検索キーワードとその関連語への重みづけによる重要なWebページの上位配置

大澤, 快至 / OHSAWA, Yoshiyuki

(発行年 / Year)
2008-03-24
(学位授与年月日 / Date of Granted)
2008-03-24
(学位名 / Degree Name)
修士(理学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

検索キーワードとその関連語への重みづけによる 重要な Web ページの上位配置

Ranking Preferable Web Pages Using Weighted Search Keywords and the Relevant Terms

大澤　快至

Yoshiyuki Ohsawa

法政大学大学院 情報科学研究科 情報科学専攻

E-mail: yoshiyuki.ohsawa.gh@gs-cis.hosei.ac.jp

Abstract

The search engine based on keyword such as, 「Yahoo! JAPAN」 or 「Google」 is one of the popular search engines used in Web searching for information. However, the current search engines have the problems of outputting too many search results and not searching for information by including the input keywords and their relevant words. This research is focused on obtaining the words or terms (the synonym, the hypernym, and the hyponym) related to the input keywords from WordNet.OWL and giving them weight coefficient. WordNet.OWL contains thesaural words and terms which are written in the form of OWL (Web Ontology Language). Web pages including more input keywords and the relevant terms of the input keywords are assumed more important. Therefore, this research is aimed at solving the above-mentioned problems by displaying ranked Web pages according to their importance. Individual web page importance is analyzed by firstly obtaining the keywords from user's input and the relevant terms from the WordNet.OWL, then counting the appearing frequency of these keywords and the relevant terms, and finally calculating the score of each Web page according to the evaluation function. Web pages outputted from an ordinary search engine are ranked according to the calculated scores. The pages with higher scores are displayed in front so as to better meet user's requirement and preference. In the implemented system, JSP on Tomcat is used by input and output of search results and Yahoo! JAPAN Web service is used by search of Web pages.

Supervisor: Runhe Huang, Professor

1. まえがき

現在の Web の現状と問題点について以下に示す.

1.1. 現在の Web

World Wide Web は人々のコミュニケーションやビジネスのやり方を変えた[1]．Web は，先進社会を知識社会へと変化させた中心のツールと言える．現在の Web 上のコンテンツは人間が利用する目的で作成されている．Web 上のコンテンツがデータベースから機械的に生成されたものであったとしても，ほとんどの場合，データベース内にもともとあった構造的な情報は表示の際には無くなってしまう．そのため，情報の検索や活用がごく単純なレベルに留まっている.

1.2. 検索エンジン

私たちの今日の Web の使い道は，知りたい情報を検索したり，オンラインショッピングを楽しんだり，旅行に行く際に宿を予約したりすることである．Yahoo! JAPAN，Google のようなキーワードに基づく検索エンジンは，今日 Web を使うときの主要なツールである．もし検索エンジンがなかったら，Web は今日のように大きく成功していない．しかし，検索エンジンの利用にも以下のような重大な問題点がある[1].

(1) 再現率は高いが精度が低い
たとえ主な関連するページが検索できたとしても，あまり関連しない文書も検索されてきたら役に立たないものになってしまう．検索結果が多すぎるのも，少なすぎるのと同様に良いこととは言えない.
(2) 再現率が低い
検索に対し何も答えが返ってこない，あるいは重要かつ関連するページが検索されないということが起きてしまうことがある.
(3) 結果が語彙に依存しすぎる
基本的にキーワードマッチングで検索されるため，関連するページに意味的には似ているが別の用語が使われていた場合，検索結果にはヒットしない.
(4) 結果が単一の Web ページでしかない
たくさんの文書に分散している情報が必要な場合，検索を繰り返しては，文書中から情報を手作業で抽出して一緒にしなければならない.

本研究では検索エンジンの特に(1)と(3)の問題解決に対して，入力された検索キーワード

およびそのオントロジーから得られる検索キーワードの関連語で Web 全体の検索をし，検索キーワードとその関連語を多く含むページを検索結果の上位に持ってくることでアプローチを試みる．検索キーワードを多く含むページは，実際 Web 検索をするユーザにとって本当に欲しい情報が含まれている可能性が高い．またその際，検索キーワードや関連語，重要と思われる HTML タグに重み付けを行った．そこから得られる総得点の高いページを検索結果の上位に持ってくることによって，多くの検索結果の中からユーザにとって必要な情報が探しやすくなる．

本研究では，実際の Web 検索には Yahoo! JAPAN Web サービス，検索キーワードの関連語の取得には，英語のオントロジーである“ WordNet.OWL ”を使用した．

2. 関連分野・研究

2.1. セマンティック Web 構想

以下に，現在の Web の問題を解決するための取り組みであるセマンティック Web の技術を示す．

2.1.1. セマンティック Web

現在の Web は 1989 年にヨーロッパの原子力研究所である CERN の技術者であった Tim Berners-Lee によって考え出されたハイパーテキストのシステムである[2]．現在の Web 上のコンテンツは主に HTML（HyperText Markup Language）で記述されている．そのため，ページやその中の単語など，その文書が何を意味しているのかを理解し，処理することはコンピュータにとって非常に困難なことである．そこで再び Tim Berners-Lee によって提唱されたのが，Web ページの意味を扱うことを可能とする標準やツールの開発によって World Wide Web の利便性を向上させようというセマンティック Web である．セマンティック Web は Web 上の情報を機械的に処理するための枠組みである．セマンティック Web では HTML 文書を人工知能的に解析して内容を機械に理解させるのではなく，機械的に処理可能なメタデータを HTML 文書に付与しておく．HTML 文書はあくまでも人が読む文書とし，その文書を説明するためのデータを機械処理可能な形で別に用意する．また，メタデータを活用する際，オントロジーが用いられる．セマンティック Web 上でオントロジーは概念間の関係の明確な定義の集まりという意味で利用される．同じ対象物であっても，ページによって用いる用語や異なる場合がある．オントロジーを利用することにより，用語の違いの吸収や用語間の関係を反映した推論処理が可能になる．図 1 にメタデータやオントロジーの Web ページとの関係を示す．

