Vol.2014-HCI-157 No.26
Vol.2014-GN-91 No.26
Vol.2014-EC-31 No.26
2014/3/15

視覚的コンテキストによる検索結果提示とナビゲーションの可能性

丸山 一貴[1,a)]　井桁 正人[2]　寺田 実[3]

概要：Web 検索エンジンで調べ物を行う場合、クエリを入力し、検索結果ページ (SERP) に列挙されたページ (候補ページ) を順次閲覧することになる。しかし実際に候補ページを開いてみると、クエリは広告欄やインデックスの羅列に含まれるだけで、その候補ページでは言及されていないことも多い。こうしたケースは、該当する文字列がページ内でどのようにレンダリングされているかをユーザが見ればすぐに判別できることが多い。我々はこのような、レンダリング後の Web ページにおけるクエリの表示状態を視覚的コンテキスト (VP) と呼び、これを用いて SERP を改善することを提案してきた。本論文では、新たな実装方法であるヘッドレスブラウザを用いた VP の生成について述べると共に、VP のみで構成される SERP と、VP を単位とした Web ページナビゲーションについて論じる。

1. はじめに

現在は様々な情報が Web ページを介して提供され、一般には Web ページが一次情報源となっている。従って、情報検索もまた Web ページを対象として行うこととなる。Web の情報検索を行う際は、Google 等の検索エンジンを利用して、見つけたいと考えているページ（以下、目的ページという）が含んでいそうな単語（以下、クエリという）を入力する。検索エンジンが提示する検索結果ページ（以下、SERP という）に並んでいる、目的ページの候補（以下、候補ページという）を順にクリックし、所望の内容を含むページが見つかればタスクは完了となる。現在では広く一般的になったこのタスクには、以下のような問題点がある。

- 見つけたいと考えている目的ページは未知の場合が多く、SERP を見ただけでは判断できず、候補ページを 1 つずつ確認する必要がある。
- 候補ページの内容が、検索エンジンがキャッシュしたときとは内容が異なっていることもあり、情報検索タスクの実行時には既にクエリを含んでいないことがある。
- 目的ページが既知の場合で、SERP からそれを特定できたとしても、そのページを開いただけではクエリのある場所にすぐには移動できない。

我々はこれまで、第 3 点の支援として ScoutView[1] を、第 1、第 2 点の支援として PatchworkVision[2] を提案してきた。これらの提案では、候補ページ中に登場するクエリを実際にブラウザでレンダリングし、クエリ周辺をクリッピングした画像を利用する。この画像は、候補ページでクエリがどのように扱われているかを視覚的に示す情報、すなわち視覚的なコンテキストを持っていると考え、我々は Visual Patch（以下、VP という）と呼んでいる。一般的な検索エンジンにおける SERP では、スニペットと呼ばれる文字ベースの情報により、候補ページ中のクエリに関する情報が断片的に示されるが、VP はこれを置き換えるものであると考えている。

本論文では、VP の生成方法を従来方式とは変更し、ヘッドレスブラウザを使用した方法について紹介する。また、VP を用いた SERP の提示方法について、従来の候補ページ単位に提示する方式から、クエリが候補ページ内で与えられている DOM エレメントを単位としてグループ化する方法について述べる。これらを踏まえて、VP を新たな単位として Web のナビゲーションを行う可能性について論じる。第 2 章では、従来方式の VP 生成と SERP について、第 3 章ではヘッドレスブラウザを用いた VP の生成方法について述べる。第 4 章では VP の DOM エレメントによる分類方法について説明し、第 5 章では分類例と考察、VP に基づく Web ナビゲーションについて述べる。第 6 章

[1] 明星大学
2-1-1, Hodokubo, Hino, Tokyo 191–8506, Japan
[2] 楽天株式会社
4-12-3, Higash-Shinagawa, Shinagawa, Tokyo 140–0002, Japan
[3] 電気通信大学
1-5-1, Chofugaoka, Chofu, Tokyo 182–8585, Japan
[a)] kazutaka@acm.org

