平成17年度

筑波大学第三学群情報学類

卒業研究論文

題目 ブラウジングしているページを
自動連想検索するインタフェース

主専攻　　情報科学主専攻

著者　　小澤 崇記

指導教員　田中 二郎　志築 文太郎　三末 和男　高橋 伸

要　旨

ブラウジングしている Web ページに関連する情報を知りたいと思ったとき、通常用いられる検索エンジンを用いては、満足のゆく情報をすぐに得ることは難しい。キーワード型検索エンジンでは、ユーザが自ら検索クエリを考え入力する必要があり、ディレクトリ型検索エンジンでは、どのカテゴリにそのページが属しているかを考えながらより詳しいカテゴリへ潜っていくという探索を行う必要があるからである。

本研究では、以上の問題点を解決するために次の二つの特徴を持つインタフェースを提案する。まず、ブラウジングしているページと、ディレクトリ検索エンジンに登録されている各カテゴリとの類似度を自動で算出する。そして、類似度の高いカテゴリに登録されている Web ページ群をユーザがブラウジングしているページと併せて提示する。その結果、ユーザは通常のブラウジングが今まで同様に行えると同時に、ブラウジングしている Web ページに関連する質の高い Web ページを、検索を行うことなく必要に応じて得ることが可能でなる。本研究では、提案インタフェースを備えたプロトタイプシステムとして「あ〜る Navi」を実装した。

目 次

図 目 次

表 目 次

第1章　序論

インターネットが一般家庭にも普及するようになり誰でも触れることができるようになった。それ故に、誰でも Web 上に情報を発信することも受信することもできるようになったため、インターネット上の情報量及び Web ページの数は急速に増加し続けている。そんな中、インターネットは情報を収集する際には欠かせない道具となり、世界中のありとあらゆる情報にインターネットを通じて容易に触れることができるようになった。インターネット上の膨大な量の情報の中から、ユーザは有益な情報を取捨選択するために検索エンジンを使うことが多い。キーワード型検索エンジンの最大手である Google には 80 億以上もの URL がインデックスとして存在する。さらに検索クエリに関して、英語圏の Web サーチエンジン Excite の場合、検索クエリの長さは平均 2.21 語である [1] という報告がなされている。また、日本語では複合語が用いられるため，検索語数はより小さくなる。日本語 Web ページを主な検索対象とする Web サーチエンジン ODIN において、1ヶ月に用いられた検索質問に含まれるクエリは平均 1.40 単語であり、検索質問の 7 割以上が 1 つのクエリのみから構成されるという事実もある [2]。

以上のことから、検索クエリが短いことにより絞込みの条件が少なくなるため、検索エンジンを用いて検索を行う場合、検索結果として提示される Web ページの数は膨大になることが多い。膨大な数の Web ページの中からユーザが自身にとって有益な情報を取捨選択するためには時間がかかる。また、ディレクトリ型検索エンジンは、登録されている Web ページはキーワード型検索エンジンと比べると少ないが、どのカテゴリにブラウジングしているページが属しているかを考えながらより詳しいカテゴリへ潜っていくと言う探索を行う必要がある。ブラウジングしているページに関連する情報を見つけ出すために、ユーザは膨大な Web ページの中から目的の Web ページを検索することをブラウジングしているページに関連する情報を知りたいと思った度にする必要がある。何度も何度も似たような作業を繰り返し、なおかつ、かかった時間に見合った質の情報が見つかるとは限らない。そのため、ブラウジングしているページに関連するページを見つけ出すことはますます困難になっていくであろうと予想される。

本研究の目的

本研究の目的は、ブラウジングしているページに関連するページの検索の現状を理解し、その問題点を解決するシステムを開発することである。そのために本研究では、ブラウジングしているページの解析手法、そのページに関連するカテゴリの算出手法を提案し、類似度の

高いカテゴリに登録されている質の高い Web ページ群を取得する。さらに、ユーザがブラウジングを今まで同様に行えると同時に、検索を行うことなく必要に応じて取得することが可能になるように、ブラウジングしているページに関連するカテゴリとそのカテゴリに登録されている Web ページ群をブラウジングしているページと併せて提示する手法を提案する。

本論文の構成

本論文の構成を述べる。第 2 章では我々が日常的に行っている Web 検索及び検索インタフェースについて考察し、その問題点を既存のインタフェースと比較しながら述べる。第 3 章では、問題解決手法として関連ページ提示システムを提案し、その特徴と利用する要素技術について説明する。第 4 章では、関連ページ提示システム「あ〜る Navi」のシステムの流れと詳細について説明し、第 5 章ではシステムの実用例を挙げる。第 6 章では関連研究について触れ、第 7 章で結論を述べる。

第2章　Web検索の現状と問題点

本章ではまず日常のWeb検索の現状を考察し、さらにその問題点について述べることで本研究の目的をより明確にする。

2.1　Web検索の現状と考察

Webページをブラウジングしている最中に興味を惹かれる情報を見つけたとき、その情報に関連する情報やより詳しい情報を知りたくなることがある。例えば、友人のWebページをブラウジング中にレビューされていた本に興味を引かれたとき、その本に関連する情報（売っているお店、他の人のレビュー、同じ作家の別の本等）を知りたくなる。その際、検索エンジンを用いて関連があると思われるキーワードを抜き出し検索を行う人が多い。例えば、Google[4]に代表されるキーワード型検索エンジンであれば、キーワードを入力し“検索”ボタンをクリックすることで検索結果を得ることができる。また、Googleには“I’m Feeling Lukey！”ボタンをクリックすることで会社名などの一般的な検索をする場合は検索結果最高位のページに到達することができる機能もある。これらの機能は検索クエリが適当であるならば関連する情報を手に入れることができる。しかし、必ずしも最も質のいいWebページが検索結果の最高位にあるとは限らない上、検索クエリとして適当なものをユーザが考え付けるかという問題もある。検索クエリが適当であったかどうかWebページのタイトルと要旨だけで判断することは難しい。なので、上位いくつかのページを検索の度に見てそのキーワードが適当であったかどうか判断することになる。一回の検索でブラウジングしているページに関連する情報を手に入れられなかった場合、再度検索クエリを考え検索をする必要がある。この一連の作業の流れを図に表すと図2.1のようになると考えられる。Yahoo![5]に代表されるディレクトリ型検索エンジンであれば、一番上位のジャンルカテゴリを選択し、より詳しいカテゴリへと潜っていくという探索を行う。または、最初にキーワードを入力しカテゴリを絞り込んでからより詳しいカテゴリに潜っていく。もしブラウジングしているページが属すると思われるカテゴリがなかった場合や、ユーザが考えているカテゴリと実際に分類されているカテゴリが違った場合、再度上位のカテゴリに戻り探索をする必要がある。この場合の動作の流れも図に表すと図2.1のようになると考えられる。