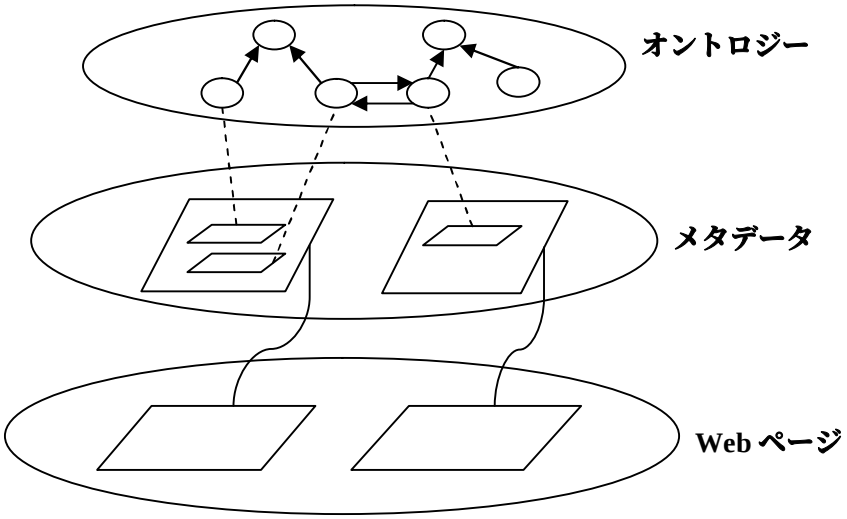

図 1. メタデータやオントロジーの Web ページとの関係図

2.1.2. 階層的なアプローチ

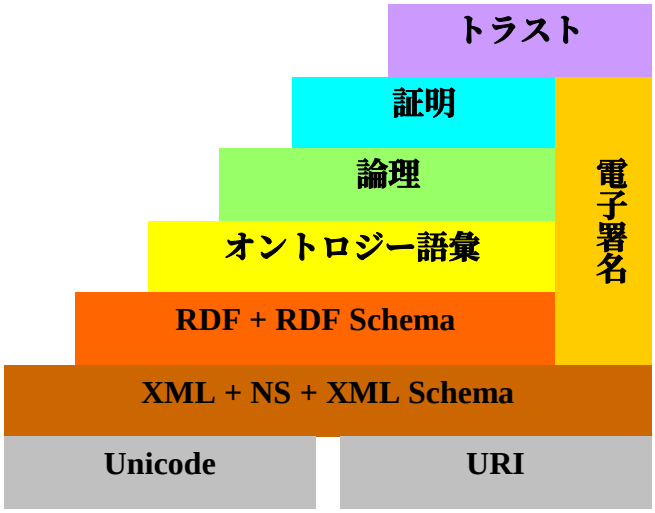

図 2. セマンティック Web のレイヤーケーキ

セマンティック Web の発展は 1 つずつ階層を重ねていくような形で段階的に進められている．たくさんの段階を同時に解決しようとすると，多くの人が関連するため，合意を得るのが難しくなるからである[1]．図 2 は，セマンティック Web の設計と構想の主要な階層を表している．メタデータは誰でも記述でき，また複雑な構造のデータも記述できなくてはならない．さらに，将来に渡って利用可能であることを保証する必要もある．このような目的のために Web リソースのためのメタデータ記述形式としては RDF（Resource

Description Framework）が定められている．RDF は主語（Subject）と述語（Predicate）と目的語（Object）の RDF トリプル（RDF Triple）の集合として定義される．Web 上のあらゆるリソースは URI（Uniform Resource Identifier）により表され，RDF トリプルの主語として記述対象となる．目的語も文字列だけでなく URI も許可することにより，別の RDF トリプルと結合することができ，複雑なデータ構造を表すことができる．RDF トリプルの述語が意味の表現，またはハイパーリンクである．述語として用いる語彙は，いろいろなグループによって付与されるメタデータの語彙と重複しないために，URI を用いる．また，構造化された Web 文書を，ユーザ定義語彙で書くための言語である XML（eXtensible Markup Language）がある．XML は，Web を介して文書をやり取りするのに特に適している．RDF は，XML に基づく構文を持っている．

RDF Schema は Web 上のオブジェクトを階層的に組織化するための基本モデル要素である．基本となる要素は，クラスとプロパティ，サブクラス関係とサブプロパティ関係，および定義域と値域の制約である．RDF Schema は RDF に基づいている．

RDF Schema はオントロジーを書くための基本的な言語と見ることができる．しかし，RDF Schema を拡張して，Web 上のオブジェクト間のより複雑な関係を表現できるオントロジー言語として OWL（Web Ontology Language）がある．もちろん，OWL も XML の構文に基づいて記述されている．論理層はオントロジー言語をさらに強化し，応用に特化した宣言的な知識を記述できるようにするためのものである．証明層は，実際に結果を導きだす課程および Web 言語での証明の表現形式と証明の妥当性確認に関するものである．トラスト層は，最も高レベルで重要な概念である．Web はユーザがその運用と提供される情報の質について信用して初めて性能を発揮することができるからである．

セマンティック Web に必要なメタデータを活用するためには，Web ページにメタデータが付与または Web ページとは別に持っていなければならないが，世の中にある Web ページの数に対して，メタデータが付与されている Web ページは少ない[2]．メタデータの付与を今後普及させていくには，

・Web ページの内容に対して，形態素解析（意味を持つ最小単位の単語への分割）などを行い，何が書かれているのかを判断して，自動でメタデータを作成してくれるアプリケーションの登場
・Web ページの製作者にとってのメタデータを付与することのメリットの明確化

などが必要である．したがって，セマンティック Web の実現はまだ難しいとされている．このことから少しでも Web を使いやすいものにする研究は，今現在非常に有効であると思われる．

2.2. オントロジー

以下に，語彙の厳密な意味や関係の記述に必要なオントロジーの概要とオントロジー言語 OWL について記述する．

2.2.1. オントロジーの概要

オントロジーという言葉は，「存在論」という哲学の一分野という意味で使われていたものを，コンピュータ科学の分野では対象とする世界に存在しているものごとを体系的に分類し，その関係を記述したものという意味で使うようになった[3]．オントロジーに記述された情報を利用することによって，機械に単語の意味を理解させることが可能となるため，セマンティック Web の分野に利用されている．

オントロジーは，有限個の用語リストと，それらの間の関係記述から構成される．例を図 3 に示す．

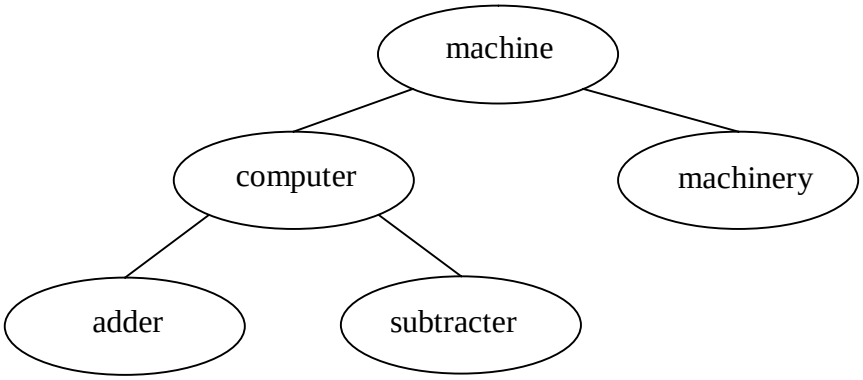

図 3．機械オントロジーの例

クラスの階層関係は典型的な関係である．図 3 の例で言えば，すべての「computer」は，「machine」でもあるということである．よって，「computer」は「machine」のサブクラスと言える．

シソーラスは，しばしばオントロジーと同じような意味で使われることがある．シソーラスの構造は，用語の相互関係を「上位概念」「下位概念」「同義語」といった体系が基づいて分類した意味ネットワーク構造であるが，上位概念，下位概念という階層関係よりは同義語関係に中心をおいた意味の分類システムである．シソーラスにはオントロジーにみられるような，推論機構が組み込まれていない．シソーラスは，静的な辞書であり，手続き的，宣言的なアルゴリズムを対象にしない[4].

2.2.2. オントロジー言語 OWL

Web オントロジー言語に求められる共有性，発展性，相互運用性，矛盾の検出といった要件を満たすものとして，W3C のワーキンググループで開発されているのが，OWL という言語である[5]．OWL は，用語・語彙とそこに含まれる各要素の関連の明確な表現を目的としており，これまで開発されてきたオントロジー言語「DAML+OIL」の改良版である．OWL のオントロジーは，RDF のトリプルの集合で構成される．RDF のトリプルとは，リソースの関係を主語，述語，目的語の 3 つの要素で表現したものである．OWL 言語仕様では，どんな RDF トリプルが OWL の語彙を構成し，それによって何を意味するかを定義する．OWL は一般に RDF の XML 構文によって記述され，次のような構成要素を含む（いずれも 0 回以上任意の回数，記述が可能）．

(1) バージョン情報と他のオントロジーのインポートを記述するヘッダ

ヘッダは owl:Ontology 要素として記述し，バージョン情報と他のオントロジーのインポートを示すことができる．また owl:imports 要素で目的語として示されたオントロジーのグラフをインポートし，主語オントロジーのグラフに加えることができる．これによってオントロジーを再利用することができるため，拡張性や相互運用性に重要な機能である．

(2) クラスを定義するクラス公理

「ウェブに存在するもの」の概念であるクラスは owl:Class 要素によって表現し，rdfs:subClassOf 要素（参照クラスのサブクラス）や owl:equivalentClass 要素（参照クラスと同じインスタンスを持つクラス）などでクラス公理を構成する．またそのクラスが持つプロパティの制約条件を与えることで，クラスをより詳細に定義にすることができたり，owl:intersectionOf プロパティ（論理積）や owl:unionOf プロパティ（論理和）などを用いることによって，別のクラスのインスタンスの和集合，差集合などの論理的な組み合わせとしても定義できる．

