Omniget：第三者情報を提示するブラウザ内ブラウザ

周 安平
東京大学情報理工学系研究科
zhou@cl.ci.i.u-tokyo.ac.jp

田中 久美子
東京大学情報理工学系研究科
kumiko@i.u-tokyo.ac.jp

1　はじめに

インターネット上に大量の情報が流れている昨今、多様な情報源に手軽に同時アクセスしたいというニーズが増えてきた。このニーズに応じて、関連する複数情報を同時に提供するウェブサービスや、複数のページを混合させる処理を行ってユーザに提供するマッシュアップウェブサービスが増えてきている。とはいえ、これらのサービスを利用するにはそのサービスサイトにアクセスすることが前提となっているものが多く、依然 1 つのウェブページを閲覧しながら他のウェブサービスを同時に利用することは困難である。本論文ではこの問題を解決するため、Omniget を提案する。Omniget は、閲覧中のウェブドキュメントに関連する複数の第三者情報をブラウザ内に提示する機能を提供する。

Omniget は、ブラウザの Add-on として実装されるクライアントと、専用プロトコルでクライアントと通信し付加情報を提供するサーバとで構成され、クライアント・サーバ方式により実現される。クライアントはユーザの特定のマウス操作に反応し、利用するサーバの種類に応じて必要なテキストや URL といった情報を送信し、サーバから返信される付加情報を閲覧中のウェブページにポップアップなどの形式で表示する。この様子はブラウザ内の子ブラウザとみることができる。また、元のウェブページの表示を編集することもできるので、本システムを他の様々な用途に用いることも可能である。

Omniget システムの実現手法を説明する前に、まず Omniget によるサービス例を紹介する。

2　Omniget サービス

Omniget システムを利用することによって、閲覧中のドキュメント内容に応じた様々なサービスを受けることができる。ここでは翻訳サービスとセキュリティ警告サービスを紹介する。

2.1　翻訳サービス

図 1 は Omniget 翻訳サービスを利用した際の画面である。ここでは日本語のウェブページを閲覧しており、マウスによるテキスト選択の結果としてポップアップにそのテキストの英語、中国語、韓国語の三つの翻訳が出ている。これらの翻訳文は 3 つの Omniget 翻訳サービスサーバと通信して得られた結果であり、1 つのポップアップに融合され表示されている。また、利用するサービスの数や種類を随時に変更することができる。

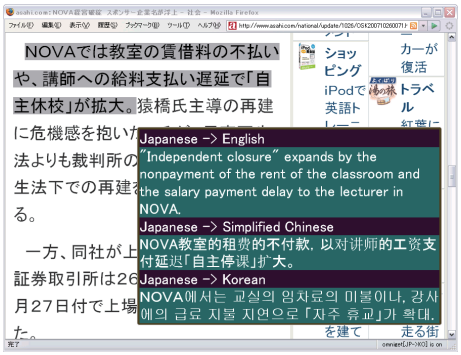

図 1　Omniget 翻訳サービス：日本語から英語、中国語、韓国語へ

本サービスの翻訳部分は既存の翻訳サイトを利用している。Omniget サーバはユーザの選択したテキストを受信すると、そのテキストを翻訳サイトに問い合わせて翻訳文を取得し、それを Omniget クライアントに送信する。

この例でのインタラクションはハイライト (§3.2) を利用したが、マウスオーバーなどを用いることも可能である。マウスオーバーの場合はマウスの指している単語または文を翻訳する。また、単語の難易度や、文の難易度、それにユーザのレベルから、ユーザの一番翻訳したい範囲を推定し、動的に翻訳範囲を決定するための研究も行っている。

2.2　ウェブセキュリティサポート

ネットサーフィン中に、マルウェアを含んだ有害サイトにアクセスすると、ウイルスの感染や、個人情報の流出などの被害にあう恐れがある。またフィッシングサイトにだまされ、クレジットカード番号や金融機関の情報を入力させられるケースもある。これら悪意のあるサイトへのアクセスを防止する策として、ブラウザ上で有害サイトのリンクを無効化するという方法がある。Omniget はこの方法によるサービスの提供が可能である。

本サービスを有効にした場合はウェブページを表示する際に、すべてのリンクは無効化される。ユーザがリンク上にマウスを動かすと、Omniget クライアントが自動的にそのリンクの安全性を Omniget セキュリティサー

バに問い合わせ、リンク先が安全と判断された場合のみ当該リンクが有効になる。安全でないと判断された場合、そのリンクを警告用フォントに変え、ポップアップで警告を表示する。またリンク先を専用の警告ページに変え、そこに詳細情報を表示することも可能である。