ブラウジングしている最中に興味を惹かれる情報を見つけ（検索要求の発生）、頭の中でどのようなキーワードが検索クエリとして適当であるか考え入力し（検索クエリの作成）、“検索”ボタンを押すかカテゴリを潜る（検索クエリの実行）。得られた検索結果と今ブラウジングしていたページとを見比べそのWebページで満足できるか考える（検索結果の収集）。検

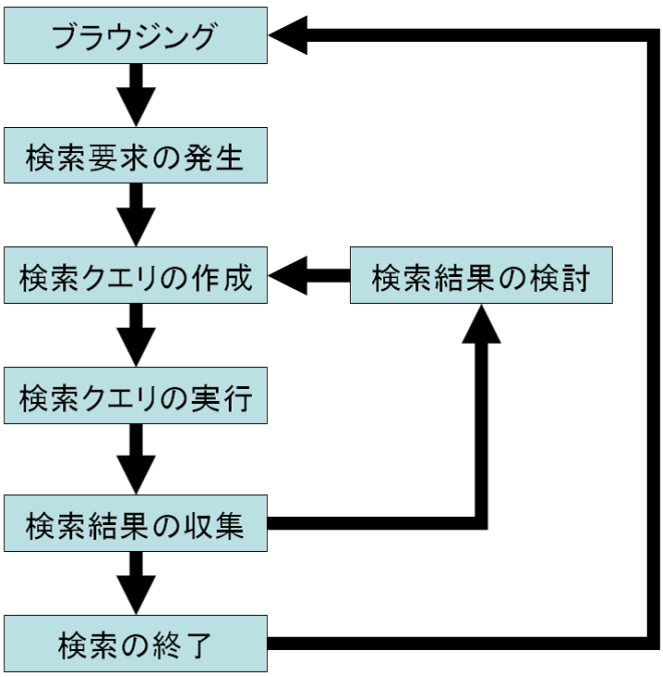

図 2.1: ユーザの検索要求が発生してからブラウジングに戻るまで

索結果に満足できた場合は検索を終了しブラウジングに戻る。検索結果に満足できなかった場合、検索結果やブラウジングしていたページから検索クエリを検討し再度キーワードを入力する。検索エンジンを使い慣れている人であれば、一回でブラウジングしているページに関連する Web ページにたどり着けるか、もしくは検索クエリを作成し実行し、検索結果を収集し検討するサイクル数が少ない。しかし、検索エンジンを使い慣れていない人の場合、関連するであろうキーワードを考え付けずに何度も試行錯誤を重ねても目的のブラウジングしている Web ページに関連する Web ページに到達できない場合もある。それは、Web 検索で用いられる検索クエリが短く、絞込みの条件が少なくなるために検索結果が膨大なものになるからである。

以上のような Web 検索の現状から一般的な検索エンジンが提示するような入力インタフェース及び検索結果提示インタフェースは不十分であると考えた。

2.2 関連ページ検索における既存インタフェースの問題点

ブラウジングしている Web ページに関連するページを検索する際、ユーザはブラウジングしている Web ページを眺めその中身と今まで移動してきた Web ページの中身等を頭の中で整理する。そして、どのようなキーワードが検索クエリとして適当であるか考える必要がある。しかし、「この Web ページに関連したページを見たい」と言う検索要求の「この Web ページ」、

つまりブラウジングしているページがどんなページであるかという情報は抽象的なものである。また、「この Web ページ」の内容が必ずしも一つのジャンルに絞られるとは限らない。たとえば、近年注目されてきたブログ等のページであれば様々な話題に触れている場合がある。その場合、ユーザは興味があるもの全てに関してキーワードを考え検索を行う際に、検索エンジンと「この Web ページ」とを見比べるために何度もページ間を行き来する必要がある。複数の Web ページを見比べるときは一画面に収まっていて同時に見ることができるほうが情報を収集しやすい。そのうえ、「このページ」に載っていたキーワードが「このページ」に関連する Web ページを検索するための検索クエリとして適当でない場合もある。そのページ特有の言い回しや単語などが適当でない例として挙げられる。

また、検索をして行き着いたページにユーザは満足できないかもしれない。その時は、再度同様の検索を行う必要がある。興味が引かれる情報があった度に検索をする。それでも、かかった時間に見合った質の Web ページにたどり着けるとは限らない。

これらの問題点を解決し、ユーザがブラウジングしている Web ページに関連するページを必要に応じて得ることができるようにするには、ユーザがブラウジングしているページに関連する情報を自動で検索し、ブラウジングを今までと同様に行えるようにブラウジングしているページと併せて提示する必要がある。

第3章　関連ページ提示システム

本章ではまずブラウジングしているページに関連するページを検索する際における既存インタフェースの問題点を改善する方法として、関連ページ提示システムを提案する。そして、システムを実現するためのインタフェースとしての目標とシステムの目標を掲げた。次に提案システムと他の類似インタフェースとの相違点を述べ、さらに提案システムに用いた要素技術をおのおの簡単に説明する。

3.1　関連ページ提示システムのための提案

本研究では、前章に述べた関連するページの検索における既存インタフェースの問題点を改善する方法として、関連ページ掲示システムを提案する。提案システムのインタフェースとして既存インタフェースの問題点を改善するために、以下の 2 点が必要であると考えた。

- ユーザが検索を行うことなくブラウジングしているページに関連する情報を得ることができるようにする
- ユーザのブラウジングは今までと同様に行える状態で、必要に応じてブラウジングしているページに関連する情報を得ることができるようにする

ユーザが検索を行うことなくブラウジングしているページに関連する情報を得ることができるようにするため、検索クエリを考え実行し、検索結果を収拾し検討する一連のサイクルを図 3.1 のようにシステムが自動で行う。つまり、システムは、ブラウジングしているページに関連する情報を自動で検索し、その検索結果を自動で提示する。また、ブラウジングを今までと同様に行える状態にしたまま、ユーザが必要に応じてブラウジングしているページに関連する情報を得ることができるようにブラウジングしているページと併せて提示する。そして、上記の二つの目標を達成し、提案システムを実現するためには、システムは以下の 2 点が必要であると思われる。

1. ユーザがブラウジングしているページに関連の深いカテゴリを自動で算出
2. カテゴリをツリーを利用してサイドバーに提示

ブラウジングしているページに関連する情報を検索し、検索結果を動的に分類する場合、テキストや要旨などから抜き出すのが一般的であるが、抜き出された物の中に単体では関連があるかどうか分からない物も含まれる場合がある。そのため、あらかじめ分類されている状