(3) プロパティを定義するプロパティ公理

プロパティは，ウェブに存在するものの関係を定義する部分である．例えば「子供がいる」というプロパティがあるときに，「親がいる」という反対の関係を示すときに用いる．プロパティには，あるオブジェクトを別のオブジェクトと関連づける個体値型プロパティと，オブジェクトをデータ型値に結びつけるデータ値型プロパティがある．

(4) 個体 (Individual)クラスのインスタンスによる事実の記述

2 や 3 で述べてきたクラスやプロパティの定義は，推論などを行うためのルール集のような役割を果たす．これを用いて，実際に存在するものを具体的に描くのがインスタンスである．抽象構文ではこのインスタンスを記述する部分を事実（Fact）と呼ぶ．OWL ではインスタンス，つまり個体は，必ず何かのクラスに属する．それらは owl:sameAs プロパティ（2 つの固体が同一）などを用いて表現される．

英語圏ではすでに，英語のシソーラスである WordNet1.7.1[6]を基に，OWL により記述された英語のオントロジー「WordNet.OWL」が公開されている．しかしながら今のところ，Web 上に公開されている日本語のオントロジーは無い．したがって，現在様々な手法を用いて日本語オントロジーを構築する動きが盛んである．

まず，対訳辞書を用いることにより，日本語と WordNet を結びつけて日本語のオントロジーを構築する研究がある[7]．ここでは，和英辞書 EDICT を使用することにより，ある日本語に対して，WordNet の URI 参照と同じクラスを RDF の形式で返す試みが行われている．

また，対訳辞書を用いる方法だと訳語が複数存在するため，訳語の曖昧性が出てきてしまう．この問題解決のため，Web ディレクトリを用いた 2 言語オントロジーを構築する研究がある[3]．ここでは，Yahoo! カテゴリの英語版と日本語版を用いて，ある英単語はどのカテゴリに適合するのかを仮定し，それぞれの特徴語を結びつけることにより，対訳辞書を用いる方法よりも的確な日本語を抜き出す実験を行っている．

実用化段階の日本語オントロジーは，今現在まだ公開されていないため，本研究では，すでに公開されている英語のオントロジー「WordNet.OWL」を利用した．

3. マイアプローチ

以下に現在の Web 検索の問題解決に対して，本研究のアプローチ手法を示す．

3.1. 重要度の高いページの上位配置

ユーザが本当に欲しい情報が含まれているページは，検索キーワード多く出現するページの可能性が高い．例えば，ユーザが computer について調べたいと思っている場合，検索キーワードとして“ computer ”と入力をする．一般的なキーワードマッチングでの検索の場合，“ computer ”という単語を含んでいるページであれば，何度も検索キーワードが出現するページはもちろん，1 回しか出現しないページも検索結果として表示される．絶対に 1 回しか検索キーワードが出現しないページに必要な情報が書かれていないとは言えないが，何度も出現するページの方が重要度は高いと思われる．

一般的な検索エンジンでは，ページのランク付け（スコアリング）をしており，ランクの高い順に結果が表示されるようになっている．ランク付けの基本的な考え方は「ユーザにとって重要と思われる文書を上位に表示する」ことである．ランク付けの手法の中で，上述のような文書中の検索キーワードの出現頻度で得点をつける方法は一般的に行われている．その他の手法も以下に示す[8]．

・HTML タグの解析

<title>タグや<h1>タグを重視し，重視されたタグにキーワードが入っていた場合，高い得点を与える方式である．

・TF・IDF 法

文書中の単語の出現頻度を“TF(Term Frequency)”と呼ぶ．また，全文書中 N のうち，該当する単語が n 個あるとき，その比の対数($log(N/n)$)を“IDF(Inverse Document Frequency)”と呼ぶ．TF と IDF の積である“TF・IDF 値”用いることにより，単語の重要度を計算する[9]．IDF はある単語を例にとったとき，様々な文書に出現する場合には小さくなり，特定の文書のみに出現する場合には大きくなる．よって TF・IDF 値が，広く使われる一般的な単語は低くなり，その文書には何が書かれているのか特定するために役立つと思われる単語は高くなる．

・ページランク方式

Google などで採用されており，「重要度の高いページからリンクされているページは重要である」という考えに基づいてランク付けを行う方式である．

本研究は，一般的な検索エンジンと同様，ユーザにとって必要な情報が含まれていると思われるページを上位に持ってくることによって，ユーザがそれらのページを見つけやすくすることを目標とする．その際，検索キーワードが何度も出現するページはもちろんのこと，新たな手法としてオントロジーから得られる検索キーワードの類義語などの関連語が何度も出現するページも上位に持ってくるようにする．また，単純に単語の数をカウントしていくのではなく，Web ページのソース解析を行い，重要と思われる HTML タグの中に書かれた内容の中に検索キーワードやその関連語が含まれていた場合，さらに高い得点を与えるようにする．それぞれに得点を与えることにより，より重要度の高いページが上位に配置されるよう考慮する．

3.2. システムの流れ

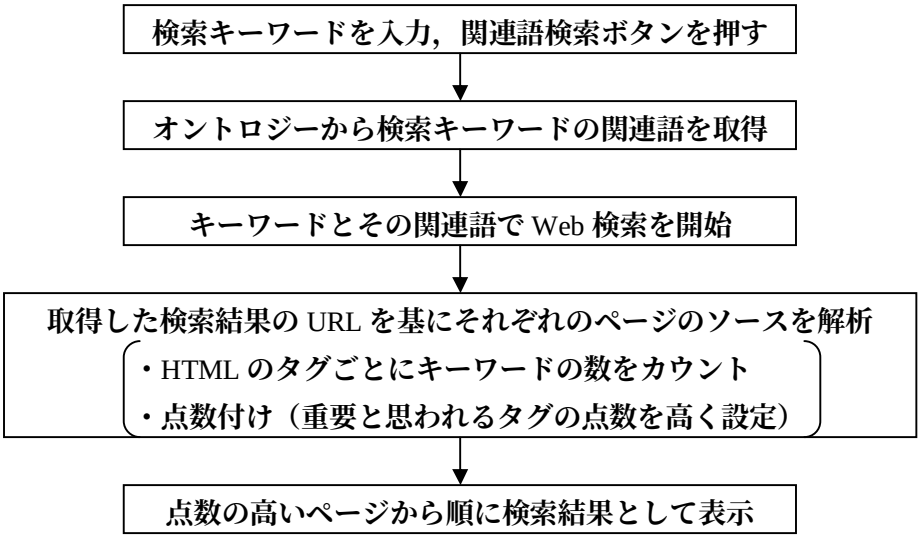

図 4. 開発するシステムの流れ

図 4 は開発するシステムの流れである．まず，検索キーワードからその関連語を取得する．次にキーワードと類義語で Web 検索を行う．検索結果の URL を利用し，そのページを解析，得点の高いページを上位に配置し，検索結果として表示する．図 5 に開発するシステムのユースケース図を示す.

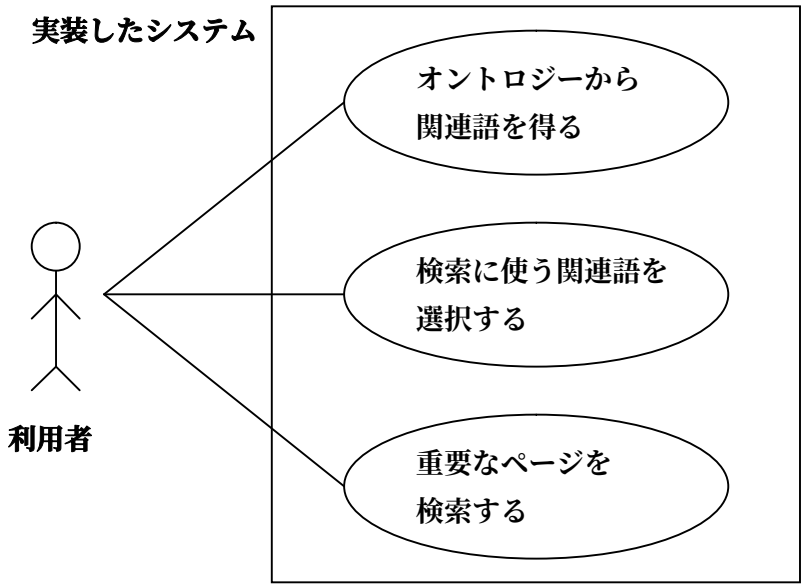

図 5. 開発するシステムのユースケース図

検索キーワードの関連語を取得する際に，英語のオントロジーである WordNet.OWL を利用する. 検索キーワードを入力する際, Ajax を用いてキーワードの候補を出すことにより，ユーザがキーワードの入力をしやすいようにした．また，上位語や下位語などの関連語も取得する．Web 検索には，Yahoo! JAPAN Web サービスを利用し，検索結果の表示ページは JSP で作成した.

