ではあるが、wget が速かった。

よって、今回のシステムでは、継続してメンテナンスが行なわれている wget を利用することとした。

2.2. 検索エンジン

次に検索システム本体であるが、これについては、フリーソフトウェアであり、かつ速度の点で優れているといわれている、

- Namazu
- Freya[6]

の二つを試用した。[7]

どちらも収集した情報を元に、ファイルとそのファイル内に記述されている単語についての対応データー(インデックス)を予め作成しておき、検索要求に対しては、そのインデックスを参照して、情報を応答するものである。

それぞれの長所・短所は以下の通りと考える。

- Namazu について
  - 長所
    - 各種フィルターを用いることにより、HTML 形式やテキスト形式以外の文章も検索の対象とすることができる。(例 PDF 形式や MS-Word 形式)
    - 利用者が多く、持続的に開発が続けられている。
  - 短所
    - 文書中の単語を抽出するために、与えられた文章を単語に切り分ける(分か

ち書き)機能を持つソフトウェアが、別途必要である。

・Freyaについて
  ・ 長所
    ・ ODIN[8]で使われていたものであり、速度並びに扱えるデーター量の点で実績がある。
    ・ 単語の抽出はN-gram方式によるため、分かち書きのためのソフトウェアが必要ない。
  ・ 短所
    ・ 現在は個人でメンテナンスされており、ここ数年はバージョンアップがなされていないことから、今後の発展性に乏しい。
    ・ HTML 形式とテキスト形式にのみ対応している。

両者の最も大きな違いはそのインデックスの作成方法にある。

Namazuは、与えられた文を予め用意した辞書とつき合わせながら単語に分割し、その単語をインデックスに記録する。また、辞書に記載の無い単語は適当に分割した上でインデックスに登録する。

一方、FreyaはN-gramによる方法、つまり、文中のある文字に着目し、そこから始まる長さN(自然数)の文字列を抽出するといった方法を用いている。よって理論的には任意の語を抽出してインデックスに登録できるようになっている。

この違いは、それぞれのソフトウェアを利用する場合の、検索方法に影響を与える。

Namazuの場合は、基本的に「単語」単位でインデックスが作成されているので、検索も単語単位での検索の必要がある。この場合、複合語に対しては、それを and/or などによって組み合わせて行なう方法が基本となる。これは利用者にはわかりやすい反面、ノイズを増加させやすく、目的の情報を得るためには絞込みの手間がかかることも多い。

一方のFreyaの場合は、複合語もそのままインデックスに登録されるので、検索式にも任意の語(や文)を指定することが可能となり、ノイズも小さく抑えることができる。(ちなみに、現在www.meiji.ac.jpで提供されている検索システムも、これに似た方式を採っている。)

これだけであればFreyaの方がNamazuよりも勝っているともいえるが、実際には、Namazuは複合語が検索式に指定された場合、それを内部で分割して(正確さには欠けるものの)フレーズ検索を行なうようになっているので、Freyaのノイズの小ささは、それほど大きなアドバンテージにはなっていない。

また、N-gram方式の場合、無意味な文字列を単語としてインデックスに登録することもありえるので、多少の無駄が生じる。

さらに、WWWはその性質上、どのような形式のデーターでも存在できるので、HTML形式やテキスト形式以外のデーター形式も検索可能なシステムが望ましい。

以上のことから、扱えるデーター形式の豊富さ、今後の発展性、検索サービスの継続性といった点を重視し、Namazuを採用することとした。

2.3. 分かち書き用ソフトウェア

Namazuを採用することにしたため、分かち書きのためのソフトウェアの用意が必須となる。現在Namazuで利用できるものは以下の二つである。

- Kakasi
- ChaSen[9]

Kakasiは入力された日本語の文章を、漢字で始まる単語とその読み方が記載されている「辞書」を基に、「ひらがな(カタカタ、Roma-ji)」の文章に変換することを目的としたソフトウェアであり、その変換の過程で文章に対して分かち書き(というよりは語の識別)を行なう。Namazuは、この機能を利用することにより、単語を抽出、インデックスを作成する。

一方、ChaSenは辞書を基に分かち書きを行なった上で、各単語の品詞などの情報を利用者に提供する(形態素解析を行なう)ためのものである。

これらを比較した場合の、それぞれの長所・短所は以下の通りと考える。

・Kakasiについて
  ・ 長所
    ・ 辞書の保守が容易である。(基本的に、漢字で始まる単語とその読みからなっている。)
  ・ 短所
    ・ 辞書を基にした単純な「最長一致法」によるため、単語の認識が正確さに欠ける。
    ・ 「ひらがな」を分割しない。このため、ひらがなで始まる単語等は認識しない。