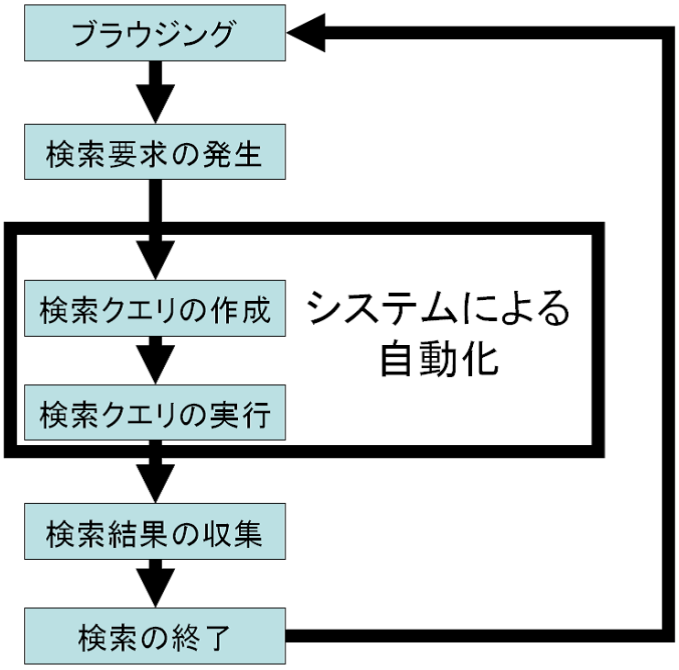

図 3.1: 本研究のシステムを使った際のユーザの検索要求が発生してからブラウジングに戻るまでの流れ

態のカテゴリを用いることによって単体でも意味のあるキーワードとしてユーザに提示する。1 はブラウジングしているページを即座に自動で分析し、登録されているカテゴリとの類似度を算出し、その値が高いものをユーザが操作することなく自動で取り出すことを目的とする。2 は 1 によって得られたカテゴリをユーザのブラウジングが今まで同様に行え、なおかつ、必要に応じて関連する Web ページを得ることができ全体像も理解できるように提示することを目的とする。

さらに、この様な機能を実装したシステムを利用することで、ユーザはブラウジングしているページがどのようなカテゴリに関連しているか理解し、関連する質の良いページを収集する効率を向上させることができるようになる。ここで言う「質の良いページ」とはユーザがブラウジングしているページが関連するカテゴリにおける信頼度の高いオフィシャルページを指す。

3.2 提案手法と他の類似インタフェースのとの相違点

ここで、提案システムと他の類似インタフェースとの相違点について述べることで本手法の特徴を明らかにする。他の類似インタフェースとしては、検索結果をクラスタリングして検索結果と同時に提示する Vivisimo[6] という検索エンジンが挙げられる。Vivisimo はキーワー

図 3.2: Vivisimo の検索結果掲示インタフェース

ド検索して得られた URL をタイトルでクラスタリングし、その結果を URL と同時にユーザに提示している。図 3.2 に検索クエリ「アーチェリー」として Vivisimo を用いて検索を行った検索結果掲示インタフェースを示す。

「アーチェリー」で検索した結果、全検索結果 115,470 件のうち上位 181 件をクラスタリングした結果を左側に表示している。Yahoo!等のディレクトリ型検索エンジンに比べれば掲示されている数は多いが、クラスタリング結果はタイトルや簡単な説明からクラスタリングしているので、「ようこそ」や「スケジュール, ケンタウルス」等検索クエリと関連があるかどうか定かでない項目ができてしまっている。さらに検索エンジンである以上、キーワードを入力しなければならず、ブラウジング中のページと検索結果を画面を切り替えて見比べなければならない。

ここで本研究と Vivisimo との相違点を表 3.1 にまとめてみる。まず、分類の方法であるが、Vivisimo では動的に分類を行うが、本研究ではあらかじめ取得したカテゴリのデータを使うため静的であり、単体でも意味の通る項目が提示される。また、Vivisimo が提示する Web ページは全 Web ページからキーワード検索した結果であるが、本研究ではカテゴリに登録されている Web ページを用いる。これにより、ブラウジングしているページに関連する質のいい Web ページを提示することができる。Vivisimo は検索エンジンであるため、ユーザは検索クエリを作成し実行し、検索結果を収集し、検討する必要があるが、本研究はシステムが自動でブラウジングしているページに関連のあるカテゴリと Web ページを提示するため、ユーザが検索する必要はない。また、ブラウジングしているページに関連する情報を入手するまで、Vivisimo では検索を繰り返すため時間がかかってしまうが、本研究では、ブラウジングしている Web ページと同時に提示するため時間はかからない。さらに、Vivisimo の場合ブラウジングしているページと検索結果の内容を比較するためにページを切り替える必要があるが、ブラウジ

	Vivisimo	本研究
分類のタイミング	動的	静的
提示される Web ページ	全 Web ページから検索	登録されている Web ページ
ユーザが行う操作	検索クエリの作成と実行＋検索結果の収集と検討	検索された結果の収集のみ
ブラウジングしているページに関連する情報を入手するまでの時間	時間がかかる	すぐに手に入る
通常のブラウジングが今までと同様に行えるか	行えない	行える

表 3.1: 本研究と Vivisimo との相違点

ングしている Web ページと併せて提示しているため本研究では通常のブラウジングが今までと同様に行うことができる。

3.3 本システムで利用する要素技術

本節では、ウェブページの分析、関連するカテゴリの算出をするための手法について説明する。

3.3.1 類似度

文書検索において、その文書が他の文書とどのくらい類似しているかを計る度合いであり、値が大きいほど二つの文書が似た内容であるということを表している。

3.3.2 形態素解析

文書の特徴を分析するにはその文書がどのような語句を含んでいるか調べる必要がある。そのために用いられる技術が形態素解析である [7]。形態素解析とは文書の文字列を、単体で意味が通る最小の文字列に分解し、品詞、語形変化、読みなどの情報を追加する処理である。表 3.2 は「音楽とリズムは魂のもっとも深いところに至る道を持っている。」という文章に形態素解析器「MeCab」[8] を使って形態素解析を行った例である。

このように形態素解析を行って得られた語句から、もっとも特徴が現れると思われる名詞だけを抜き出し、出現頻度を計測した物を、文書の特徴とし、カテゴリの特徴を求めるために用いた。