3.3. 使用するツール

以下に本研究で使用するツールを示す.

3.3.1. Yahoo! JAPAN Web サービス

Web サービスは，従来の Web アプリケーションによる Web 上のサービスとは異なる．Web サービスの定義には，以下のように様々な表現がある[10].

・インターネット経由で XML 技術によって連携される，URI で識別されたソフトウェア・アプリケーション

・インターネット上で，アプリケーションから利用できるサーバ・プログラム
・インターネット上の RPC（Remote Procedure Call：遠隔手続き呼び出し）
・インターネットまたはイントラネット上で，標準のデータ形式，標準の通信規約を使用してシステムを連携させる技術
・XML や SOAP（Simple Object Access Protocol）などの標準技術を利用したシステム間の連系技術

今回 Web 検索を行う際，Yahoo! JAPAN の Web サービス[11]を利用した．Web サービスは，SOAP と呼ばれるプロトコルを用いて，メッセージの送受信を行う技術という意味で使われることが多いが，Yahoo! JAPAN の Web サービスは，SOAP を使わずに REST（REpresentational State Transfer）というスタイルを用いている．Yahoo! JAPAN の Web サービスは開発者が，Yahoo! JAPAN のコンテンツやサービス，技術にアクセスして，新しいアプリケーションを作成するための Web サービスである．例えば，データとサービスを合わせて，デスクトップアプリケーションを作成したり，それらを用いた別のサイトを提供したりできる．また，アプリケーションのリソースとして，Yahoo!検索など，Yahoo! JAPAN のサービス，技術に興味をもつ開発者や研究者のために提供されている．検索の他に，カテゴリ，オークション，ミュージック，地図情報，テキスト解析，家電ナビ，ニュースなどの Web サービスが無料で提供されている．HTTP リクエストを送ると，結果を XML 形式で返してくれるため，様々なプログラミング言語から簡単に利用できる．

本研究では，Yahoo!検索のウェブ検索の Web サービスを利用した．検索キーワードをクエリとして検索キーワードを HTTP 形式で送ると，タイトルやサマリー，URL などが XML 形式で返される．本研究では，検索の入出力に JSP を使用したため，Java から利用した．

3.3.2. WordNet.OWL

WordNet は英語の概念辞書である．英単語が synset と呼ばれる同義語のグループに分類され，簡単な定義や，他の同義語のグループとの関係が記述されている．また，データベースやソフトウェアは，自由にダウンロードして利用することができる．プリンストン大学の認知科学研究所によって心理学者である同大学教授の George A. Miller の主導のもとで運営されている．現在，WordNet のデータベースには約 15 万語が収録されており，全体で 20 万 3000 の単語と意味の組み合わせがある．WordNet では，名詞，動詞，形容詞，副詞を文法上の扱いが異なることから，区別している．synset は同義語あるいは熟語をグループにまとめている．ほとんどの synset は他の synset との意味的な関係が番号によって示されている．この関係の種類は品詞によって異なっており，以下に示す通りになっている．[6][12]

・hyponymOf

すべての Y が X の種類の 1 つであるなら Y は X の下位語である．

・hypernymOf

すべての X が Y の種類の 1 つであるなら Y は X の上位語である．

・entailsTo

主語 S に対して動詞 V が伴うことを示している．

・similarTo

主語 S に対して動詞 V が意味の点で同様であることを示している．

・Meronym

Y が X の一部であるなら，Y は X の meronym **である．**

・Holonym

X が Y の一部であるなら，Y は X の holonym **である．**

・causedBy

主語 X に対して動詞 Y が引き起こされることを示している．

・attributeRel

形容詞 A と X の属性関係を定義する．

・groupWith

動詞 V と動詞 V´ が同様の意味の集まりであること示している．

・antonymOf

X と Y が反意語であることを示している．

・seeAlso

X と Y が同意語であることを示している．

・participleOf

形容詞 A は動詞 V の分詞であることを示している．

・pertainsTo

同じ意味を持つ 2 つの異なる品詞間関係を示している．たとえば，形容詞 A を名詞形にしたものや，副詞 Adv を形容詞形にしたものがこれに当たる．

synset に含まれる語句は同じ意味を持った同義語であるため意味的な関係は synset **内全体に適用されるが，　単独の語句が他の語句と反意語や派生語などの関係を結ぶこともある．また，**WordNet **には語彙が属する** synset **の数の情報も含まれている．ある単語がいくつかの** synset **に属している，つまりいくつかの意味を持っている場合，その単語は同じ意味の他の単語よりも一般的に用いられていることが多い．**WordNet **ではこのような関係を頻度点** (frequency score)**と呼ばれる数値で表している．サンプルの文書の中には全ての単語に** synset **等の意味を表すタグを付与しているものがあり，単語が特定の意味で出現している頻度によって頻度点が計算されている．**

WordNet.OWL はオントロジー言語である OWL によって WordNet を記述したものである．

3.3.3. Ajax

Ajax（Asynchronous JavaScript XML）は，Web ブラウザ内で非同期通信とインターフェースの構築などを行う技術の総称である[13]．HTTP 通信を行うための JavaScript の組み込みクラスである XMLHttpRequest による非同期通信を利用し，通信結果に応じてダイナミック HTML で動的にページの一部を書き換えることを可能にする．Ajax は，別途プラグインをインストールする必要がなく，Web ブラウザさえあれば利用できる．また，Ajax はそれ自体が技術なのではなく，JavaScript や XML，DOM（Document Object Model）など既存の技術の集合体である．

Ajax は近年，次世代 Web として注目されている Web2.0 の中心技術の 1 つである．Web2.0 の概念の明確な定義はなく現在も議論が行われているが，その中で Web のコンテンツをユーザが自分の好きなように変えるという考え方がある．Ajax では，Google や Amazon，Flickr のように開発者がプログラムしやすいようにデータや API を公開する Web サービスの実現に一役買っており，Web2.0 の新しい世界を実現する足がかりになると考えられる[14]．

従来の Web アプリケーションではブラウザで操作をすると，サーバにリクエストが送られる間はページ遷移が発生して，ユーザはブラウザの操作が不可能になる．そしてサーバが HTML をブラウザに渡し，表示されて，初めてユーザは再び操作を行うことが可能になる．つまり，従来の Web アプリケーションでは，ブラウザへの表示とサーバとの通信が同期的かつ密接に関連しており，分離されていない．

一方 Ajax を使った Web アプリケーションは，従来の Web アプリケーション同様に HTML を取得し，埋め込まれている JavaScript を得る．この JavaScript が Ajax エンジンになる．ユーザがブラウザで操作を行うと，その操作に新しいデータが必要ならば，Ajax エンジンによってサーバにデータのリクエストを行う．Ajax の特徴は，このサーバへのリクエストの間もページ遷移が発生せず，ブラウザが制御されないことである．ユーザはブラウザの操作を行うことができ，必要な部分のみの表示変更を Ajax エンジンが行う[14]．

