商用検索エンジンの検索結果では取得できないランキング下位部分の収集・解析

Gathering and Analysis of Unlisted Search Engines' Results

舟橋 卓也[1] 上田 高徳[1]
平手 勇宇[2] 山名 早人[3]

Takuya FUNAHASHI Takanori UEDA
Yu HIRATE Hayato YAMANA

現在，検索エンジンは Web の入り口として広く利用されている．既存の商用検索エンジンの多くは検索結果として Web ページのランキングを行い，ランキングが高い Web ページから順に返す．しかし，既存の商用検索エンジンはランキング上位一定件数しか検索結果として返さない．そのため，ユーザはランキング外となった Web ページを知ることができない．検索結果においてランキング外となった Web ページを取得することができれば，ユーザ自らが検索結果を解析することにより，検索エンジンの特徴解析などの調査を行うことができる．そこで本稿では，検索エンジンにおいて取得検索結果件数の制限により取得できないランキングが下位の Web ページを UnListed Search Result(ULSR)と名づけ，ULSR の取得を目指した．提案手法では，クエリ中に「-」(マイナスオペレータ)を用いて検索問題を排他集合に細分化することにより，検索結果として ULSR を網羅することができるクエリを生成し，ULSR の収集を行った．また，提案手法により収集された ULSR の傾向を知るために，ランキング上位の Web ページ集合との比較を行った．ULSR の内容を解析した結果，ULSR には上位の検索結果には現れない様々なトップレベルドメインを持つ Web ページが存在するという特徴があることがわかった．

When searching for information on the Internet, it becomes general to use search engines. Most of the existing commercial search engines rank web pages and show them in the order from the highest ranked web page to the lowest. However, these conventional commercial search engines are able to show only a limited number of high ranked web pages. That means low ranked web pages are not shown. If users get unranked web pages, they can analyze these pages and may find useful web pages. Therefore, this paper focuses on those that are not ranked high enough to be on a search engine, which we call "UnListed Search Result, ULSR." We will propose an idea how to get them from the search engine and analyze web pages in ULSR. In this paper, we divide query's condition into many small queries by using '-' (minus operator) in order to gather all ULSR. After analyzing ULSR, we have confirmed that it had a characteristic that ULSR consists of web pages from various top level domains.

[1] 学生会員 早稲田大学大学院基幹理工学研究科修士課程
{takuya, ueda}@yama.info.waseda.ac.jp
[2] 正会員 早稲田大学メディアネットワークセンター
hirate@yama.info.waseda.ac.jp
[3] 正会員 早稲田大学理工学術院
yamana@yama.info.waseda.ac.jp

1. はじめに

Web の規模が膨大になった現在，Web 空間から求める情報を取り出すため検索エンジンが広く利用されるようになった．日本においては Yahoo! JAPAN[16]や Google[6]などの商用検索エンジンが広く利用されている．これらの検索エンジンは，ユーザからクエリを受け取ると，そのクエリを含む Web ページ群を，あらかじめ収集してある Web ページの中から検索し，ランク付けをしてユーザに提示するシステムとなっている．

商用検索エンジンを用いると，クエリに一致する Web ページ群を容易に取得できる一方，取得できる Web ページ数に制限が生じる．例えば Yahoo! JAPAN や Google，MSN[11]では，検索結果のランキング上位 1,000 件までしか結果を取得することができない．本論文ではこのような検索エンジンから取得できる検索結果の最大値を取得限界数と呼ぶ．検索エンジンが返す検索結果は取得限界数までしか得ることができないため，ランキング外となった Web ページについて，ユーザはその内容を知ることができない．検索エンジンのランキング外となった Web ページを取得することができれば，ユーザ自らが検索結果を解析することによって，検索エンジンのランキング動向調査や著作権違反ページの網羅的な抽出など、様々な用途に利用できると考えられる．

従来，Deep Web を対象として網羅的にデータを収集する方式が研究されている[2] [8][14]．Deep Web とは，クエリフォームのバックエンドに存在するデータベースから，クエリフォームを通じて動的に生成される Web ページ群を指す[10]．検索エンジンはバックエンドに持つデータベースからクエリフォームを通じて，ユーザから与えられたクエリの条件に一致する Web ページを検索し，動的に検索結果ページを生成してユーザに提示を行う．そのため，検索エンジンが返す検索結果は Deep Web の１つであると考えることができる．既存の Deep Web 収集手法は一般的な Web サーバの配下にあるデータベースを対象としたものであり，検索エンジンが一般的に持つ「排他集合を求める検索」をサポートしていない．つまり，Deep Web を対象とした場合，網羅的にデータを抽出するためには，クエリに追加する単語をどのように選択するかが課題となっている．これに対して検索エンジンを対象とした場合は，排他集合を求める検索を利用することができ，効率的なデータ抽出が可能となる．