Vol.2014-HCI-157 No.26
Vol.2014-GN-91 No.26
Vol.2014-EC-31 No.26
2014/3/15

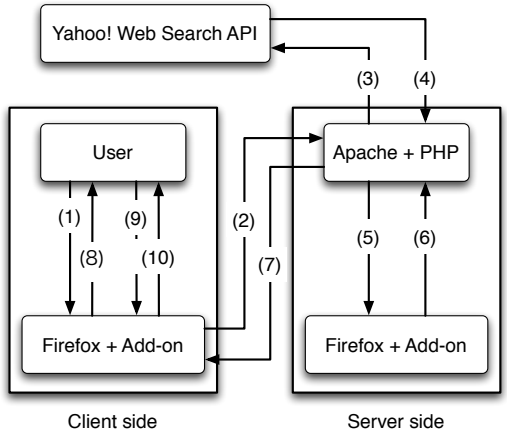

図 1 従来方式のシステム構成図
Fig. 1 Components of our prior system.

はまとめと今後の課題である。

2. 従来方式

従来方式は**図 1** のような、サーバ・クライアント方式により実装されている。サーバ及びクライアントの双方には、我々の開発したアドオンをインストールした Firefox が動作している。これらを仲介し、Yahoo!検索 API と連携して Web 検索を行うための PHP コードを組み合わせて実装してきた。図中の矢印はシステムの処理手順を示しており、以下の通りである。

(1) ユーザが検索クエリと VP クエリを入力
(2) クライアント側ブラウザがサーバサイドの PHP コードに転送
(3) PHP コードが Yahoo!検索 API に検索クエリを送信
(4) Web 検索結果として URL 群を受信
(5) 各 URL をサーバ側ブラウザで表示し、VP クエリを用いてページ内検索を実行（クエリ部分がハイライトされる）して、クエリ周辺をクリッピングして VP を保存
(6) 生成した VP 群を PHP コードに通知
(7) PHP コードが SERP を生成して、クライアント側ブラウザに送信（**図 2**）
(8) クライアント側ブラウザが SERP をレンダリングし、ユーザに提示
(9) ユーザが SERP 内の VP を 1 つクリック
(10) クライアント側ブラウザが該当ページをロード、レンダリングした上で、VP 位置まで自動スクロール

図 2 では、ノート PC のサイズについて調べるというタスクを行っている。ページを探すための検索クエリには PC のモデル名と “寸法” を、ページ内検索のための VP クエリには “寸法” のみを設定することで、図のような結果が出力される。図中では合計 6 個の VP が表示されているが、破線で囲んだ左上の VP と、右下の VP が表の一部であると一目で分かり、情報源として確度の高い仕様表ではないかと推定することができる。

この従来方式による提示は、学生 9 名による評価実験により概ね有用であるとの評価を得たが、一方で以下のような問題点もあった。

- 検索 API サービスの終了に伴う、システム運用の困難
- 処理時間が数十秒程度と長く、多数の VP を生成して提示することが困難
- 仮に多数の VP を提示できても、認知的負荷が高くユーザの判別が困難

本論文では、第 1 点を解決するためにサーバサイドの構成を変更し、ヘッドレスブラウザである PhantomJS[3] 及び CasperJS[4] を使用して実装し直した。これに伴い、複数の VP クエリに対応することも可能となっている。また、第 3 点を解消するために、VP が含むクエリがページ内でどのような DOM エレメントとなっているかを利用して分類する方法を示す。

なお、第 2 点は SERP に含まれる多数の候補ページをダウンロード、レンダリングすることによる遅延と、画像である VP の保存時間に起因しており、本論文では改善の対象としない [*1]。