Ajax の問題点としては，大きく 2 つある[14]．1 つ目の問題点は，異なるドメイン間の通信が不可能な点である．これはセキュリティ上の制限からであるが，サーバサイドにある CGI を仲介してデータを取得したり，script タグを利用して外部ページから動的に JavaScript コードを読み込んだりすることで，この問題を解決することは可能にはなっている．2 つ目の問題点は，クロスブラウザの問題である．ブラウザ間に仕様の差がある場合， Ajax の振る舞いがブラウザによって異なる場合がある． また， JavaScript 自体の動作がブラウザによって異なったり，ブラウザのローカルキャッシュが Ajax の動作に影響を与えたりする可能性もある．

本研究では，Ajax を用いる際に，Ajax のフレームワークである prototype.js[15]を使用し

た．prototype.js は Sam Stephenson によって書かれた JavaScript ライブラリであり，Ajax のデファクトスタンダードとして多くの開発者から認知されている．prototype.js を利用することで，上述のクロスブラウザの問題からある程度解放されるという利点も持つ．

今回実装したシステムでは，検索キーワードの入力の際のキーワードの候補を出すのに Ajax を用いた．また，Web ブラウザには Internet Explorer7.0 を用いた．

3.3.4. DOM・SAX

Web オントロジー言語 OWL は，XML 形式で記述されている．よって WordNet.OWL を読み込んでいく際に，XML 文書を操作する作業が必要である．XML 文書を利用したり操作するための標準化されたインターフェースで，一般的によく使われるものには，DOM と SAX がある[16]．

DOM（Document Object Model）は，ドキュメントの内容や構造にアクセスしたり変更したりするための，プラットフォームやプログラミング言語に直接関係しないモデルである．DOM の 1 つの重要な目的は，多種多様な環境とアプリケーションで使うことができるプログラミングの標準インターフェースを提供することである．DOM そのものは仕様なので，実際のプログラミングでは DOM を実装したソフトウェアコンポーネントやモジュールを使用する．それによって，様々なプログラミング言語から，XML 文書の構造を調べたり，ドキュメントを作成したり，ドキュメントの内容や要素を追加，変更，削除することができる．正しく記述された XML 文書は，ツリー構造で記述されている．DOM は，XML 文書の内容全体をこのツリーの形でそのままメモリ上に保存して，DOM で定義されているインターフェースを使用してその各要素にアクセスをする．実際に DOM でドキュメントの内容にアクセスをする際には，メソッドやプロパティを使用する．

XML 文書全体をメモリに読み込む DOM に対して，SAX（The Simple API for XML）はドキュメントを読み込みながら処理を行うことが可能である．そのため，反応が早く，メモリ消費量の少ないプログラムを作成することが可能である．また，ドキュメント全体の読み込みが終わるのを待つ必要がないので，ネットワークなどを介してドキュメントを読み込みながら作業を行うことも可能である．SAX も DOM と同様に，SAX そのものは仕様なので，実際のプログラミングでは，SAX をサポートする XML パーサーを使用する．SAX は XML 文書を読み込みながら，イベントを発生させ，そのイベントを処理することでドキュメントにアクセスしたり操作をする．DOM は XML 文書の変更や新規作成もサポートしている一方，SAX はそういう機能はない．よって，SAX で XML 文書や HTML 文書を変更したり生成したりするときには，フィルタのように結果を随時出力するプログラムとして作成するか，生成したり変更するドキュメントの内容をメモリ上に保存するデータ構造を独自に作るか，ドキュメントの生成部分に DOM を用いる必要がある．

Web ブラウザ上で JavaScript による SAX パーサーを構築する議論は現段階ではほとんど

されていないため，本研究では，Ajax によってキーワードの候補を出す際に DOM を使用した．また，検索キーワードの類義語，上位語，下位語を取得する際には JSP を用いるため，SAX を使用した．本研究で用いる WordNet.OWL は 70MB の XML 文書であるため，あらかじめ分割することにより，アクセスの高速化を試みた．

3.3.5. JSP・Apache Tomcat

本研究では，検索結果の入出力の際，JSP（Java Server Pages）を使用した．また，サーブレットコンテナとして Apache Tomcat を使用した．Apache Tomcat は Apache Software Foundation で開発された Web コンテナであり，Web サーバと連携して実行できる Java Servlet と JSP の仕様を実装している．JSP の実行がリクエストされると，サーバである Apache Tomcat は JSP ソースファイルをサーブレットのソースコードに変換する[17]．そしてそのソースコードをその場でコンパイルして実行し，結果をクライアントに返信する．本研究では，Tomcat 5.5.23，JDK 1.5.11 を使用した．

4. 実装

開発するシステムの実装の詳細を以下に示す．

4.1. 実装するファイル

図 6 は実装するファイルの関係図を示している．

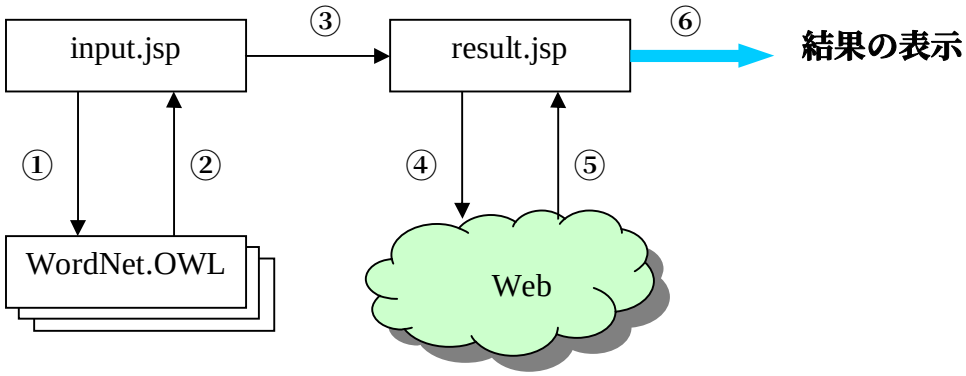

図 6．ファイルの関係図

・input.jsp

検索キーワードを入力，いくつかのファイルに分割された WordNet.OWL より取得した関連語を表示し，ユーザによって選択された関連語と検索キーワードを result.jsp へ送る．

・WordNet.OWL

英語のオントロジーで，OWL の形式で記述されている．

・result.jsp

input.jsp からの検索キーワード受け取り，Yahoo! JAPAN Web サービスを利用して，Web 検索を行う．取得した URL から，それぞれの Web ページのソースを取得し，Web ページが含む検索キーワードの数をカウントし，得点計算を行う．計算結果より，得点の高い順にソートし，結果をブラウザに表示する．今回は英語のオントロジーを利用しているため，英語で書かれたページの検索を行うこととする．

4.2. WordNet.OWL の読み込み

WordNet.OWL の構成を，上から順に大きく分けると以下①～⑦のようになる．

① DOCTYPE 宣言（ENTITY 宣言）

例：<!DOCTYPE owl [<!ENTITY wn ‘http://www.unine.ch/knowler/wordnet#’>]>

② 品詞（Noun など）や意味的関係（hyponymOf，similarTo など）の定義や説明

例：
```
<owl:Class rdf:about="&wn;Adverb">
    <rdfs:subClassOf rdf:resource="&wn;LexicalConcept"/>
    <owl:disjointWith rdf:resource="&wn;Nouns_and_Verbs" />
    <owl:disjointWith rdf:resource="&wn;Nouns_and_Adjectives" />
</owl:Class>
```

③ 番号によって表記されている語彙の品詞分け

例：
```
<wn:Noun rdf:about="&wn;c112947045" />
<wn:Verb rdf:about="&wn;c200001742" />
```

④ 語彙の一般的な表記とそれを ENTITY と結びつけたもの（WordObject）の定義

例：
```
<wn:WordObject rdf:about="&wn;word_Amazon_River">
    <wn:label>Amazon River</wn:label>
</wn:WordObject>
```

⑤ 語彙を番号によって表記するための定義

例：
```
<rdf:Description rdf:about="&wn;c107428489">
    <wn:wordForm rdf:resource="&wn;word_America" />
    <wn:wordForm rdf:resource="&wn;word_U.S.A." />
    <wn:wordForm rdf:resource="&wn;word_United_States_of_America" />
</rdf:Description>
```