そこで本論文では，検索エンジンにおいて取得検索結果数の制限により取得できないランキングが下位の Web ページを UnListed Search Result(ULSR)と名づけ，ULSR の取得を，多くの検索エンジンで利用可能な 「-」（マイナスオペレータ）を利用して行う手法を提案する．マイナスオペレータとは，「-」の直後に入力された条件に一致しない検索結果集合を返すオペレータであり，キーワードや URL の一部を条件として指定することができる．マイナスオペレータを用いることにより，検索エンジンにおいて排他集合を求める検索を行うことができる．本論文ではマイナスオペレータを用い，任意の条件を含む検索結果と含まない検索結果，それぞれを取得し，網羅的な検索結果の収集を行った．

また本論文では，ULSR が取得限界数以内の検索結果（Listed Search Result，LSR）と異なった傾向をもっているのか調査を行うために，ULSR の解析を行った．具体的には，ULSR と LSR の間にトップレベルドメイン分布の差異が存在しているのか，確認を行った．

以下，次のような構成をとる．まず，２節では，本研究の関連研究である Deep Web について述べる．次に，３節で提案する ULSR の収集手法と，収集した ULSR の解析手法について述べる．４節で検索エンジンに提案手法を適用した実験結果を示し，最後に５節にてまとめを述べる．

2. Deep Web

2.1 Deep Web の定義

Deep Web とは，検索エンジンのクローラによって収集されていない Web ページの総称である[10]．クローラでは収集することが難しい Web ページとして，次の(1)～(4)のような Web ページが挙げられる．(1)専用のオンラインクエリフォームを通じてデータベースから動的に生成される Web ページ，(2)自分以外の Web ページから静的なリンクが存在しない Web ページ，(3)スクリプト言語を通して動的に動く，Flash 等で構成された Web ページ，(4)パスワード認証が必要な Web ページ．(1)～(4)のような Web ページは，現在のクローラが一般的に行っている「起点 Web ページから静的なリンク構造を辿って Web ページを収集する手法」では収集することが難しく，Deep Web となりやすい．(1)～(4)の中でも特に(1)は WebDB[17]とも呼ばれ，Deep Web の主要な構成要素となっている．そのため，近年の研究では WebDB のみを指して Deep Web と呼ぶ場合がある[2][8]．本研究においても，WebDB を指して DeepWeb と呼ぶ．

2.2 Deep Web の取得

Bergman による研究[10]では，Deep Web は容量にして，表層に存在する Web ページ全体の 500 倍存在していることが示された．表層 Web の 500 倍もの容量を持つと言われている Deep Web の中には，表層 Web には含まれていない情報が存在していると考えられている．そのため，Deep Web 内の情報を取得するための研究が行われている[2] [8][14] ．

ほとんどの Deep Web サイトにおいては，DB から情報を取得するために，キーワード検索を行うシンプルなクエリフォームが設けられている[14]．ここで Deep Web サイトとは，Deep Web を持つ Web サイトを指す．ユーザがそのシンプルなクエリフォームにキーワードの入力を行うと，Deep Web サイトが対象とする主なトピックスに関して全文検索を行い，一致するものを結果として返す．シンプルなクエリフォームは多くの Deep Web サイトに設置されてため，そのクエリフォームを通じて Deep Web を取得しようという試みが行われている．シンプルなクエリフォームを通じ Deep Web を取得する手法は一般的に次の手順で行われている．(1)クエリ辞書 Q を作成する，(2)Q からクエリ q を選択する，(3)データベース D に q をサブミットする，(4)D から q への応答として m 件の検索結果を得る，(5)[オプション]検索結果を用いて Q を更新する，(6)終了条件を満たすまで，(2)から(5)を繰り返す

Ntoulas ら[1]は，Amazon[1]や Open Directory[12]を収集対象として，事前に用意した Web ページ集合に出現する単語を Q として用いた．Q を更新するために，検索結果として取得した Web ページに現れる単語を Q へ追加している．Ntoulas らの研究では q として，Q の中から出現率が高く，かつストップワードではない単語を使用している．その結果，対象とした多くの Deep Web サイトにて 9 割以上の高い網羅率を得ることができたと報告されている．

また，Wu ら[14]は DB 内の１つの属性値をノード，属性値が同じタプルに存在することをエッジと見なしたグラフ(Attribute-Value-based Graph, AVG)を用いると，クエリフォームを通じた Deep Web の収集は Web グラフの巡回問題と同様に考えることができると論じている．AVG では，多くのタプルに属するハブとなるフィールド要素が存在している場合が多いため，ハブの要素を考慮して q を選択することによって時間的効率を高め収集することが可能となる．

Álvarez ら[8]は，ラジオボタンなども含む様々なクエリフォームを通じて Deep Web の収集を行うことを目的として，クエリフォームの構造を同定するクローラ DeepBot の構築を行った．DeepBot は高い精度でクエリフォームの構造を同定することができるものの，同定したクエリフォームにクエリをサブミットする機能は乏しく，Álvarez らは論文中において，[8]と[1]は補完しあうことができる論文であると述べている．