・ChaSenについて
  ・ 長所
    ・ その目的上、分かち書きがKakasiよ

りも格段に正確であり、より良質のインデックスの作成が期待できる。

・短所
  - 一度に処理できる文章サイズに仕様上の上限がある。
  - 辞書の保守は、単語に対する品詞等を確定する必要等があるため難しい。

Kakasi は先にも記したとおり文中の「漢字」を「ひらがな等」に変換することを目的としている。よって、単純に語の分割可能な位置を見つけることができればよく、それが分かち書きとして不正確であっても変換に支障がなければよい。つまり、Kakasi をもとにインデックスを作成した場合、必ずしも妥当なインデックスになるとは限らないという問題がある。(後述)

もう一方の ChaSen だが、こちらが分かち書きを行なう際に利用する辞書には単語に対する「コスト」が定義されていて、ChaSen はこれを利用して分割位置の違いによる文や語のコストを求めることにより、分割位置を決定する方法を採っている。このため、かなり正確な分かち書きが可能であり、当然、インデックスの品質は良いと考えられる。

良質なインデックスを用意するということは検索システムを構築する上で重要な点であり、その意味で Kakasi よりも ChaSen の方が格段の望ましいことは間違いない。しかし、ChaSen はそのシステムの仕様上、Namazu と組み合わせた場合に大きいデーターの処理ができないため、安定した運用、特に自動化には不向きである。(読み込むデーターが大きいとダウンするので、処理そのもののやり直しとなる。)

ChaSen を利用することに伴うこの問題を解決する手段として、例えば Namazu のインデックス作成部分に手を入れ、

- 分かち書きを実行する直前にサイズを確認し、ChaSen で処理できないサイズのデーターのみを例外的に Kakasi で処理する。
- ChaSen が大きいデーターを受け付けられるようにする。

等は可能である。

しかし、前者の場合、文書によって単語の抽出方法が異なってしまう点が問題であるし、後者については、結局はいたちごっこになってしまう。また、インデックスの品質が良くも悪しくも辞書に影響される以上、単語の同定に使われる辞書の保守は容易であることが望ましい。

よって、安定運用と辞書の保守の容易さ、ならびにサービスの継続性を重視し、当面は Kakasi を採用することとした。

3. システム構築と運用テスト

3.1. システム構築

当初、検索システムは www.meiji.ac.jp 上に構築する予定であったが、wget による情報収集を行なうことにしたため、将来的にはディスクがかなり必要になることが予想された。

そこで、www.meiji.ac.jp と検索システムは分離することとし、専用の検索システム用マシン(以下、検索サーバー)を用意して、そこへ

- wget により収集した情報を蓄積する
- Namazu 本体も、そのマシン上に実装する

こととした。また、検索サーバーのセキュリティー面を考慮して、それ自体は MIND アクセスレベル 1(設定上は 2)での運用が可能となるようにした。具体的には前記のソフトウェアの導入の他に、以下作りこみを行なった。

- 検索サーバーへのアクセスは www.meiji.ac.jp が行なう。
- 検索サーバーは基本的に MIND 内からの要求のみに応答するようにする。

- 現行検索システムは検索結果として、ページを作成した部書名を表示するので、この機能を加える。

こうしてでき上がったシステムの概要を図 1 に、このシステムの外観となる検索用のページと検索結果の表示のページを図 2,3 に示す。

3.2. 運用テスト

現在このシステムでは、およそ 400MB、24,540 件のデーターが検索対象として登録されているが、検索を実行しても、ストレスを感じることなく応答が得られ、速度の面では申し分ないといえるシステムであると考えている。

このシステムでは wget による情報の収集に最も時間が費やされる。常に新鮮なインデックスを用意するためには、wget を頻繁に実行することが必要になるが、対象となる WWW サーバーに CGI により生成されるページが多数ある場合には、そのサーバーに高い負荷を与えてしまう。それを防ごうと wget の実行間隔を空けた場合には、インデ

ックスの鮮度が落ちることになる。

幸いなことに、このシステムは MIND 内の WWW サーバーのみを対象としているので、サーバー上の情報の構成や更新頻度に関する事前・事後のリサーチが可能である。よって、これを行なうことで wget の動作と実行のスケジュールを決定し、WWW サーバーへの負荷を低減とインデックスの鮮度を維持することとした。