音楽	名詞, 一般		音楽	オンガク, オンガク
と	助詞, 並立助詞		と	ト, ト
リズム	名詞, 一般		リズム	リズム, リズム
は	助詞, 係助詞		は	ハ, ワ
魂	名詞, 一般		魂	タマシイ, タマシー
の	助詞, 連体化		の	ノ, ノ
もっとも	副詞, 一般		もっとも	モットモ, モットモ
深い	形容詞, 自立	形容詞・アウオ段, 基本形	深い	フカイ, フカイ
ところ	名詞, 非自立, 副詞可能		ところ	トコロ, トコロ
に	助詞, 格助詞, 一般		に	ニ, ニ
至る	動詞, 自立	五段・ラ行, 基本形	至る	イタル, イタル
道	名詞, 一般		道	ミチ, ミチ
を	助詞, 格助詞, 一般		を	ヲ, ヲ
持っ	動詞, 自立	五段・タ行, 連用タ接続	持つ	モッ, モッ
て	助詞, 接続助詞		て	テ, テ
いる	動詞, 非自立	一段, 基本形	いる	イル, イル
。	記号, 句点		。	。,。

表 3.2: 「音楽とリズムは魂の最も深いところに至る道を持っている。」を形態素解析した例

3.3.3 tf・idf 法

文書を形態素解析することで、その文書の中の特徴語がどれぐらいの頻度で出現するかを求めることはできるが、全文書における一つの文書中の単語がどれほど重要であるかまでは分からない。そこで、tf・idf 法という手法が用いられる。

tf・idf 法を用いることによって、「ある単語の、その文書における文書集合全体を考慮した相対的な重要度」を算出することができる。文書 D_i の中の単語 t_j の重み（重要度）w_{ij} を以下の計算式で求める。

$$w_{ij} = tf_{ij} \times idf_j$$

tf_{ij} とは、局所的重みとも呼ばれる文書 D_i の中での単語 t_j の出現頻度を表現している。文書 D_i に単語 t_j が多く出現すればするほど、tf_{ij} は大きな値となる。

idf_j とは、大域的重みとも呼ばれ、単語 t_j が全文書集合の中に出現すればするほど小さな値となり、珍しい単語であれば大きな値となる。

まとめると、ある文書 D_i における単語 t_j の重み（重要度）は、単語 t_j が文書 D_i においてよく出現し、かつ文書集合中において出現する文書が少なければ大きくなるといえる。tf・idf の計算法については、様々なものが考えられるが、本研究で用いた計算式については、次章で述べる。

3.3.4　ベクトル空間法

ベクトル空間法とは、文書やクエリ、カテゴリの内容を多次元空間上のベクトルとして表現する手法である。これには tf・idf 法を用いて得た重みを適用する。m をカテゴリ集合全体の単語数、w_{kj} をカテゴリ C_k 中の単語 t_j の重みとすると、カテゴリ C_k はベクトル c_k で表現される。

$$c_k = [\, w_{k1}\ w_{k2}\ w_{k3} \cdots w_{km} \,]$$

このようなベクトルをカテゴリの数だけ計算し、ユーザが見ているページとの類似度を算出するために行列計算を行うこととなる。

第4章　「あ～るNavi」

関連ページ提示システムとして「あ～る Navi」を実装した。ソフトウェアのプログラミングには XUL、JavaScript、Ruby を用いた。実行環境及び実装に使用した装置は以下である。

コンピュータ　CPU:Celeron 2.53GHz, Memory:0.99GB

OS　Microsoft Windows XP

ブラウザ　Mozilla Firefox ver.1.5

4.1　システムの概要

「あ～る Navi」はサーバクライアント方式である。サーバプログラムとクライアントプログラムの相互関係を図 4.1 にまとめておく。本システムでは、クライアントであるブラウザからユーザがブラウジングしている Web ページの URL がサーバプログラムに送られる。同時にシステムに登録されているカテゴリをツリーで提示する。サーバプログラムは送られてきた URL から HTML ファイルを取得し HTML タグを取り除き、形態素解析器にかけテキストの名詞だけを抜き出す。そして、Web ページの特徴ベクトルとして各単語の出現頻度を計算し、前処理によってあらかじめ計算しておいたカテゴリの特徴ベクトルとの類似度を算出しクライアントに返す。最後にサーバから受け取った類似度を元にブラウザはツリーで提示されているカテゴリのうち類似度の高いカテゴリをハイライトして表示する。

サーバプログラム　プログラムは Ruby で実装され約 200 行である。
プログラムの流れは以下の通りである。

1. クライアントから送られてきた URL を HTML ファイルへ変換し HTML タグを取り除く
2. 形態素解析器にかけ名詞だけを抜き出す
3. 特徴ベクトルを作成する
4. カテゴリの特徴ベクトルとの類似度を算出する
5. 類似度をクライアントに送る

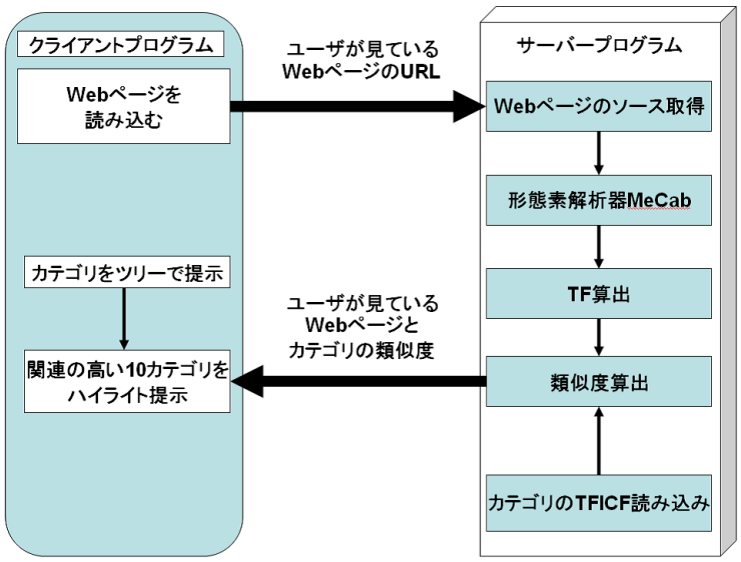

図 4.1: システムの流れと相互関係

クライアントプログラム　プログラムは XUL と JavaScript で実装され計約 42500 行である。以下の流れで Mozzila Firefox の拡張機能として動作する。

1. システムに登録されているカテゴリをツリーで提示する
2. サーバプログラムにユーザがブラウジングしているページの URL を送る
3. サーバプログラムからブラウジングしているページとカテゴリとの類似度を受け取る
4. 類似度を元にカテゴリをハイライト提示する

4.2　前処理

類似度計算を高速化するために各カテゴリの特徴ベクトルを事前に算出しておく。

4.2.1　カテゴリ内の Web ページの取得

カテゴリのデータは株式会社 Spline が無料で提供している CustomDir[9] というディレクトリデータベースを用いた。

各カテゴリに登録されている URL から HTML ファイルを取得し、HTML タグを取り除きテキストデータのみを抽出する。この処理には Ruby[10] の HTML スキャナである HTMLSplit[11] を用いた。