2.3 既存研究の問題点と本研究との関連

本研究が取得対象とする ULSR は，Deep Web の取得手法を応用して取得することが可能である．現在の商用検索エンジンは，背後にデータベースを持ち，かつクエリフォームを通じてユーザに対して動的に生成された検索結果ページを提示するシステムとなっている．そのため，2.2 で示した手法を用いることにより，クエリフォームを通じて ULSR を取得することが可能となる．しかし，既存の Deep Web 収集手法では，Deep Web を汎用的に収集することを目的としているため，多くの Deep Web サイトが備えているシンプルな検索構文を用いた収集手法を提案している．それに対して検索エンジンでは，検索エンジンが一般的に備えている特殊な検索構文を利用することが可能である．本論文では，こうした検索エンジンが一般的に備える特殊な検索構文を用いた ULSR の取得を行う．

3 提案手法

3.1 提案手法の概要

提案手法では既存の Deep Web 取得手法に加えて，検索エンジンに特化し，ULSR の取得を行う．現在，多くの検索エンジンは「排他集合を求める検索」をサポートしている．排他集合を求める検索とは，任意の条件に一致する集合と，条件に一致しない集合をそれぞれ収集する検索である．排他集合を求める検索は，ほとんどの検索エンジンにおいて「-」(マイナスオペレータ) と呼ばれる演算子により実現されている．マイナスオペレータとは，「-」（マイナス）の直後に入力された条件に一致しない検索結果集合をユーザに返すオペレータであり，条件としてキーワードや URL の一部を指定することができる．「-」を用いることにより，検索エンジンにおいて排他集合を求める検索を行うことができる．本論文では，「-」を用いて「排他集合を求める検索」を行い，論理的に完全性のある検索結果を取得する手法を提案する．

3.1.1 完全性のある検索結果の取得

ここでは，完全性のあるクエリの生成方法について述べる．ここで「完全性がある」とは，検索エンジンにインデックス化されているページに対して，入力したクエリに一致する Web ページを論理的にすべて取得できることを指す．完全性のあるクエリを使用することができれば，検索時に使用するクエリの内容に依存しない ULSR の収集が可能となる．

まず，クエリ *query* に対する完全性のある検索結果を $S(query)$とする．このとき，$|S(query)|$を$S(query)$の結果件数であると定義をすると，$|S(query)| \leqq N$ であれば，検索エンジンに *query* を投げるだけで $S(query)$を取得することが可能である．このとき，N は取得限界数であり，Yahoo! JAPAN や Google，MSN では 1,000 となる．$|S(query)| > N$ であれば，次の(1)式を満たす制約条件の集合 Cを用いて，入力クエリ *query* に制約をつけて複数回検索を行い，$S(query)$ の部分集合を排他的に取得すればよい．

$$S(query) = \bigcup_{c \in C} S(query \wedge c) \ldots (1)$$

ここで，$|S(query \wedge c)| > N$となるような cが存在した場合，$query \wedge c$を新たなクエリ *query'*として生成する．そして，*query'*に関して同様に(1)式を満たす制約集合を作成する．このように再帰的に(1)式を満たす制約を付加していくことで，完全性のある検索結果が取得可能となる．

提案手法では，(1)式を満たす制約集合として，トップレベルドメイン (TLD) による制約と，マイナスオペレータ (「-」) を用いた制約を利用した．「-」を利用した制約では，inurl 構文が利用可能な検索エンジンでは inurl 構文を，inurl 構文が利用不可能な検索エンジンではキーワードを利用している．TLD の種類は ICANN[7]によって定められており，全世界で一定数しか存在しないことが保証されている．したがって,TLD すべてを制約集合 Cとした場合 Cは式(1)式を満たす．また，新たに URL u を選び，「u」と「-」を用いた「-u」を制約に加えることにより得られる集合は次の(2)式を満たす，

$$S(query) = S(query \wedge u) \cup S(query \wedge \neg u) \ \ldots (2)$$

よって，$C = \{u, \neg u\}$としたときにも，(1)式を満たしている．したがって，提案手法は論理的に完全性のある検索結果を取得することが可能である．ただし，実際の検索エンジンではクエリ中に指定することができる条件数には限りがある．そのため，入力クエリに加えた条件数が，検索エンジンが 1 クエリで受け付けるクエリの条件数を超えた場合，完全性を満たさない．