3. VP の生成

第 2 章の図 1 で述べた、従来方式の手順における第 3 ステップから第 5 ステップまでを、ヘッドレスブラウザを用いた実装に変更する。

3.1 CasperJS

PhantomJS は GUI を持たない WebKit レンダリングエンジンであり、JavaScript によって動作を記述、制御することができる。DOM や CSS を扱ったり、ロードしたページに JQuery のような JavaScript ライブラリをインジェクトすることも可能である。従って、従来方式においてブラウザアドオンで行っていたような、ページ内検索やハイライトも実現可能である。また、一般の検索エンジンを通常のブラウザと同様にアクセスすることができるため、検索 API サービスから独立できる。

本論文では、PhantomJS を使用してより高機能なメソッド等を提供する CasperJS を使うこととした。CasperJS では標準の機能として以下が提供されており、これらを利用して従来方式の手順を置き換えた。

- 指定した URL にアクセスし、コンテンツをメモリ内でレンダリングする。
- ページ内のリンクやフォームをクリックし、ページ遷移する。
- ページ全体、もしくは、ページ内の指定要素をキャプ

[*1] レイテンシの大部分はネットワーク転送であると言えるため、CPU コア数を超えて並列化を行うことで、通常の Web ページ閲覧にかかる時間に近づけることが可能と考えられる。

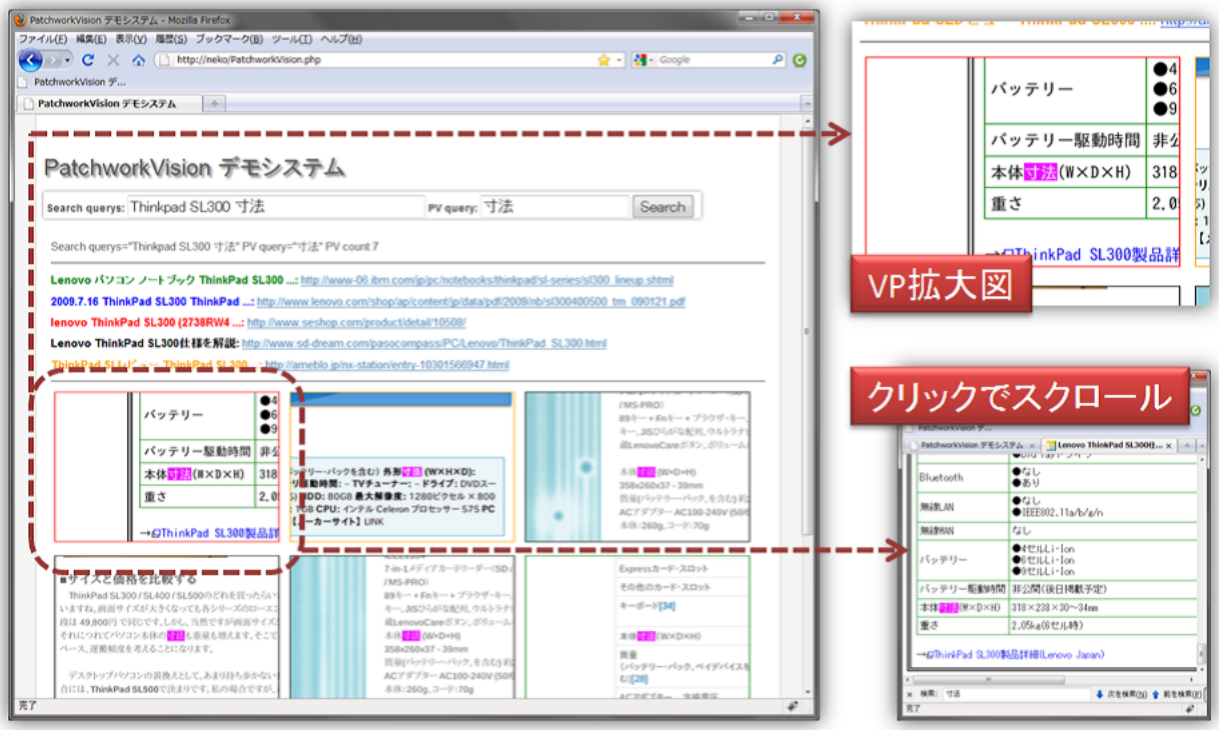

図 2 従来方式の SERP
Fig. 2 Search engine result page of our prior system.