⑥ 語彙の意味や説明

例：<rdf:Description rdf:about="&wn;c107428489">

```
    <wn:glossaryEntry>
      North American republic containing 50 states - 48 conterminous states in North
      America plus Alaska in northwest North America and the Hawaiian Islands in the
      Pacific Ocean; achieved independence in 1776
    </wn:glossaryEntry>
  </rdf:Description>
```

⑦番号によって表記されている語彙の意味的関係（hyponymOf，similarTo **など）**

例：

```
<rdf:Description rdf:about="&wn;c107428489">
    <wn:hyponymOf rdf:resource="&wn;c107168879" />
  </rdf:Description>
```

本研究では，検索の際に使用するキーワード候補，または類義語や上位語，下位語などの関連語を WordNet.OWL **から取得する. キーワード候補取得の際には** Ajax **による** DOM **を, 類義語などの関連語取得の際には** JSP **による** SAX **を用いて** XML **形式の** Web **オントロジー言語** OWL **で記述された** WordNet.OWL **を読み込んでいく．今回のシステムを実装するにあたり,** 70MB **ある** XML **形式の** WordNet.OWL **ファイルを読み込むのは,** DOM **を用いた場合, 一旦メモリ上に文書全体を保存するためにメモリの消費がとても大きく，**SAX **を用いる場合にも，何度もファイルを一番上から読み込むのは無駄である．よってそれらを防ぐために，今回のシステムを実装に必要な部分だけを抜き出し，あらかじめ** WordNet.OWL **を以下のように分割した.**

・wn_word_A.xml～wn_word_Z.xml

構成部分①と④を組み合わせたものであり，④をさらに語彙の 1 **文字目を索引として，**A～Z **を分類したファイルである.**

・wordform.xml

構成部分①と⑤を組み合わせたものである.

・hyponymOf.xml

構成部分①と⑦を組み合わせたものであり, 意味的関係が書かれている⑦の「hyponymOf**」の部分だけを抜き出したものである.**

以下, WordNet.OWL **を読み込んでいく際のプログラムの流れを, 検索キーワードを“** CPU **”とした場合を例にとって説明する.**

4.2.1. 類義語の取得

まず，検索キーワードが入力された瞬間に，検索キーワードを入力したテキストボック

スに入っている文字列と同じ文字列を上述の WordNet.OWL の構成部分④より探していく．検索キーワードと前方一致した wn:label 要素の内容をキーワードの候補として出力する．この例の場合，候補として“ CPU ”と“ CPU board ”が出力される．

```
<wn:WordObject rdf:about=”&wn;word_CPR”><wn:label>CPR</wn:label></wn:WordObject>
<wn:WordObject rdf:about=”&wn;word_CPU”><wn:label>CPU</wn:label></wn:WordObject>
<wn:WordObject rdf:about=”&wn;word_CPU_board”>
        <wn:label>CPU board</wn:label></wn:WordObject>
<wn:WordObject rdf:about=”&wn;word_CRO”><wn:label>CRO</wn:label></wn:WordObject>
```

図 7．WordNet.OWL の④の一部分

次に検索キーワードの類義語の取得をする．検索キーワードと要素の内容が一致した wn:label 要素の親要素（wn:WordObject）の属性（rdf:about）の値である「&wn;word_CPU」を抜き出し，これを基に類義語の検索をする．&wn;は構成部分①で宣言されている ENTITY の参照部分であり，WordNet.OWL では「http://www.unine.ch/knowler/wordnet#」として宣言されている．格納された rdf:about 属性の値と同じものを，構成部分⑤より探していく．

```
<rdf:Description rdf:about="&wn;c102607679">
   <wn:wordForm rdf:resource="&wn;word_mainframe" />
   <wn:wordForm rdf:resource="&wn;word_central_processor" />
   <wn:wordForm rdf:resource="&wn;word_central_processing_unit" />
   <wn:wordForm rdf:resource="&wn;word_C.P.U." />
   <wn:wordForm rdf:resource="&wn;word_CPU" />
   <wn:wordForm rdf:resource="&wn;word_processor" />
</rdf:Description>
```

図 8．WordNet.OWL の⑤の一部分

「&wn;word_CPU」を探していくと，wn:wordForm 要素の属性（rdf:resource）の値と一致する．また，このとき一致した要素と同じ階層の要素の属性値から「&wn;word_」を除いたものが類義語となっている．この例では，“ CPU ”の類義語は，“ mainframe ”“ central_processor ”“ central_processing_unit ”“ C.P.U. ”“ processor ”と分かる．

4.2.2. 上位語・下位語の取得

さらに，上位語や下位語の取得をする．属性値が一致した要素の親要素（rdf:Description）

の属性（rdf:about）の値「&wn;c102607679」が，WordNet.OWL の中で“ CPU ”や“ mainframe ”という語彙群を番号で表した時の表記である．これと同じものを，構成部分⑦の hyponymOf の意味的関係を記述してある部分より探していく．

```
<rdf:Description rdf:about="&wn;c102607679">
    <wn:hyponymOf rdf:resource="&wn;c102855671" />
</rdf:Description>
```

```
<rdf:Description rdf:about="&wn;c102607679">
    <wn:hyponymOf rdf:resource="&wn;c103043877" />
</rdf:Description>
```

図 9．WordNet.OWL の⑦の一部分

「&wn; c102607679」を探していくと，rdf:Description 要素の属性（rdf:about）の値と一致したものが 2 つある．ここでは，rdf:Description 要素の属性値「&wn; c102607679」が下位語であり，rdf:Description 要素の子要素（wn:hyponymOf）の属性値「&wn; c102855671」「&wn; c103043877」が上位語である．この例では「&wn; c102607679」の下位語は見つからなかった．得られた上位語の番号を，再び構成部分④より検索していく．

```
<rdf:Description rdf:about="&wn;c102855671">
  <wn:wordForm rdf:resource="&wn;word_electronic_equipment" />
</rdf:Description>
```

```
<rdf:Description rdf:about="&wn;c103043877">
  <wn:wordForm rdf:resource="&wn;word_hardware" />
  <wn:wordForm rdf:resource="&wn;word_computer_hardware" />
</rdf:Description>
```

図 10．WordNet.OWL の④の一部分

「&wn; c102855671」「&wn; c103043877」を探していき，一致した属性値（rdf:about）の要素（rdf:Description）の子要素（wn:wordForm）の属性値から「&wn;word_」を除いた部分が，上位語となっている．つまりこの例では，“ CPU ”の上位語は“ electronic_equipment ”“ hardware ”“ computer_hardware ”と分かる．

4.3. 検索方法

Web 検索には Yahoo! JAPAN Web サービスを利用する．検索キーワードとその関連語を一緒に検索をするときには OR 検索を用いる[18]．OR 検索は，例えば単語 A と単語 B があ

った場合，A または B が含まれるページが検索される．また検索キーワードとして複数の入力があった場合，それらの間には AND 検索を用いる[18]．上と同じように単語 A と単語 B があった場合，A と B の両方が含まれるページが検索される．よって例として入力された検索キーワードが単語 C と単語 D であった場合，単語 C の関連語を c1, c2，単語 D の関連語を d1 とすると，今回のシステムでは，「（C OR c1 OR c2）AND（D OR d1）」といった形式で検索を行う．

また本研究では一度に取得できるページ数の限界である 100 件ごとに総得点を計算して出力される Web ページの順番を入れ替えることとする．

4.4. スコアリング

本研究では，検索の際の入力キーワードとそのオントロジーから得られた類義語や上位語，下位語などの関連語の数を Web ページのソース解析によりカウントしていく．その際，入力キーワードと関連語には重み付けをすることにより，検索キーワードと関連語の間に差をつけることとする．ユーザが入力したキーワードは，ユーザがそのキーワードを含む Web ページを必要としているということなので，重要度が一番高い．それとは逆に類義語は，入力したキーワードと意味は似ているが，完全に同じではない語もある．それらが多く含む Web ページを上位に表示してもユーザが意図しない結果が出ることがある．