Omniget システムでは複数のサービスを同時に利用することができるので、翻訳サービスとセキュリティサポートを同時に利用することが考えられる。Omniget システムはアイデア次第で様々なサービス提供に応用することができる。例えば、記事で取り上げられている会社や商品に関連する株価、ネットショッピングのユーザが望む製品関連情報、商品推薦や口コミ、価格などの情報が手軽に入手可能になる。

3 システムデザイン

3.1 Omniget クライアントとサーバ

最近では、マッシュアップサーバ形式を利用した翻訳サービスをよく見かける。例えば rikai.com[5] の英日翻訳サービスを利用するには、まず、rikai.com のホームページにアクセスする。次に、翻訳したい英語のページの URL をフォームに入力する。最後に GO を押すと、英語ページと翻訳情報をマッシュアップしたページが表示される。このようなサービスはローカルマシンに専用アプリケーションをインストール必要がない反面、サービスを利用するには専用サイトにアクセスしないといけない。これはユーザの負担になる可能性がある。さらに、クロスドメインなどのセキュリティの制限 [4] によって、元ページにあった機能がマッシュアップで生成したページでうまく動作しない可能性がある。またユーザの好みで複数のサービスを利用する場合にも向いていないと考えられる。

これらの不便さを解消する選択肢としてはクライアント側に専用アプリケーションをインストールする方法が考えられる。Omniget クライアントはブラウザの Add-on として実装しており、ユーザがそれをインストールして使用する。Add-on 方式では元ページの機能には影響を及ぼさず、ユーザの好みに応じて複数サービスへの対応も可能である。ただし、Add-on はブラウザの種類 (Firefox、Internet Explorer、Opera、Safari 等) に応じてそれぞれ開発する必要があるという点が問題として指摘されている。この問題を解決するための手法として Bookmarklet[3] がある。ロスアラモス研究所の図書館システムプロジェクト [2] では Bookmarklet を利用して、ユーザ間の情報共有を実現した。ただし、Bookmarklet はユーザの好みの保存が難しく、またシームレスに利用できないという問題もある (ページを更新する毎に Bookmarklet を再実行しないと提供する機能が有効にならない)。常に有効にしなくてよいサービスであれば、Bookmarklet でも十分である。我々は Omniget Bookmarklet も開発している。ただし本論文で説明する Omniget クライアントは Add-on 方式で実装される。

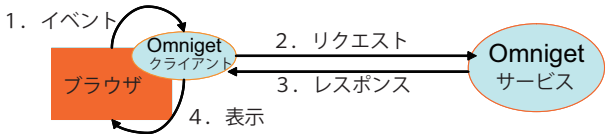

図 2　クライアント・サーバ通信

Omniget サーバは専用プロトコルで Omniget クライアントと通信するサーバであり、Omniget プロトコルを用いて第三者が実装可能である。

3.2 Omniget プロトコル

Omniget プロトコルは Omniget パケットを転送することを目的とした、リクエスト-レスポンス型のプロトコルである。下位セッション層のプロトコルとして HTTP を使用する。

3.2.1 Omniget パケット

Omniget パケットは次の 3 つの要素で構成される：

ID:　パケットの識別番号である。毎回の通信は異なる ID を使い、ペアとなるリクエストパケットとレスポンスパケットのみ同じ ID を持つ。

アクションラベル:　インタラクションの種類を示すラベルである。リクエストとレスポンスはそれぞれ異なるアクションラベルの種類がある。

データコンテンツ:　通信の主体である。

3.2.2 リクエストフェーズ

Omniget クライアント・サーバ通信は 4 ステップとなり (図 2)、イベント検出とリクエスト送信はリクエストフェーズとする。Omniget イベントには下記の 3 種類がある：

MOUSE-STOP:　マウスが動き状態から閾値時間以上の静止状態になる時に発生する。

MOUSE-OVER:　マウスが別の DOM[6] 要素から新たな DOM 要素上に移動し、静止時間が閾値を超えた場合に発生する。MOUSE-STOP のサブセットである。

MOUSE-HIGHLIGHT:　マウスでテキスト文章の一部を選択する時に発生する。

また、キーイベントに関してはセキュリティを考慮 ([1]) し、Omniget イベントに含まれていない。Omniget イベントが検出されると、アクションラベルにイベント名を、データコンテンツにその他の必要情報 (テキスト文章、リンク URL 等) を入れ、Omniget パケットとしてまとめてサーバに送信する。