4.2.2 カテゴリ内の分析

カテゴリ内の Web ページの分析には前章で説明した要素技術である形態素解析と tf・idf 法を用いる。Web ページ毎の単語の出現頻度、全カテゴリ中の単語の出現頻度、各カテゴリの特徴ベクトルを前節で述べた手法を用いて算出する。形態素解析には、京都大学情報科学研究科で開発されたシステム「MeCab」を用いた。なお、MeCab は日本語のみの形態素解析を目的としていて、他言語の語句は全て未知語として処理される。次に形態素解析を行った情報を基にして、名詞だけを抽出し、Web ページをベクトル空間法で表現する。これには tf・idf 法の tf を用いた。さらに、カテゴリごとの単語の出現頻度を求めて、各カテゴリをベクトル空間法で表現する。Web ページ D_i における単語 t_j の出現頻度を tf_{ij} とし、n をカテゴリ内の Web ページ数とすると、カテゴリ C_h の中の単語 t_j の出現頻度 tfc_{hj} を以下の計算式で求める。

$$tfc_{hj} = \sum_{i=1}^{n} tf_{ij}$$

idf_j は、単語 t_j の全文書集合中における出現する文書数を基にした値であるが、本研究では全カテゴ集合中における出現するカテゴリ数を基にした icf_j を用いた。前節で求めた tfc_{hj} を用いて、単語 t_j のカテゴリ C_h における重み（重要度）を以下の計算式で求める。

$$w_{hj} = tfc_{hj} \times icf_j$$

icf_j に関しては以下の計算式で求める。

$$icf_j = \log \frac{N}{cf_{t_j}} + 1$$

icf_j において 1 を加えるのは cf_{t_j} が N であるとき、icf_j が 0 となってしまわないようにするためである。cf_{t_j} が小さい、つまり単語 t_j を含むカテゴリが少ないほど値が大きくなることを示している。対数をとるのは、icf_j が過度に変化するのを防ぐ目的がある。

単に tf・icf を用いると、長い Web ページを持つカテゴリに含まれる単語ほど重みが高くなってしまうという問題点がある。そのため、正規化を行った。正規化にはコサイン正規化を用いた。tfc_{hj}、icf_j を用いてコサイン正規化における正規化係数は以下のようになる。全カテゴリに含まれる単語の総数を m としたとき、m 次元ベクトル（$tfc_{h1}icf_1$,$tfc_{h2}icf_2$,$\cdots$,$tfc_{hm}icf_m$）の向きを変化させずに、ベクトル長を 1 にする処理である。

$$n_h = \sqrt{\sum_{j=1}^{m} (tfc_{hj} \times icf_j)^2}$$

まとめると、カテゴリ C_h 中の単語 t_j の重み w_{hj} は次式で求めることが可能となる。

$$w_{hj} = \frac{tfc_{hj} \times icf_j}{n_h}$$

こうした計算を行って、カテゴリそれぞれに対して m 次元の特徴を現すベクトルが与えられた。

4.2.3 類似度計算高速化のための処理

類似度計算を高速化するためにカテゴリの特徴ベクトルのうち、0 であるものを取り除く処理を行った。類似度計算において、各カテゴリとユーザがブラウジングしているページの特徴ベクトルの内積を取るため、どちらかが 0 であれば計算の必要はない。そのため、0 でないベクトルのインデックスと値の組だけを別ファイルに保存する。

4.3 Web ページ分析と類似度算出部

4.3.1 Web ページの取得

ユーザが Web ページをブラウジングする。すると、そのブラウジングしているページの URL がサーバプログラムに送られる。

4.3.2 HTML ファイルへの変換

次に送られた URL から Web ページを分析する。まずカテゴリ内のウェブページの取得と同様の手法で、URL から HTML ファイルを得て、HTML タグを取り除きテキストデータのみを抽出する。

4.3.3 類似度計算

HTML ファイルの分析には、前章で説明した要素技術である形態素解析とベクトル空間法を用いる。まず、取得したテキストデータを形態素解析し、名詞だけを抜き出し、各単語の出現頻度を求める。Web ページ一つだけの分析であるため、idf を計算する必要はない。この際、長い Web ページであればあるほど単語一つの重みが高くなってしまうので、コサイン正規化を行う。この場合の正規化係数は以下のようになる。

$$n = \sqrt{\sum_{j=1}^{m} (tf_j)^2}$$

全カテゴリに含まれる単語の総数を m とすると、単語 t_j の Web ページ中の重み w_j は

$$w_j = \frac{tf_j}{n}$$

となる。これにより Web ページの特徴ベクトルが与えられる。各カテゴリの特徴ベクトルと Web ページの特徴ベクトルの内積を類似度とした。

4.3.4　類似度計算高速化のための処理

類似度計算を高速化するために、カテゴリの特徴ベクトルと同様の処理を Web ページの特徴ベクトルにも行った。

4.3.5　類似度計算結果の考察

ここでは、類似度算出結果の考察を行う。本システムを用いて「あだち充の屋根裏部屋」[12] と「ゆんフリー写真素材　テーマ：大地」[13] いう 2 つの Web ページの類似度を計算し、類似度の高い上位 10 カテゴリを抜き出した結果を表 4.1 に示す。前者は文字が多く、後者は文字が少なく画像が主である。これを見ると、以下のようなことが分かる。

1. 文字の多い Web ページであれば、関連のあると思われるカテゴリの類似度が高くなる
2. 文字が少ないと全体的に類似度が低くなり関連のあると思われるカテゴリが分かりにくくなる
3. 文字の多い Web ページでも 2 番目以降上げられるカテゴリは、関連があるかどうか怪しくなる

1. に関しては望んでいた結果といえる。しかし、2. に関しては予想以上に類似度の差が出ていなかった。本研究の分類手法では、Web 文書内のテキストデータを形態素解析することで Web 文書の特徴としているので、文字が少ない Web ページだと特徴が少なく類似度に差が出てこなくなってしまう場合がある。3. に関しては少しでも関連があればそれは意外な Web ページを発見する手助けになると思われる。それでも、ジャンルが多岐にわたるときは全てを全体像と共に見える状態にしておくことはツリーで表示する場合縦に長くなり、関連がかえってつかみづらくなってしまう。

文章が少ない Web ページであると類似度が全体的に低くなってしまう問題と関連するカテゴリを提示するときに縦に長くなってしまう問題をインタフェースとして考えたとき、前者については画像などの文字が少ない Web ページに関してはユーザが見る時間はほぼそんなに長くないため無視できると考える。後者に関しては、最も類似度が高いカテゴリだけを見える状態にしておき、2 番目以降はそのカテゴリを含む親カテゴリをハイライトすることで解決できると考えた。