チャする。

3.2 Google 検索と SERP のナビゲーション

図 3 は、Google のトップページで特定のクエリにより検索し、表示されたトップ 10 の SERP と、次の 10 件の SERP を取得する方法を示している。コードは JavaScript で書かれており、CasperJS（つまり、WebKit レンダリングエンジン）を制御するためのコードが書かれている。通常の Web ページを作る際はページコンテンツの一部として JavaScript を使用するが、ページ内で動作する（WebKit が解釈する）JavaScript コードは `this.evaluate()` により動作させることができる。

3.3 クエリのハイライトとキャプチャ

図 4 は、JQuery [*2] 及び JQuery 用ハイライトプラグイン [*3] を用いることで、ページ内のクエリをハイライトした上で、その部分をキャプチャする例を示している。実際には、クエリの座標を取得し、その周辺にマージンを設定して、クエリを含むより広い範囲の矩形を指定してキャプチャする。

VP 生成時には、第 4 章で述べる分類のために、ハイライトしたクエリがどのような DOM エレメントとしてページ内で記述されているかを記録する。例えば**図 5** の場合は

```
// Google トップページにアクセスし、Nexus 7 で検索
casper.start('http://www.google.co.jp/',
  function() {
    this.fill('form[action="/search"]',
      { q: 'Nexus 7' }, true);
});

// 検索結果ページから、次の 10 件にアクセス
casper.then(function(){
  var next = 'http://www.google.co.jp'
    + this.evaluate(function(){
      var nextLinks =
        document.querySelectorAll('a.fl');
      var hrefs =
        Array.prototype.map
          .call(nextLinks, function(e){
            return e.getAttribute('href');
        });
      for(var i = 0; i < hrefs.length; i++){
        if(hrefs[i].match(/start=10/)){
          return hrefs[i];
        }
      }
      return "";
  });
  this.open(next);
});
```

図 3 CasperJS による Google 検索の例
Fig. 3 Usage example of CasperJS for searching by Google.

2 つのクエリを含んでおり、それぞれの親要素を順に辿って、以下のような情報を記録する。

*2 http://jquery.com/
*3 http://johannburkard.de/blog/programming/javascript/highlight-javascript-text-higlighting-jquery-plugin.html

Vol.2014-HCI-157 No.26
Vol.2014-GN-91 No.26
Vol.2014-EC-31 No.26
2014/3/15

```
// 各ページにライブラリをインジェクト
var casper = require('casper').create({
  clientScripts: ["./jquery-2.0.3.js",
                  "./jquery.highlight-4.js"]
});

// url にアクセスして、Nexus 7 をハイライト
casper.open(url).then(function() {
  this.evaluate(function(query){
    $(':root').removeHighlight().highlight(query);
    $('.highlight').css("background-color","yellow");
  }, {
    query: 'Nexus 7'
  });
});

// ハイライト箇所の 1 つ目をキャプチャ
casper.then(function(){
  this.captureSelector('.highlight');
});
```

図 4 CasperJS による VP 生成の例
Fig. 4 Usage example of CasperJS for clipping query.

図 5 2 つのクエリを含む VP の例
Fig. 5 VP example including two queries.