また普段 Web 検索をする際，より必要な情報が含まれている Web ページのみを検索結果として得るために，2 語以上の複数のキーワードを入力して検索することが多い．このとき，1 つ目のキーワードは検索の主となる単語を入力し，2 つ目以降はどちらかと言えば，1 つ目のキーワードの補足的な役割を持つキーワードを入れることが多い．よって 1 つ目のキーワードに，より強い重みを与えることとする．本研究で与えた重みを表 1 に示す．

表 1．入力キーワードとその関連語の重み

	キーワード 1 つ目	キーワード 2 つ目以降
入力キーワード	1.0	0.8
関連語	0.75	0.6

上述の重みと合わせて，Web ページにある，入力キーワードとその関連語をカウントしていく際，重要と思われる HTML タグの中に含まれている場合は，より強い重み付けをし，高い得点を与える．本研究で重みを与える HTML タグを以下に示す[19]．

・<title>タグ

Web ページのタイトルが記載されているため，そのページ何について記述されているかを表している．最も重要なタグである．

・<h1>～<h6>タグ

heading 要素は見出しとして使用するタグであるので，その Web ページの文章の内容を表している．数字が大きくなるほど，小さな見出しになるので重要度が下がっていく．<h5><h6>は通常の<body>タグの文字より小さくなるため，重要度は高くない．

・<meta>タグ

このタグの中に書かれた内容は，実際にページを閲覧する際には表示されないが，Web ページのメタ情報として記述される．name 属性で記述する内容を指定する際，Web ページの作成者が何を書いてもいいのだが，一般の検索エンジンのヒット率向上のため，属性の値として，“ keywords ”（Web ページの内容のキーワードを記述する）や“ description ”（Web ページの内容を記述する）が使われている場合が多い．端的に内容を示している可能性が高いが，記述された内容は表示されないため，Web ページの内容とは関係の無い単語を並べる作成者もいるため，そこまで重要度を高めにしない．

・ タグ

中央表示するためのタグである．中央表示をするということは，見出し的要素が強いと考えられるため，重要度が高い．

・, , <b>, <i>, <big>タグ

文字を強調するためのタグであるので，強調される分重要度が高い．やは<b>や<i>よりもより強調したい時に使用する．

・<a>タグ

他のページやサイトへのリンクを貼る際に使用するタグである．他のページにリンクを貼るということは，そのページに関する重要な内容があると考えられるため，重要度が高い．

・<ul>, タグ

箇条書きにする際に使用するタグである．箇条書きには，重要な内容が簡潔にまとめられていると考えられるため，重要度が高い．

・alt 属性

属性ではあるが，画像，フォーム，アプレットが表示できない場合に代替テキストを指定するため，キーワードのカウントに含める．Web ブラウザが Internet Explorer の場合，マウスカーソルを当てると小さなポップアップテキストが表示されるが，一見表示されないため，そこまで重要度を高くしない．

・title 属性

属性ではあるが，ある要素の補足情報を付け足すときに記述されるため，キーワードのカウントに含める．alt 属性と同様，マウスカーソルを当てると小さなポップアップテキストが表示されるが，一見表示されないため，そこまで重要度を高くしない．

実際の重み付けであるが，以上に記述したことと，今回提案する手法として，そのタグ

が使われているページ数とタグの出現頻度から考慮して重みを決めることとする．本研究では<title>タグに着目した．<title>タグには，Web ページのタイトルが記述されるため，最も重要なタグといえる．またほぼすべてのページに出現し，そのページの中では 1 度しか使用されない．よって<title>タグの重みが一番大きくなるように以下のように式を立て算出することにした．

$$w = f_t \times \frac{1}{A_t} \quad (1)$$

f_t ：タグ t が使われているページ数の割合(%)

A_t ：タグ t の平均使用回数

表 2 は，検索キーワードとして様々な分野から 11 語の単語を選び出し，それぞれの Yahoo! JAPAN の検索結果の上位 100 件，つまり合計 1100 件の Web ページについて式(1)を使って計算したものである．例えば<h1>タグについて，1100 件中 399 件のページ内で使われており，使われているページの中での平均使用回数は 1296(回)／399(件)＝3.248120301 であった．式(1)を使って計算すると 399／1100×100×（1／3.248120301）＝11.16729798 となる．

表 2．重要な HTML タグについての(1)式の計算結果

<title>タグ	93.27701049
<h1>タグ	11.16729798
<h2>タグ	5.796671889
<h3>タグ	3.03725307
<h4>タグ	1.834352373
<h5>タグ	1.49271012
<h6>タグ	0.821022727
<meta name=”Keywords”>タグ	58.09773075
<meta name=”Description”>タグ	55.46627099
タグ	5.048885077
タグ	3.820335146
タグ	1.794817332
<b>タグ	4.071973267
<i>タグ	1.069920765
<big>タグ	0.278409091
<a>タグ	1.03307372

タグ	4.744776119
タグ	3.386518244
alt 属性	6.042158071
title 属性	1.652856904

<title>タグの数値について，今回は整数値が得られなかった．原因を調べてみると，<title>タグが 2 回使われているページや 1 度も使われていないページがあった．また，Web ページ 1100 件について調べたが，1030 件ほどの結果しか得られなかった．これはすでにリンク切れのページやサーバからの応答が返って来ないために接続できなかったものを含んでいるからである．

ところで表 2 のデータをそのまま重みに当てはめるのには，問題点がいくつかある．それを以下に示す．

(1) 同じように使われるべきではあるが，認知度の低いタグに関しては，数値が低くなりすぎてしまう（<big>タグと<b>タグ，<i>タグなど）
(2) 認知度が低く使われるページは少ないが，そのページごとでの出現回数が少ないと数値が大きくなりすぎてしまう（<h5>タグ，<h6>タグなど）
(3) <title>タグの重みが大きすぎる

まず，同じような重要度で使われるタグに関しては，それらの平均を取ることにした．次に，<h5>タグや<h6>タグ，alt 属性，title 属性などそこまで重要度が高くない要素に関しては，<body>タグの重みと同様にすることとする．<title>タグの重みが大きくなりすぎてしまう問題に関しては，それぞれ 10 倍したものの対数を取ることで解決する．また<meta>タグに関しては，先述の通り，Web ページ閲覧の際には表示されないため，重要度を少し下げることとする．以上のことから，最終的に重みを決定する式を以下に示す．

$$w' = \log\left(f_t \times \frac{1}{A_t} \times 10 \right) \tag{2}$$

f_t ：タグ t が使われているページ数の割合(%)
A_t ：タグ t の平均使用回数

式(2)より本研究で与えた HTML タグの重みを表 3 に示す．

表 3. HTML タグの重み（小数第 2 位を四捨五入）

<body>タグ(基準)	1.0
<title>タグ	6.8
<h1>タグ	4.7
<h2>タグ	4.1
<h3>タグ	3.4
<h4>タグ	2.9
<h5>タグ	1.0
<h6>タグ	1.0
<meta name="Keywords">タグ	4.4
<meta name="Description">タグ	4.4
タグ	3.9
タグ	3.3
タグ	3.3
<b>タグ	2.9
<i>タグ	2.9
<big>タグ	2.9
<a>タグ	2.3
タグ	3.7
タグ	3.7
alt 属性	1.0
title 属性	1.0

それぞれのタグに含んでいるキーワードおよび関連語をカウントし，表 1 の重みと表 3 の重みを掛け合わせて，総得点が高い順に検索結果として表示する．

5. 実行結果

以下にシステムの実行結果を示す．今回は入力キーワードに“ cpu ”とした場合と“ java ”とした場合をテストケースとした．フレームを用いて，左にキーワードの入力，関連語の取得を行う input.jsp，右に得点の計算を行い，検索結果の出力を行う result.jsp を表示した．