3.2.3 レスポンスフェーズ

クライアントから送られてきたパケットはサーバ側で解析され、それに対応するレスポンスパケットが生成される。レスポンスパケットのアクションラベルにより

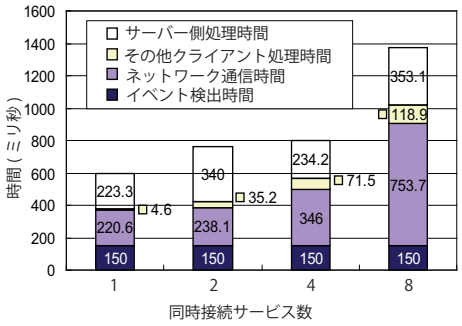

図 3　インターネットに置いたサーバとの通信時間

クライアントが返送結果を表示する方法が伝わる。アクションラベルは次の 3 種類がある：

INCOMPLETE:　サーバがリクエストを処理できなかったことを示す。

POP-UP:　データコンテンツをポップアップで表示する。

MODIFY:　DOM 要素のスタイル属性や内容を編集するようにクライアントに指示する。具体的にはフォントやサイズの変更、リンクやボタンの無効化、削除等がある。

データコンテンツ部分には翻訳文などのサーバアウトプットが含まれる。最後に、クライアントはサーバから返送されたパケットの指示に従い、ブラウザ画面サイズなどを考慮してユーザに情報を提示する。

4　レスポンスタイム評価

クライアント負荷実験とユーザ実験を行い本システムを評価した。この節で Omniget システムの応答時間の測定によりクライアントの負荷耐久性能を評価した。応答時間は次の 4 つの遅延の合計となる：

- Omniget イベント検出時間
- イベント検出以外のクライアント側処理時間
- ネットワーク通信時間
- サーバ側処理時間

Omniget イベントの中、MOUSE-STOP と MOUSE-OVER イベントはマウスが閾値時間以上止まる必要があるので、ユーザにとって長く感じる可能性がある。今度の実験ではデフォルト閾値の 150 ミリ秒で MOUSE-STOP イベントを実験した。

評価には §2 で説明に用いた翻訳サービスを使用した。実験タスクとしてはブラウザ上 MOUSE-STOP イベントで 30 単語ぐらいの英文を Omniget 翻訳サーバに送り、翻訳結果を画面に表示することである。同時接続サーバ数を 1、2、4、8 の 4 通りでそれぞれ 50 回を実験し、平均応答時間求めた。

Omniget サーバが LAN 上にある場合とインターネット上にある場合をそれぞれ測定した。前者の場合、主な応答時間はイベント検出時間とサーバ側処理時間に占められ、8 サーバの場合でもトータル応答時間は 1 秒以内に収まっている。

図 4　テストセットの 1 記事

サーバはインターネット上にある場合の応答時間は主に通信時間に左右され、8 サーバでは、その半分以上は通信時間になる。トータル応答時間は 8 サーバではおおよそ 1.4 秒となり、4 サーバまでは 1 秒以内に収まっている (図 3)。一般ユーザが許容できる時間である。また 4 つぐらい以上の情報を同時に画面に表示するには画面配置の観点から困難であり、4 つ前後の Omniget サービスの同時利用を想定すれば十分である。

5　ユーザ実験

Omniget のポップアップユーザインタフェースの有効性を検証するため、Omniget を利用する場合と既存の翻訳サイト (TW と略す) を利用する場合の比較実験を行った。両サービスが使用する翻訳エンジンが同じものであるから、ユーザインタフェースの違いを評価することができる。

5.1　実験条件

表 1　実験コース: システムとテストセットの組み合わせ

	テストセット X	テストセット Y
コース 1	Omniget(1st)	TW (2nd)
コース 2	TW (1st)	Omniget (2nd)
コース 3	Omniget (2nd)	TW (1st)
コース 4	TW (2nd)	Omniget (1st)

テスト用ドキュメントセット 2 つ (X と Y) を用意した。ドキュメントセット、テストシステム、実験順番の全組み合わせは表 1 に示した 4 つのコースとなる。

テストセットは Google ニュース (US) をもとに作成した。1 つのテストセットは 9 つの記事でを構成される。普段ウェブでニュースを閲覧するときの感覚で閲覧させるためにニュースページのスタイル、写真、リンクをそのままにして、被験者の記事の理解度を測定するための 2 択問題を記事毎に用意した。図 4 は 1 つの記事のサンプルである。

被験者 12 人を 4 つの実験コース (表 1) に 3 人ずつ割り振った。全員母国語が英語でない大学院生である。ユーザの翻訳単位を観察するため、Omniget を利用する

表 2　Omniget と TW のパフォーマンスの比較

	時間 (秒) (avg./std. dev.)	精度 (%)	利用回数
Omniget	683 / 125	88.9%	19.8
TW	733 / 142	83.3%	6.3
比較	Speedup 6.6%	up 5.6%	3.2 倍