- H1 DIV DIV DIV DIV BODY HTML
- H2 DIV DIV DIV BODY HTML

4. VP の分類

第 3 章で述べた方法により、与えられたクエリにより Google 検索を行い、SERP で得られた各候補ページを順次レンダリングし、ページ内に含まれるクエリをハイライトすることが可能となった。ここではこのようにして得られた VP を、DOM エレメントの階層情報を用いて分類することにより、構造上、同種の視覚的コンテクストを持つ VP をグループ化する。結果として、第 2 章で述べた、多数の VP を提示することによる認知的負荷の軽減を目指す。

本論文では試みとして、クエリの構造を「直近 3 世代分の親要素」とすることとした。但し、DIV 及び SPAN タグはそれ自身が構造を示すとは考えにくいので、除外する。従って、図 5 の例では、1 つの目のクエリは H1 BODY HTML がその構造であり、2 つの目のクエリは H2 BODY HTML がその構造である。図 5 の VP は 2 つのクエリを持つことから、自身の特徴を表すデータとしてこれら 2 つの構造を持つと言える。このような、VP が含むクエリの構造データを、その VP の VP タグと呼ぶことにする。

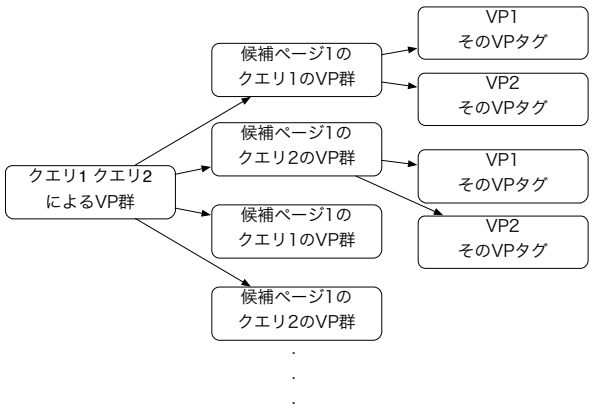

図 6 1 つの Web 検索に対応する VP 群
Fig. 6 VPs according to a certain web search with two queries.

ある Web 検索を行う場合、複数のクエリを用いることが一般的である。本論文で提案するシステムに複数のクエリを用いると、結果として**図 6** のようなデータ構造ができあがる。

5. 分類例と考察

本章では、第 4 章で述べた VP タグを用いた分類について、具体例を示しながら有効性を検討する。

VP タグの定義や分類方法と同様に、多数の VP をどのように提示するかはいくつもの選択肢があり得る。ここでは、以下のような仮定・方針に基づいて VP を整列した。

- Google が提示する候補ページは、上位のものから順に信頼度が高い。
- 信頼度の高い候補ページに含まれる VP と、同じ VP タグを持つ VP は、他の候補ページにあるものもグループ化して表示する。
- VP を並べる場合は、より上位の候補ページにあるものを先に提示する。
- 同一の候補ページにある VP を並べる場合は、より後ろのクエリに基づく VP を先に提示する [*4]。

5.1 分類例 1

“安倍総理 支持率” という 2 つのクエリで検索し、前述の方法で VP 群を整列させた。実装したシステム上は、SERP を次々にクリックしながら多数の候補ページを取得することができるが、実験時点では 12 番目の候補ページに接続できずシステムが正常終了しなかったため、上位 10 件の候補ページを対象とした。生成された VP 群の情報を**表 1** に示す。

この例における最上位の候補ページは、スレッド型掲示

[*4] 複数クエリで検索する際、後方のクエリほどより具体的な内容であることが多いという、我々の経験則に基づく。

Vol.2014-HCI-157 No.26
Vol.2014-GN-91 No.26
Vol.2014-EC-31 No.26
2014/3/15

表 1 分類例 1 のクエリで生成された VP 群
Table 1 The number of VPs of example 1.

VP の総数	190
最も多く VP を持つ候補ページの VP 数	78
クエリ “安倍総理” の VP 数	14
クエリ “支持率” の VP 数	176

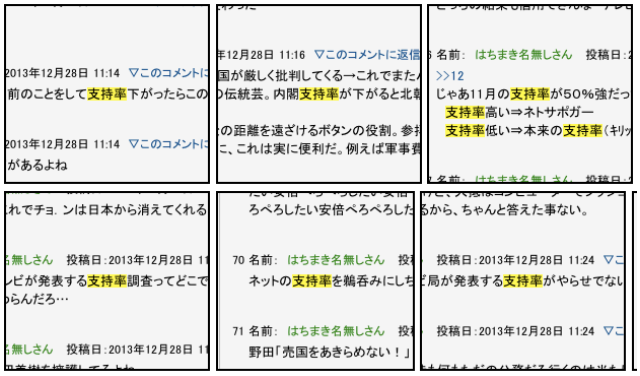

図 7 分類例 1 での最上位サイト関連 VP 群
Fig. 7 A part of VPs grouped to the top site of example 1.