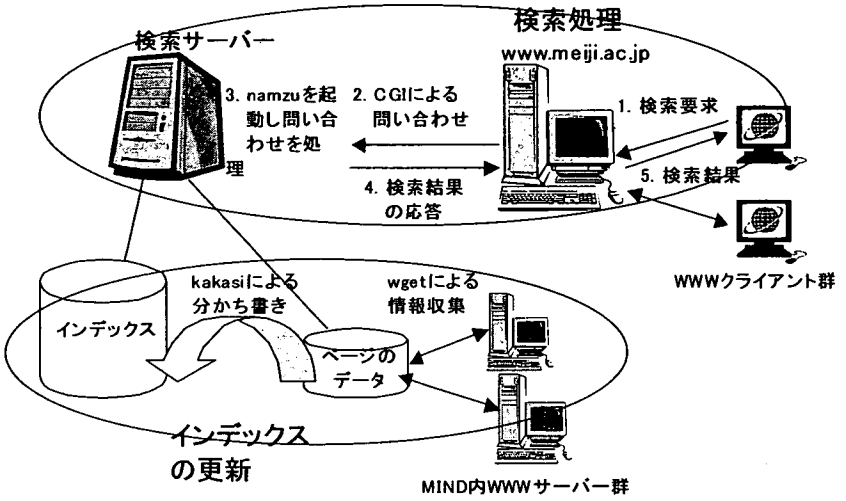

図 1 システム概要

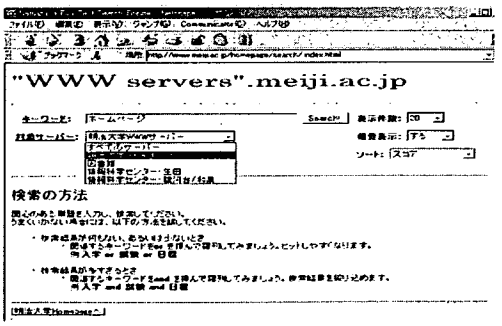

図 2 検索用ページ

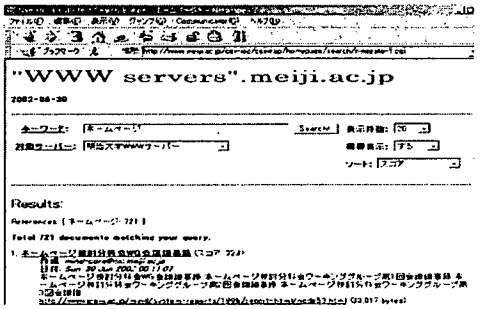

図 3 検索結果

4. このシステムの欠点とその対策

4.1. 辞書のメンテナンスとの関係

検索システムの性能を評価する際に使われる言葉に、「再現率」と「適合率」という言葉がある。「再現率」とは全体のなからからどれだけ該当するとみなすべきデーターを抽出できるかをさす言葉で、検索漏れがない場合に再現率は 1 となる。また「適合率」は、検索結果として提示されたデーターのすべてが、利用者が意図した検索結果として妥当なものである場合(つまりノイズが無い場合)に 1 となる。

この二つはどちらも 1 ならば理想的であるが、実際にはどちらかを重視してシステムを構築しなければならないのが実情である。

例えば、「川崎市役所」という文字列に対しては、Kakasi は最初に「川崎市」を抽出し、ついで「役所」と抽出する。このため、利用者が「川崎市役所」を検索する目的で「川崎 and 市役所」として検索してしまうと、「データーが無い」旨の応答がなされてしまう。(図 4)

この場合、「川崎市役所」あるいは、「川崎市 and 役所」なら検索結果を得られるが、そもそも、後者の and 検索のような位置でこの語を分割する方はいるだろうか？いなければ、結局のところ「データは無い」ということになる。(再現性が悪い。)

これを「川崎 and 市役所」で検索できるようにしたい場合には、辞書から「川崎市」を削除すればいいことになる。そうすれば、「川崎市役所」でも「川崎 and 市役所」でも検索可能にり、検索漏れが減る(再現性は良い)。