図 1 に本収集手法の疑似コードを示す．図 1 の疑似コードにおいて，*search*(*query*)はクエリ *query* にて検索を行い，検索結果 *result* を返す関数，|*result*| は検索結果の取得件数，TLD.Count は TLD の総数，TLD[i] は i 番目 （0≦i <TLD.Count）の TLD，getNextUseQuery(*result*)は検索結果セット *result* を入力とし，*result* を解析した結果，次に「-」を使用し検索に用いるキーワードを選択する関数である．この擬似コードにおける入力は，ユーザが入力したキーワードである *input_query*，出力は，search 関数にて取得した検索結果 *result* をマージした R となる．

3.1.2 検索エンジンの特別構文と性能に関する仮定

提案手法では，検索エンジンにおいて利用可能な特別構文を用いる．特別構文とは，クエリ内で使用することにより，Web ページ内の特定の部分の検索や特別な情報の検索を行うことが可能なコマンドである．提案手法では site 構文と inurl 構文を用いる．site 構文とは，「site:」の文字列の直後にドメイン名を続けることによって，そのドメイン内に存在する Web ページのみを検索対象とすることができる構文である．また，inurl 構文とは，「inurl:」の直後に任意の文字列を続けることによって，書き込んだ文字列を URL 内に含む Web ページのみを検索対象とすることができる構文である．この 2 つの特別構文は，Yahoo! JAPAN と Google で利用可能である．MSN では site 構文を利用できるが，inurl 構文を利用することができないため，特別構文を用いずに全文検索を用いて本

```
//インプット　：検索文字列「input_query」
//アウトプット：検索結果集合「R」

// 入力クエリで検索
result ← search(input_query);
R ← result;
// 今回の検索で取得した結果数|result|が取得限界数に達した場合
if(|result| > 取得限界数){
        //  TLDの指定により検索問題を細分化
        for(i=0; i < TLD.Count; i++){
                query = input_query;
                deepSearch(query + " site:" + TLD[i]);
        }
}

deepSearch(query){
        //指定されたクエリで検索
        result ← search(query);
        R ← result ∪ R;

        //今回の検索で取得した結果数|result|が取得限界数に達した場合
        //検索結果から次に使用する検索条件を選定し
        //その条件に合致するか，合致しないかで検索
        if (|result| > 取得限界数){
                //検索結果から次に使用する検索条件を決定
                useCondition = getNextUseQuery(result);
                //検索条件に合致する場合
                deepSearch(query + useCondition);
                //検索条件に合致しない場合
                deepSearch(query + " -" +useCondition);
        }
}
```

図 1．収集手法の疑似コード
Fig. 1. Pseudo-Code of Gathering Method

文中に出現する任意の文字列の指定を行う．

site 構文を利用することによって，対象とする Web ページの TLD を指定することができる．しかし，Web ページの中にはホスト名として IP アドレスが直接指定されている場合もある．そのような場合，検索エンジンは IP アドレスの下位 8bit 部分，10 進数で 0 から 255 までを当該サイトの TLD とみなす．このことは，検索エンジンにおいて「site:.1」など，TLD として IP アドレスを指定する検索を行うことによって確認できる． inurl 構文では，URL 中に含まれる部分文字列を指定できる．ただし，指定できる部分文字列には制限があり，URL 中「.」「/」「:」「?」などで区切られた文字列を単位として指定できる．

本論文では，対象とする検索エンジンに対して，次のような仮定を行う．(1)site 構文，マイナスオペレータを検索に使用できる，(2)入力するクエリの条件数が，一度の検索で入力可能なクエリの条件数を超えない限り，検索エンジンは与えられたクエリの条件に対して，常に一致する検索結果を返す，(3)クエリを入力して取得した検索結果件数は，その検索結果数が取得限界数に達しない限り完全性のある検索結果が得られる．この(1)～(3)の仮定を満たす検索エンジンを対象とした場合，提案する収集手法により網羅的に収集できることになる．しかし，実際の検索エンジンにおいて(2)の条件は，一般的に満たされない．その原因としては次の(a)～(d)の理由が考えられる．(a)検索エンジンの DB は時間とともに変化している，(b)検索エンジンはクエリフォームの背後に複数の DB を持っており，どの DB から検索結果が返されるかわからない，(c)検索結果はクエリで指定する語順によっても変化する，(d)検索エンジンはクエリで指定した検索条件を正確に処理できない場合がある．つまり，実際の検索エンジンにおいては，(a)～(d)の理由から，提案手法を用いても完全性のある検索結果を取得できない可能性がある．この点については 5 節に述

べる通り今後の課題とする．

3.1.3 検索に使用する部分文字列の作成

提案手法では，TLD の指定の他に，inurl 構文が利用可能な検索エンジンにおいて URL の部分文字列を指定することによって検索問題の細分化を行う．また inurl 構文が利用できない検索エンジンでは，任意のキーワードを検索に利用することによって検索問題の細分化を行う．