図 8 分類例 1 での第 2 位サイト関連 VP 群
Fig. 8 A part of VPs grouped to the 2nd site of example 1.

板のまとめサイトであり、安倍総理の支持率に関する話題を扱ったスレッドだったために極端な偏りが現れたと考えられる。最上位の候補ページに含まれる VP にグループ化された VP 群の一部を**図 7** に示す。このグループに分類された VP は、この最上位候補ページに含まれる VP だけであった。

第 2 位の候補ページはマスコミによる公式のニュースサイトであり、支持率に関するニュース記事が多く含まれていた。VP 群の一部を**図 8** に示す。このグループに分類された VP は、合計 7 つの URL から供給されており、5 つはニュースサイト（4 種類のサイト）、2 つは異なるブログサイトであった。

“安倍総理 支持率” というクエリは、支持率の現状や推移などについて調べたいと考えられ、その場合は図 8 のグループを提示するのが妥当と考えられる。図 7 がより上位となってしまったことは改善の余地があるが、同じ URL

表 2 分類例 2 のクエリで生成された VP 群
Table 2 The number of VPs of example 2.

VP の総数	84
最も多く VP を持つ候補ページの VP 数	28
クエリ “suface 2” の VP 数	76
クエリ “寸法” の VP 数	8

図 9 分類例 2 での最上位サイト関連 VP 群（クエリ 1）
Fig. 9 A part of VPs grouped to the top site of example 2 (query 1).

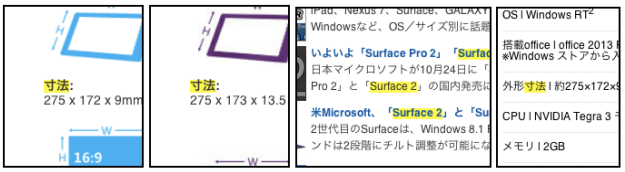

図 10 分類例 2 での最上位サイト関連 VP 群（クエリ 2）
Fig. 10 A part of VPs grouped to the top site of example 2 (query 2).

から生成された VP や、画像的に似通っている VP 群を重ねて表示するなど、ユーザインタフェースの工夫により解決できる可能性が高いと考えている。

5.2 分類例 2

“”surface 2” 寸法” という 2 つのクエリで検索し、VP 群を整列させた。分類例 1 と条件を揃える目的で、こちらも上位 10 件の候補ページを対象としている。生成された VP 群の情報を**表 2** に示す。

この例における最上位の候補ページは、Microsoft 社による Surface 2 の公式ページであった。公式ページでは “Surface 2” を含む箇所が動的な HTML として構成されていると見られ、**図 9** に示すように VP を正常に切り出せていないケースが見られた。また、このクエリを含む箇所はハイパーリンク（A タグ）になっており、関連する VP としてニュースサイトの記事の一覧などがリストされている。

一方、“寸法” を含む箇所の周辺には、通信販売サイトの仕様表が表示されており、関連の深い情報が近くに配置されていることが分かる。

Vol.2014-HCI-157 No.26
Vol.2014-GN-91 No.26
Vol.2014-EC-31 No.26
2014/3/15

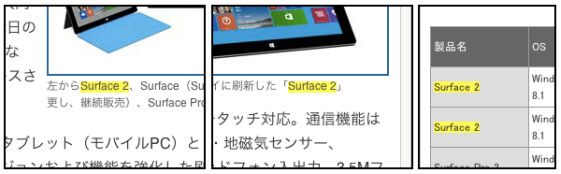

図 11 分類例 2 での最 5 位サイト関連 VP 群
Fig. 11 A part of VPs grouped to the 5th site of example 2.