ところが今度は、「横浜市役所勤務の川崎さん」も「川崎 and 市役所」に該当してしまうため適合性が悪くなってしまう。

また、辞書に「川崎市役所」を追加した場合には、利用者が最初から「川崎市役所」と検索式に指示しないかぎり、該当データーは絶対に検索対照とはならなくなってしまう。

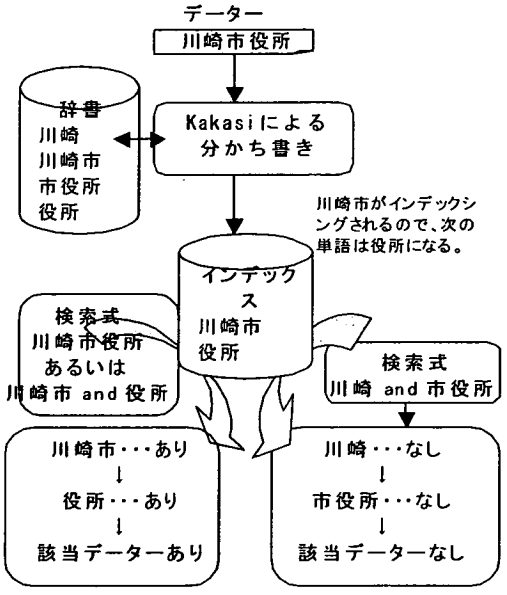

図 4 辞書更新前

4.2. 不正確な分かち書きへの対策

以上に記した具体的な例にあるような問題は、Kakasi による分かち書きに関する問題である。これらはすでに記したように、

1) 複合語は辞書から削除する。
2) (逆に)複合語は辞書に追加する。

といった方法である程度は解消できる。特にシステムが収集する情報が特定の分野に特化したものであれば、用語等が明確であるため辞書のメンテナンス自体はある程度簡単になと思われる。しかし今回のような汎用的な検索システムの場合は、辞書の保守に関しては経験によるしかなく、その作業はそれほど簡単なことでもない。

また、

3) 複合語はそれ自体をインデックスに登録した上で、さらにそれをいろいろなパターンで分割したときの結果もインデックスに登録する。

という方法が、おそらくは理想的であると思われるが、Kakasi や Namazu のソースコードを大幅に変更しなければならず、また、最終的には分割の際には辞書に頼る必要がでてしまう。

実は、「複合語の分割位置」に関する問題の大半は、ChaSen を用いることにより実用的には問題ない程度に解消できる。それは Chasen が形態素解析を目的としているために分かち書きが相当に正確だからである。しかし、ChaSen を利用したインデックスの作成も先に記したように安定運用の点で問題があるし、仮にそれが解消されたとしても「分かち書き」によるシステムである以上、Kakasi のそれよりも良さそうにだというだけで、インデックスの作成の問題そのものが解消されているわけではない。

よって当面は利用者には、「最初は、複合語は分割せずに検索を行い、必要に応じて徐々に検索範囲を広げていくと、検索がうまくいくことが多い」等のアドバイスを行うとともに、入力された検索式を基に辞書の保守をうまく行い、MIND 内の情報がより適切に提示されるようにするより他にないと考えている。

4.3. インデックスに登録される言葉の統一

今まで触れてこなかったが、もう一点、単語の正規化の問題がある。具体的にいうと「コンピューター」と「コンピュータ」は、意味としては同じものであるがシステム的には異なるものであるため、それぞれの単語で検索した場合の検索結果が異なるという問題である。

一部の検索システムでは、この問題には、例えば長音記号は一律に削除する等といった方法で対応しているらしいが、統一的な方法というものは無いらしい。

これについては、Namazu(あるいは Kakasi)にはその機能がないので、実装するならば、別途作り込みを行う必要がある。

5. 今後について

現在は ChaSen に若干手を入れて、ChaSen によるインデックスの作成・システムの安定運用を試している。今後はさらに、システムの大幅な変更も視野にいれつつ、

- 4.2. 記載の方法 3) によるインデックスの作成
- 同義語への対応

等について、試験的に取り組んでみたいと考えている。

以上で、Namazu を用いた検索システムの構築の報告を終る。

Reference

1 「なまず」。http://www.namazu.org 参照。

2 「かかし」。http://kakasi.namazu.org 参照。

3 http://www.gnu.org/software/wget/wget.html 参照。

4 http://www.mechatronics.mech.tohoku.ac.jp/~kumagai/bins/kuma/kumabins.html 参照。

5 http://www.ff.iij4u.or.jp/~rewsirow/WWWcp/WWWcp.html 参照。

6 http://www.ingrid.org/ja/project/freya/ 参照。

7 実はこれらのほかに freeWAIS-sf も試したが、WAIS プロトコル自体が Internet では殆ど使われなくなっていることから除外した。

8 国内では老舗の検索エンジン。現在は休止中 (http://odin.ingrid.org/)。

9 「茶筅(ちゃせん)」。http://chasen.aist-nara.ac.jp/ 参照。