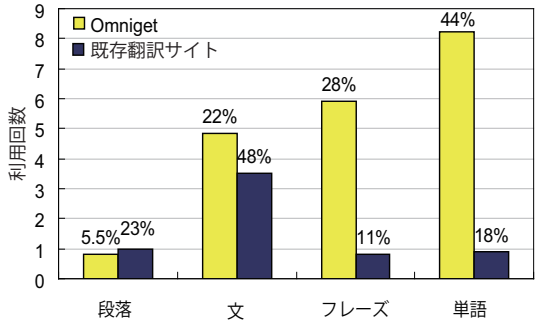

図 5　被験者が選択されたテキストのサイズ

ときハイライトインタラクションのみを有効にした。

実験の流れ：

1. 該当するシステムの説明を読む (Omniget もしくは TW)。
2. サンプルページで練習する。
3. 1 テストセットを読解する。
4. 3 分間休憩を取る。
5. 2 つ目のシステムとテストセットに対して 1 から 3 を行う。

1 セットのテスト時間は概ね 10 から 20 分程度である。

5.2　結果

Omniget と TW システムそれぞれのテストセットの平均完成時間、正解率、補助ツール利用回数を表 2 にまとめた。Omniget を利用した場合は TW より 6.6 %のスピードアップが得られ (有意水準 3 %)、正解率も 5.6 %あがった。この結果は Omniget システム及びポップアップインターフェースの有効性を示している。

表 2 の「平均の利用回数」列にを示したように、Omniget の利用回数は TW の 3 倍以上であるが 6.6 %のスピードアップが得られており、Omniget によるオーバーヘッドが少ないことが分かる。

翻訳されたテキスト長さの分布を図 5 に示した。TW システムは主に文単位で使われているのに対し、Omniget の利用形態は段落以外のところにより分散している。ユーザはより短いテキストで Omniget 翻訳システムを利用する傾向が見える。ただし、TW システムを使う場合は毎回翻訳サイトのアクセスにかかるコストが大きいことから、まとめて、文単位で翻訳されることが多かった。また、グラフから単語単位以外の翻訳ニーズが 50 %以上あることが分かり、Google マウスオーバー辞書などのように単語だけに対応した辞書の限界も伺える。

実験後アンケート調査による評価では Omniget が全項目において勝っている。特に使いやすさとユーザビリティで評価された。またコメントから Omniget の小さい学習・利用オーバーヘッドも評価されていることが分かる。

6　まとめ

本論文は閲覧中のウェブドキュメントに関連する複数の第三者情報をブラウザ内に提示するシステムである Omniget を提案し、インタラクティブなユーザインタフェースをクライアント・サーバ方式で実現した。その上で、クライアントの負荷実験とユーザ評価実験を行った。Omniget クライアントが同時に 4 つのサービスを利用した場合でも、実時間応答が確認された。ユーザ評価実験では Omniget のオーバーヘッドの低さとユーザインタフェースの良さが検証された。またユーザ実験によると、ユーザは、節や句など比較的短い単位ではあるが単語よりは長い単位で Omniget 翻訳を多く利用する傾向にあることが判明した。この長さであれば翻訳誤りも混入しにくく、Omniget 翻訳がユーザにとって読解の助けとなることが分析された。

現在、Omniget をパッケージ化する作業を行っている。このため、セキュリティを考慮した Omniget の実装が必要である。また、Omniget サービスとして新しい可能性を追求している。

参考文献

[1] R. Atterer and A. Schmidt. Tracking the interaction of users with ajax applications for usability testing. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 1347–1350, New York, NY, USA, 2007. ACM Press.

[2] M. D. Giacomo, D. Mahoney, J. Bollen, A. Monroy-Hernandez, and C. M. R. Meraz. Mylibrary, a personalization service for digital library environments. In *Proceedings of the Second DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries. European Research Consortium for Informatics and Mathematics (ERCIM)*, 2001.

[3] Web 2.0 glossary. `http://www.collaboration20.com/wiki/show/Web+2.0+Glossary`.

[4] C. Jackson and H. J. Wang. Subspace: secure cross-domain communication for web mashups. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pp. 611–620, New York, NY, USA, 2007. ACM Press.

[5] T. D. Rudick. Rikai.com. `http://www.rikai.com/`.

[6] L. Wood, M. Champion, S. Byrne, G. Nicol, P. L. Hégaret, J. Robie, and A. L. Hors. Document object model (DOM) level 2 core specification. W3C recommendation, W3C, Nov. 2000. http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113.