まず，URL からの本手法で使用している inurl 構文で指定可能な部分文字列作成手順について述べる．本手法において，URL の部分文字列は次のような手順で作成される．①URL から http://, https://を取り除く．②記号で URL を分割する．②において表れている記号とは，RFC1738[15]で規定されている URL 内で使用できる記号である．具体的には $ - _ . + ! * ' ( ) , ; / ? : @ = & の 18 個の記号を指す．それらの記号のうちいずれかが現れると，その記号を取り除き URL を分断する．

続いて，inurl 構文を利用できない検索エンジンにおいて，検索に利用するキーワードの作成手順について述べる．inurl 構文を利用できない検索エンジンにおいては，タイトルとスニペットを Mecab[9]を利用して形態素解析を行い，その結果名詞と判断された形態素を文字列として使用する．

3.1.4 部分文字列辞書からのクエリ選択手法

検索結果を分割するための部分文字列の選択にあたっては，直前に検索した結果から部分文字列辞書を作成し，その辞書から選択する．本手法において，次に使用するクエリとして最も理想的なクエリは，現在の検索結果数を２分割することができるクエリである．網羅的に検索結果を取得するだけならば，どのような部分文字列選択を行ったとしても検索結果を網羅的に取得することができる．しかし，実際には検索エンジンにおいて一度の検索で入力できるクエリの条件数には制限があるため，入力するクエリの条件数を可能な限り少なくする必要がある．よって，本手法において次の検索に使用する理想的なクエリは現在の検索結果数をちょうど２分割することができるクエリとなるが，２分割することのできるクエリ選択手法として本論文では，以下の 3 手法を考える．

１）ランダム選択
部分文字列辞書の中から，これまでにクエリとして使用していない部分文字列をランダムに選択する．

２）最頻選択
部分文字列辞書の中から，これまでにクエリとして使用しておらず，辞書作成時に使用した検索結果の中で最も出現頻度が高い部分文字列を選択する．

３）中央値選択
部分文字列辞書の中から，これまでにクエリとして使用しておらず，辞書作成時に使用した検索結果中での出現頻度が取得した検索結果数の２分の１の値に最も近い部分文字列を選択する．

3.1.5 検索エンジンの API 利用上の制約

提案手法は検索エンジンの検索 API を通して，適用することを想定している．日本において広く利用されている Yahoo! JAPAN，Google，MSN の３社はいずれも検索 Web サービスを用いた API を提供している．しかし，それらの検索 API の利用において，検索エンジンは表 1 に示す各種制約を設けている．例えば，Yahoo! JAPAN の検索 API では５万回／日の制約がある．

提案手法において検索問題が理想的に分割できた場合，

表 1. 検索エンジンの API 利用における制約
Table 1. Limitation of Accessing Search Engines' API

	Yahoo! JAPAN	Google	MSN
１日当たりの検索制限回数	50,000	1,000	25,000
１検索あたりの最大検索結果数	50	10	50
アクセス制限方法	IP アドレス	ライセンスキー	IP アドレス

1,000 件の検索結果を取得するためには 40 回の検索が必要となる．つまり，Yahoo! JAPAN において本手法が理想的に適用されたと仮定すると，１日で 1,250,000 件の検索結果が取得できる．また MSN においては１日に２万５千回の検索が可能であるので，理想的には１日に 625,000 件の検索結果が取得できる．これら検索結果件数は理想的な状況下の取得件数であるため，実際にはこれらの数字よりも少ない数になる．

3.2 Unlisted Search Result の解析

3.1 において提案した手法を用いて取得した ULSR に対して解析を行う．従来，検索エンジンの検索結果をクラスタリングする手法がいくつか提案されている[4][5][13]．しかし，これらの手法は，少ない情報をリアルタイムにクラスタリングすることに着眼点を置いている．そのため，高々1,000 件ほどの検索結果しか対象としておらず，今回のように ULSR のような大規模(数万～数十万件)の検索結果集合に対応できない．また，本論文の解析では，ULSR と，LSR の比較を行い，その違いや特徴を抽出することが目的となる．そこで，本論文では，スニペットクラスタリングは行わず，単純に検索結果の Web ページに現れたトップレベルドメイン(TLD)の分布の比較を行うこととした．

4. 評価実験

4.1 実験概要

本項では，3 節にて提案した ULSR 収集手法を用いた実験を行う．まず， 3.1.4 にて提案した３手法のうち，どの手法が効率よく Unlisted Search Result を収集できるかについて実験を行う．続いて，その実験結果において最も効率よく取得できた手法を用いて，検索エンジン間の違いを知るために日本で広く利用されている検索エンジンである Yahoo! JAPAN，Google，MSN において ULSR の取得を試みる．

4.2.1 検索パラメータ

本実験において設定した，検索エンジンにおける検索パラメータについて説明する．本実験は，対象とする検索エンジン各社が提供している検索 Web サービス API を用いて実験を行った．Yahoo! JAPAN が提供している API を用いて検索を行った際に設定した検索パラメータを表 2 に示す．また，Google，MSN における検索パラメータの設定も表 2 の設定に準ずるパラメータ設定を行った．