4.4　関連カテゴリ提示部

本システムでは、ユーザがブラウジングしている Web ページを分析して、関連カテゴリ掲示部で類似度の高いカテゴリをツリー内でハイライトして提示する。以下に関連カテゴリ提示部の詳細について述べる。

	あだち充の屋根裏部屋		ゆんフリー写真素材　テーマ：大地	
順位	カテゴリ名	類似度	カテゴリ名	類似度
1	Japanese/アート/コミック/作家別/あ行/あだち充	0.173394691853174	Japanese/レクリエーション/旅行/旅行記/北アメリカ	0.0233080845288375
2	Japanese/地域/アジア/日本/滋賀/市町村/彦根市/健康	0.0954584494620425	Japanese/レクリエーション/旅行/ガイドとディレクトリ/北アメリカ	0.0116710245367587
3	Japanese/ゲーム/テーブルゲーム/ボードゲーム	0.0924196218595997	Japanese/地域/アジア/日本/東京/旅行・観光	0.0101240917488282
4	Japanese/地域/アジア/日本/山梨/市町村/都留市	0.0919939805563525	Japanese/レクリエーション/イベントとテーマパーク/遊園地・テーマパーク/ディズニー	0.00753518468911212
5	Japanese/地域/アジア/日本/兵庫/市町村/新宮町	0.0863936709137147	Japanese/社会/歴史/地域別/北アメリカ/アメリカ合衆国	0.00682765543440325
6	Japanese/地域/アジア/日本/神奈川/市町村/南足柄市/健康	0.0815993547833912	Japanese/地域/アジア/日本/東京/区/江東区/教育/大学・短大	0.00663274107068687
7	Japanese/レクリエーション/アウトドア/スキューバダイビング/ショップとガイド/日本/滋賀	0.0714210133792899	Japanese/スポーツ/ウォータースポーツ/サーフィン	0.0065726156285564
8	Japanese/科学/自然科学/ニュースとメディア	0.054270486501781	Japanese/スポーツ/サッカー/AFC（アジア）/日本/東京/FC 東京	0.00607437957184021
9	Japanese/レクリエーション/アウトドア/登山・クライミング/団体/地域別/栃木	0.0509951131686643	Japanese/アート/ビジュアルアート/コンピュータ・グラフィックス/イベント	0.00596046985861984
10	Japanese/ビジネス/不動産	0.0471948586234184	Japanese/アート/音楽	0.00582051390396685

表 4.1: 二つのウェブページにおけるカテゴリとの類似度

4.4.1 関連カテゴリ掲示画面

図 4.2 に本システムにおいて「Project Team Doga」[14] というソフトウェアのプロジェクトの Web ページをブラウジングしている場合の関連カテゴリ掲示画面を示す。カテゴリが多岐にわたり、ツリーが長くなりすぎてしまったため、分割して横に並べてある。本研究では、ディレクトリの全体像が分かるように CustomDir に登録されているカテゴリをユーザがブラウジングしているページとの類似度を自動算出して、高い順に濃淡でハイライトして提示している。また、類似度が 10 番目に高いカテゴリまではカテゴリ名の左に [Rn] のマークがつき、何番目に類似度が高いのかを提示する。ユーザがまったく操作をすることなくブラウジングしているだけで、システムが自動で関連カテゴリ提示場面は更新される。また、通常のブラウジングが今までと同様に行えるようにするためにサイドバーに提示する。このことにより、ユーザはブラウジングしているページと関連ページを画面を切り替えることなく比較することができる。類似度の高い 10 個のカテゴリをハイライトする際、分野が多岐に渡り、ツリーが縦に長くなってしまう問題がある。その問題を解決するために、もっとも類似度の高いカテゴリ以外で、子孫のカテゴリに類似度の高いカテゴリが含まれていて、なおかつ展開されていないカテゴリは、カテゴリ名の左に矢印を提示する。図 4.2 は図 4.3 の様に一画面で提示される。さらに、子孫のカテゴリに類似度の高いカテゴリを含む際に表示される矢印はカテゴリが展開されると一つ下のカテゴリに移動する。また、カテゴリが畳み込まれると矢印は一つ上のカテゴリに移動する。例えば、図 4.4 を例に取ると、類似度が高いカテゴリを含む展開されていないカテゴリとしてカテゴリ「社会」が挙げられる。カテゴリ「社会」が展開されていないときは「社会」の左に矢印が表示され、展開されるとその下の「いろいろな人々」に矢印が移動する。これはつまり、類似度が高いカテゴリは「社会」の「いろいろな人々」に含まれており、他のカテゴリには類似度が高いカテゴリが含まれていないことを表している。「いろいろな人々」を展開すると「出会い・交流」に矢印が移動し、さらに展開することで「異文化」に辿り着ける。これに加えて、子孫のカテゴリに類似度の高いカテゴリが含まれていて、なおかつ展開されておらず、自身も類似度が高い場合は、濃淡のハイライトに加えてカテゴリ名の左に矢印が表示され、展開されると何番目に類似度が高いかを表す [Rn] のマークが表示されるよう組み合わせた。このような表示インタフェースによってユーザはより情報の関連性をつかみやすくなり、効率よく情報を収集することが可能となる。

4.5 類似度計算高速化のための課題

本システムでは、まずブラウザがユーザがブラウジングしているページの URL をサーバに送り、その URL から HTML ファイルを得て分析し、類似度計算を行っている。この際、URL が送られてくるたびに全ての処理を実行していては時間がかかってしまうため現実的ではない。処理時間を短縮し、実用性のあるシステムにする必要がある。

類似度計算についてはブラウジングしているページによって変化するが、カテゴリの特徴ベクトルは普遍であるので、カテゴリの特徴ベクトルを作成し、配列に挿入までを前処理として事前に処理しておくことで、全体の処理時間が短縮できると考える。