第 2 位から第 4 位までの候補ページでは、VP はほとんど分類されなかった。これは、既出の VP を重複して表示しない設定としていることによっており、これらの候補ページに含まれる VP のほとんどが、最上位サイトに関連していたことを示している。SERP から候補ページを順次辿るという従来型の Web ページナビゲーションを行った場合、第 2 位から第 4 位までをも見る必要があるが、本提案方式ではこれらのページの抜粋を最上位サイトに関連づけて提示することができている。つまり、ユーザが手動でナビゲーションを行う手間を削減することに成功したと言える。

第 5 位は IT 系ニュースサイトの紹介記事であり、ここに分類された VP 群の一部を**図 11** に示す。この例では仕様表と思われるものと、インライン画像のキャプションとが近くに配置されている。いずれも HTML のテーブルを利用しているためと考えられるが、インライン画像に丁寧なキャプションが付けられていることから機器の紹介記事であることが伺える。特定の IT 機器の寸法について調べている場合は紹介記事も有用である可能性が高いと考えられ、一般的な SERP で採用されている文字ベースのスニペットよりも、関連性を直感的に把握することができると言える。

5.3 VP に基づく Web ナビゲーション

Web という仕組みが生まれて以来、ナビゲーションはページ単位で行うことを原則としてきた。# を用いたアンカーによりページ内の特定位置にナビゲーションすることは可能だが、見出しと言った単位でしか適用されていない。単一の Web ページには複数の情報が含まれており、中には広告や、他の記事へのインデクスを羅列したものもある。文字ベースで検索し、文字ベースのスニペットに頼って、ページ単位でナビゲーションを行う現在の方法には限界があることは明らかである。

この課題を解決するため、我々は VP のような「Web ページ内の特定箇所が含む情報に基づくナビゲーション単位」が必要であると考えている [5]。本論文で述べた手法により、多数の VP を生成しても分類はある程度可能であると考えられ、VP のみを並べた SERP を画像検索結果のように利用することで、有用な情報検索と新しい Web ナビゲーションに繋げることができる。

6. まとめ

本論文では、従来手法で行ってきた VP 生成手法を変更し、ヘッドレスブラウザを用いる方法で実現できることを述べた。VP 生成時にはクエリの DOM 構造を合わせて保存することにより、ページ内の構成に基づいて VP を分類することが可能になった。2 種類の具体的な検索事例について実際に VP の分類を行い、有効な場合と改善が必要な場合を考察した。

今後の課題として、第 5.1 章で述べた通り、VP の画像的な類似度による分類尺度を導入することが挙げられる。また、第 5.2 章の例に対応するため、クエリの周辺にリンクが集中している場合は単なるインデクスと考えて重要度を下げるような、ヒューリスティックスの導入も検討する必要がある。

謝辞 本研究は JSPS 科研費 23700107 の助成を受けたものです。

参考文献

[1] 井桁正人，寺田 実，丸山一貴：ScoutView: Web ページにおけるナビゲーション支援インタフェース，情報処理学会研究報告，Vol. 2009-HCI-133, No. 8 (2009).
[2] 井桁正人，丸山一貴，寺田 実：PatchworkVision:視覚的コンテキストを用いた検索結果の提示，第 18 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 (WISS2010)，pp. 159–160 (2010).
[3] PhantomJS: PhantomJS, http://phantomjs.org/.
[4] CasperJS: CasperJS, a navigation scripting and testing utility for PhantomJS and SlimerJS, http://casperjs.org/.
[5] Maruyama, K., Igeta, M. and Terada, M.: Search Engine Result Page with Visual Context and Already Rendered Snippets, *Proceedings of the 7th International Conference on Web Informational Systems and Technologies (WEBIST2011)*, pp. 340–345 (2011).