また本手法を適用するにあたって，クエリの条件としてドメインの指定と，Yahoo! JAPAN，Google において URL に含む部分文字列の指定を行う必要がある．ドメインの指定，URL の部分文字列の指定には，それぞれ 3.1.2 で述べた site 構文，inurl 構文を用いた．また，入力クエリは「”」（ダブルクォーテーション）で囲むことによってフレーズ検索を使用し入力クエリに厳密に一致するものだけを検索の対象とした．

表 2. Yahoo! Japan における本実験の検索パラメータ
Table 2. Parameter Set for Yahoo! JAPAN Web Search

パラメータ名	値	効果
type	all	全てのクエリ文字を含む検索結果を返す
format	any	全てのファイルの種類を対象とする
adult_ok	1	アダルトコンテンツを含める
similar_ok	1	同じコンテンツを別の検索結果とする
language	ja	日本語で書かれた結果を取得する
results	50	一度の検索で 50 件の検索結果を取得する

表 3. 提案手法間における収集結果の比較
Table 3. Results Comparison of Gathering Methods

	平均取得率		
概算 Hit 件数	ランダム	最頻	中央
1001-10000	0.502	0.659	0.659
10001-100000	0.059	0.155	0.231
100001-1000000	0.022	0.056	0.058
1000000-10000000	0.003	0.013	0.016
全平均	0.147	0.221	0.241

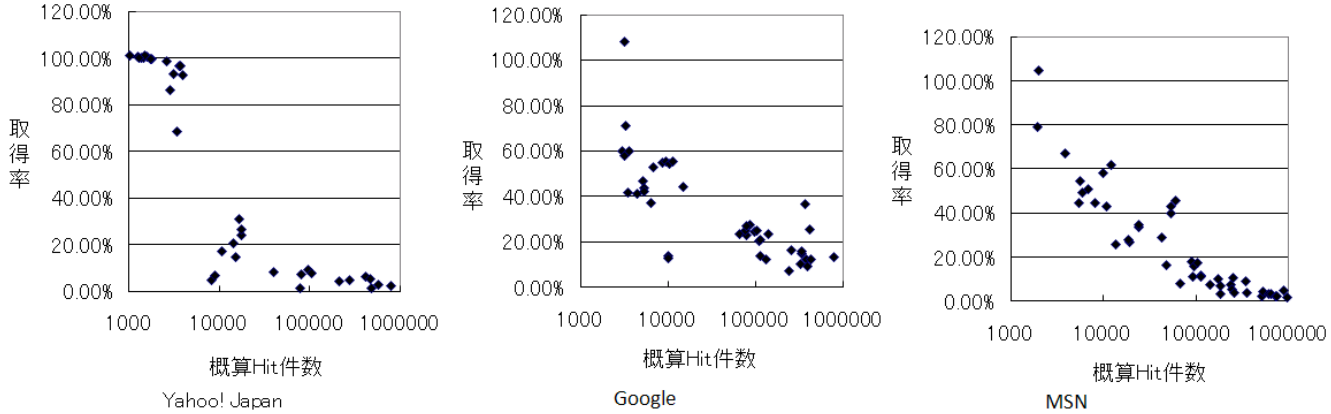

図 2. Yahoo! JAPAN, Google, MSN における概算 Hit 件数と取得率の関係
Fig. 2. Relation between gathered rate and estimated hit count by using Yahoo! JAPAN, Google, MSN

4.2.2 初期入力のクエリ選択

本項では，各実験において使用する初期入力クエリの選択方法について述べる．

3.1.4 で提案したクエリ選択手法の比較実験においては，結果がクエリの知名度に影響されないよう，次のように使用するクエリの選択を行った．なお，概算検索結果数は Yahoo! JAPAN のものを採用した．(1) Wikipedia[18]の日本語項目の中から，ランダムに 1,000 項目を選択，(2) 選択した項目に対して，Yahoo! JAPAN から時間間隔をおいてそれぞれ 5 回ずつ概算 Hit 件数を算出，(3) それら項目の中から，概算 Hit 件数の平均が 1,001 ~ 10,000, 10,001 ~ 100,000, 100,001 ~ 1,000,000, 1,000,001 ~ 10,000,000 となるグループを作成し，それぞれのグループの中から 5 つずつ，合計 20 個のクエリをランダムに選択した．

また，USRL の取得実験においては，次のように使用するクエリの選択を行った．(1)Wikipedia の日本語項目の中からランダムに 20,000 項目を選択，(2)選択した項目を用いて，3 つの検索エンジンにおいて検索を行い，1,000,000 件以内に検索結果が収まる項目のリストを検索エンジンごとに作成 (3)それぞれの検索エンジンにおいて使用するクエリを，(2)にて作成した各検索エンジンが持つリストからランダムに 60 項目選択した．