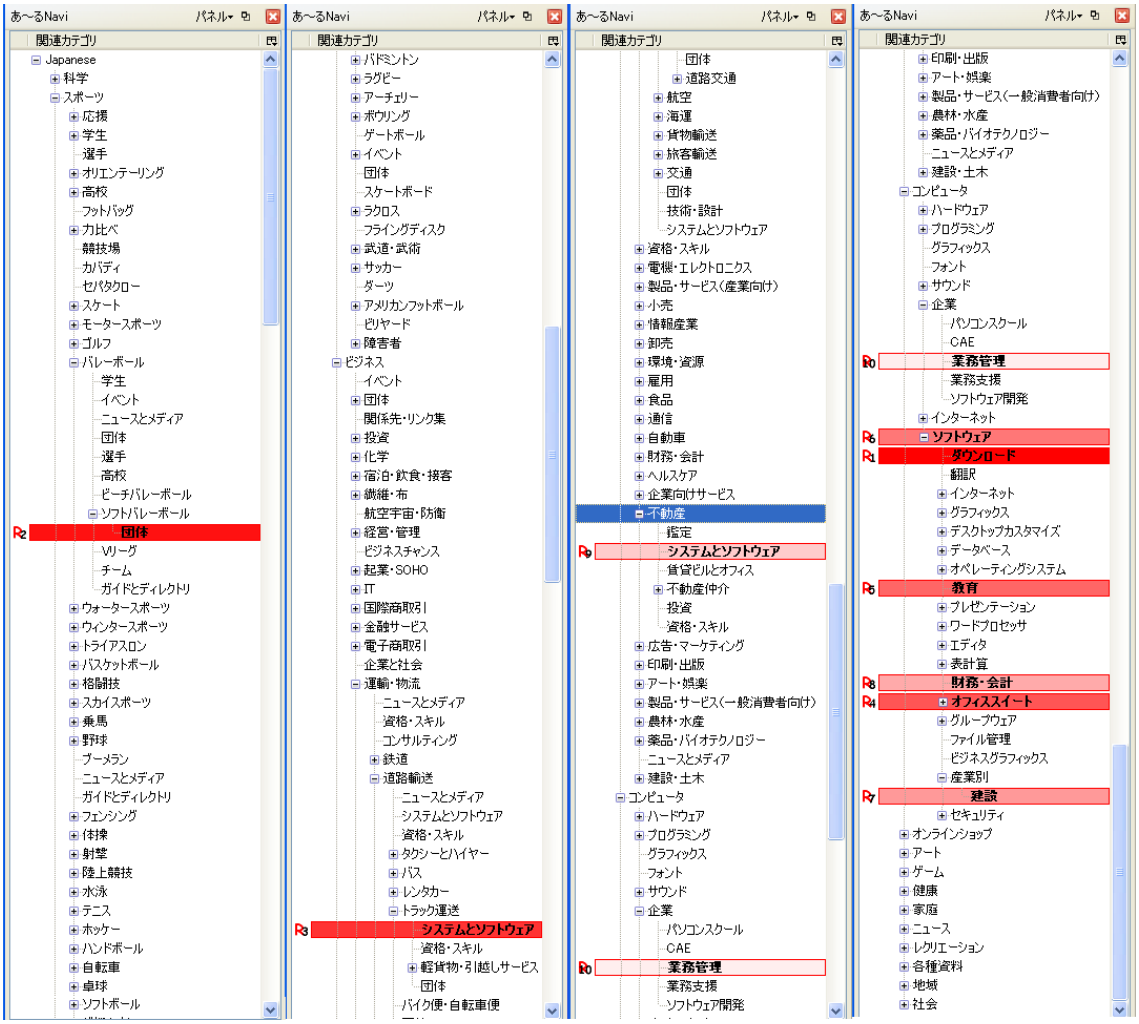

図 4.2: 「あ〜る Navi」の提示画面（展開）

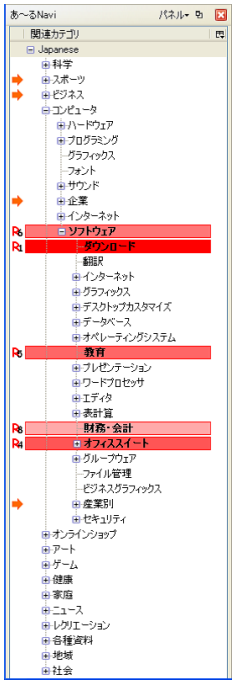

図 4.3: 「あ〜る Navi」の提示画面（畳み込み）

図 4.4: ハイライトの遷移

処理を行うタイミングは、ユーザが最初にブラウジングしたときの類似度をデータベースに蓄積しておき、以降のブラウジング時にデータベースに登録されている URL であれば処理を行わずにデータベースから取り出すと言う方式が考えられる。また、定期的に Web を巡回してデータベースに URL と類似度を登録し、ブラウジング時に呼び出して利用すると言う方式も考えられる。前者の方法は Web ページが更新されたとき、特にブログなどの情報が変わるページの場合、類似度が一定であるとは限らない。後者の方式は Web の規模が膨大であるために類似度を保存しておく記憶容量が膨大になると言う問題点がある。これらの問題については今後検討する必要があると思われる。

第5章　システムの実用例

ここでは、具体的な例を使いながら本研究のシステムの実用例を示す。あるユーザが「Project Team Doga」というページを見ているとする。システムはユーザがブラウジングしているページをアルゴリズムに従って自動で分析してどのカテゴリに当てはまるかをサイドバーに提示する。提示された情報の中からユーザは関連する質のいい情報を入手する。

図 5.1 を見ると、「あ〜る Navi」は一番関連のあるカテゴリとして「コンピュータ」の下の「ソフトウェア」の下の「ダウンロード」を挙げており、その他にも同じ階層の「オフィススイート」、「教育」、「財務・会計」、上の階層である「ソフトウェア」にも関連があることが分かる。もし、一般の検索エンジンなどを使うのであれば、「ソフトウェア」等のキーワードを入力して検索し大量の Web ページが提示される。Web 全体から Web ページを検索しているため、必ずしも質のいい Web ページに到達できるとは限らない。もし、質のいい Web ページでなかった場合、もう一度検索画面に戻って他の URL をクリックして Web ページを見るか、他のキーワードを入力して再検索する必要がある。本研究のシステムでは、関連するカテゴリをクリックすると図 5.2 のようにあらかじめ登録されている質のいいページを提示されるため、キーワードを入力する必要も再検索をする必要もない。一番上のページを選ぶと図 5.3 のように変わる。「あ〜る Navi」が提示するカテゴリもブラウジングに合わせて動的に変化していることが分かる。ココで、さらに教育関係のソフトウェアも関連があることも見れる。教育のカテゴリを選ぶと図 5.4 のように登録されている質のいいページが提示される。一つサイトを選んで移動すると「あ〜る Navi」もページの遷移と同時に提示する関連カテゴリも図 5.5 のように変更する。このように、ユーザはブラウジングしているページに関連する質のいい Web ページを効率よく収集することができる。