図 11. 入力キーワードを“ cpu ”とした場合の検索結果

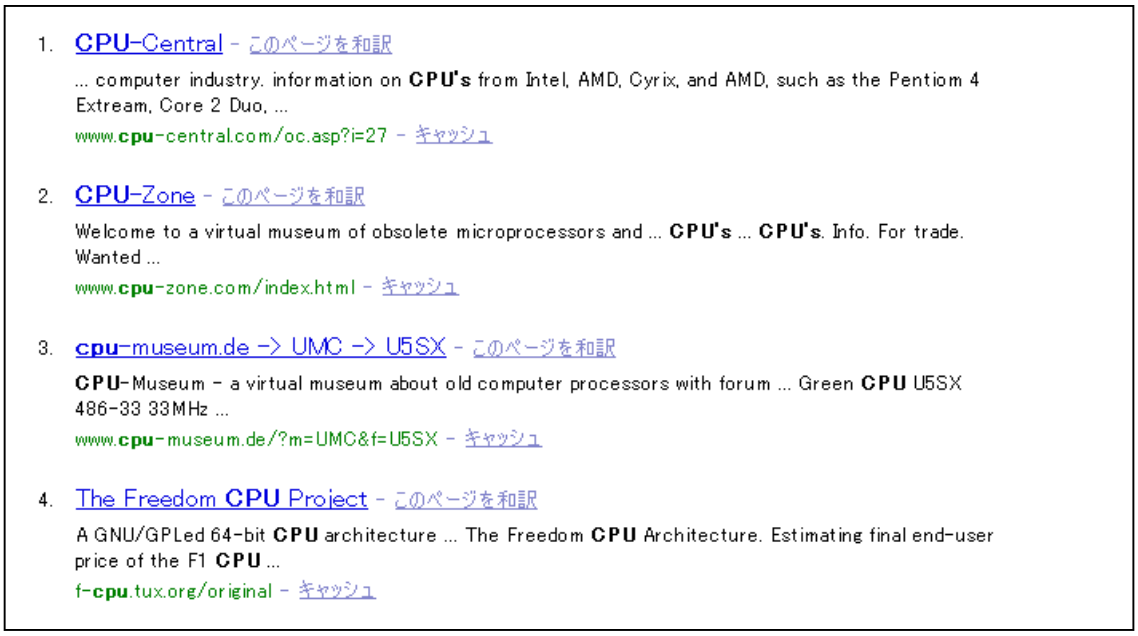

図 12. Yahoo! JAPAN で入力キーワードを“ cpu ”とした場合の検索結果

図 13. 入力キーワードを“ java ”とした場合の検索結果

図 14. Yahoo! JAPAN で入力キーワードを“ java ”とした場合の検索結果

6. まとめ・評価

本研究では，検索キーワードの類義語，上位語，下位語といった関連語を英語のオントロジーファイルである WordNet.OWL から取得，それらを検索語として Web 検索を行い，それぞれに付けられた重みからスコアリングを行い，総得点の高い，つまり重要な Web ページの上位表示するシステムを実装した．

今回実装したシステムの実行した結果と，Yahoo! JAPAN で検索結果を比較すると，今まで検索できなかったページができるようになった．また，検索結果の順位が変わったことが確認できた．例えば，“CPU”と入力して検索した場合，ユーザは CPU とはどういうものか知りたいことが多い．今回のシステムの検索結果ではそれらのページが Yahoo! JAPAN の検索結果よりも上位に表示された．よって，本研究の目的はある程度達成されたといえる．しかし今回のシステムを利用した場合，関連語の取得に約 8 秒，関連語を 5 つとして検索すると，検索結果の出力には平均して 200 秒ほど掛かる．この検索時間はとても実用的とはいえない．多少のスコアリング時のアルゴリズムの改善の余地はあるものの，一件ごとに Web ページをインターネットから取得しているため，Web ページ取得先のサーバの処理速度なども大きく影響する．ゆえに，今回のシステムで普段私達が使っている検索エンジンの検索速度に近づけるのは難しいことがわかった．

今回関連語の取得には WordNet.OWL という英語のオントロジーを利用したが，今後実用化段階の日本語オントロジーが公開されれば，そこから関連語を取得して，日本語の Web ページを検索することが可能である．

7. 今後の課題

今後の課題としては，まず検索結果で表示される Web ページの順番に大きく影響する，重みの部分の改善が考えられる．一番簡単な方法として挙げられるのは，まずある単語を例にとり，無作為に選んだいくつかの Web ページをサンプルとする．そして，<body>タグを基準とし，<title>タグなどにその単語が書かれている場合，<body>タグにいくつ書かれている場合に相当するかを考えることである．しかし，この方法は人が 1 つ 1 つ判断しないといけなく，困難であると予想されるため，別の方法を考える必要があるかもしれない．

次の課題として 6 章でも触れた検索時間の改善がある．実際に使っている検索エンジンでは，「クローラー」あるいは「スパイダー」と呼ばれるロボット（プログラム）を用いて World Wide Web 上の Web ページの情報を収集し，検索アルゴリズムが扱いやすいデータに変換した上で，インデックス（データベース）に格納するという作業を行っているので，Web 検索の高速化が可能である[20]．しかし，今回実装したシステムは大規模な検索エンジンではないため，クローリングしてデータベースに格納するということはしなかった．今回のシステムでも，検索ページを増やすことに伴って，さらに必要な検索結果出力の高

速化のためには，そのようなデータベースを持ったり，いくつかのコンピュータで並列処理をする必要があるだろう．それぞれのコンピュータにより計算した後，計算結果を照合し，得点の高いものから順に出力をすれば良い．

また将来的には，オントロジーはそれぞれのコンピュータにダウンロードをしてから，そのオントロジーファイルにそれぞれがアクセスをして検索するということは想像しづらいため，サーバ上に置いてそこにアクセスをして関連語などを取得できるようにする必要があると考えられる．

参考文献

[1] Grigoris Antoniou, Frank van Harmelen, 訳:ジャストシステム知識活用研究グループ Gnosis, *CD-ROM で始めるセマンティック Web*, June 2005.

[2] 斎藤信男, 荻野達也, *セマンティック Web 入門*, November 2004.

[3] 木村 文則, 前田 亮, 越田 高志, 宮崎 純, 植村 俊亮, *Web ディレクトリを用いた 2 言語オントロジーの構築*

[4] 斉藤孝, *意味論からの情報システム ユビキタス・オントロジ・セマンティックス*

[5] ウェブのオントロジー言語 OWL -- ウェブに存在するものとその関係の定義
http://www.kanzaki.com/docs/sw/webont-owl.html

[6] Knowledge, Information and Data Processing Group. "WordNet.OWL"
http://taurus.unine.ch/GroupHome/knowler/wordnet.html

[7] 神崎正英. *"日本語ウェブ・オントロジーの試み"*
http://www.kanzaki.com/docs/sw/jwebont.html

[8] Wikipedia:「全文検索」
http://ja.wikipedia.org/wiki/Semantic_Web

[9] 汎用連想計算エンジン GETA
http://geta.ex.nii.ac.jp/

[10] 岩本のぞみ, *事例でわかる Web サービス・ビジネス*, January 2004

[11] Yahoo! デベロッパーネットワーク
http://developer.yahoo.co.jp/

[12] Wikipedia:「WordNet」
http://ja.wikipedia.org/wiki/WordNet

[13] Wikipedia:「Ajax」
http://ja.wikipedia.org/wiki/Ajax

[14] 【IT 用語】Ajax -Asynchronous JavaScript + XML-
http://mikilab.doshisha.ac.jp/dia/research/report/2005/0822/004/report20050822004.html

[15] prototype.js v1.5.0 **の使い方**

http://www.imgsrc.co.jp/~kuriyama/prototype/prototype.js.html

[16] **日向俊二**, ***独習 XML 第 2 版***, 2004

[17] Wikipedia:「JavaServer Pages」

http://ja.wikipedia.org/wiki/JavaServer_Pages

[18] Yahoo! **検索 ヘルプ - 演算子（+,-など）の使い方**

http://help.yahoo.co.jp/help/jp/search/search-07.html

[19] HTML **コード攻略**

http://www.accessup2.com/link7.htm

[20] **住 太陽の**「SEO **検索エンジン最適化**」

http://www.searchengineoptimization.jp/