4.2.3 クエリ選択手法の比較と評価

3.1.4 で提案した 3 種類のクエリ選択手法について評価を行う．3.1.4 では理想的なクエリ選択手法として，検索結果を 2 分割できる手法としているが，本評価では，最終的に得られた検索結果数により評価を行う．これは，うまく 2 分割できることにより，より多くの検索結果を得ることができると考えるからである．まず，取得率 r を次のように定義する．

$$r = \frac{\text{実際の結果取得件数}}{\text{概算総 Hit 件数}}$$

ここで，「実際の結果取得件数」は，あるクエリ選択手法を用いた際に実際に取得できたユニークな URL 数である．また「概算総 Hit 件数」はユーザが入力したクエリのみを用いて検索したときの概算 Hit 件数とする．

3 種類のクエリ選択手法各々について，取得率を求めた結果を表 3 に示す．表 3 中の平均取得率は，概算 Hit 件数に示した範囲に収まるクエリにおける取得率の平均を表わす．実験の結果，取得率 r は $r_{ランダム} = 0.147$，$r_{最大値} = 0.221$，$r_{中央値} = 0.241$ となった．この結果より，3 種類の手法の内，最も網羅的に収集できるのは，クエリ選択に中央値を用いる収集手法となり，その平均取得率は 24.1%となった．しかし，3 つの手法何れも概算取得件数が大きくなるにつれて，本手法を用いて実際に取得できた Web ページ数は大幅に減少しており，再現率もそれに伴って低くなっている．この理由については，5 節で議論する．

4.2.4 中央値検索を用いた ULSR の収集

4.2.3 で最も取得率が高い中央値選択手法により部分文字列を選択し ULSR の収集を行った． 対象とした検索エンジンは，Yahoo! JAPAN，Google，MSN である．4.2.2.にて準備したクエリを用いて，ULSR を収集した結果を図 2 に示す．図 2 に示すように，Yahoo! JAPAN では概算 Hit 件数が少ない場合（具体的には 4000 より少ない場合），高い取得率が得られている．しかし，概算 Hit 件数が高くなると取得率の急速な低下が発生している．Google では，他の 2 つの検索エンジンと比べ，概算 Hit 件数が大きい場合でも高い取得率を維持していることが読み取れる．MSN では，概算 Hit 件数が 20 万件付近まで徐々に取得率の低下が発生している．また Google や MSN では，取得率が 100%を超えるクエリが存在する．これは，概算総 Hit 件数よりも実際に取得した検索結果数が多いことを示す．つまり，Google や MSN では，概算総 Hit 件数より 5～10%総検索結果数が増える場合があることを示す．

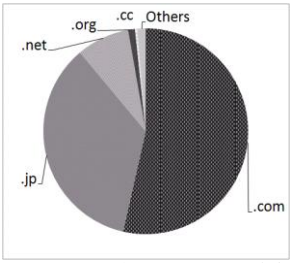

図 3. 日本語で書かれた Web 全体の TLD 分布（[19]のデータを元に作成）
Fig.3 TLD Distribution of All Japanese Pages

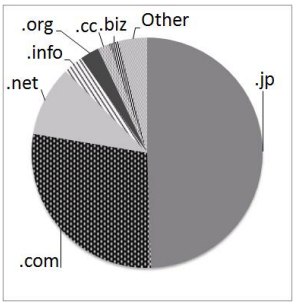

図 4. ULSR の TLD 分布
Fig. 4. ULSR TLD Distribution

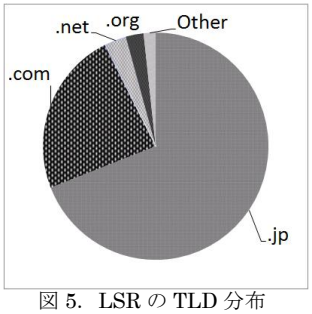

図 5. LSR の TLD 分布
Fig. 5. LSR TLD Distribution

4.3 Unlisted Search Result の解析

本実験にて取得した ULSR と LSR の TLD の比較を行う. 4.2 において取得した ULSR に対し，クエリ毎の TLD の分布を計算し，その分布の平均をグラフ化したしたものを図 4 に示す. また, 逆に LSR に対し, クエリ毎の TLD 分布を計算し，その分布の平均をグラフ化したものを図 5 に示す．参考として童らによる研究[19]を元に，日本語で書かれた Web ページの TLD 分布をグラフ化したものを図 3 に示す．図 3，図 4，図 5 を比較すると, ULSR の jp ドメインの分布と日本語で書かれた Web 全体の jp ドメインの分布を ULSR の jp ドメインの分布と比較すると似ていることが分かる．また，ULSR には，通常の検索結果や日本語で書かれた Web 全体において現れにくい TLD が多く現れていることが分かる．

5. まとめと今後の課題

本稿では，検索エンジンの取得限界数によって取得できない検索結果（UnListed Search Result, ULSR）を収集する手法を提案した．提案手法を Yahoo! JAPAN，Google，MSN に適用し，評価を行った．ULSR の解析を行った結果，ULSR には様々な TLD が出現することがわかった. つまり検索エンジンの内部には様々な TLD を持つ Web ページがインデックス化されているのに対し, LSR においては jp ドメインと com ドメインが 9 割以上を占めていることが確認された.