図 5.1: 「Project Team Doga」ブラウジング中

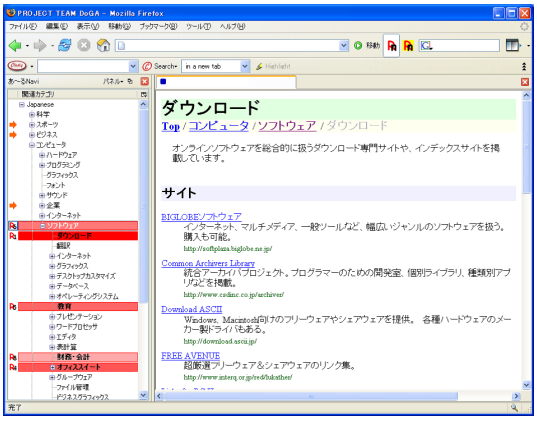

図 5.2: カテゴリ「ダウンロード」選択時

図 5.3: ダウンロードに関する質のいいページ

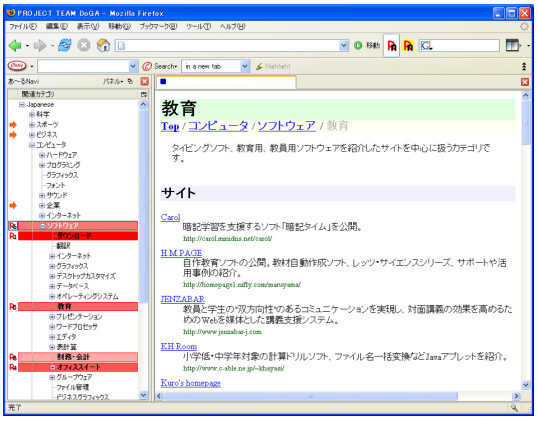

図 5.4: カテゴリ「教育」選択時

図 5.5: 教育ソフトウェアに関連するページ

第6章　関連研究

大坪五郎の開発した Goromi[15] は Web 上の情報を流し見するためのインタフェースであるが、漠然とした要求に応えるという点で本研究と共通する点がある。Goromi は「何か面白い物を見たい」と言う漠然とした要求に対して班受動的に Web 上の情報を閲覧できるインタフェースであり、キーワードを入力し、Google から帰された検索結果中の Title と要約を用いて関連する単語を抽出し Web ページのタイトルと載せられている画像と同時に関連のある単語を提示する。また、Web ページのタイトルと乗せられている画像が自動的にしたから上に移動するのでユーザは操作なしに流れる情報を眺め続けられる。これは、Web 上の情報をぼんやり眺めるには適していると思われるが、ユーザがブラウジングしているページに関連するページを検索するには不向きであると考える。さらに、本研究とは、キーワードを入力する必要がある点と Web 上の情報を垂れ流しにすると言う点が異なる。また、奥村穂高らによる Carta[16] ならびに、松田耕史による Seezle[17] はユーザがキーワードを指定しなくてもシステム側が関連キーワードを提示する、と言う点で本研究と共通する点がある。前者は検索キーワードを入力すると、関連キーワードをノードとして提示し重ね合わせたりすることで検索クエリを作り出し、その検索クエリを下に検索を行うことができる。後者は検索結果を 2 次元平面上に配置しキーワードと文書の関連を色づけと言う形で表現している。両者とも直感的に検索クエリを作成できると言う点で優れている。しかし、両者とも検索を主としていて通常のブラウジングを今まで同様に行うことができない点が異なるといえる。大坪五郎による Gards[18] は常に変化し続けるユーザの興味に対応すると言う点で本研究と通ずる点がある。Gards は比較的少数の操作履歴データのみを用いて次々と候補を表示し、ユーザが満足できる解に到達することを目指している。ユーザが行う操作を直感的かつ単純にすることでユーザが情報を得るためのコストを減らしている。Gards は対象が昼食が取れる場所であることに対し、本研究ではブラウジングしているページに関連した情報である点が違う。

第7章　結論

本稿ではまず現在の Web と Web 検索の現状を考察し、既存の検索インタフェースの問題点を述べた。次の問題解決の手法として関連カテゴリ掲示システムの提案と実装を行い、システムの活用例を述べた。本システムによってユーザがブラウジングしているページに関連した質の高いページの収集を支援することが可能となる。

今後は掲示インタフェースの改良や類似度算出時間の短縮などを行い、ユーザにとってさらに使いやすいシステムに改良すること、さらにシステムのユーザ評価を行って開発に役立てたいと考えている。

謝辞

本研究を進めるにあたり、指導教官である田中二郎教授に様々な助言やご指導をいただきました。深く感謝いたします。また、志築文太郎先生、三末和男先生にはチームミーティングなどにおきまして適切なご指導をいただきました。深く感謝いたします。有益な助言、ご指導を下さった高橋伸先生に心から感謝いたします。また、田中研究室の皆様には研究の全般にわたり丁寧なアドバイスをいただきました。本当にありがとうございました。

参考文献

[1] Barnard J.Jansen, Amanda Spink, and Tefko.Saracevic. Real users, and real needs:A study and analysis of usr queries on the web. Information Processing and Management, Vol. 36, No. 2, pp. 207-227, 2000.

[2] 原田昌紀, 佐藤進也, 風間一洋. 索引篩法 - 大規模サーチエンジンのための高速なランキング検索法. DEWS, 2003.

[3] Inc. Internet Systems Consortium. http://www.isc.org/index.pl?/ops/ds/.

[4] Google. http://www.google.co.jp/.

[5] Yahoo!. http://www.yahoo.co.jp/.

[6] Vivisimo. http://vivisimo.com/.

[7] 形態素解析・構文解析入門. http://www.unixuser.org/euske/doc/nlpintro/.

[8] MeCab: Yet Another Part-of-Speech and Morphological Analyzer. http://mecab.sourceforge.jp/.

[9] CustomDir Web Site - ディレクトリサイト開発支援ツール/CustomDir. http://www.customdir.net/.

[10] オブジェクト指向言語 Ruby http://www.ruby-lang.org/ja/.

[11] htmlsplit / htmlrepair. http://www.nslabs.jp/htmlsplit.rhtml.

[12] あだち充の屋根裏部屋. http://tokyo.cool.ne.jp/vin2/index.html.

[13] ゆんフリー写真素材　テーマ：大地. http://fount-k.com/ tomo/jp/cat36-1.html.

[14] PROJECT TEAM DoGA. http://doga.jp/.

[15] 大坪五郎. Goromi - Web 上の情報を「流し見」する方法. 第 12 回インタラクティブシステムとソフトウェアに関するワークショップ, 2004.

[16] 奥村穂高, 田中二郎. 重ね合わせノードによる Web のキーワード検索. 情報処理学会第 60 回 (平成 12 年前期) 全国大会, 2000.

[17] 松田　耕史. 統計的手法による Web 検索補助システム “Seezle”の開発. 平成 15 年度未踏開発ソフトウェア創造事業, http://www.ipa.go.jp/jinzai/esp/gaiyo/3-7.html.

[18] 大坪五郎. Gards - 変化し続ける興味に対応する情報推薦. 第 13 回インタラクティブシステムとソフトウェアに関するワークショップ, 2005.