今回の収集手法では ULSR の取得を行うことができたものの，取得できた検索結果数は検索エンジンが返す概算総 Hit 数が大きくなると取得率が低下する．これは，3.1.2 で仮定した(2)が満たされていないためであると考えられる．しかし，検索エンジンの検索アリルゴリズムの多くは公開されておらず，現時点では議論することができない．今後，取得率を高める手法について検討を行っていく予定である．また，今回の解析では ULSR と通常の検索結果の間に異なる傾向があることは確認できたが，ULSR に有益な情報が含まれているのかどうかについての解析は行っていない．今後，ULSR に有益な情報が含まれているか否かについての調査を行うことと，有益な情報が含まれているのならばその情報をどのような手法で抽出するのかが今後の研究課題となる．

【文献】

[1] Amazon.com: http://www.amazon.com/
[2] A. Ntoulas, P. Zerfos and J. Cho: “Downloading Textual Hidden Web Content through Keyword Queries,” In Proc. of JCDL2005, pp.100 - 109, Denver, Colorado(2005.7).
[3] B. He, M. Patel, Z. Zhang and K.C.C. Chang: “Accessing the Deep Web: A Survey,” CACM, Vol.50, pp.94- 101(2007.5).
[4] F. Geraci, M. Pellegrini, P. Pisati and F. Sebastiani: “A Scalable Algorithm for High-Quality Clustering of Web Snippets,” In Proc. of the 2006 ACM Symp. on Applied computing, pp.1058-1062, Dijion, France(2006.4)
[5] G. Mecca, S. Raunich and A. Rappalardo: “A New Algorithm for Clustering Search Results”, Data & Knowledge Engineering, Vol.63, Issue.3, pp.504 - 522(2007.12)
[6] Google: http://www.google.co.jp/
[7] ICANN: http://www.icann.org/tlds/
[8] M. Álvarez, J. Raposo, A. Pan, F. Cacheda, F. Bellas and B. Carneiro: “Crawling the Content Hidden Behind Web Forms”, In Proc. of Int. Conf. on Computational Science and Its Applications, Vol.4706, pp.322 -333(2007)
[9] Mecab: http://mecab.sourceforge.net/
[10] M.K. Bergman: “The Deep Web: Surfacing Hidden vValue.”, J. of Electronic Publishing, Vol.7, No.1(2001.8)
[11] MSN: http://www.msn.com/
[12] Open Directory Project: http://www.dmoz.org/
[13] P. Ferragina and A. Gulli: “A Personalized Search Engine based on Web-snippet Hierarchical Clustering”, In Special interest tracks and poster Proc. of Int. Conf. on the World Wide Web, pp.801- 810(2005.5)
[14] P. Wu, J.R. Wen, H. Liu and W.Y. Ma: “Query Selection Techniques for Efficient Crawling of Structured Web Sources”, In Proc. of the 22nd Int. Conf. on Data Engineering, p.47(2006.4)
[15] RFC 1738: http://tools.ietf.org/html/rfc1738
[16] Yahoo!JAPAN: http://www.yahoo.co.jp/
[17] W.S. Li, J. Shim, K.S. Candan and Y. HARA: “WebDB: A Web Query System and its Modeling, Language, and Implementation”, In Proc. of IEEE Advances in Digital Libraries, pp.216-227(1998)
[18] Wikipedia : http://ja.wikipedia.org/wiki/
[19] 童芳，平手勇宇，山名早人：“全世界の Web サイトの言語分布と日本語を含む Web サイトのリンク・地理的位置の解析”，DEWS2008，A2-3(2008.3)

舟橋 卓也 Takuya FUNAHASHI
早大・基幹理工学研究科修士課程在学中．DBSJ 学生会員.
上田 高徳 Takanori UEDA
早大・基幹理工学研究科修士課程在学中．ACM, IEEE，情報処理学会，電子情報通信学会，DBSJ 各学生会員.
平手 勇宇 Yu HIRATE
2005 早大・理工学研究科修士課程修了．2008 早大・理工学研究科博士課程修了．博士（工学）．2006 年より同大学 MNC 助手．ACM，IEEE，情報処理学会，DBSJ 各会員.
山名 早人 Hayato YAMANA
1993 早大・理工学研究科博士課程了. 博士（工学）. 1993-2000 電総研．2000 早大・理工助教授。2005 同大理工学術院教授，現在に至る．IEEE，ACM，IEICE，IPSJ，DBSJ 各会員.